3D Reconstruction of Neural Circuits from Serial EM Images

Nina Maack

München 2008

3D Reconstruction of Neural Circuits from Serial EM Images

Nina Maack

Dissertation

zur Erlangung des Doktorgrades
der Naturwissenschaften (Dr. rer. nat.)
der Fakultät für Biologie
der Ludwig–Maximilians–Universität München

Angefertigt am Max-Planck-Institut für Neurobiologie
Abteilung "Neuronale Informationsverarbeitung"

vorgelegt von
Nina Maack
aus Landsberg/Lech

München, den 29.04.2008

Erstgutachter: Professor Dr. Alexander Borst
Zweitgutachter: Professor Dr. Benedikt Grothe
Tag der mündlichen Prüfung: 20.06.2008

Table of Contents

List of Figures

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
API	Application Programming Interface
BSE	Backscattered Electrons
DTW	Dynamic Time Warping
EM	Electron Microscope
EMD	Elementary Motion Detector
ESEM	Environmental Scanning Electron Microscope
GUI	Graphical User Interface
LMCs	Large Monopolar Cells
LPTCs	Lobula Plate Tangential Cells
NLDF	Nonlinear Diffusion Filter
NP	Non-deterministic Polynomial time
OpenGL	Open Graphics Library
PB	Phosphate Buffer
RAM	Random Access Memory
SBFSEM	Serial Block-Face Scanning Electron Microscopy
SDF	Signed Distance Function
SE	Secondary Electrons
SEM	Scanning Electron Microscopy
TEM	Transmission Electron Microscopy
XML	Extensible Markup Language

Abstract

A basic requirement for reconstructing and understanding complete circuit diagrams of neuronal processing units is the availability of electron microscopic 3D data sets of large ensembles of neurons. A recently developed technique, "Serial Block Face Scanning Electron Microscopy" (SBFSEM, Denk and Horstmann 2004) allows automatic sectioning and imaging of biological tissue inside the vacuum chamber of a scanning electron microscope. Image stacks generated with this technology have a resolution sufficient to distinguish different cellular compartments, including synaptic structures. Such an image stack contains thousands of images and is recorded with a voxel size of 23 nm in the x- and y-directions and 30 nm in the z-direction. Consequently a tissue block of 1 mm^3 produces 63 terabytes of data.

Therefore new concepts for managing large data sets and automated image processing are required. I developed an image segmentation and 3D reconstruction software, which allows precise contour tracing of cell membranes and simultaneously displays the resulting 3D structure. The software contains two stand-alone packages: Neuron2D and Neuron3D, both offering an easy-to-operate graphical user interface (GUI).

The software package Neuron2D provides the following image processing functions:

- Image Registration: Combination of multiple SBFSEM image tiles.
- Image Preprocessing: Filtering of image stacks. Implemented are Gaussian and Non-Linear-Diffusion filters in 2D and 3D. This step enhances the contrast between contour lines and image background, leading to a higher signal-to-noise ratio, thus further improving detection of membrane borders.
- Image Segmentation: The implemented algorithms extract contour lines from the preceding image and automatically trace the contour lines in the following images (z-direction), taking into account the previous image segmentation. They also permit image segmentation starting at any position in the image stack. In addition, manual interaction is possible.

To visualize 3D structures of neuronal circuits the additional software Neuron3D was developed. The program relies on the contour line information provided by Neuron2D to implement a surface reconstruction algorithm based on dynamic time warping. Additional rendering techniques, such as shading and texture mapping, are provided.

The detailed anatomical reconstruction provides a framework for computational models of neuronal circuits. For example in flies, where moving retinal images lead to appropriate course control signals, the circuit reconstruction of motion-sensitive neurons can help to further understand the neural processing of visual motion in flies.

1 Introduction

The ability of animals to navigate in the world relies heavily on the processing of visual information. Whenever an animal moves through its environment or an object moves past it the visual system is challenged with motion. By examining large sections of the changing retinal image it is possible to make good estimates of ones self motion in comparison to both the landscape and moving objects of interest. The optic flow, describing the continuous changes of the retinal images, is not explicitly represented in the brightness patterns of the retinal images. In contrast, the information is computed from the temporal brightness changes in the retina. Therefore it is highly interesting how the brain is organized to analyze visual patterns and how behaviorally relevant motion information is extracted.

1.1 Neural Circuit Reconstruction

The reconstruction of complete circuit diagrams of neuronal processing units is necessary to understand the connectivity in neuronal networks and the principles of brain design. In addition to the detailed wiring diagrams, activity patterns related to computational processes of neuronal populations are required. These are obtained with optical and electrical recording techniques (e.g. Buzsaki 2004) and are used to relate the circuit connectivity to observed signals. Finally the aim is to understand how electrical activity in neuronal circuits generates behavior. So far only one organism's wiring diagram exists: that of the nematode *Caenorhabditis elegans* (White et al., 1986). Further reconstructions of complete circuit diagrams of neuronal processing units, e.g. the vertebrate cortical columns, hippocampal circuits or entire invertebrate nervous systems still must be obtained.

One representative for circuit reconstruction of invertebrate nervous systems are flies, where neural processing of visual motion is one of the sensory components involved in flight control. To understand the neural mechanisms leading from moving retinal images to appropriate course control signals, the underlying neural network of the fly visual system needs to be analyzed. Anatomically, the fly visual system is divided into three layered

structures (fig. 1.1A): lamina, medulla and lobula complex. Each of these structures is composed of repeated columns, where each column corresponds to the relative position of facets in the eye and consists of stereotyped, layer specific ensembles of neurons (fig. 1.1B). The lobula complex is divided into the lobula and the lobula plate. In the lobula plate, one prominent group of 60 neurons, the lobula plate tangential cells, are individually identifiable (LPTCs, fig. 1.1C and blue neuron 1.1B). These neurons are motion sensitive visual interneurons. With their large dendrites, LPTCs spatially integrate the output signals of columnar neurons (fig. 1.1B-C) and respond to visual motion within large parts of the visual field in a directionally selective way. So far most of the neural cell types, e.g. LPTCs, are well-characterized at a resolution accessible with light microscopy (Strausfeld, 1984). However the presynaptic processing of LPTCs underlying local motion detection is unknown. To understand this presynaptic circuit it is necessary to identify the spatial arrangement and synaptic connectivity of these cells within and across the columnar network.

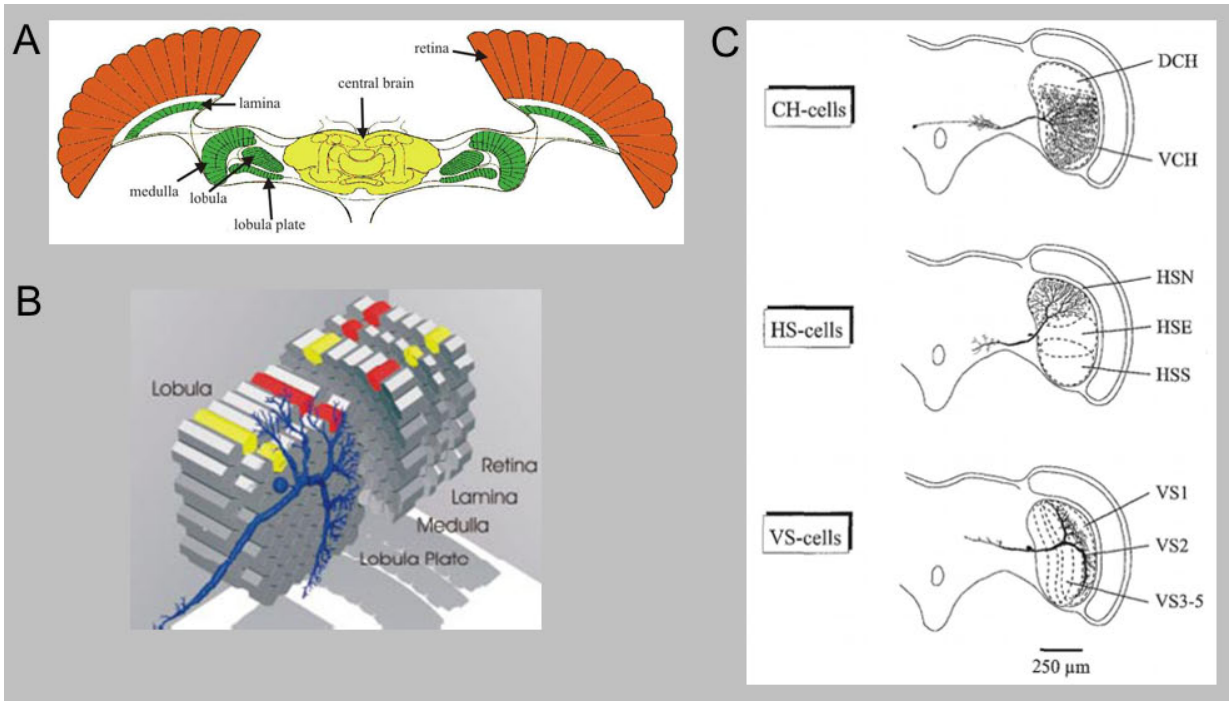


Figure 1.1: Visual system of the fly
(A) Schematic horizontal cross-section through the fly head. The three layers of the optic ganglia are indicated in green (modified from Borst and Haag 2002). (B) 3D schematic of the visual ganglia together with a LPTC, displayed in blue. The columns in each layer can be seen to represent the facets of the retina in a one-to-one fashion leading to a retinotopic projection of the visual surround onto the dendrites of LPTCs (modified from Borst and Haag 2002). (C) Subset of lobula plate tangential cells (Neurobiotin filled cells from *Calliphora vicina*, modified from Borst and Haag 1996).

1.2 Microscopy Techniques

The reconstruction of complete neuronal circuits is based on the anatomical knowledge of single neurons, as well as their pre- and postsynaptic connections. This implies that both high resolution and a large field of view must be combined, as for example vesicles of a chemical synapse have a diameter of approximately 25 nm. The techniques available for the reconstruction of large tissue volumes are dictated by the required resolution.

3D light microscopy, such as confocal (Conchello and Lichtman, 2005) or two-photon microscopy (Denk et al., 1990), is limited by the wave length of the light and hence axons or dendrites with a diameter significantly lower than the resolution limit cannot be properly resolved (lateral and axial resolution of about 200 nm). A higher lateral and axial resolution of about 100 nm are obtained with technologies such as the stimulated emission depletion (STED, Hell 2003), where the nonlinear interaction between two light sources is used to gain a higher resolution. However, so far only electron microscopy (EM) techniques are suitable to reconstruct neural circuits in their entirety with a resolution sufficient to distinguish tiny processes, such as synaptic connections. Traditionally, three dimensional reconstruction of neural tissue has been achieved by serial section transmission electron microscopy of ultrathin sections (SSTEM, Stevens et al. 1980). An alternative is serial block-face scanning electron microscopy (SBFSEM, Denk and Horstmann 2004), in which image acquisition is automated and subsequent-section alignment is irrelevant. The third possibility to reconstruct 3D neural tissue is serial section electron tomography (SSET, Soto et al. 1994), which is based on the principle of reconstructing a 3D structure from multiple 2D projections at varying angles. The method is similar to computed tomography (CT) scans in medical radiology (Frey et al., 2006). In the following sections, 1.2.1 to 1.2.2, only serial section transmission electron microscopy and serial block-face scanning electron microscopy will be described further, as serial-section reconstruction in comparison to SSET is better suited for tracing continuity over long distances throughout the depth of tissue.

1.2.1 Serial Section Transmission Electron Microscopy

For the past 30 years serial section transmission electron microscopy (SSTEM, Stevens et al. 1980) has been well-established for studies of single neurons in 3D with a lateral resolution of less than 10 nm and a section thickness of about 50 nm. SSTEM is a conceptually simple technique (fig. 1.2A, Reimer 1993). Briefly, the fixated, embedded tissue block is

serially sectioned with a diamond knife on an ultramicrotome. The staining process can be performed before (block staining) or after sectioning (section staining). Ribbons of sections are transferred onto grids for imaging with a conventional transmission electron microscope (TEM, fig. 1.2B), in which the thin specimen is irradiated with an electron beam of uniform current density. The electron energy ranges from 60-150 keV. Electrons are emitted from the electron gun in vacuum mode by thermionic emission from tungsten hairpin cathodes or by field emission from pointed tungsten filaments. A two-stage condenser-lens system permits variation of the illumination aperture and the area of the specimen illuminated. The electron-intensity distribution behind the specimen is imaged with a three- or four-stage lens system, onto a fluorescent screen. The image can be recorded by direct exposure of a photographic emulsion inside the vacuum chamber or digitally by CCD (charge-coupled device) or TV cameras. The image contrast is due to the increased elastic scattering of electrons in areas containing the heavy metal stain (Reimer, 1993).

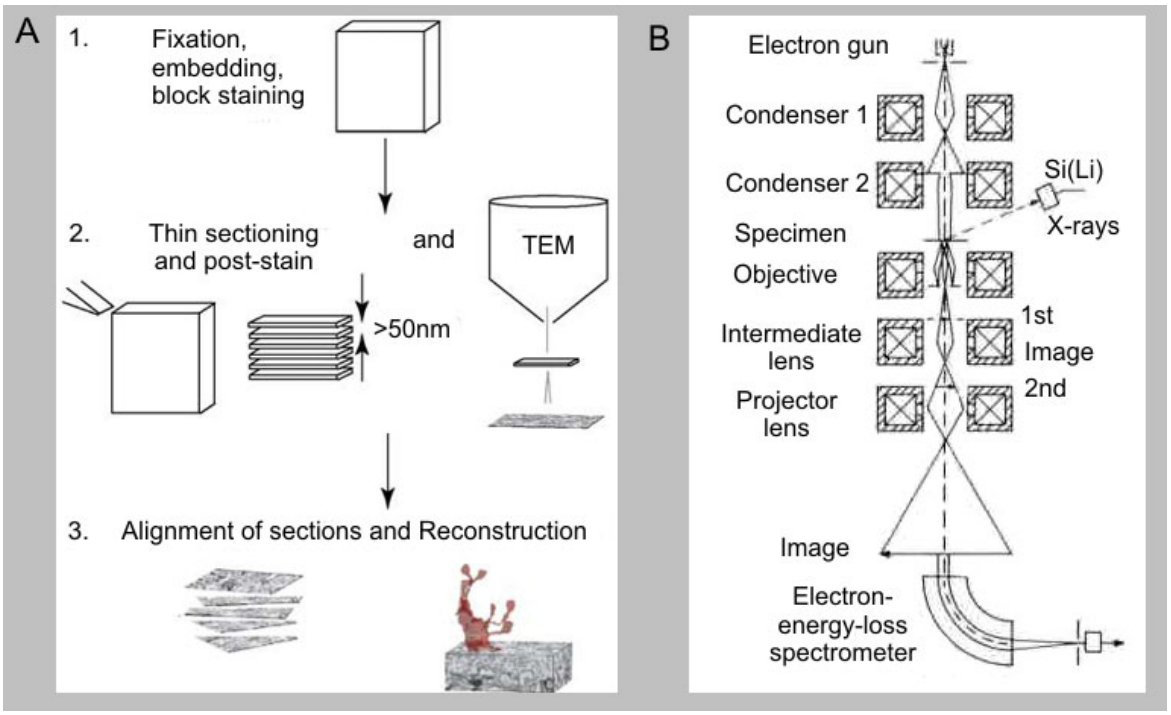


Figure 1.2: Technique of SSTEM and TEM
(A) Schematic diagram of the steps involved in the acquisition of tissue volumes using SSTEM (modified from Briggman and Denk 2006). (B) Schematic ray path for a TEM, equipped for additional x-ray and electron energy-loss spectroscopy (modified from Reimer 1993).

Generally the sectioning and transfer processes of SSTEM are susceptible to many prob-

lems including loss of sections, uneven section thickness, debris on sections and geometrical distortion. Even if the sections can be successfully imaged, distortions, which vary locally and between sections, hamper automated approaches for the alignment and reconstruction of fine neuronal processes or complete circuits.

1.2.2 Serial Block-Face Scanning Electron Microscopy

A recently developed technique in EM serial-reconstruction is serial block-face scanning electron microscopy (SBFSEM, Denk and Horstmann 2004; Leighton 1981). SBFSEM automates the process of sectioning and imaging blocks of tissue by incorporating a custom-made microtome into the vacuum chamber of a scanning electron microscope (SEM). Unlike a TEM, the images in an SEM are generated from electrons scattered off the surface of an embedded tissue sample, which allows the imaging of block faces. One major component of an SEM is the electron column (fig. 1.3A), which consists of an electron gun and two or more electron lenses. The electron lenses influence the paths of electrons traveling down an evacuated tube. At the base of the column are typically pumps that produce a vacuum of about 10^{-6} Pa. The electron gun generates electrons and accelerates them to an energy in the range of 0.1 - 30 keV (Goldstein et al., 2003). The beam emerges from the final lens into the specimen chamber, where it interacts with the specimen to a depth of approximately 1 µm and generates signals used to form an image. The prevalent image mode in the SEM is the detection of secondary electrons (SE), which are emitted when the primary electron beam strikes the sample surface. The SE signal depends strongly on the orientation of the surface, leading to topographic images that characteristically resemble obliquely illuminated solid objects. Since the microtome-cut block surface is devoid of topographic features, very little contrast is generated (Kuzirian and Leighton, 1983). Therefore for SBFSEM the image formation is based on a second type of generated electrons, the backscattered electrons (BSE). The BSE signal is strongly depended on the mean atomic number (Z) of the sample material, since atoms with a high atomic number scatter more effectively than light ones (Goldstein et al., 2003). Hence, the backscattered coefficient, η, provides information about the material composition of a specimen (fig. 1.3B).

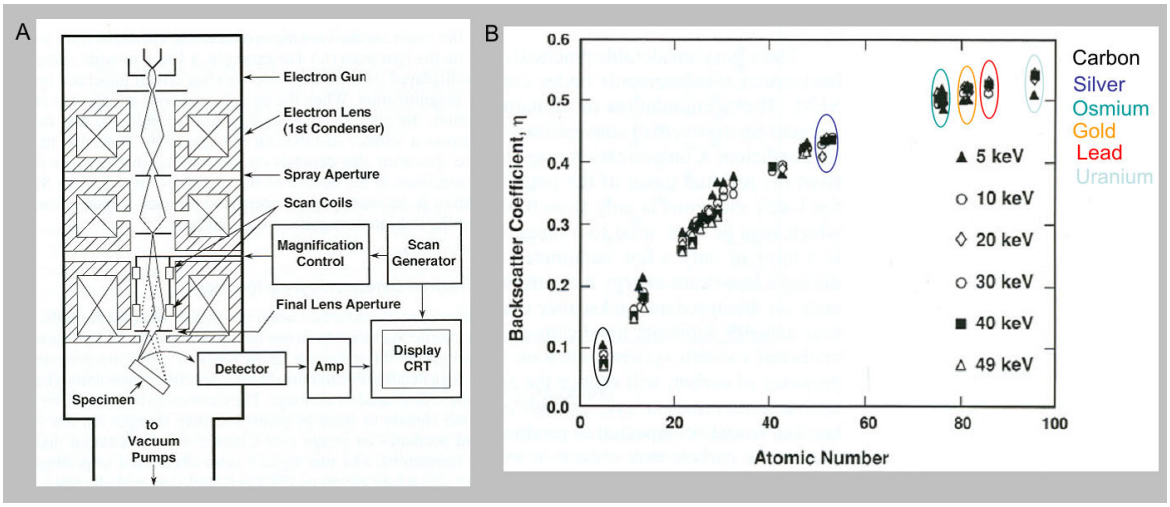


Figure 1.3: Schematic of SEM and BSE coefficient
(A) Schematic drawing of the electron column, showing the electron gun, lenses, the deflection system and the electron detector (modified from Goldstein et al. 2003). (B) Backscattered electron coefficient as a function of atomic number plotted for a range of beam energies from 5 to 49 keV. Backscatter coefficient of selected chemical elements are marked (modified from Goldstein et al. 2003).

One major requirement for a conventional SEM is that both the electron-optic column and the specimen chamber are kept under high vacuum of about 10^{-6} Pa. Although this condition ensures that the electron beam can travel unimpeded from the source to the sample, many nonconducting specimens, such as biological tissue, cannot be observed in their natural state in the SEM. As it is required of SBFSEM to image uncoated block surfaces, an environmental scanning electron microscope (ESEM, Donald 2003) instead of a conventional SEM is used. Hereby, a gaseous environment is maintained around the sample during imaging, although the electron gun itself is kept at the standard pressure of around 10^{-6} Pa. This is achieved by the use of differential pumping within the column, creating a series of different pressure zones, with increasing pressures as the sample chamber is reached (fig. 1.4). In addition the gaseous environment around the sample compensates charge build-up at the surface of the specimen. Hereby charging, which occurs when the energy from the primary electrons is retained by the sample instead of being shed to an electrical ground, is avoided by gas ionization in the sample chamber. The resulting ionization of the gas molecules with the emitted electrons creates additional electrons, amplifying original secondary electron signal and positive ions. The positive ions are attracted to the sample surface, that is negatively charged from the primary electron beam. The resulting neutralization suppresses charging artifacts. Consequently ESEM allows imaging of samples in low-pressure gaseous environments and high relative humidity

(up to 100%). To avoid the scattering of electrons, the total distance electrons must travel through the gas is minimized; thus, most incident electrons do not undergo any large-angle collisions. Consequently the ESEM allows imaging of cells and other biological assemblies including whole organisms without dehydration or coating: only staining or fixation of the specimen block is necessary.

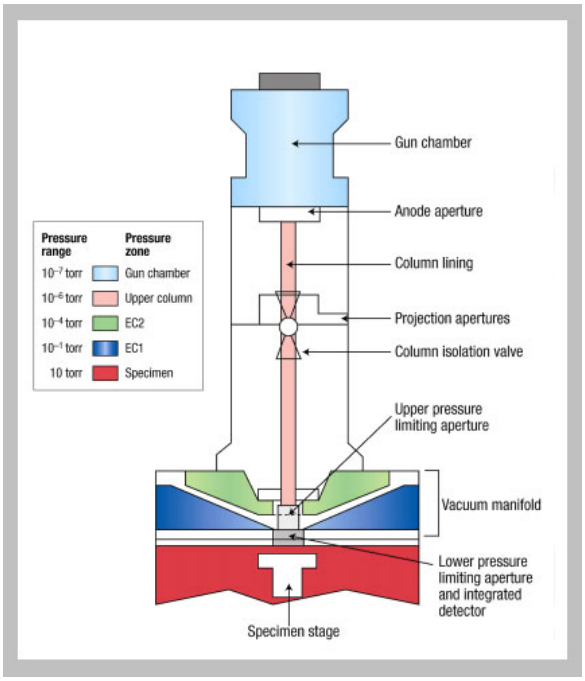


Figure 1.4: Schematic of ESEM
The different pressure zones in the ESEM column. The gun can be maintained at high vacuum, while a system of differential pumping- and pressure-limiting apertures allows the chamber to be maintained at a pressure of a few torr (modified from Donald 2003).

As the SBFSEM technology uses an ESEM, the microtome incorporated into the vacuum chamber, must be compatible with low vacuum conditions. Additionally to ensure alignment of subsequent images, which is one of the primary objectives of serial block-face imaging, the position of the block must remain stable or be returned to the same location after each image is taken. Therefore the designed knife, rather than the block, is moved for cutting. In the z-direction, sample advance rather than knife advance is used in order to keep the cutting plane, and hence the location of the block surface, constant. This removes the need for refocusing the ESEM, which contributes to image stability and registration (Denk and Horstmann, 2004). The lateral position jitter is typically below 10 nm. Adequate contrast is obtained by block staining, using for example uranyl acetate and osmium tetroxide. The maximum lateral resolution is about 10 nm and it is possible to cut series

of thousands of sequential sections with a section thickness of around 30 nm. The detailed working principle is shown in figure 1.5 as well as the mechanical design for the in-chamber microtome.

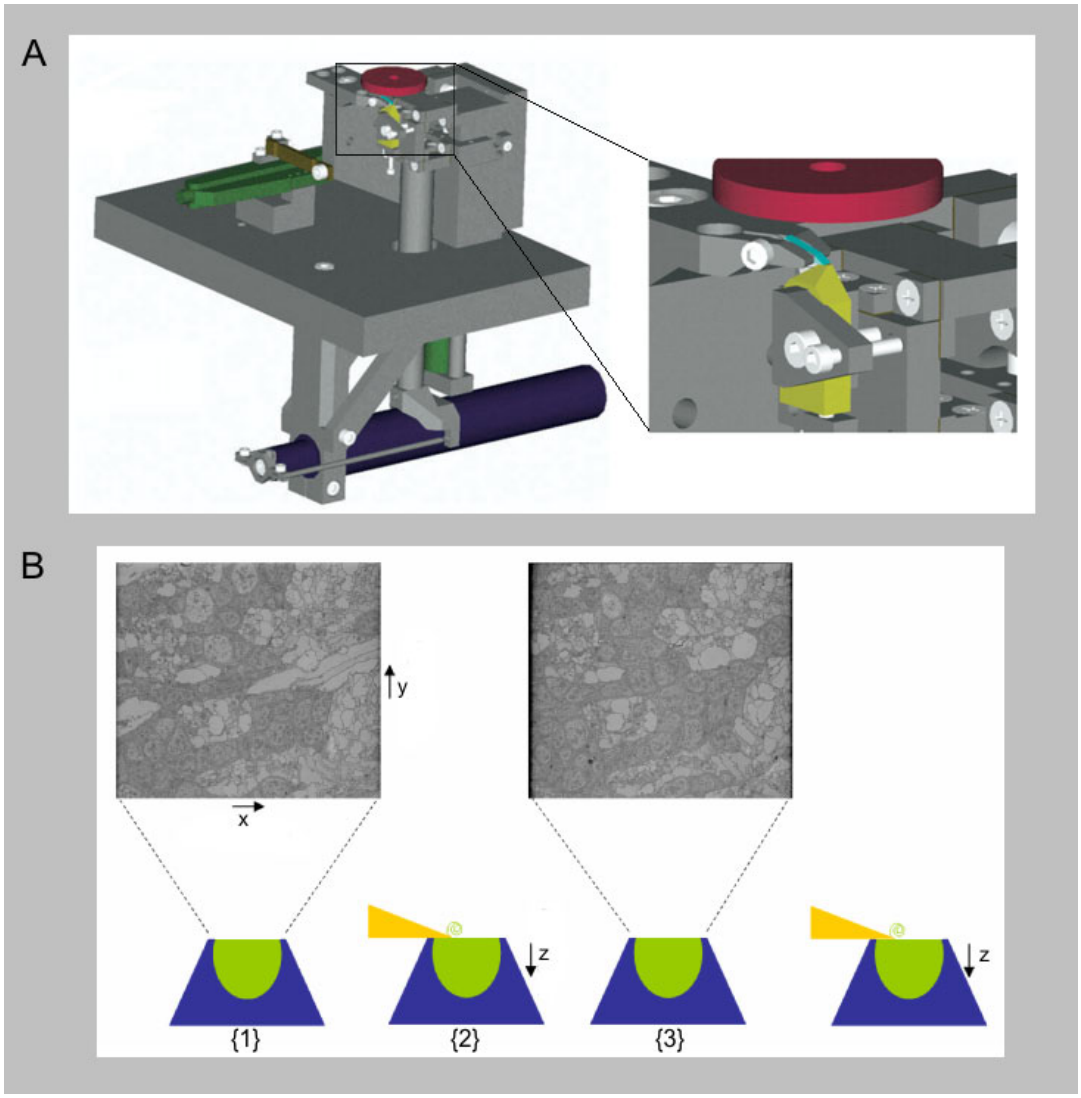


Figure 1.5: Schematic of SBFSEM and working principle
(A) The mechanical design for the in-chamber microtome is shown in an overview and a zoomed view of the diamond knife (light blue) and sample (amber). Most parts are nonmagnetic stainless steel (gray). The BSE detector (red) is depicted schematically above the sample. A piezoelectric actuator (green) drives the knife holder back and forth. A direct-current-motor-driven micrometer (dark blue) drives the sample advance via a lever (modified from Denk and Horstmann 2004). (B) Principle of SBFSEM operation: (1) an SEM image is taken of the surface of the plastic-embedded (blue trapezoid) tissue preparation (green). (2) Then with a diamond knife (yellow) an ultrathin slice is cut off the top of the block. (3) After retraction of the knife, the next picture is taken. The pictures shown are from an image stack of the outer chiasm of the blowfly, but are not successive slices (modified from Denk and Horstmann 2004).

There are several advantages of SBFSEM over traditional SSTEM (section 1.2.1). Because the images are taken directly from the block face prior to each cut, the problems of sections being distorted or lost during handling are completely avoided. Furthermore the images in raw SBFSEM datasets are already aligned and therefore automated image analysis and reconstruction are possible. Additionally, as the sectioning process is fully automated, large volumes can be imaged without significant operator involvement.

Generally, however, the area to be imaged is many times larger than the field of view of the ESEM at the required resolution. Hence it is necessary to take multiple images to cover block faces as large as 500 µm. Thus the microtome, needs to be mounted on a translation stage with a mechanical reproducibility better than the lateral resolution of 10 nm in order to maintain the alignment within and between subimage stacks.

The speed of SBFSEM is ultimately limited by the voxel dwell time necessary to achieve a reasonable signal-to-noise ratio. The voxel dwell time, τ_d, can be calculated with the following formula (Denk and Horstmann, 2004):

$$\tau_d = \frac{(R_{SN})^2 e}{\eta I_b}, \tag{1.1}$$

where R_{SN} = signal-to-noise ratio, e = elementary charge, η = backscattering coefficient and I_b = beam current.

Using the values $R_{SN} = 100$, $I_b = 1$ nA and $\eta = 0.2$ the calculated voxel dwell time τ_d is 8 µs. At this rate the time required to collect a tissue volume of 0.01 mm^3 (total brain volume of *Drosophila melanogaster*) at a resolution of 20 x 20 x 30 nm^3 would be about 2.5 months and the required memory in the range of 550 gigabyte. Using the SBFSEM technique for the analysis of larger organisms such as the blowfly *Calliphora vicina*, with a total brain volume of 1 mm^3, the scan time is about 21 years and 54 terabyte memory capacity is required. However, it is possible to analyze only interesting regions of the brain, e.g. the visual ganglia, which leads to a significant reduction of the scan time and the memory capacity.

In principle with the lateral (10 nm) and horizontal (30 nm) resolution limit of SBFSEM it is possible to identify plasma membranes (diameter of approx. 20 nm) and chemical synapses (diameter of vesicles between 20 to 30 nm). Without additional labeling electrical synapses, with a diameter of 1 to 7 nm, are currently not detectable with the SBFSEM. Shortcomings of the SBFSEM technique are the low vacuum mode, block staining (no possibility to enhance contrast after cutting) and the low R_{SN} value. Overall ultrastructural

3D anatomical data of e.g. neuronal circuits of the fly brain can be obtained with the SBSFSEM technology.

1.3 Goals

So far the fly visual system is well described at a resolution accessible with light microscopy (Strausfeld, 1984). To gain further insights into the circuits underlying visual motion processing in flies (Borst and Haag, 2002) knowledge at the ultrastructural level is necessary. To obtain detailed anatomical information of the underlying neuronal circuitry, we apply the SBFSEM technique (section 1.2.2). This technique, in comparison to other methods, allows automated sectioning and imaging of biological tissue with a resolution sufficient to trace even thin neuronal processes and visualize sites of chemical synaptic contact. As mentioned above an enormous amount of data is produced during the imaging of a specimen block with the SBFSEM technique. Therefore, new concepts for managing large data sets and for automated 3D reconstruction algorithms needed to be developed. To allow the best possible analysis of SBFSEM image stacks, it was necessary to develop custom-made software, that achieves the following requirements:

- Efficient (time and space) data handling and storage:
  Images are segmented sequentially rather than the whole stack at once.
- Semi-automated image segmentation:
  - Development of segmentation algorithms for precise contour tracing of cell membranes.
  - Segmentation of a varying number of objects per image.
  - Automated segmentation of splitting objects, branch points of e.g. dendrites.
  - Segmentation of neurons with process diameters varying from approx. 50 nm to several µm.
- Visualization of 3D structures of neuronal circuits parallel to image segmentation:
  Segmentation errors are located and corrected in an early phase of data analysis.

In addition the developed algorithms must be embedded into an easy-to-operate graphical user interface (GUI). In the following, I will describe the software I developed for the analysis of SBFSEM image stacks.

2 Materials and Methods

2.1 Specimen Preparation for SBFSEM

2.1.1 Fly Brain Preparation and Fixation

Female blowflies, *Calliphora vicina*, were briefly anesthetized with CO_2 and mounted ventral side up with wax on a small preparation platform. The head capsule was opened from behind. Finally trachea and air sacs that cover the lobula plate were removed. The tissue was prepared as described in Brotz et al. (Brotz et al., 1995). In a first step the head capsule was filled with a fixative solution (4% paraformaldehyde, 0.3% glutaraldehyde, 1% sucrose in 0.1 M phosphate buffer, pH 7.2). After a brief incubation time of 3 minutes the brain was carefully removed with a shear from the head capsule and kept in an Eppendorf cup with fresh fixative solution overnight at 4°C. Afterwards the two optic lobes were dissected and washed three times for ten minutes in 0.1 M phosphate buffer, pH 7.2, and 1% sucrose at room temperature.

2.1.2 Staining

Unless noted otherwise all steps were carried out at room temperature. Staining of samples for the SBFSEM technique was done using established procedures common for preparing TEM samples (Hayat 2000, chapter 6 p.342ff). The tissue was postfixed for 2 hours in a solution containing 2% Osmiumtetroxide (SERVA Electrophoresis GmbH, Heidelberg, Germany) reduced with 0.8% potassium ferricyanid in 0.1 M PB, pH 7.2, on a rotationary wheel. Afterwards, the specimen was washed once for 10 minutes in 0.1 M PB, pH 7.2, and three times for 10 minutes in a 25% methanol-water (double distilled) mixture. Then the tissue was subjected to a block contrast enhancement step by soaking it for 12 hours in a solution of 4% uranyl acetate (Sigma-Aldrich Chemie GmbH, Munich, Germany) in a 25% methanol-water (double distilled) mixture (Stempak and Ward, 1964) in the dark. After that the tissue was dehydrated in a methanol sequence (2x 25% for 20 min, 2x 70% for 30

min, 2x 90% for 30 min, 1x 95% for 30 min, 2x 100% for 30 min) and incubated two times for 10 minutes in propyleneoxide. These steps were followed by infiltration of the epoxy resin monomer (Epon 812 mixture, Serva Electrophoresis GmbH, Heidelberg, Germany) into the specimen. Infiltration steps were: 1:1 Epon 812/propyleneoxide mixture, for 4 hours; 3:1 Epon 812/propyleneoxide mixture, overnight at 4°C; pure Epon 812 for 3 hours. The epoxy resin was polymerized for 72 hours at 60° in flat embedding molds.

All chemicals, unless otherwise noted, were obtained from Sigma (Sigma-Aldrich Chemie GmbH, Munich, Germany).

2.1.3 Final Preparation of SBFSEM Sample

The block face of the resin block, containing the specimen, was trimmed to a pyramid of dimension 500 x 500 x 500 µm^3 with an EM high speed milling system (Leica EM Trim, Leica Microsystems GmbH, Wetzlar, Germany) and a conventional ultramicrotome (Reichert & Jung Ultracut, Leica Biosystems Nussloch GmbH, Nußloch, Germany). Light microscopy of semithin (1 µm) sections stained with toluidine blue and SEM images of the untrimmed block face were used to select the desired field of view before the final trimming step producing the desired small cutting pyramid with an edge angle of 45° (Denk and Horstmann, 2004).

2.1.4 Microscopy

The SBFSEM consists of an environmental SEM with a field-emission electron gun (QuantaFEG 200, FEI, Eindhoven, Netherlands) and a custom-made microtome (Winfried Denk, MPI for Medical Research, Heidelberg, Germany, Denk and Horstmann 2004). The microtome is located in the low-vacuum chamber of the SEM. The entire SBFSEM experiment is controlled by custom-made software (Winfried Denk, MPI for Medical Research, Heidelberg, Germany). The principles of the SBFSEM operation are:

1. An SEM image is taken of the surface of the plastic-embedded tissue preparation by detecting the backscattered electrons.

2. An ultrathin slice is removed from top of the block with a diamond knife.

3. After retraction of the knife the next picture is taken.

For a detailed description see section 1.2.2 and Denk et al. (Denk and Horstmann, 2004).

One image stack from the *Calliphora vicina* outer chiasm, which was used as the basis for the software development, was recorded by Alexander Borst (MPI of Neurobiology, Martinsried, Germany) and Winfried Denk (MPI for Medical Research, Heidelberg, Germany) under the following conditions:

- Low vacuum mode (environmental SEM): 50 Pa.
- Beam energy: 4 keV.
- Spot size: 3.5.

The detailed properties of this image stack are:

- Digital resolution: 26.40 nm/pixel.
- Scan field (x-, y-direction): 2048 x 1768 pixels, equivalent to 55 x 47 µm^2.
- Total number of images: 1000.
- Slice thickness: 50 nm.
- Extent in the z-direction: 50 µm.
- Voxel dwell time: 12.5 µs.
  The voxel dwell time is calculated with formula 1.1.

A volume reconstruction as well as some example images of this stack are represented in figure 3.1.

Most of the steps described in section 2.1 were performed by Christoph Kapfer and Marianne Braun (MPI of Neurobiology, Martinsried, Germany).

2.2 Software Development Tools

2.2.1 Hardware

The software was developed on a 32-bit workstation (Intel(R) Xeon Processor, operating system: Windows XP32) with an ATI FireGL V3100 graphics board and 2.00GB RAM. All software tests were executed on a 64-bit workstation (64-Bit-Dual-Core Intel Xeon Processors, operating system: Windows XP64) with an NVIDIA Quadro FX 3500 graphics board and 8.00GB RAM.

2.2.2 Programming Languages

Depending on the requirements of each software part, the following programming languages were used:

- Java (Sun Microsystems, California, United States; Java^{TM} SE Development Kit 6): Object-oriented programming language, used for the development of all GUIs. Additionally the following APIs were used: Java Advanced Imaging API (version 1.1.3) for image analysis, JDOM (version 1.1) for accessing, manipulating, and outputting XML data, Java Media Framework API (version 2.1.1) for movie generation and JMatLink (version 1.3.0) for connecting Java and Matlab.

- Jogl (Sun Microsystems, California, United States; Java^{TM} Binding for the OpenGL^{R} API, version 1.1.0): External OpenGL-programming library for the programming language Java, used for implementation of hardware-supported 3D graphics to model and render 3D-scenes.

- Matlab (Mathworks, Michigan, United States; version 7.5.0(R2007b)): High-level technical computing language and interactive environment for algorithm development, used for implementation of image segmentation, registration and filter algorithms.

- C (Borland Compiler, California, United States; version 5.5): Imperative programming language, used for the development of application programming interfaces (WinAPI, 32 & 64 bit).

The entire software project has been developed with Eclipse SDK (Eclipse Foundation, Inc, Canada, version 3.2.1), an open source development platform.

3 Results

3.1 SBFSEM Data

The SBFSEM data which was mainly used as the basis for software development, was taken from the blowfly *Calliphora vicina*. The image stack was taken from the outer chiasm, a brain region located between two of the five retinotopic neuropils of the fly visual system: the lamina and the medulla (fig. 3.1A). The outer chiasm contains as the most prominent structures long visual axons, which project from the lamina to the medulla, so called large monopolar cells (LMCs, Strausfeld and Douglass 2003, Meinertzhagen and O'Neil 1991). A detailed description of the specimen preparation and recording technique is summarized in chapter 2. One feature of the staining method is highlighted in the zoomed inset of figure 3.1B. Comparing the cytoplasm (★) with the membranes (←), membranes show a higher backscattered contrast and consequently appear darker. Therefore, a precise border between cytoplasm and extracellular space exists, that can be used for automated image segmentation. The volume reconstruction (fig. 3.1C) of 27 from 1000 slices, each 50 nm thick, visualizes the possibilities given by the SBFSEM technique: without any prior alignment of consecutive images in x-, y- direction the axonal structures of e.g. LMCs can be followed in z-direction.

In the following sections 3.2 to 3.3.3 the development of the custom-made software will be described in detail. In general the software contains two stand-alone packages: Neuron2D (section 3.2) which allows for a precise contour tracing of cell membranes and Neuron3D (section 3.3) which displays the resulting 3D structure.

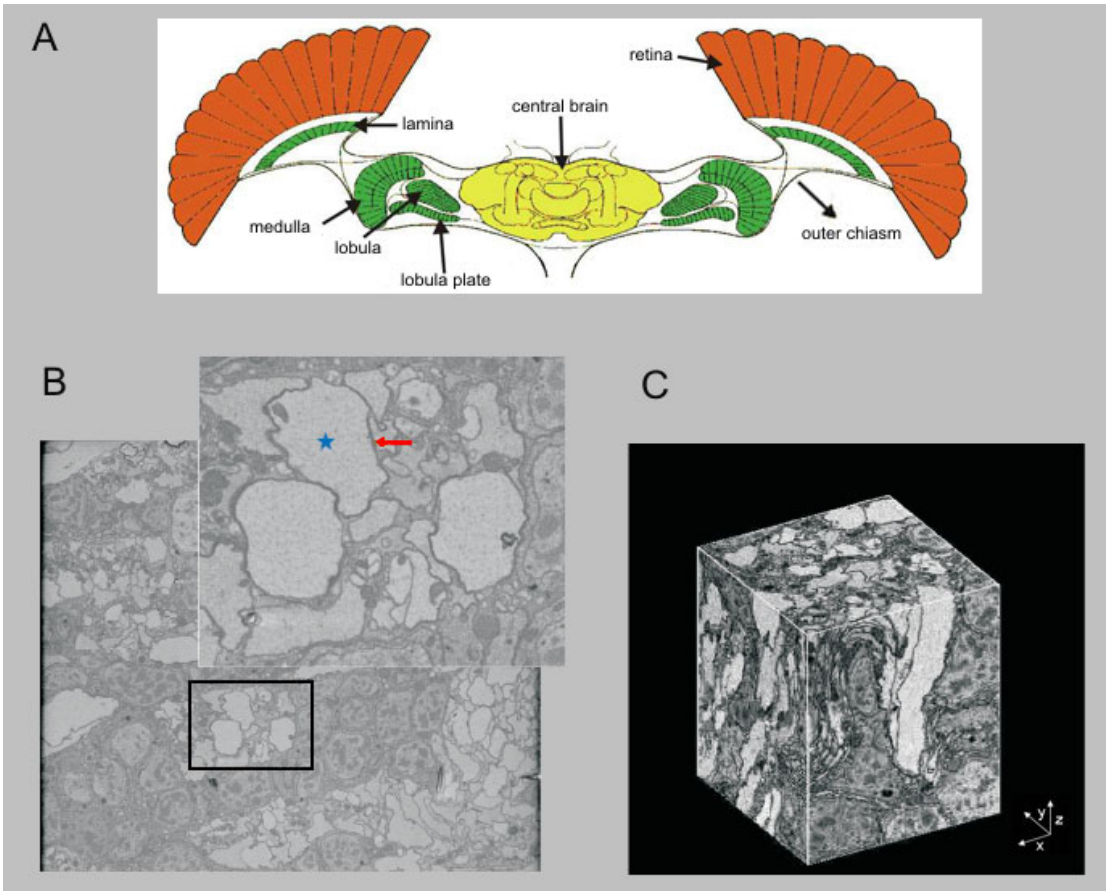


Figure 3.1: Example SBFSEM image stack
(A) Horizontal cross-section through the fly head (modified from Borst and Haag 2002): displayed in red the retina, in yellow the central brain and in green the fly visual system, which consists of three successive visual neuropils, the lamina, the medulla and the lobula-complex. The latter is divided into the lobula and lobula-plate. (B) Sample SBFSEM image, size 2048 x 1768 pixels (equal to 55 x 47 µm^2). The inset shows a magnification of large processes of lamina monopolar cells (average diameter of 3.5 µm). (C) Volume reconstruction (14 x 14 x 27 µm^3, slice thickness 50 nm), done by Alexander Borst (MPI of Neurobiology, Martinsried, Germany) with IDL functions (ITT Visual Information Solutions, Colorado, USA).

3.2 Software - Neuron2D

The software package Neuron2D provides the possibility to extract contour lines from one image and to automatically trace those in the following images (z-direction). It includes an easy-to-operate GUI and provides three main image processing functionalities (image registration, filtering and segmentation), which are described in section 3.2.1 to 3.2.3. In addition, an elementary image viewer is integrated, where images are displayed in sequential order.

3.2.1 Image Registration

The goal of image registration is to match two or more images in x-, y- direction, so that identical coordinate points in these images correspond to the same physical region of the imaged scene. These images are related through 2D coordinate transformation. The next part describes why image registration is necessary for SBFSEM image stacks and how it was realized.

3.2.1.1 Tiling Technique of SBFSEM

The total volume that can be reconstructed by SBFSEM is limited in the lateral dimension by the digital resolution available on commercial SEM instruments. To allow the acquisition of large volume the surface of the image block is tiled. This means that a block surface of 200 x 200 µm^2 is for example divided into 10 x 10 subimage stacks (fig. 3.2).

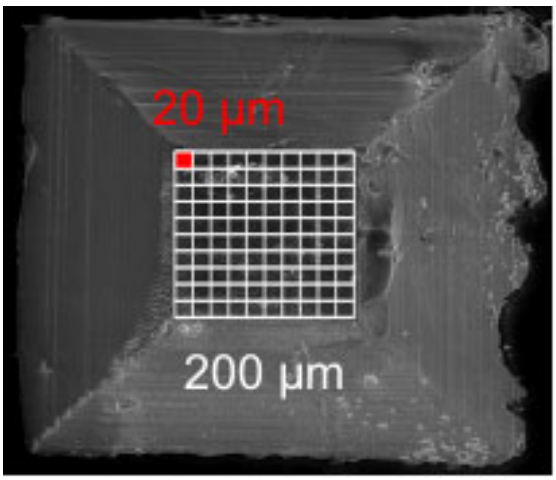


Figure 3.2: Example tiling SBFSEM
Large volume, e.g. 200 µm^2, can be acquired by tiling. Hereby each tiled image stack has a size of 20 µm^2 in x-, y-direction.

Neuronal structures are spread over more than one single tiled stack. To achieve reconstruction of those processes first the single tiled planes must be registered in x-, y-direction. Therefore it was necessary to implement a software function to reassemble the original surface.

3.2.1.2 Overview of Image Registration Methods

As image registration is used in different areas, such as remote sensing, medical imaging, computer vision etc., its application can be divided into four main groups (Brown, 1992):

- Multiview Analysis: Images of the same scene are acquired at different viewpoints. The aim is to gain a larger 2D view or a 3D representation of the scanned scene.
- Multitemporal Analysis: Images of the same scene are acquired at different time points and possibly under different conditions. Thus changes in the scene which appeared between consecutive image acquisitions can be found and evaluated.
- Multimodal Analysis: Images of the same scene are acquired by different sensors. The integration of the information obtained from different source streams to gain more complex and detailed scene representation is accomplished with this analysis.
- Scene to model registration: Images of a scene and a model of the scene are registered. The aim is to localize and compare the acquired image in the scene/model.

The image acquisition with the SBFSEM technique is classified as a multiview analysis. Because, as above described, images of the same scene are acquired at different viewpoints. Furthermore standard image registration algorithms are categorized into two main methods: area based and feature based methods (Zitova and Flusser, 2003). In the following the original image is called reference image and the image to be mapped onto the reference image is referred to as the target image.

Area based registration is focused on the global structure of the image (Fonseca and Manjunath, 1996). Consequently this method is applied if the images do not contain many prominent details and the distinctive information is provided by graylevels/colors rather than by local shapes and structure. Therefore one principal limitation is, that reference and target image must have similar intensity functions (identical or at least statistically dependent). The second limitation is that only transformations in x-, y-direction and small rotations between images are located by the algorithm. One prominent representative of the area-based method is the normalized cross-correlation and its modifications (Pratt, 1974). The calculated normalized cross-correlation gives a measure of degree of similarity between the reference and target image. The maximum value indicates the pixel shift in x-, y-direction between the two images. Disadvantages are that the normalized cross-correlation is not invariant with respect to imaging scale, rotation, and perspective distortions. Further approaches of the area-based registration are Fourier methods and mutual information

methods, where a measurement of statistical dependency between images from different modalities is calculated (Viola and Wells, 1997). Fourier methods are used when images were acquired under varying conditions or are corrupted by frequency-dependent noise. One representative is the phase-correlation method (Kuglin and Hines, 1975), which is based on the Fourier Shift Theorem (Bracewell, 1965) and was originally proposed for the registration of translated images. It computes the cross-power spectrum of the sensed and reference image and looks for the location of the peak in its inverse:

$$(\Delta x, \Delta y) = \underset{(x,y)}{argmax} \left\{ F^{-1} \left\{ \frac{F(I)F(T)^*}{|F(I)F(T)^*|} \right\} \right\} \tag{3.1}$$

where I = reference image and T = target image.

Such as in the normalized cross-correlation the peak indicates the pixel shift between the two images.

Feature based methods are the second main method of image registration algorithms (Zitova and Flusser, 2003). These are typically applied when the local structural information is more significant than the information carried by the image intensities. They allow registering of images of completely different nature (such as aerial photographs, maps) and can handle complex between-image distortions. The crucial point is to have discriminative and robust feature descriptors that are invariant to all assumed differences between the images. Features are for example significant regions, lines or points, which can be detected in both images. This kind of registration method is based on algorithms using spatial relations, invariant descriptors or relaxation approaches (Zitova and Flusser, 2003). By selecting appropriate features, feature based methods are very robust to image rotations and zooming.

Tiled image stacks acquired with the SBFSEM technique have the following properties: predefined arrangement of the image tiles (stored in image file name), user-defined overlap (2 - 10%) of successive images, possibly brightness changes, small translational shifts in x-, y-direction between images and no image rotations/zooming. To cover these requirements, I implemented for the image registration function the above decribed phase-correlation method, which estimates efficiently the translative movement between two images and is robust against brightness changes. The phase-correlation method has been chosen as for feature-based matching methods it is necessary to have significant local structural image information, which is not given for the small overlapping region of SBFSEM image stacks. Therefore a member of area-based methods is chosen, where image registration is based

on image intensity values. Furthermore, the phase-correlation method is robust against frequency dependent noise and non-uniform, time varying illumination disturbances. As one of the advantages of SBFSEM is, that the lateral position jitter is less than 10 nm, only small translational shifts between subimage stacks are expected and those can be calculated with the used phase-correlation method.

3.2.1.3 Implementation - Image Registration

The implementation of the phase-correlation algorithm was based on formula 3.1. The predefined arrangement of the image tiles and the user-defined overlap of successive images allows calculating the image alignment between two images only on a small extract of the images. In detail images are considered as matrices and the registration of the images starts at the top-left tile and images are added row by row to the complete plane. For the alignment calculation of the top-left tile with the following tile in the row only the last/first columns of the matrices, depending on the overlap, are considered. For example tiles with a matrix size of 2048 columns to 1768 rows are cropped by an overlap of $2 \cdot 2\%$ to a matrix size of 80 columns to 1768 rows, where for the top-left tile the last 80 columns and for the following tile of the row the first 80 columns are considered. Consequently for the image registration of multiple subimage stacks, such as 10 x 10 subimage stacks, an acceleration of the computational speed was achieved. An overview of the calculation process is given in the algorithm 3.1, written in pseudocode.

The algorithm was implemented in Matlab and the corresponding GUI in Java. The GUI offers the possibility to select image tiles, specify the overlap (in %) of successive images and to define further parameters, such as the arrangment of tiles in rows and columns.

3.2.1.4 Results - Image Registration

The image registration was tested with different numbers and arrangements of image tiles as well as varying image brightness. Even though successive images had considerable brightness changes in the overlapping region, the images were aligned correctly.

In figure 3.3 one example of the image registration function is displayed. The block surface was divided into 4 x 4 images, each with a size of 512 x 442 pixels and 2% overlap. Figure 3.3B shows a magnification of the red outlined images in figure 3.3A. In the magnification the correctness and accuracy of the implemented phase-correlation algorithm is displayed. For the calculation of the image alignment only the blue outlined overlap region is considered.

Algorithm 3.1: Image Registration (uses phase-correlation algorithm)

Input: *imageArray*=selected image tiles, *ov*=overlap in %
Output: *outImg*=registered image

begin
  imgA=first element of imageArray;
  cropSize=two-times (*imagesize* times *ov*);
  for *x=2 to size of imageArray* **do**
    imgB=image at position *x* of imageArray;
    Images are cropped depending on their position to each other in the image mosaic - left, right, upper or lower image boundary:
    imgACrop=crop image *imgA* to *cropSize*;
    imgBCrop=crop image *imgB* to *cropSize*;
    Calculate phase-correlation between cropped images:
    peak=maximum of phase-correlation(*imgACrop*, *imgBCrop*);
    imgA=translate(defined by *peak*) and add image *imgB*, without overlapping region, to image *imgA*;
  end
  outImg=imgA;
  return *outImg*;
end

At the moment, intensity adjustment between image edge pixels (image blending) is only provided for borders of the finally aligned image.

So far the image registration function of the software Neuron2D provides a robust and accurate solution for the alignment of image tiles. The existing test data volume had only small translational shifts and no projective distortion. As these problems might occur in following SBFSEM image stacks, it might be useful to extend the image registration function, in the future, with the implementation of the scale-invariant feature transform (SIFT) algorithm (Lowe, 1999). This approach transforms an image into a large collection of local feature vectors, each of which is invariant to image translation, scaling, and rotation. Unlike previous local feature approaches, the SIFT algorithm is invariant to illumination changes and is not sensitive to projective distortion.

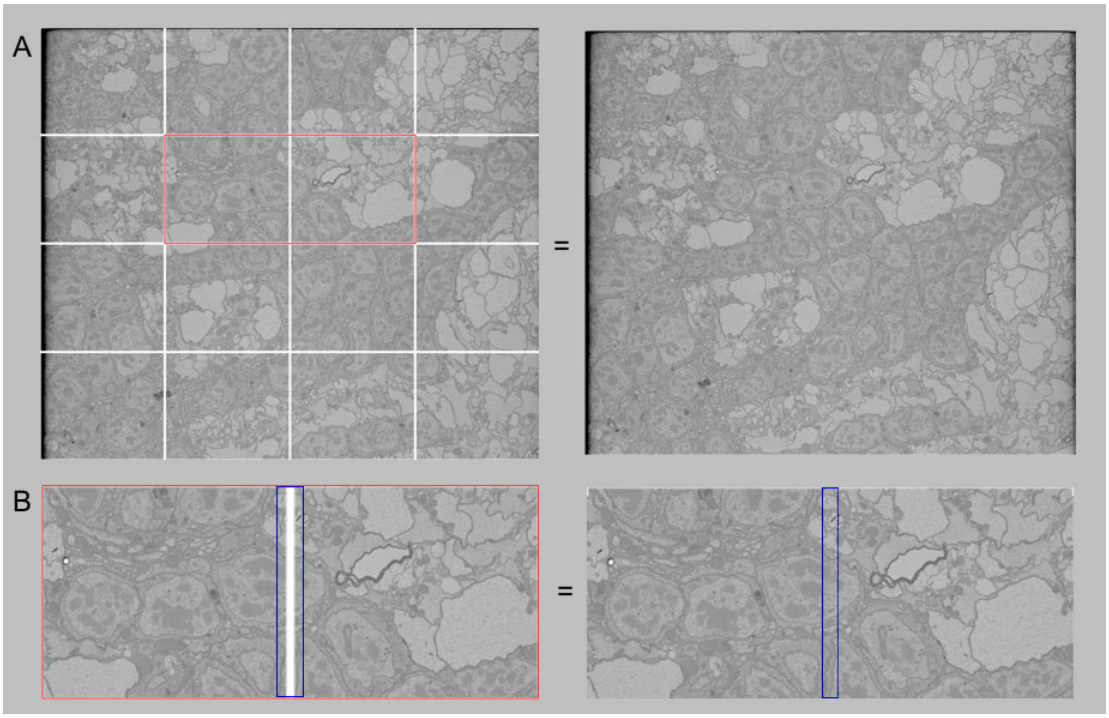


Figure 3.3: Results image registration
(A) 4 x 4 Artificial image tiling, image size of 512 x 442 pixels and an overlap of 2%. (B) Magnification of the red bordered images. Only the blue bordered image sections are used for the image registration calculation.

3.2.2 Image Preprocessing

As the main aim of the software package is to segment and finally extract contour line information from SBFSEM image stacks, the signal-to-noise ratio must be enhanced. Consequently patterns (e.g. edges or membranes) are clearly identifiable in contrast to the back- foreground. The main function of the image preprocessing step is therefore the usage of selected image filtering algorithms. The image contrast of SBFSEM image stacks depends mainly on the staining procedure and the recording conditions. Hence different image filtering algorithms were implemented (see section 3.2.2.2) to provide the user with the opportunity to choose the most efficient (time vs. quality) filtering method for each image stack.

3.2.2.1 Overview of Image Filtering Algorithms

Image filtering algorithms can be divided into the following common filter techniques (Gonzalez and Woods, 2002):

- Low-pass filter, low frequencies are passed.
- High-pass filter, high frequencies are passed.
- Band-pass filter, a limited range of frequencies are passed.
- Band-reject filter, all frequencies except a limited range are passed.
- All-pass filter, all frequencies are passed, but the phase relationship among them is altered.
- Notch filter is a specific type of band-reject filter that acts on a particularly narrow range of frequencies.

Furthermore, image filtering consists of two main classes: linear and nonlinear filtering. Linear filtering, convolution, is an operation where the same filter mask is applied to each pixel of the image (Gonzalez and Woods, 2002). Consequently the calculation of the filter process is fast. However, during image blurring patterns, like membranes, are reduced. This can be a major drawback to image quality. The simplest type of linear filters is the mean filter. The idea is to replace each pixel value in an image with the average value of its neighbors, including itself. This has the effect of eliminating pixel values which are unrepresentative of their surroundings, but as a consequence edges are smoothed. The most prominent representative of linear filters, especially the low-pass filters, is the isotropic Gaussian filter. The Gaussian filter uses a normal distribution for calculating the transformation to be applied to each pixel in the image. The equation of the Gaussian distribution in N dimensions is:

$$G(r) = \frac{1}{(2\pi\sigma^2)^{\frac{N}{2}}} e^{\frac{-r^2}{(2\sigma^2)}} \tag{3.2}$$

where r = radial variable and σ = filter width.

When applied in two dimensions, $N = 2$ and $r^2 = x^2 + y^2$, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. Values from this distribution are used to build the filter kernel $g(x, y)$ which is applied to the original image $I(x, y)$:

$$C(x, y) = g(x, y) * I(x, y) \tag{3.3}$$

Using the Gaussian distribution as filter kernel, the 'weighted average' of each pixel's neighborhood is weighted towards the value of the central pixels. Therefore boundaries and edges are better preserved, than using other, more uniform linear filters such as the above described mean filter.

Nonlinear image filters are the second main class of filter algorithms. Hereby the filtering process depends on local properties of image patterns, e.g. edges. Thus edges are preserved, but the filtering is time-consuming (Gonzalez and Woods, 2002). One member of this group is the median filter (Yin et al., 1996), which is an extension of the previously described mean filter. It considers each pixel in the image in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the pixel value with the average of neighboring pixel values, it replaces it with the median of those values. The median is calculated by first sorting all the pixel values from the surrounding neighborhood into numerical order and then replacing the pixel being considered with the middle pixel value. By calculating the median value of a neighborhood rather than the mean value, the median filter has two main advantages over the mean filter:

- The median is a more robust average than the mean and so a single very unrepresentative pixel in a neighborhood will not affect the median value significantly.

- Since the median value must actually be the value of one of the pixels in the neighborhood, the median filter does not create new unrealistic pixel values when the filter straddles an edge. For this reason the median filter is much better at preserving sharp edges than the mean filter.

An alternative nonlinear filtering process is the nonlinear diffusion filter. This technique, based on the use of partial differential equations, has been extensively studied since the early work of Perona and Malik (Perona and Malik, 1990). It is a noise-removing yet edge preserving filtering technique which, at each location, sets the degree of filtering in dependence to an estimate of the intensity-gradient at that location. Thereby strong edges are preserved but regions that do not have strong edges are smoothed substantially. 2D nonlinear diffusion filtering is based on the following formula, where I(x,y,t) is the image at time t:

$$\frac{d}{dt}I(x,y,t) = \frac{d}{dx}\left[c(x,y,t)\cdot\frac{d}{dx}I(x,y,t)\right] + \frac{d}{dy}\left[c(x,y,t)\cdot\frac{d}{dy}I(x,y,t)\right] \tag{3.4}$$

where c(x,y,t) is the diffusivity function.

For nonlinear diffusion filters the diffusivity function is image dependent. The algorithm of Perona and Malik (Perona and Malik, 1990) uses a diffusivity function $c(x, y, t)$ that is based on the derivative of the image at time t. Consequently the diffusion near the edges in the image is controlled. The result is a diffusion process that encourages intraregion smoothing while inhibiting interregion smoothing. The extension of formula 3.4 to the third dimension is demonstrated in the following formula, where I(x,y,z,t) is the image at time t:

$$\frac{d}{dt}I(x,y,z,t) = \frac{d}{dx}\left[c(x,y,z,t)\cdot\frac{d}{dx}I(x,y,z,t)\right] + \frac{d}{dy}\left[c(x,y,z,t)\cdot\frac{d}{dy}I(x,y,z,t)\right] + \frac{d}{dz}\left[c(x,y,z,t)\cdot\frac{d}{dz}I(x,y,z,t)\right] \tag{3.5}$$

where c(x,y,t) is the diffusivity function.

3.2.2.2 Implementation - Image Preprocessing

The implementation of the image preprocessing function is divided into two steps. First, intensity values within each image are normalized to have the same median and interquartile range. Next, each image is spatially filtered. For the filtering the program gives the user the choice between Gaussian broadening and nonlinear diffusion (Perona and Malik, 1990), both implemented in 2D and 3D. For the implementation of the nonlinear diffusion filter a publicly available Matlab filtering toolbox (D'Almeida, 2000) was used. The two implemented filter algorithms were chosen, because the user has now the possibility to select the most efficient filtering techniques, depending on the contrast of the image stack. High-contrasted image stacks are thus filtered with the fast, smoothing Gaussian filter (2D or 3D). Images with less contrasted contour lines are filtered with the time-intensive, edge preserving nonlinear diffusion filter (2D or 3D). The reason for the additional 3D implementation of both algorithms is based on the SBFSEM data. Considering image stacks as volume data, filtering including the third dimension leads to higher noise reduction and a better preservation of edges. The GUI allows the setting of user-specified parameters, e.g. median value or filter size, for each filtering technique.

As a final preprocessing step the cross-correlation between any two consecutive images is calculated. This allows the detection and elimination of corrupted images, which can result, for example, when debris has gotten onto the block face.

3.2.2.3 Results - Image Preprocessing

The quality of the image preprocessing step mainly depends on the original image contrast. In figure 3.4 example results of the image preprocessing function are displayed. In both charts the first illustration represents the normalized, unfiltered image. Figure 3.4A allows a comparison between the normalized image and 2D/3D Gaussian filtered images. As previously described a Gaussian filter smooths the image linearly and therefore edges are not well preserved. The inset highlights this disadvantage: contour lines with a strong contrast are preserved, but those with a weak contrast disappear in the filtered image. Including the third dimension the result of the Gaussian filter is improved as weakly contrasted edges are better preserved against smoothing. Regarding the nonlinear diffusion filtered images there is almost no difference between the 2D and 3D version, as the original images were already highly contrasted. However, in contrast to the Gaussian filtered images edges, independent from the original contrast, are not smoothed and consequently are clearly distinguishable from other parts of the image. Comparing the calculation time 2D nonlinear diffusion filtering takes about twice as long as 2D Gaussian filtering.

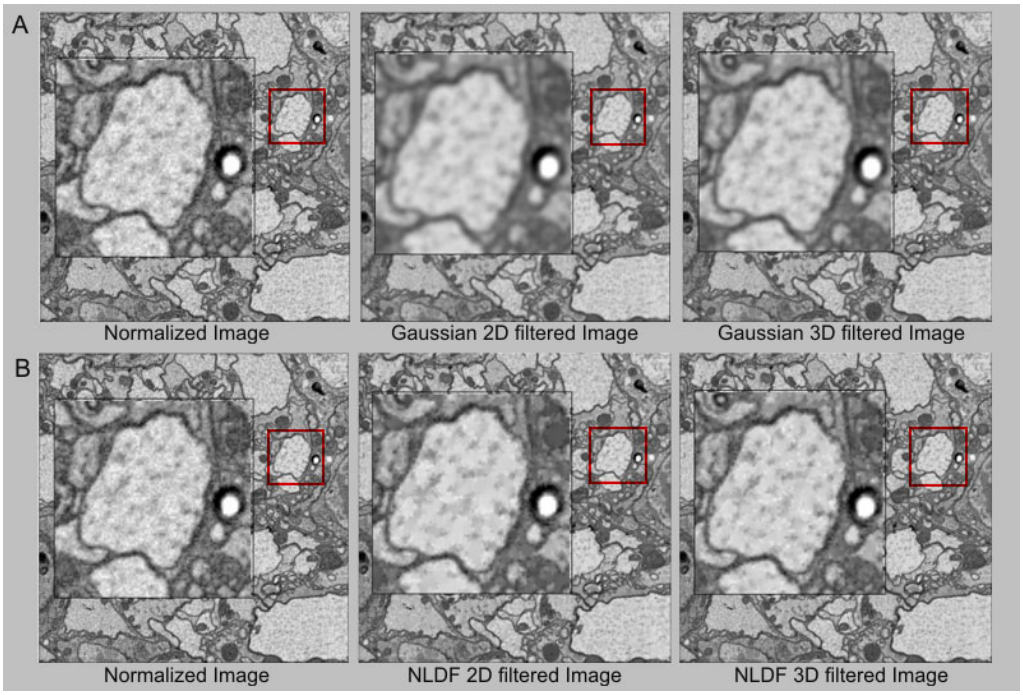


Figure 3.4: Example image preprocessing
(A) Comparison of the normalized, unfiltered image (median 150, spread 80) with a Gaussian 2D ($\sigma = 1$ pixel, window size $n = 3$ x 3) and 3D ($\sigma = 1$ pixel, window size $n = 3$ x 3 x 3) filtered image. (B) Comparison of the normalized, unfiltered image (median 150, spread 80) with a NLDF 2D and 3D (cube size: 8 images) filtered image.

As the loss of contour lines leads to problems in the following membrane structure detection, it is recommended to choose for each image stack independently the most efficient, implemented filtering technique.

Finally the performance of filters is sensitive to the choice of user-specified parameters. Therefore, it is advisable to optimize parameters on, or even learn the filter parameters from manually labeled images (Vollgraf et al., 2004). The contrast parameter λ and the speed of the diffusivity m are example parameters of the nonlinear diffusion filter which could be optimized further.

3.2.3 Image Segmentation

Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images and to finally extract a set of contours (Shapiro and Stockman, 2001). Practical applications of image segmentation are face and fingerprint recognition as well as medical imaging (computer-guided surgery, study of anatomical structure etc.). Several general-purpose algorithms and techniques have been developed for image segmentation. Since there is no general solution, these techniques often must be combined with domain knowledge in order to effectively solve an image segmentation problem. Established techniques are for example (Shapiro and Stockman, 2001):

- Clustering Methods
- Histogram Based Methods
- Edge Detection Methods
- Region Growing Methods
- Level Set Methods
- Neural Networks Segmentation

The implementation of the Neuron2D software function "image segmentation" is divided into two approaches: a contour propagation (section 3.2.3.1) and an edge detection (section 3.2.3.2) method. Such as in the preprocessing step the user must choose, depending on the image gray value distribution, the most promising algorithm. Here, two different gray level value distributions, induced by the used image staining method, are considered. On the one hand images can be divided into fore- and background regions depending on the

gray value. For these stacks foreground regions, especially the cytoplasm of neurons, are overall brighter than the background and a precisely identifiable border separates fore- and background regions of the image (fig. 3.5A). On the other hand the gray value distribution of an image stack can be homogeneous. Hence fore- and background regions are only separated by an border or edge (fig. 3.5B). For this reason the favored image segmentation algorithm for image A is the contour propagation (section 3.2.3.1) algorithm and for image B the edge detection algorithm (section 3.2.3.2).

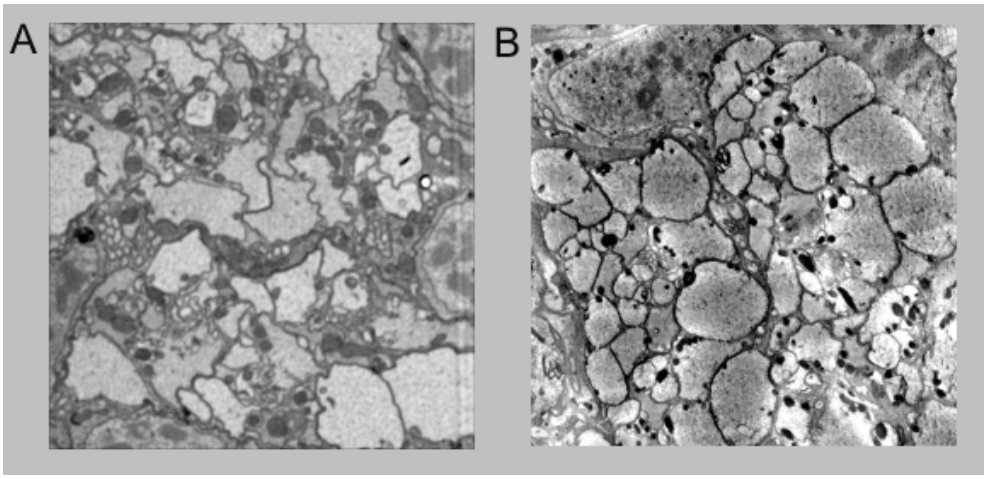


Figure 3.5: Different gray value distributions of SBFSEM image stacks
(A) Gray values of foreground regions are brighter than background, allowing contour line detection based on gray value threshold (contour-propagation method). (B) Homogeneous gray value distribution, consequently no difference between fore- and background regions exists. Segmentation based on edge detection method.

To allow segmentation of large image stacks, as well as to enable user interaction, images are segmented sequentially rather than the entire stack at once. I assume that the objects are continuous across adjacent images — an assumption that can be problematic for processes that run at an oblique angle to the slicing plane. To ensure continuity, the information from the pixel-intensities of the current image with that from the segmentation of the previous image must be combined. Therefore, a *prior* was used that favors such segmentations. In other words, the segmentation of one image is propagated into the next.

3.2.3.1 Contour-Propagation Algorithms

For the contour propagation segmentation, algorithms were chosen that do not represent the boundaries of objects explicitly (e.g. splines) but rather implicitly as the zero-level set of (usually) a signed distance function ϕ. I seek to segment the image $I : \Omega \rightarrow \mathbb{R}$, where

$I(x)$ is the gray-scale value of pixel x, into foreground and background regions. In the level-set framework (Osher and Fedkiw, 2003; Sethian, 1999), one tries to find a function $\phi : \Omega \rightarrow \mathbb{R}$, such that the set $\Omega_+ = \{x : \phi(x) > 0\}$ is the foreground, i.e. the set of pixels that are inside within a boundary, or neuron, and $\Omega_- = \{x : \phi(x) < 0\}$ is the background. The contour separating objects and background is given by $\Gamma = \{x : \phi(x) = 0\}$. Note that images with multiple objects can be segmented with a single segmentation function, ϕ, by defining objects to be connected components of the foreground regions.

As this embedding is not unique, ϕ is sometimes constrained to be a signed distance function (SDF), i.e. such that its absolute value gives the distance to the closest boundary. However, this does not have to be the case. For example, given a statistical model for the segmentation, one could interpret $\phi(x) + t$ (where t is a scalar offset) to be the log of the probability that pixel x belongs to the foreground. In this case, $\phi(x)$ indicates not only what region a pixel is assigned to, but also represents a measure of the confidence in the assignment.

Compared to an explicit representation, an implicit representation has the advantage that it can easily deal with changes in topology (such as splitting or merging of objects, fig. 3.6) and can readily be extended to higher dimensions.

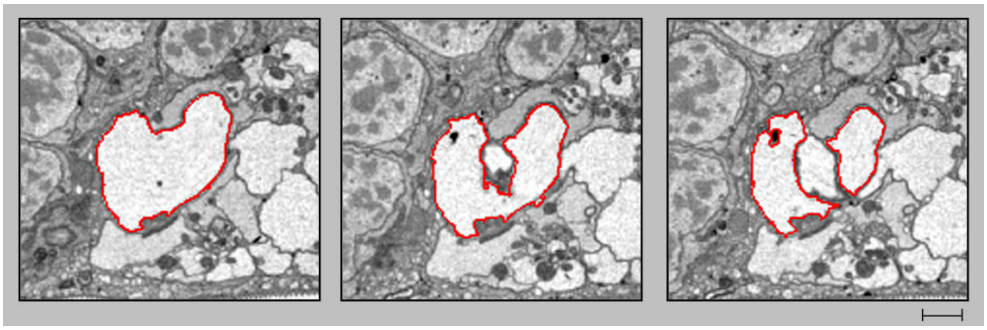


Figure 3.6: Splitting of objects
The used region-based approach makes it possible to segment splitting objects without reparametrization. In this example, the object is split into two constituents, each of which is tracked in subsequent images. Scalebar: 55 pixels (1.47 µm).

Here the focus is on region-based methods, which do not rely on detecting edges in the image but exploit differences in the distribution of pixel intensities in fore- and background regions. In practice, an artificial time variable, t, is often introduced and ϕ_t is evolved to minimize some energy function, $E(\phi, I)$, (Cremers et al., 2006).

Algorithms that have been developed for the analysis of images obtained from confocal– or multi-photon microscopy are often based on the assumption that neuronal structures can

be modeled as tube-like structures (Koh et al., 2002; Al-Kofahi et al., 2002; Schmitt et al., 2004) with approximately circular cross-sections. As the processes observed in SBFSEM data can have shapes very dissimilar to circles, this assumption was not made.

Level-set methods are commonly used for segmentation of three-dimensional objects for bio-medical applications (Whitaker et al., 2001) as well as in other domains such as synthetic aperture radar (SAR) images (Huang et al., 2005). The implemented algorithms differ from existing ones in the way that information is propagated from one image to the next, and in the exact form of the energy-functions used. Furthermore, many algorithms segment the three-dimensional data-blocks directly and need time-consuming computations, and are thus not well suited for online user interaction. This approach is related to the one of Jurrus et al. (2006) who used a propagation scheme based on Kalman filters and explicit contours rather than level-sets to represent objects.

Statistical Model

Let I_n be the image n, and ϕ_n the corresponding segmentation that is to be found. Also, suppose that the previous image is already segmented, such that ϕ_{n-1} is known. The aim is to find the *most probable* segmentation, ϕ_n, given I_n and ϕ_{n-1} or, in mathematical terms, such that $P(\phi_n|I_n, \phi_{n-1})$ is maximal (Cremers et al., 2006). By Bayes' Rule, I obtain

$$P(\phi_n|I_n, \phi_{n-1}) \propto P(I_n|\phi_n, \phi_{n-1})P(\phi_n|\phi_{n-1}). \tag{3.6}$$

$P(\phi_n|\phi_{n-1})$ can be interpreted as a *prior* on the possible segmentations ϕ_n, and can be used to ensure continuity of objects between adjacent images. This helps tracing of structures through multiple images. In addition, other priors can be used to favor smooth contours, or to incorporate prior knowledge about the likely shapes of segmented objects. Ideally, these priors would be learned from manually labeled images, but for the lack of an extensive training set, the functional forms for the prior distributions were chosen.

For simplicity, I assume that image I_n is independent of the segmentation of the previous image, ϕ_{n-1}, given the actual segmentation, ϕ_n, i.e. $P(I_n|\phi_n) = P(I_n|\phi_n, \phi_{n-1})$. For a given (signed) distance, d, $P(I_n(x)|\phi_n(x) = d)$ gives the probability distribution for the intensities of all pixels that are on the outside (or inside, depending on the sign of $\phi(x)$), and have a distance d to the closest boundary. For convenience, I will from now on write ϕ instead of ϕ_n to denote the segmentation function of the current image, n. The term ϕ_0 is used instead of ϕ_{n-1} to denote the segmentation function of the previous image. Maximizing

the posterior probability, $P(\phi|I)$, is equivalent to minimizing its negative logarithm, which leads to the energy function

$$E(\phi|I) = -\log(P(I|\phi)) - \log(P(\phi|\phi_0)) = E_I + E_\pi. \tag{3.7}$$

The total energy is written as a sum of an image-dependent part E_I and a prior-dependent part E_π. In the following, two possible forms of the probability distributions, $P(I|\phi)$ and $P(\phi|\phi_0)$ are specified in detail.

Simple Segmentation Algorithm

A normal distribution is assumed for the pixel-intensities $I(x)$ with mean $\alpha\phi(x) - \beta$ and variance σ^2, i.e. on average the intensity of pixels is linearly related to their signed distance to the boundary. An additional assumption is that the difference of the signed distance functions of two adjacent images is normally distributed. As mentioned above, this favors segmentations that are similar to the segmentation of the previous images, with the aim of improving the tracking of structures through multiple images. The corresponding probability distributions are:

$$P(I|\phi) \propto \exp\left(-\int_\Omega \left(\frac{I - \alpha\phi - \beta}{2\sigma}\right)^2 dx\right) \tag{3.8}$$

$$P(\phi|\phi_0) \propto \exp\left(\|\frac{\phi - \phi_0}{2\eta}\|^2\right) = \exp\left(-\int_\Omega \left(\frac{\phi(x) - \phi_0(x)}{2\eta}\right)^2 dx\right). \tag{3.9}$$

This leads to an energy function of the form:

$$E(\phi|I) = \int_\Omega \left(\frac{I - \alpha\phi - \beta}{\sigma}\right)^2 + \int_\Omega \left(\frac{\phi - \phi_0}{\eta}\right)^2, \tag{3.10}$$

which attains its minimum for

$$\phi = \frac{\alpha\eta}{\eta\alpha^2 + \sigma} I + \frac{\sigma}{\eta\alpha^2 + \sigma}\phi_0 - \frac{\alpha\beta\eta}{\eta\alpha^2 + \sigma} \tag{3.11}$$

Therefore, the function ϕ, which characterizes the current segmentation, is merely a linear combination of the intensity values of the current image I, the previous segmentation function ϕ_0, and a scalar offset (fig. 3.7).

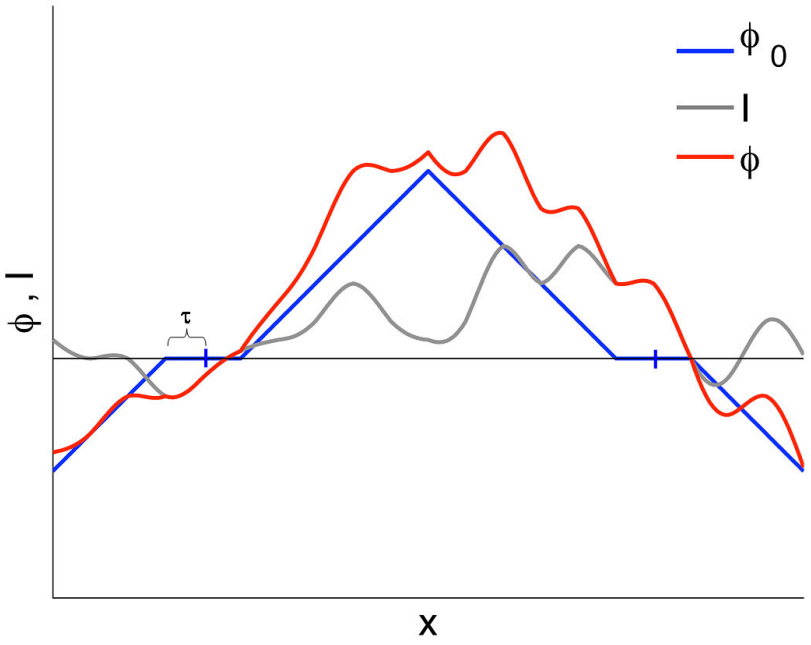


Figure 3.7: Schematic illustration of the algorithm
The segmentation function ϕ of the image (red) is the weighted sum of the intensity value of that image (gray) and a term that depends on the segmentation of the previous image (blue).

Although this model might be too restrictive to provide a good fit to the actual image, using such a simple form has two advantages: First, the minimum of the energy function, $\min E(\phi|I)$, has a simple closed-form solution, which permits rapid calculation of the boundary. Second, the parameters α, β, σ and η can be interpreted and adjusted while the effect on the segmentation is visually judged. The parameter α re-scales the distances relative to intensities and β is an offset: If an intensity value of a pixel is larger than β, then (ignoring the prior) it is more likely that the pixel belongs to an object than to the background. The parameters σ and η control the relative weight of the image and the prior-dependent contribution. Discrepancies between two adjacent segmentations are only penalized if they exceed a threshold, τ, as small variations between adjacent images are to be considered as normal. Thus, the term $(\phi(x) - \phi_0(x))^2$ was replaced by $|\phi(x) - \phi_0(x)|_\tau^2$, where

$$|\phi(x) - \phi_0(x)|_\tau^2 = \begin{cases} (|\phi(x) - \phi_0(x)| - \tau)^2 & \text{for } |\phi(x) - \phi_0(x)| \geq \tau \\ 0 & \text{else} \end{cases} \tag{3.12}$$

Then the form of the solution is as before, just with ϕ_0 replaced by $|\phi_0|_\tau sign(\phi_0)$.

Alternative Segmentation Algorithm

The algorithm described above does not attempt to achieve segmentations with smooth boundaries. In addition, it makes a rather restrictive assumption about the intensity distribution of pixels in the image. Therefore a second segmentation algorithm with a different energy function that also contains a term favoring smooth segmentations was implemented.

This algorithm assumes that the pixel-intensity distributions of the background and foreground, P_- and P_+, are different but might overlap. Furthermore, I assume that the probability that a given pixel has a certain intensity only depends on the region that the pixel belongs to, but not on its distance to the closest boundary. Therefore P_+ and P_- must be estimated from manually segmented images. One could use the histograms of intensities in these images directly (Cremers and Rousson, 2006), but I chose to approximate them by Gaussians, with specific means and variances, $\mu_\pm$ and $\sigma_\pm^2$. This has the advantage of specifying the distribution using a small number of parameters, which can be manually optimized by the user. I then get

$$\begin{aligned} E_I &= \int_{\Omega_+} \frac{1}{2}\log(\sigma_+)(I(x)-\mu_+)^2 dx + \int_{\Omega_-} \frac{1}{2}\log(\sigma_-)(I(x)-\mu_-)^2 dx \\ &= c_+ \int_{\Omega_+} (I(x)-\mu_+)^2 dx + c_- \int_{\Omega_-} (I(x)-\mu_-)^2 dx, \end{aligned} \tag{3.13}$$

where $c_\pm = \frac{1}{2}\log(\sigma_\pm)$. The cost function E_I is low if the pixels inside the objects are close to μ_+, and the pixels outside are close to μ_-. The prior-dependent energy E_π is a sum of the two terms E_{cont} and E_{smooth}:

$$E_\pi = E_{cont} + E_{smooth} \ . \tag{3.14}$$

The first term, E_{cont}, ensures that segmentations of adjacent images are similar, and is identical to the corresponding term in the simple algorithm.

$$E_{cont} = \lambda_{cont} \int_\Omega |\phi(x) - \phi_0(x)|_\tau^2 dx \tag{3.15}$$

The second term E_{smooth} penalizes the length of the boundary between foreground and

background, thus favoring smooth boundaries.

$$E_{smooth} = \lambda_{smooth} \int_{\Omega} \delta(\phi(x)) |\nabla \phi| dx \tag{3.16}$$

The choice of the parameters λ_{cont} and λ_{smooth} determines the relative importance of the two energy terms. In particular, the smoothness term can be switched off by choosing the weight λ_{smooth} to be 0. If $\lambda_{cont} = 0$, this model is similar to the (piecewise continuous) version of the Mumford-Shah function (Mumford and Shah, 1989; Chan and Vese, 2001). In this model, the means, $\mu_{\pm}$, are known a priori and do not have to be optimized further. I am primarily interested in the regions where the actual segmentation ϕ is close to 0 rather than in calculating its exact value for all x. Therefore, the following constraint was added for all pixels x:

$$|\phi(x) - \phi_0(x)| < \hat{\tau}, \tag{3.17}$$

where $\hat{\tau}$ is some positive scalar value. This constraint is computationally convenient, as it allows to restrict all calculations to the region of the image where $|\phi_0| < \hat{\tau}$. The parameter $\hat{\tau}$ is chosen such that it exceeds the largest difference expected between successive images.

A (local) minimum of the energy function E can be found by an iterative procedure: Starting with an initial guess, $\phi^{(0)}$, the segmentation function ϕ is repeatedly updated via the evolution equation:

$$\phi^{(k+1)} = \phi^{(k)} + t_{step} \frac{d\phi}{dt}, \tag{3.18}$$

where t_{step} is the step-size. The update-direction $d\phi/dt$ can be found by calculating the gradient of the function E with respect to the segmentation function ϕ (Chan and Vese, 2001):

$$\frac{\partial E}{\partial \phi} = \frac{d\phi}{dt} = -\delta(\phi)(\log(P_+(I)) - \log(P_-(I))) - \frac{\partial}{\partial \phi} \log(P(\phi|\phi_0)). \tag{3.19}$$

With the particular distributions $P_{\pm}$ and the energy function $E = E_I + E_{cont} + E_{smooth}$ I obtain

$$\begin{aligned} \frac{d\phi}{dt} &= \delta(\phi)(c_+(I-\mu_+)^2 - c_-(I-\mu_-)^2) \\ &\quad +2\lambda_{cont}|\phi - \phi_0|_\tau sign(\phi - \phi_0) \\ &\quad +\lambda_{smooth}\delta(\phi)\nabla.(\frac{\nabla\phi}{\|\nabla\phi\|}). \end{aligned} \tag{3.20}$$

The gradient $d\phi/dt$ is, except for the last two terms in the sum, identical to the one used in Chan and Vese (2001). I employ the commonly used smooth approximations, H_ϵ and

δ_ϵ, for the Heaviside– and δ-functions:

$$H_\epsilon(s) = \frac{1}{2}(1 + \frac{2}{\pi}\tan^{-1}(\frac{s}{\epsilon})) \tag{3.21}$$

$$\delta_\epsilon(s) = \frac{d}{ds}H_\epsilon(s) = \frac{\epsilon}{\pi(\epsilon^2 + s^2)}, \tag{3.22}$$

where ϵ is a positive parameter. The algorithm is initialized with the segmentation function of the previous image, i.e. by $\phi^{(0)} = \phi_o$. Since this initial value is usually reasonably close to the desired solution, the algorithm converges very quickly. In addition, only ϕ has to be updated close to the boundary Γ, and not on the whole domain Ω.

Even if the segmentation ϕ is initialized to be a signed distance function, this property is not preserved by the update step. Therefore, ϕ must be *reinitialized* every few iterations to ensure stability of the algorithm. Reinitialization creates a SDF function that is consistent with the given segmentation (Osher and Fedkiw, 2003; Sethian, 1999).

The algorithm described in this section is more flexible than the simple one, but also computationally more demanding, as each iteration takes as long as one run of the simple algorithm.

Segmentation of the First Image

For the segmentation of the very first image in a stack, it is not possible to rely on information from the preceding slice, so a slightly different strategy must be used. To avoid the time consuming manual retracing of contour lines, the first image is segmented with one of the two algorithms above (ignoring the term E_π), and objects are defined to be the connected components of foreground regions Ω_+, which I call $o_1 \dots o_n$. For each object o_i, the correspondence to a neuronal process must be determined. Three coefficients for each object (mean intensity, area, and "roundness", i.e. area divided by square of circumference) are extracted. After that a classification in this three-dimensional space with logistic regression, using the labeled data as a training set, is performed (Hosmer and Lemeshow, 2000). The weighted sum of these three coefficients is represented by the color of the contour line: a continuous color scale is used from black for very small values to red for very high values of the weighted sum. Finally the GUI allows the adjustment of the weights of the regression via sliding-bars. With this procedure a reasonable initialization is obtained quickly, which can subsequently be fine-tuned and further corrected manually. A manual correction is in average only necessary for 10% of all detected regions. Figure

3.8 displays a GUI screenshot of the segmentation of the first image.

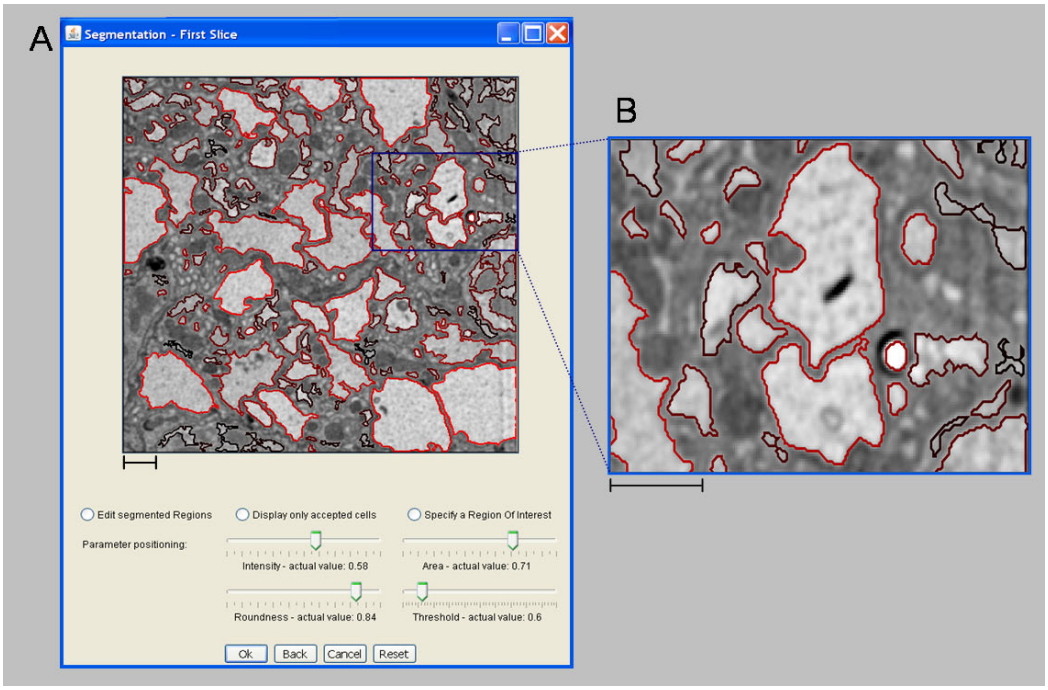


Figure 3.8: Segmentation of the first image (screen shots)
The overview (A) shows the image and the parameter adjustment GUI. In the zoomed image (B) it can be seen how even small structures can be detected. For each object of the first segmented image the mean intensity, area, and "roundness"(area divided by square of circumference) are extracted. The weighted sum of these three coefficients is represented by the color of the contour line: A continuous color scale is used from black for very small values to red for very high values of the weighted sum. Scalebars 1 µm.

3.2.3.2 Canny Segmentation Algorithm

The algorithms described in section 3.2.3.1 are based on different gray value distributions between fore- and background regions. Therefore, for the segmentation of images with a homogeneous intensity distribution. Hence fore- and background regions are only separated by an border or edge (fig. 3.5B), a different algorithm needed to be implemented.

The basic idea is to use the Canny edge detection operator (Canny, 1986) to detect a wide range of edges in the images. Similar to the previous contour propagation algorithms a distance function was used to favor segmentations that are similar to the segmentation of the previous images, with the aim of improving the tracking of structures through multiple images. Thus the segmentation of the previous image ϕ_{n-1} again was used as a *prior* for the actual segmentation, ϕ_n, of image I_n. The difference is that instead of a signed an unsigned distance function was used, as no difference between in- and outside regions exists. The

distance function d calculates the distance of all pixels to the closest boundary. Therefore, the function ϕ_n is a linear combination of the intensity values of the currently segmented image I_n and the previous segmentation function ϕ_{n-1}. Discrepancies between two adjacent segmentations are only penalized if they exceed a threshold τ, as small variations between adjacent images are to be considered as normal.

$$|\phi_n(x) - \phi_{n-1}(x)|_\tau^2 = \begin{cases} (|\phi_n(x) - \phi_{n-1}(x)| - \tau)^2 & \text{for } |\phi_n(x) - \phi_{n-1}(x)| \geq \tau \\ 0 & \text{else} \end{cases} \tag{3.23}$$

The segmentation of image I_n is based on the Canny edge detection algorithm (Canny, 1986). The following steps are performed:

1. The gradient of $I_n(x, y)$ is computed by convolving $I_n(x, y)$ with the first derivative of the 2D Gaussian distribution $G(x, y)$ (formula 3.2):

$$\nabla_x f(x, y) = \left(I_n * \frac{\partial G}{\partial x} \right)(x, y), \quad \nabla_y f(x, y) = \left(I_n * \frac{\partial G}{\partial y} \right)(x, y). \tag{3.24}$$

$$M(x, y) = \sqrt{(\nabla_x f(x, y))^2 + (\nabla_y f(x, y))^2} \tag{3.25}$$

and

$$\theta(x, y) = \arctan\left(\frac{\nabla_y f(x, y)}{\nabla_x f(x, y)} \right) \tag{3.26}$$

2. Definition of the threshold M:

$$M_T(x, y) = \begin{cases} M(x, y) & \text{if } M(x, y) > T \\ 0 & \text{otherwise} \end{cases} \tag{3.27}$$

where T is chosen such that all edge elements are kept while most of the noise is suppressed. Non-maxima pixels in $M_T(x, y)$ are suppressed to thin the edge ridges. Therefore it is calculated if each non-zero element of $M_T(x, y)$ is greater than its two neighbors along the gradient direction $\theta(x, y)$. If that is true $M_T(x, y)$ is kept unchanged, otherwise, it is set to 0.

3. The previous result is thresholded by two different thresholds τ_1 and τ_2, where $\tau_1 < \tau_2$, to obtain two binary images T_1 and T_2. Comparing T_1 and T_2 it is seen that T_2 has

less noise and fewer false edges but larger gaps between edge segments.

4. Edge segments in T_2 are linked to form continuous edges. Each segment is traced in T_2 to its end and for its neighbors it is searched in T_1. The located edge segments in T_1 are used to bridge the gap until reaching another edge segment in T_2.

An overview of the segmentation process is given in the following algorithm 3.2, written in pseudocode:

Algorithm 3.2: Canny Segmentation Algorithm

Input: *images*=previous and actual image for the segmentation,
thresholdCanny=parameter for the Canny filter function
Output: *outImg*=segmented image

begin

I_{n-1}=previous segmented image;
imgRegionArray=store all segments of I_{n-1};
I_n=actual image;
imgCanny=calculate canny edge function for I_n with *thresholdCanny*;

for *x=1 to size of imgRegionArray* **do**

imgDist=calculate distance function for segment x of *imgRegionArray* in I_{n-1};

imgCanny=add *imgDist* to *imgCanny*;

Perform boundaries calculation:
imgBound=define boundaries of *imgCanny*;

Test if new boundary is complete:
if *imgBound not closed* **then**
(Complete *imgBound* with corresponding line segment from region x of *imgRegionArray*)

outImg=add *imgBound* to *outImg*;

end

return *outImg*;

end

Segmentation of the First Image

Such as in the contour propagation algorithms for the segmentation of the very first image in a stack, it is not possible to rely on information from the preceding slice, so a different strategy must be used. In that case the regions, which should be segmented, must be marked manually by the user and these are traced and segmented in the following slices. The manual selection process is based on B-spline curves and described in detail in section 3.2.3.3. The following figure 3.9 gives a short overview of the implemented Canny Segmentation algorithm and its correctness:

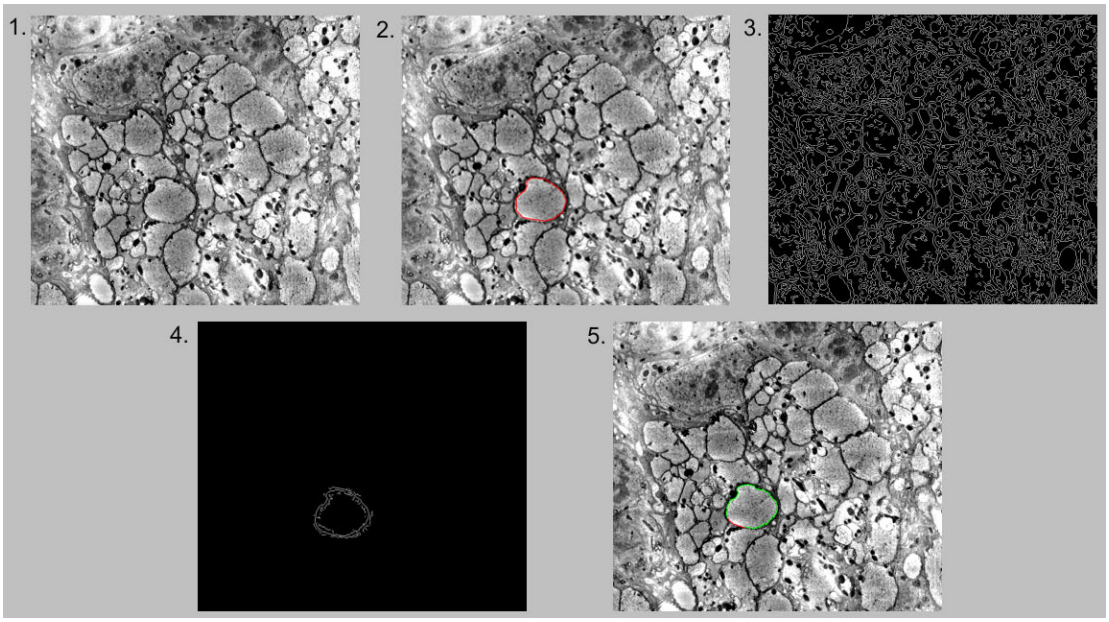


Figure 3.9: Example Canny segmentation algorithm
1. First image I_{n-1}. 2. Manual selected region by the user, displayed in red. 3. Next image, I_n, filtered with the Canny algorithm. 4. Unsigned distance function of I_{n-1} added to filtered image I_n. 5. Final segmented region of image I_n is displayed in green, if the resulting boundary is not closed the corresponding line segment, displayed in red, is copied from the previous segmentation of image I_{n-1}.

3.2.3.3 Manual Interaction

During image segmentation it is possible to insert new regions, that are tracked in the following images in addition to the previously selected ones. The user can insert points along the desired contour using mouse-clicks. To calculate the actual contours, these control points are interpolated using B-spline curves (Schoenberg, 1946)(de Boor, 1978): The B-Spline curve C(u), $u \in [\tau_p, \tau_{n-p+1}]$ with order p, is defined by control points

$P_i (i = 1, ..., n-p)$ and B-Spline basis functions $N_{i,p,\tau}\,(i = 1, ..., n-p)$ with breakpoints $\tau = (\tau_1, ..., \tau_n)\,, \tau_i \leq \tau_{i+1}$.

$$C\,(u) = \sum_{i=1}^{n-p} P_i N_{i,p,\tau}\,(u) \tag{3.28}$$

The B-Spline basis functions are defined by the recursion formula of de Boor/Cox/Mansfield (de Boor, 1978):

$$N_{i,1,\tau}\,(u) = \begin{cases} 1 & : \quad u \in [\tau_i, \tau_{i+1}] \\ 0 & : \quad \text{else} \end{cases} \tag{3.29}$$

and

$$N_{i,p+1,\tau}\,(u) = \frac{u - \tau_i}{\tau_{i+p} - \tau_i} N_{i,p,\tau}\,(u) + \frac{\tau_{i+p+1} - u}{\tau_{i+p+1} - \tau_{i+1}} N_{i+1,p,\tau}\,(u) \tag{3.30}$$

The behavior of B-spline curves is local in the sense that changes in one point only changes the shape of the curve between nearby control points. Therefore, manipulating the contour line by adding, deleting or dragging control points changes only the local behavior of the curve and not the global behavior, which allows for fast re-computation of the curve. I used functions from Matlab's Spline Toolbox.

To correct for mistakes in the segmentation process, contour lines can be adjusted manually. In this case, the contour of a selected region was approximated by a B-spline curve, which can then be manipulated by moving the control points. All manual corrections in one image are taken into account for the segmentation of subsequent images.

Using B-spline curves for the implementation of the manual interaction instead of snakes allows segmenting of regions with destructed contour lines or image distortions. Snakes, or interactive deformable contours, are a model-based feature localization and tracking technique (Kass et al., 1988). Their use for contour-based segmentation of e.g. medical images was proposed e.g. in Carlbom et al. (1995). Snakes differ substantially from manual contouring approaches. Snakes are dynamic models made of simulated elastic material. They are attracted to image boundaries through forces. This improves both the speed and the accuracy of the segmentation and contouring. As mentioned above the manual interaction is mainly necessary for the segmentation of regions that are not clearly identifiable (destructed contour lines or false fusion of regions) and therefore B-spline curves allow a more promising user interactive manual correction as snakes.

3.2.3.4 Quantification of Segmentation Algorithms

The performance of the described algorithms depends on the properties of the original image stack, e.g. the image contrast and number of corrupted images. To quantify the performance I used the described image stack from the *Calliphora vicina* outer chiasm (see chapter 2 and section 3.1). As for this image stack objects are separated from the background by their pixel intensities the implemented contour propagation algorithms (section 3.2.3.1) were used. An additional assumption was that the objects do not vary too strongly between slices. This assumption is justifiable as the average slice thickness is in a range of 30 nm. Consequently consecutive images have minor variations. The possibility to reconstruct a volume of SBFSEM image stacks (fig. 3.1C) allows analyzing of SBFSEM data in all dimensions. Therefore regions lying vertical to the slicing plane can be analyzed in a second segmentation step, where images are not segmented in z-direction but e.g. in x-direction.

I filtered the image stack with the Gaussian filter, standard deviation $\sigma = 1$ pixel (section 3.2.2) and extracted an image cube of $512 \times 512 \times 250$ pixels in x-, y- and z-direction. The extracted image cube covers a tissue volume of $14 \times 14 \times 13$ µm^3. To demonstrate the performance of the algorithm for processes with different properties, I selected regions with area size ranging from 55 up to 7800 pixels in the initial slice, and different "roundness factors" (see fig. 3.10). I analyzed this data set with both methods described in chapter 3.2.3.1:

- Using the simple algorithm (3.2.3.1), it took, on average, 3 seconds to segment one image. User interaction consisted of eye-inspection of the segmentation and, if necessary, manual correction. In the image stack displayed above, the user had to correct 2% of 2000 regions manually.

- Alternative algorithm (3.2.3.1): Average processing time per image amounted to 5 seconds plus user interaction. Out of 2140 regions, 1.5% regions had to be corrected manually.

For both segmentation methods the quality of the segmentation, and hence the need for user interaction mainly depends on the contrast difference between the object and the surrounding background as well as the complexity of the contour lines. For example, region 15 (fig. 3.10), which has a high contrast and does not deviate from one image to the next, could be segmented and traced through all slices without any user interaction. Regions 3 and 7 in figure 3.10 have complicated contour lines which are very variable between images

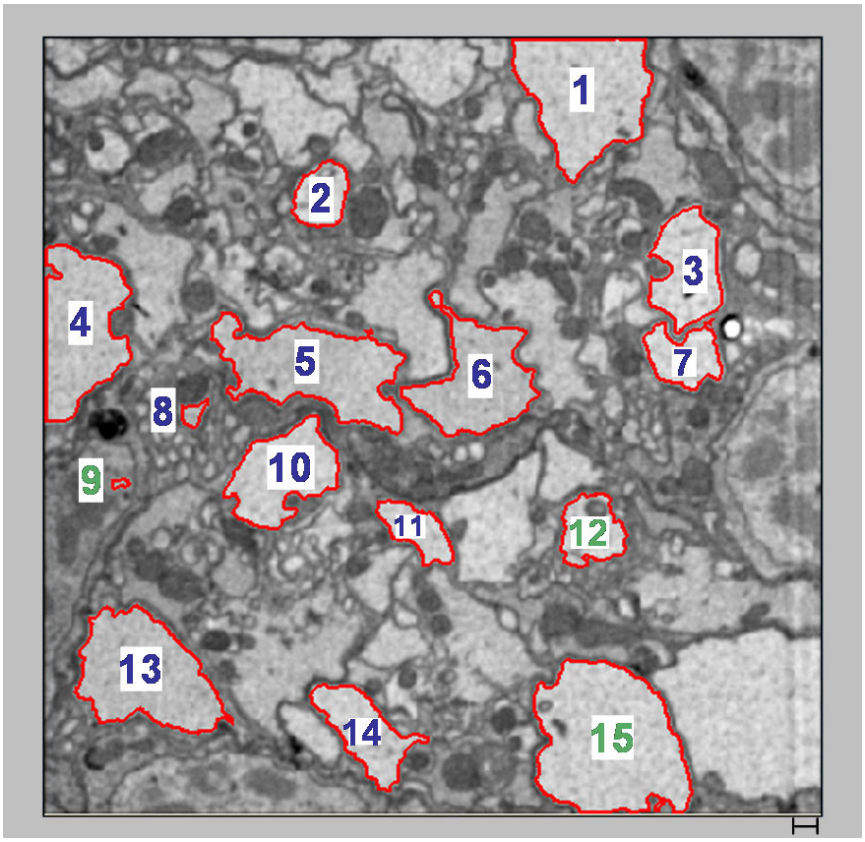


Figure 3.10: Quantification of performance
Fifteen regions (labeled in red) were segmented for the performance test. The number of pixels per region ranges from 55 pixels (region 9, labeled in green) up to 7800 pixels (region 15, labeled in green). The roundness factors range from 0.3217 (region 12, labeled in green) up to 0.9069 (region 9, labeled in green). Scalebar: 20 pixels (540 nm).

(fig. 3.11). Much more user interaction was required for these two objects than for others, but they could be reconstructed using a combination of algorithmic user-interaction and manual corrections (see fig. 3.19 for the three-dimensional reconstruction and fig. 3.11 and fig. 3.6).

Independently of the method used, the total time taken per image (algorithmic segmentation plus user interaction) was about 15 seconds for 15 selected regions. A purely manual tracing of the selected regions (fig. 3.10) takes about 12 minutes per image. The user had to trace the boundaries for each region manually by marking corner-points with mouse-clicks. I, thus, have a speed-up by a factor of about 50 by algorithmic over manual segmentation.

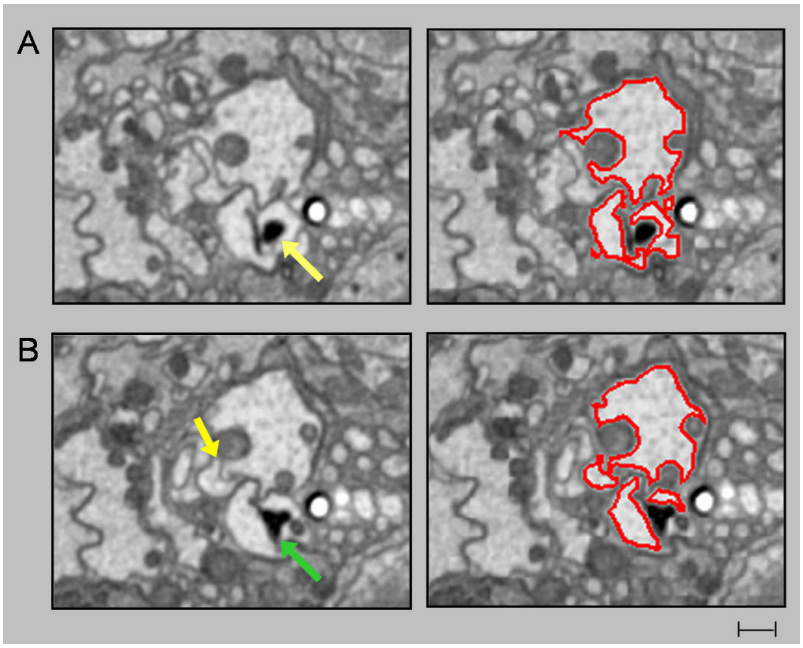


Figure 3.11: Typical errors made by the algorithm
Two typical situations in which the algorithm failed to give a satisfactory segmentation, and had to be corrected manually. Left side: original image section, right side: segmentation results, labeled in red. Scalebar: 28 pixels (750 nm). (A) Image 120: The structure surrounding the black dot (yellow arrow) has a ragged contour, and is not successfully segmented by the algorithm, which splits it up into two regions. (B) Image 129: The object at the lower left of the region I want to segment is separated by only a thin and light membrane (yellow arrow), and thus incorrectly joined to the large region. As before, the region indicated by the green arrow is separated into two regions.

As so far only few SBFSEM images with a homogeneous gray value distribution exist (fig . 3.5B), the canny segmentation algorithm (section 3.2.3.2) cannot be quantified in the same way as the contour propagation algorithms. To test the usability of the algorithm, I filtered the image stack again with the Gaussian filter, standard deviation $\sigma = 1$ pixel (section 3.2.2) and extracted an image cube of $512 \times 512 \times 30$ pixels in x-, y- and z-direction. The three regions labeled in figure 3.12 were selected to test the canny segmentation.

The canny segmentation is based on detecting edges in the image. Therefore manual interaction is necessary when contour lines are not clearly identifiable. This might occur due to brightness changes in the image (fig. 3.13, indicated by the yellow arrow) or ragged contour lines (fig. 3.13, indicated by the green arrow) e.g. produced from irregular staining of the membrane, lead to false segmentation results.

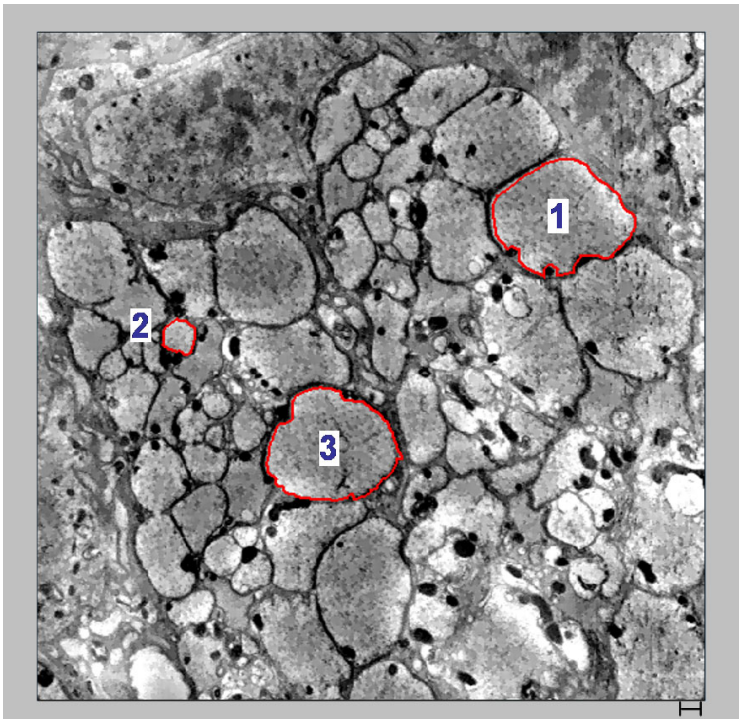


Figure 3.12: Test Canny segmentation
Three regions with a different circumference were selected for the test of the canny segmentation algorithm. Scalebar: 20 pixels (540 nm).

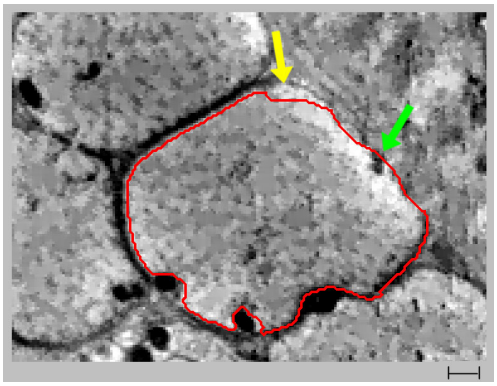


Figure 3.13: Typical errors made by the Canny segmentation algorithm
The membrane of region 1 (fig. 3.12) is not clearly identifiable due to brightness changes (yellow arrow) and the algorithm failed. The contour line is destructed by the the black dot (green arrow) and is not successfully segmented by the algorithm. Scalebar: 20 pixels (810 nm).

Summing up, the implemented segmentation algorithms offer a robust and fast segmentation of SBFSEM image stacks (section 3.3.3) with optional user interaction. The need for user interaction mainly depends on the quality of the original image contrast. Compared

to purely manual tracing, processing time is speed-up by a factor of about 50.

3.2.3.5 Data Storage

Segmentation results are stored in an XML (Extensible Markup Language) document, which facilitates the sharing of structured data across different information systems. In this case the contour line information is exchanged between the two software packages Neuron2D and Neuron3D. Key concept of XML is a generic framework for storing any amount of text or any data whose structure can be represented as a hierarchical structured tree. The only indispensable syntactical requirement is that the document has exactly one root element. Additionally tree knots are elements, attributes and comments. The XML document used for the software packages has the following output structure (fig. 3.14).

Furthermore the usage of databases to store XML-based information is possible. Therefore two different approaches exist. On the one hand native XML databases are used and XML documents are stored directly. On the other hand the usage of hybrid databases is common. These are comparable to object relational databases. Those databases receive XML documents, but store them in tables. Examples for the hybrid solution are the Microsoft SQL-Server as well as Oracle databases. The general task of a database is to efficiently govern data and to allow access to several elements. For the SBFSEM data the integration of a database might be useful, because complete regions of the fly brain are analyzed and reconstructed in the future. Consequently the efficient storage, exchange and access of the reconstructed data can be realized and in addition multiuser operations are provided.

Project_neuron

stack

imageSize	info
513x513	Pixel Size: 26.4 x 26.4nm, Slice thickness: 50nm

slice

number	z_coordinate
0100	1.8900
0101	1.9089
0102	1.9278

spline

name	parent	x_coordinates	y_coordinates	splineCenter
neuron_4.0_0100	4	111,112,112,112,112,112,112,112,112,113,..	199,198,197,196,195,194,193,192,191,190,190,..	173,220
neuron_9.0_0100	9	264,265,266,267,268,269,270,271,271,272,273,..	311,310,310,309,309,308,307,306,305,305,304,..	297,321
neuron_3.0_0100	3	311,312,313,314,315,316,317,318,319,320,321,..	3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,..	357,38
neuron_8.0_0100	8	120,121,121,122,122,123,123,124,125,125,126,..	300,299,298,297,296,295,294,293,292,291,290,..	157,290
neuron_10.0_0100	10	124,125,125,126,126,126,127,127,127,128,129,..	28,27,26,25,24,23,22,21,20,19,18,17,17,16,..	160,50
neuron_2.0_0100	2	325,326,326,326,327,327,327,327,328,329,330,..	435,434,433,432,431,430,429,428,428,428,429,..	373,467
neuron_7.0_0100	7	237,238,239,240,241,242,243,244,245,246,247,..	233,233,233,233,233,233,232,232,232,232,232,..	281,219
neuron_1.0_0100	1	403,404,405,406,407,408,409,410,411,412,413,..	423,422,421,421,420,420,420,420,420,419,419,..	466,445
neuron_6.0_0100	6	3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,..	140,140,140,140,139,139,139,139,139,139,139,..	23,192
neuron_5.0_0100	5	22,23,24,25,25,25,26,26,27,27,27,27,27,27,27,..	431,430,429,428,427,426,425,424,423,422,421,..	67,423
neuron_4.0_0101	4	102,103,104,105,106,107,108,109,110,111,112,..	208,207,208,208,209,210,210,210,210,210,211,..	166,220
neuron_9.0_0101	9	263,264,265,266,267,268,269,269,270,271,272,..	312,311,311,311,310,309,308,307,306,305,304,..	297,320
neuron_3.0_0101	3	311,312,313,314,315,316,317,318,319,320,321,..	3,3,3,3,3,3,3,3,3,3,2,2,3,3,3,3,3,3,3,3,3,..	357,38
neuron_8.0_0101	8	119,120,121,122,123,123,124,124,125,125,125,..	300,299,298,297,296,295,294,293,292,291,290,..	157,292
neuron_10.0_0101	10	124,125,125,126,126,126,127,128,129,130,131,..	27,26,25,24,23,22,21,20,19,18,17,17,17,17,..	160,52
neuron_2.0_0101	2	325,326,326,326,327,327,327,327,327,328,329,..	436,435,434,433,432,431,430,429,428,428,428,..	372,467
neuron_7.0_0101	7	237,238,239,240,241,242,243,244,245,246,247,..	233,232,232,232,233,233,233,232,232,232,232,..	281,219
neuron_1.0_0101	1	403,404,405,406,407,408,409,410,411,412,413,..	422,421,421,421,420,420,420,419,419,419,419,..	465,446
neuron_6.0_0101	6	3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,..	140,140,140,139,139,139,139,138,138,138,138,..	22,193
neuron_5.0_0101	5	22,23,24,24,25,25,26,26,26,27,27,27,27,27,27,..	431,430,429,428,427,426,425,424,423,422,421,..	66,422
neuron_4.0_0102	4	103,104,105,106,107,108,109,110,111,112,113,..	208,207,207,207,208,208,208,208,208,208,207,..	171,219
neuron_9.0_0102	9	264,265,266,267,268,269,270,271,272,273,274,..	311,310,309,308,307,306,306,305,305,304,303,..	297,319
neuron_3.0_0102	3	311,312,313,314,315,316,317,318,319,320,321,..	3,3,3,3,3,3,3,2,2,2,2,2,2,3,3,3,3,3,3,3,3,..	357,38
neuron_8.0_0102	8	118,119,120,121,122,123,124,124,125,125,126,..	300,299,298,297,296,295,294,293,292,291,290,..	157,290
neuron_10.0_0102	10	124,125,125,125,126,127,128,129,130,131,..	27,26,25,24,23,22,21,20,19,18,17,17,17,18,18,..	159,47
neuron_2.0_0102	2	325,326,326,326,326,326,326,326,326,326,327,..	442,441,440,439,438,437,436,435,434,433,432,..	367,466
neuron_7.0_0102	7	237,238,238,238,239,240,241,242,243,244,245,..	235,234,233,232,232,232,232,232,232,232,232,..	281,218
neuron_1.0_0102	1	402,403,404,405,406,407,408,409,410,411,412,..	422,421,420,420,420,419,419,419,419,418,418,..	466,446
neuron_6.0_0102	6	3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,..	141,140,140,139,139,139,139,139,138,137,137,..	23,192
neuron_5.0_0102	5	23,24,24,25,26,27,28,29,30,30,30,31,32,32,32,..	407,406,405,405,404,403,402,401,400,399,398,..	65,421

Figure 3.14: Example XML output file
Root element contains the project name, element "stack" contains the image size and general information, element "slice" represents each segmented image, with the image number and the corresponding z-coordinate, element "spline" represents each segmented neuronal structure per image, attributes are name, parent (important for branchpoints), x-, y-coordinates and spline center.

3.3 Software - Neuron3D

The software package Neuron3D was developed to visualize neuronal structures in 3D. The reconstruction is based on the contour line information provided by the XML-files generated by the software package Neuron2D. It includes an easy-to-operate GUI and provides rendering techniques such as shading and texture-mapping. Additionally the reconstructed anatomical data can be exported to 3D visualization programs such as Amira (Amira, 2006).

3.3.1 Surface Reconstruction

The surface reconstruction in the case of SBFSEM data is based on a given collection of planar contours representing cross-sections through the neurons. To reconstruct the triangular surface from given contours an algorithm solving this problem has first to decide which contours of two successive slices should be connected by the surface (branching problem). Second which vertices of the assigned contours should be connected for the triangular mesh (correspondence/tiling problem). As the information for the branching problem is in this case already provided by the image segmentation step of the software Neuron2D and stored in the XML-files (information about which contours are connected in successive slices and about existing branching points), only the correspondence problem must be solved for the surface reconstruction. Used data for the surface reconstruction of SBFSEM data are only points of contour lines. An alternative might be to also use information from the extended level-set algorithms used for the image segmentation (section 3.2.3, Osher and Sethian 1988). However implementing additional segmentation algorithms based on e.g. neuronal networks the algorithm for the surface reconstruction must be reimplemented. Therefore the surface reconstruction of SBFSEM data is based on points of contour lines as this information is always provided by segmentation algorithms.

3.3.1.1 Overview of Surface Reconstruction Algorithms

One of the first solutions to the construction of surfaces from contour data was proposed by Keppel (Keppel, 1975). In this approach surfaces are constructed from elementary triangular tiles, each defined between two consecutive points on the same contour and a single point on an adjacent contour. Constructing this surface is shown to be equivalent to finding a minimum path in a directed graph. However the method depends on the fact that there is a one-to-one relationship between contours. A different solution was proposed by

Boissonnat (Boissonnat, 1988) which is based on a Delaunay triangulation between each pair of slices. For a d-dimensional Euclidean space E and a set M of N points $m_1...m_N$, the associated Voronoi diagram is a sequence $V(m_1)...V(m_N)$ of convex polyhedra covering E where $V(m_i)$ consists of all the points of E that have m_i as a nearest point in the set M. Thus

$$V(m_i) = \{P \in E; \forall j, 1 \leq j \leq N, \delta(P, M_i) \leq \delta(P, M_j)\}, \tag{3.31}$$

where δ denotes the Euclidean distance.

The geometrical dual of the Voronoi diagram, obtained by linking the points m_i whose Voronoi polyhedra are adjacent across a common face, is called the Delaunay triangulation of M. With this approach intersection planes between contours are identified and surface tiles are reconstructed from these intersections. Additionally the triangulation of slices with multiple contours can be handled. The complexity of the algorithm is $O(NLogN + T)$, where T is the number of tetrahedra of the triangulation.

In the last few years also algorithms for the correspondence problem of planar contours were developed which primarily were used to find an optimal match between two given protein structures or word sequences. Two main approaches in this research field are the Longest Common Subsequence approach (LCSS, Hirschberg 1975) and the Dynamic Time Warping approach (DTW, Myers and Rabiner 1981). The first one is mainly used in bioinformatics and tries to find a maximum length subsequence of two or more strings. For the general case of an arbitrary number of input sequences this problem is classified into the computational complexity theory group of nondeterministic polynomial-time hard (NP-hard) problems. When the number of sequences is constant, the problem is solvable in polynomial time by dynamic programming. The second one, Dynamic Time Warping, is a well known application for automatic speech recognition and the optimization process is also performed with dynamic programming. In the case of matching points from two contours DTW matches every point with at least one point of the other contour and in particular the first and last points of both contours must be matched to each other. The final matching of two contour lines is defined by the minimum distance between both, based on a distance function $d(x, y)$.

The surface reconstruction from two consecutive contour lines obtained from SBFSEM data is not only a simple predefined point to point connection, as contour lines can have a different number of points, no defined starting or end point and translations in x-, y-direction are possible. Therefore the reconstruction of the 3D surface is based on the DTW approach.

3.3.1.2 Implementation - Surface Reconstruction

The implementation is based on a DTW algorithm for comparison of trajectories described in Sielhorst et al. (2005). As previously mentioned DTW computes the matching that has the lowest summed distance concerning a given distance function $d(x, y)$, which can be recursively defined by:

$$\begin{gathered} DTW(A,B) = d(a_i, b_j) + \\ min(DTW(A_{i-1}, B_{j-1}), DTW(A_{i-1}, B_j), DTW(A_i, B_{j-1})), \end{gathered} \tag{3.32}$$

where A and B are contour lines with points $(a_1...a_i)$ and $(b_1...b_j)$. In detail the following steps are performed by the algorithm:

1. Calculation and sorting of the distance matrix:
   In this step the 3D Euclidean distance between every point of contour line $A = (a_1...a_i)$ to every point of contour line $B = (b_1...b_j)$ is calculated and stored in the matrix $dist(A, B)$. As start-/endpoints of successive contour lines are not necessarily aligned, the calculated distance matrix is resorted so that the new starting point is the minimal distance value of the matrix. One restriction is that both points must have an equal orientation from the midpoint of the contour line, so that a misalignment is avoided.

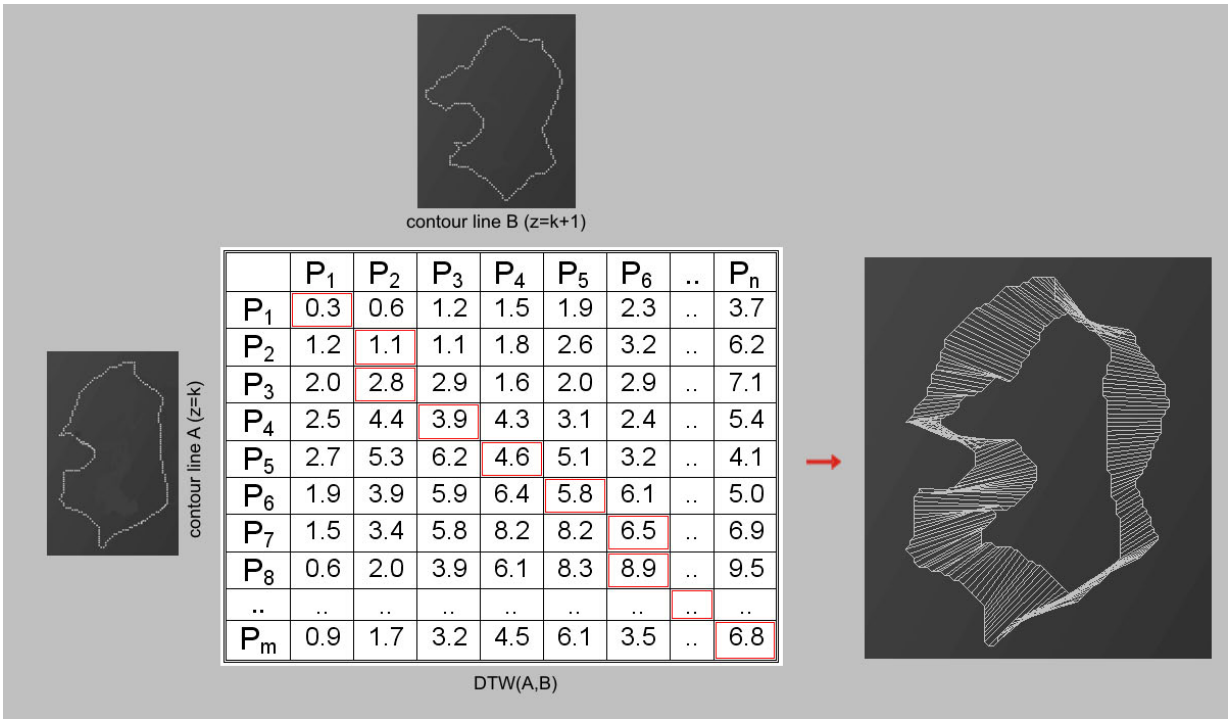
2. Calculation of the DTW function:
   The DTW matrix is filled up with the result of $DTW(A_n, B_m)$ in field(n,m), which is acquired by the value of $dist(A_n, B_m)$ and adding the minimum of the left, upper or upper-left field of the DTW matrix, see formula 3.32.

3. Calculation of optimal matching:
   For the optimal path search dynamic programming is used. Starting point is the lower-right field of the DTW matrix and in the following steps always the points a_n, b_m are stored as correspondent where the upper, left or upper-left field of the matrix is the minimum. Since all points are matched, outliers could have a too intense impact on how points are matched. Therefore in the implemented approach additional constraints were added. The first constraint considers the matching of contour lines with a different number of points. If the number of points in A is bigger than in B, $|A| > |B|$, the minimum calculation is only based on the upper and upper-left field of the DTW matrix, where columns are equal to points of B and

rows are equal to points of A. Hence for the other case, $|A| < |B|$, only the left and upper-left field of the DTW matrix, where columns are equal to points of B and rows to points of A, are considered. Adding this constraint "double matching" of points (one point is more than once connected with a point of the other contour line) is only allowed in the contour line with the smaller amount of points. As in this case each pixel of a contour line is represented by one point, this assumption leads to a better triangulation. The second constraint controls the double matching of points in the contour line with the smaller number of points. Considering the case that contour line A is smaller than B and the next minimum field of the DTW matrix is the left one, meaning point a_n might be double matched, vectors $\overrightarrow{a_n b_m}$ and $\overrightarrow{a_{n+1} b_m}$ are additionally calculated. The comparison of the orientation of both vectors and the knowledge that contour line points are oriented clockwise are used to decide if double matching the point a_n is correct, otherwise the upper-left field is chosen.

4. Quantification of warping path:
   The calculated warping path must be in the range of $length = ||A| - |B||$, if not step 3 is recalculated.

These steps are calculated for each pair of successive contour lines, so that finally a triangulation of the neuronal surface is computed. Generally the time and space complexity of the DTW algorithm is $O(N^2)$. Figure 3.15 displays the DTW calculation between two contour lines A and B. As $|A| > |B|$ the calculation of the warping path is constrained to the upper and upper-left field of the matrix. Selected elements for the final warping path are red bordered and additionally the correct triangulation of these two contour lines is displayed.



	P_1	P_2	P_3	P_4	P_5	P_6	..	P_n
P_1	0.3	0.6	1.2	1.5	1.9	2.3	..	3.7
P_2	1.2	1.1	1.1	1.8	2.6	3.2	..	6.2
P_3	2.0	2.8	2.9	1.6	2.0	2.9	..	7.1
P_4	2.5	4.4	3.9	4.3	3.1	2.4	..	5.4
P_5	2.7	5.3	6.2	4.6	5.1	3.2	..	4.1
P_6	1.9	3.9	5.9	6.4	5.8	6.1	..	5.0
P_7	1.5	3.4	5.8	8.2	8.2	6.5	..	6.9
P_8	0.6	2.0	3.9	6.1	8.3	8.9	..	9.5
..	..	..	..	..	..	..	..	..
P_m	0.9	1.7	3.2	4.5	6.1	3.5	..	6.8

Figure 3.15: Surface reconstruction
Displaying the DTW calculation between contour line A and B, with $|A| > |B|$. Red bordered matrix elements are part of the calculated warping path. Additionally the correct triangulation of these two contour lines is displayed.

3.3.1.3 Integration of Branches

The above described calculation of the surface triangulation considers only a one-to-one contour matching. Therefore, for branching contours, an additional step must be inserted. In the last decade several algorithms were developed for solving the branching problem. In the following classical branching algorithms are enumerated:

- Parametrization of Contours (Ganapathy and Dennehy, 1978):
  The parametrization of contours is a heuristic based on intercontour coherence.

- Contour Decomposition (Ekoule et al., 1991):
  The algorithm is based on the decomposition of the contour into elementary convex subcontours. Then the problem of linking one contour in a slice to several contours in an adjacent slice is examined. Consequently a new and unique interpolated contour is generated between the two slices, and the link is created using standard matching procedures.

- Contour Branching Using Minimum Spanning Trees (Meyers et al., 1992):
  The algorithm handles narrow valleys and branching structures using a minimum spanning tree (MST, Kruskal 1956) of a contour adjacency graph. The MST of a weighted graph is a spanning tree whose sum of edge weights is minimal. The minimum spanning tree describes the cheapest network to connect all of a given set of vertices.

- Angular Bisector Network (Olivia et al., 1996):
  The Angular Bisector Network (ABN) is an approximation of the Voronoi Diagram between the polygon segments of the original contours. To each edge of the contours there is one corresponding cell of the ABN that contains the points closest to this edge. The proximity information is used to identify the cells that can be triangulated in a straightforward way. Where such a straightforward triangulation is impossible, an intermediate contour consisting of the border between cells belonging to contours in different slices is inserted.

- Partial Curve Matching (Barequet et al., 2000):
  The basic idea is to partition the root contour into subcontours using shape information of branch contours. If a root contour is properly partitioned, the original double branching problem can be reduced to a single branching problem: single branching from each subcontour of the root contour to one of the corresponding branch contours.

As in this case branching contour lines are not overlapping, the implementation is based on the partial curve matching algorithm. As described above the root contour is partitioned into subcontours using shape information of the branching contours. If the root contour is properly partitioned, the original multiple branching problem is reduced to a single branching problem. Figure 3.16 shows an example of double branching. Using the partitioning line the root contour of the double branching $\langle (C_r), (C_1, C_2) \rangle$ is partitioned into two parts, and by triangulating the left subcontour to C_1 and the right subcontour to C_2 the original branching problem is solved. For the final triangulation the DTW algorithm (section 3.3.1.2) is used as the branching problem has been reduced to a partial, one-to-one, curve matching.

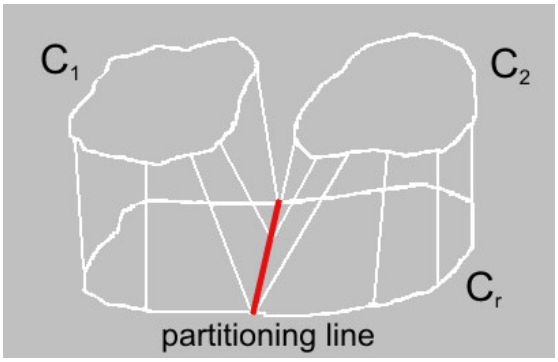


Figure 3.16: Branched contours
Using the calculated partitioning line, displayed in red, the root contour C_r is partitioned into two parts and triangulated with the subcontours C_1, C_2.

The implementation is based on the following algorithm 3.3, written in pseudocode:

Algorithm 3.3: Multiple Branching

Input: C_r=root contour at level $z = k$, $C_1, C_2...C_n$=n branch contours of C_r at level $z = k + 1$

Output: *surf*=triangulation of C_r with $C_1, C_2...C_n$

begin

partLine=calculation of partitioning line between all branch contours $C_1, C_2...C_n$;

$C_{r1}, C_{r2}...C_{rn}$=calculation of intersection points of C_r with *partLine* and divide C_r into subcontours;

for *x=1 to size of* $C_1, C_2...C_n$ **do**

Triangulation of C_{rx} *with* C_x*:*

$dtwC_x$= calculation of DTW(C_{rx}, C_x);

Add calculated $dtwC_x$ *to surf:*

surf= *surf* plus $dtwC_x$;

end

return *surf*;

end

One extension would be to allow the user to re-define branches, where re-define stands for deleting or adding branchpoints.

3.3.2 3D Image Editing

In addition to the surface reconstruction several image rendering techniques to animate or layout the calculated wireframed sketch are provided with the software package Neuron3D. To place the 3D objects within a scene and also to define the spatial relationship between objects in that scene standard transformation methods such as rotation, translation and scaling are implemented. The viewing volume, which determines how an object is projected onto the screen and which objects are clipped out of the final image is defined by the projection transformation. OpenGL provides two general classes of projection transformations: orthographic and perspective projection. The perspective projection is implemented in Neuron3D, as it creates more realistically looking scenes. Hereby the viewing volume is a frustum of a pyramid and hence the size of an object depends on the distance to the viewer. For a better orientation in the scene a coordinate system and a scale bar are optional displayed.

3.3.2.1 Surface Representation

The representation of the surface is defined by the user. One selectable feature is the final polygonal rasterization. The following standard polygon modes are implemented:

- Point: Polygon vertices which are the start of a boundary edge are drawn as points. Point attributes such as size and smoothness control the rasterization of the points.

- Line: Boundary edges of the polygon are drawn as connected line segments. The line attribute width controls again the rasterization of the lines.

- Fill: The interior of the polygon is filled. This option is predefined in Neuron3D.

Color properties are first randomly assigned to the surface but can be re-defined by the user in a later step. Additionally texture mapping is provided (Shreiner et al., 2005). In this process the user has the possibility to choose from a selection of standard textures the desired one. As texture coordinates are specified at each vertex of a given triangle of the surface the mapping must account the vertices' positions in the 3D space. Therefore to achieve the perspective correction, texture coordinates are not interpolated directly, but

coordinates are divided by their depth:

$$u_\alpha = \frac{(1-\alpha)\frac{u_0}{z_0} + \alpha\frac{u_1}{z_1}}{(1-\alpha)\frac{1}{z_0} + \alpha\frac{1}{z_1}}, \tag{3.33}$$

where u_α = texture coordinate, u_0, u_1 = endpoints, α = color component and z_0, z_1 = depth (relative to the viewer).

The texture mapping procedure is realized with standard OpenGL functions. In addition a function for polygonal surface simplification is provided by the software package. As reconstructed 3D models of SBFSEM data are acquired at a very high resolution these models have a high complexity. However, full complexity is not always required. Since the computational cost of a model is directly related to its complexity, it is useful to have simplified versions. For surface simplification several different algorithms have been formulated and are roughly categorized into three classes (Luebke, 2001):

- Vertex Decimation
- Vertex Clustering
- Iterative Edge Contraction

Since in this case surface reconstruction is based on the triangulation between two consecutive contour lines, the implemented surface simplification considers each connected contour line pair separate. Generally the implemented algorithm is based on vertex decimation. In detail neighboring normal vectors of quads are used to define the similarity of polygons. If a similarity exists the corresponding vertex is deleted and the two quads are fused. To allow a different graduation of surface simplification, the similarity determination of quads is based on a user-defined percentage value. Hereby a value of 95% similarity for example means that only vertices with equal normal vectors are deleted. Consequently the range of similarity is, depending on the percentage value, enlarged and thus different levels of surface simplification are provided. Figure 3.17 gives an impression of the implemented surface simplification based on the similarity of normal vectors.

3.3.2.2 Lighting Implementation

One of the most important aspects of 3D computer graphics is creating depth percept visualization of 3D structures in space. One possibility is to illuminate the scene. Hereby different models are used to describe the appearance of a surface to finally generate more

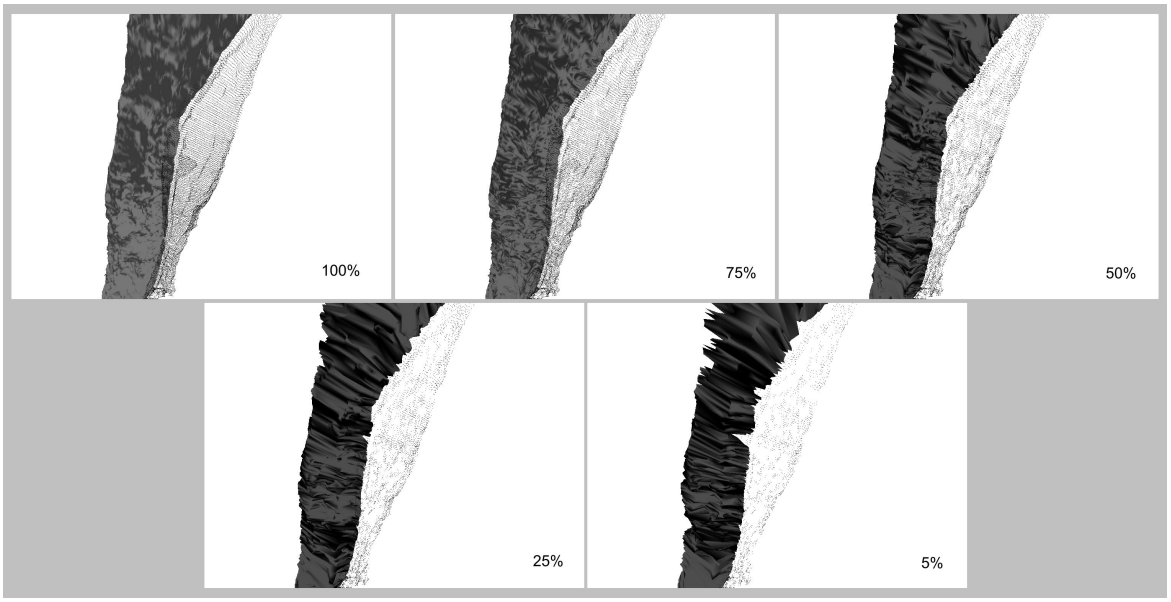


Figure 3.17: Surface simplification
Normal vectors of corresponding quads are compared and depending on the user-defined percentage value, here in the range of 100% to 5%, are defined as similar. Similar quads are fused, which results in a surface simplification.

realistic 3D objects. These models are divided into illumination, reflection and shading models. The first type, the illumination model is divided into global and local illumination. Global illumination simulates the light diffusion in a scene. Typical algorithms are e.g. radiosity and ray tracing. As these algorithms are not suited for real time rendering modern 3D computer graphics rely on local illumination. Local illumination simulates light conditions on surfaces. Hereby the brightness or color of a surface point is calculated from the orientation of a reflecting light beam. Therefore the following information is necessary: normal vectors, material and light properties and the field of view. Standard local illumination models are Lambert- , Phong-, Schlick- illumination models and per-pixel-lighting. Local illumination models must handle diffuse and specular reflections in a reasonable calculation time. The Phong model (Phong, 1975) is indeed not based on physics, but on empirical observation. Despite its lack of a theoretical grounding, it produces realistic light scenes in an efficient computing time and has been the basis for most computer graphics imagery and therefore is used for Neuron3D. The model consists of an independent ambient, diffuse and specular term:

$$I_{out} = I_{ambient} + I_{diffuse} + I_{specular}, \tag{3.34}$$

where the single terms are described in the following:

- Ambient Light:

$$I_{ambient} = I_a \cdot k_{ambient}, \tag{3.35}$$

  where I_a = intensity of the ambient light and $k_{ambient}$ = material constant.

- Diffuse Light:

$$I_{diffuse} = I_{in} \cdot k_{diffuse} \cdot \cos\phi, \tag{3.36}$$

  where I_{in} = intensity of the light beam of the point light source, $k_{diffuse}$ = empirical defined reflection factor and ϕ = angle between surface normal vector and light beam.

- Specular Light:

$$I_{specular} = I_{in} \cdot k_{specular} \cdot \cos^n \Theta, \tag{3.37}$$

  where I_{in} = intensity of the light beam of the point light source, $k_{specular}$ = empirical defined reflection factor for specular components, n = constant factor for the description of the surface properties and Θ = angle between reflection direction of omitted light beam and field of view of the observer.

As in OpenGL light is broken into red, green and blue components the user has the possibility to define these components for the light source independently. In addition the position, x-, y- and z-coordinate, of the light source is adjustable by the user.

The second model to describe the appearance of a surface is the reflection or scattering model, which describes the relationship between incoming and outgoing illumination at a given point. Descriptions of scattering are usually given in terms of a bidirectional scattering distribution function (Heckbert, 1991). Popular reflection rendering techniques in 3D computer graphics include:

- Flat shading: A technique that shades each polygon of an object based on the polygon's normal vector and the position and intensity of a light source.

- Gouraud shading: A fast and resource-conscious vertex shading technique used to simulate smoothly shaded surfaces.

- Phong shading: Simulation of specular highlights and smooth shaded surfaces, but significantly greater time complexity than Gouraud shading.

Predefined reflection model for Neuron3D is Gouraud shading (Gouraud, 1971). Reasons for this are that for flat shading, shading is constant across polygons and therefore surfaces have a staged appearance. Phong shading interpolates normals rather than colors, therefore surfaces are smoothed shaded. However, the significantly high computing time allows the rendering mostly off-line and is therefore not suitable for real time rendering. The selected Gouraud shading averages all adjacent face normals and finally interpolates the colors of the polygon interior between the vertex colors. Neighboring pixels have slightly different colors, but overall smooth lighting on polygon surfaces is achieved in a fast computation time.

The third model includes shading techniques. Shadows are essential for realistic 3D images and are a region of relative darkness within an illuminated region caused by an object totally or partially occluding the light. Without them scenes often feel unnatural and flat. In addition the relative depths of objects in the scene is hard to define. Shadow algorithms are not predefined in OpenGL. Generally the algorithms are divided into three groups: shadow mapping, shadow volume and shadow projection algorithms (Woo et al., 1990). The shadow volume algorithm is a geometry based shadow algorithm which requires connectivity information of the polygonal meshes in the scene to efficiently compute the silhouette of each shadow casting object. It is a per pixel algorithm, which performs an in shadow test for each rendered fragment. This operation can be accelerated using graphics hardware. Generally the scene complexity has direct influence on the performance of this algorithm. Shadow mapping is an intrinsic image-space algorithm, which means that no knowledge of the scene's geometry is required to carry out the necessary computations. As it uses discrete sampling it must deal with various aliasing artifacts, the major drawback of the technique.

Therefore an alternative algorithm to achieve shadow calculations, shadow projection, was implemented in Neuron3D. Blinn (1988) invented the original technique, which allows shadows to be cast on planar surfaces. The general idea is given by a plane P with $\vec{n} \cdot \vec{r} + d = 0$ and a point light source l with position vector $\vec{l} = (l_x, l_y, l_z)$. A projection matrix M is constructed that projects point p with position vector $\vec{p} = (p_x, p_y, p_z)$ onto the plane P. The point s with position vector $\vec{s} = (s_x, s_y, s_z)$, which is the projection of point p onto the plane P, can be described as:

$$\vec{s} = \vec{l} - \frac{\vec{n}\vec{l} + d}{\vec{n} \cdot (\vec{p} - \vec{l})}(\vec{p} - \vec{l}) \tag{3.38}$$

This can be converted into a projection matrix which satisfies $Mp = s$ (Moeller and Haines, 2002):

$$M = \begin{pmatrix} \vec{nl} + d - l_x n_x & -l_x n_y & -l_x n_z & -l_x d \\ -l_y n_x & \vec{nl} + d - l_y n_y & -l_y n_z & -l_y d \\ -l_z n_x & -l_z n_y & \vec{nl} + d - l_z n_z & -l_z d \\ -n_x & -n_y & -n_z & \vec{nl} \end{pmatrix} \tag{3.39}$$

To avoid anti-shadows, e.g. when the light source is between the occluder and the shadow plane, the stencil buffer is used. The implemented approach allows a fast calculation of shadows and is very effective for the usage in Neuron3D (fig. 3.18).

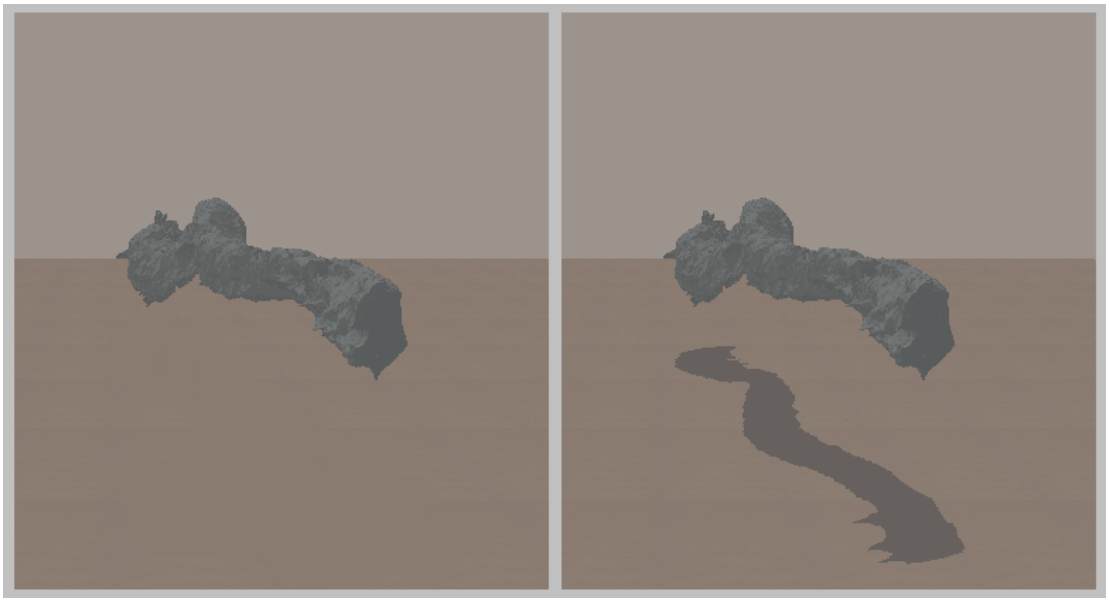


Figure 3.18: Shadow implementation
Comparison between image display without, left section, and with shadow, right section, projected to the ground.

3.3.2.3 Integration of SBFSEM Data

The integration of SBFSEM images gives the user the opportunity to correlate the reconstructed 3D structure with the original SBFSEM data. Therefore single SBFSEM images can be selected and are added in relation to the orientation of the 3D object (fig. 3.19A). Also an image cube can be generated from multiple SBFSEM slices (fig. 3.19B). Hereby

the user has the possibility to define a "slicing mode" where depending on the specified time always the first image is removed until the end of the image cube. Furthermore for the verification of the resulting 3D structure the 2D segmentation of a region can be displayed in a separate window. Hereby it is possible to edit the corresponding contour line (see section 3.2.3.3) and after confirmation of the changes the 3D structure is automatically updated.

3.3.3 Results and Data Storage

Finally the software provides three different storage possibilities:

1. QuickTime movies (file extension .mov):
   Storage of a video stream of the animated 3D scene in the QuickTime file format.

2. Portable network graphics (file extension .png):
   Screenshots can be stored in the bitmapped image format, png, which employs lossless data compression.

3. XML file (file extension .xml):
   Store screenshots of the entire rendered scene, including transformation coordinates, light conditions and so on. These files can be reloaded at a later time point to further modify the 3D scene.

To give an impression of the final visualization of neuronal structures figure 3.19 displays 3D surfaces of axonal structures in the outer chiasm of the blowfly *Calliphora vicina.*

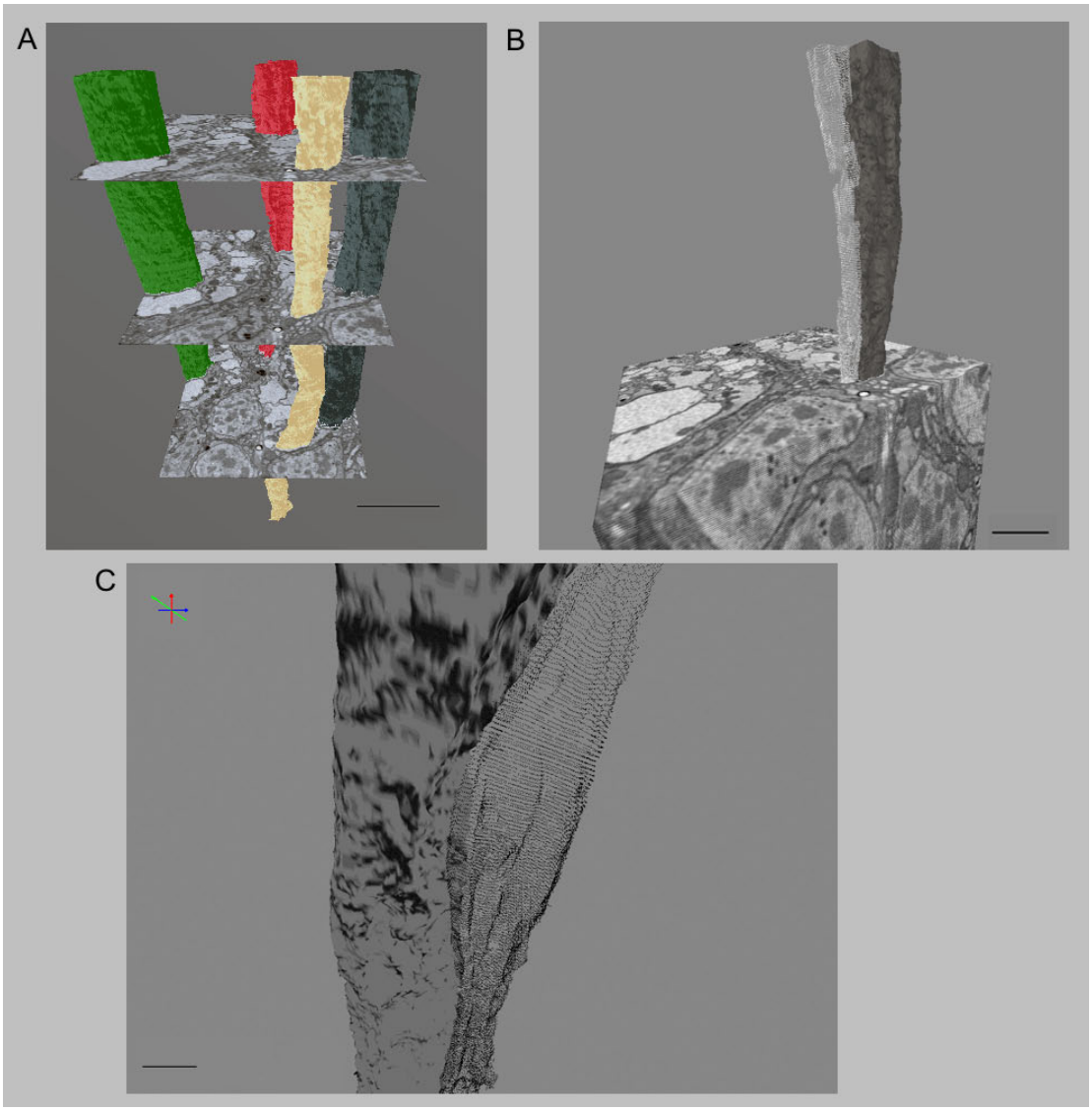


Figure 3.19: Reconstruction of axonal structures
Size of image stack: 14 x 14 x 26 µm^3, slice thickness 50nm. (A) Multiple axonal structures with interlaced SBFSEM-Images. (B) Two axonal structures lying close to each other, embedded in a SBFSEM-Image Block using the slicing mode. Average diameter of process shown in gray is 1.53 µm and in white 1.13 µm (C) Zoomed and rotated view of figure 3.19B. Scalebars: 2.6 µm A, 1.81 µm B and 0.54 µm C.

3.4 Software Usage and Evaluation

The two software packages Neuron2D and Neuron3D are stand-alone applications and therefore mainly used in single mode. However, as sometimes it is desirable and helpful to visualize the growing 3D structures during the segmentation process, it is also possible

two run both programs in parallel. Additionally both programs are executable under 32 and 64 bit operating systems.

For the evaluation of the software packages Neuron2D and Neuron3D a user test was performed. Therefore ten individuals had to achieve the following two tasks after a short introduction about how the software operates. In the first task participants had to test the functionality of Neuron2D, especially the segmentation of an image stack. The second task was performed to test the usability of Neuron3D to setup a 3D scene.

After the experiment participants had to fill out a questionnaire reporting on their experiences with the two software packages.

Task I - Software Neuron2D:

1. Create a new project, using the images in the folder "D:\imageStack\". Select the following parameters:
   - Image range: Extract a region of [100,400,612,912].
   - Filter method: Gaussian, window size: 5, standard deviation: 1.
   - Filter options: Mean: 150, variance: 80.

2. Use the function "new segmentation" of the image processing button for the segmentation of the previously filtered images.
   - First slice: image with number 0001.
   - Number of images: 30.
   - Parameters: Select method 1 - exact segmentation.
   - Select Button "Parallel 3D Reconstruction".

3. Select the following three regions (fig. 3.20), start/analyze the segmentation and try to reconstruct the 3D structure:

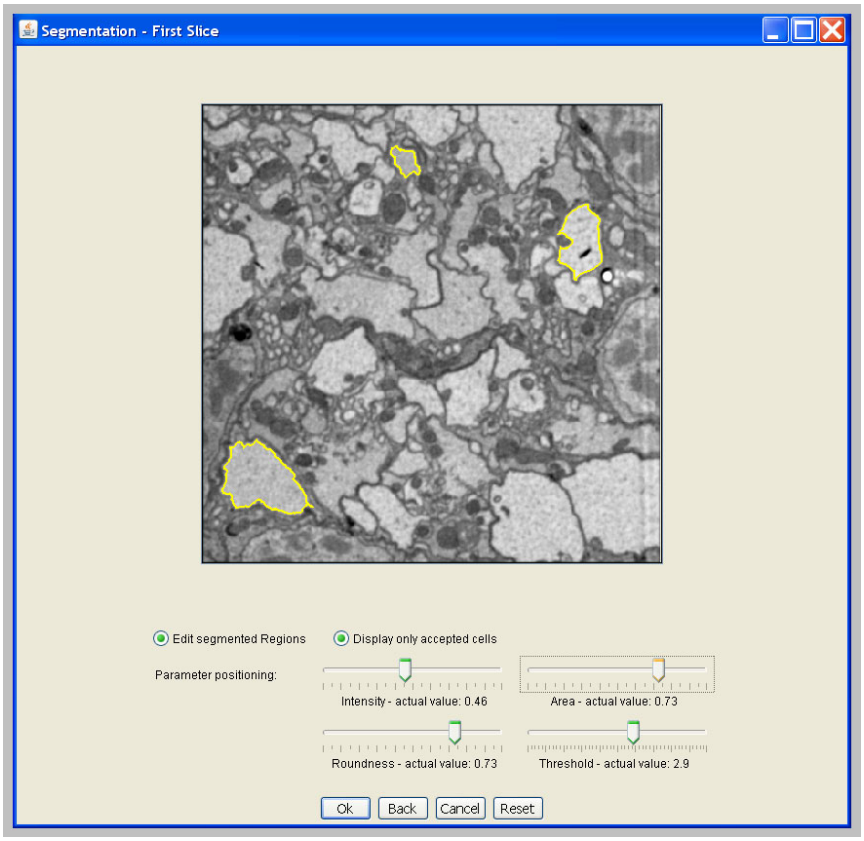


Figure 3.20: Selected test regions

4. Please fill out the following questionnaire.

Software Evaluation - Neuron2D (Scoring: 1 = best up to 6 = worst)							
Question	**1**	**2**	**3**	**4**	**5**	**6**	**Note**
How clearly laid out is the GUI in general?							
How intuitive is the usability? (e.g. name of a button vs. its function)							
How useful is the help file?							
How stable is the application? Did an error occur?							

How is your general feeling about the segmentation results? (correctness of the result vs. computing time)							
Please quantify the usability of the manual interaction.							

Task II - Software Neuron3D:

1. Load the xml-file "D:\axonalStructures.xml\".
2. Play around with the various software functions.
3. Please fill out the following questionnaire.

Software Evaluation - Neuron3D (Scoring: 1 = best up to 6 = worst)							
Question	**1**	**2**	**3**	**4**	**5**	**6**	**Note**
How clearly laid out is the GUI in general?							
How intuitive is the usability? (e.g. name of a button vs. its function)							
How useful is the help file?							
How stable is the application? Did an error occur?							
How would you quantify the implemented rendering functions (e.g. color, light, texture) for setting up a realistic 3D scene?							
How satisfied are you with the storage options (mov, png, xml)? If you wish any extensions, please note them.							

The questions in the questionnaire allowed personal estimations between 1 (best) and 6 (worst). The results of the subjective answers are summarized in the following plots (fig. 3.21).

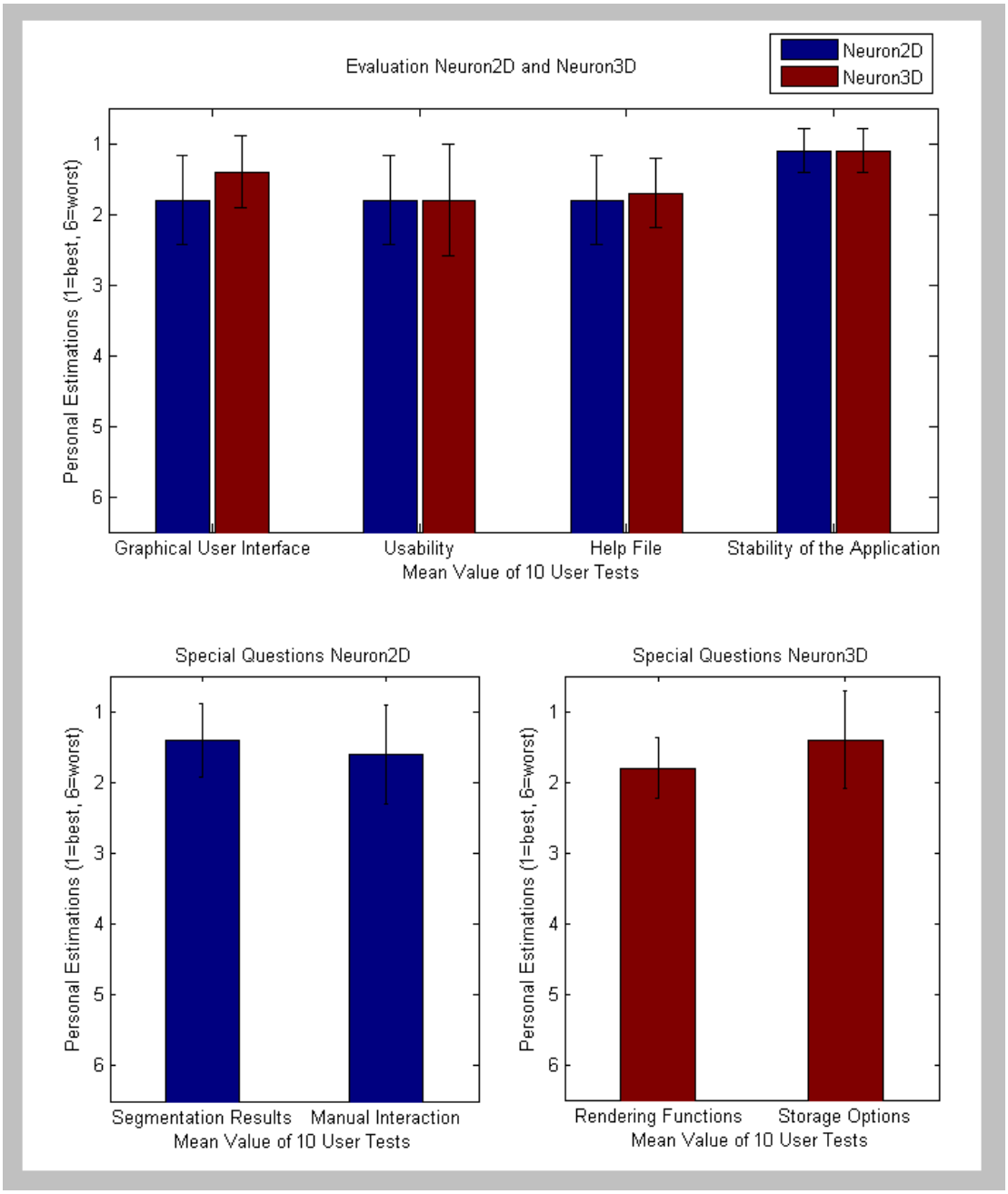


Figure 3.21: Software evaluation

The evaluation of the user test shows that both applications have a clearly arranged

GUI and are easy to use. In addition the help files are convenient and both applications run stable (fig. 3.21, upper graph).

For both software packages two additional questions, specially for each program, were asked in the questionnaire. Hereby for Neuron2D users had to quantify the segmentation results and the manual interaction (sec. 3.2.3.3) during the segmentation process. As seen in the lower left plot of figure 3.21 for both questions the mean of the personal estimation is around 1.5. The lower right plot of figure 3.21 displays the results of the two additional questions of Neuron3D. Hereby users quantified the rendering functions as useful and satisfying. Also the possible storage options are adequate. Two users mentioned that storage options should be extended with e.g. jpeg or tiff picture format.

Summing up both software packages were evaluated user-friendly and suitable to analyze SBFSEM image stacks.

4 Discussion

In this work, I described the software development for the analysis and reconstruction of SBFSEM image stacks. In the following I summarize the specific requirements that had to be met by the software to allow fast and user-friendly processing of the memory-intensive SBFSEM data. Finally I will give a short overview of possible applications of the software for answering biological questions.

4.1 Software Neuron2D and Neuron3D

The developed software package Neuron2D offers robust and fast segmentation of SBFSEM image stacks (section 3.3.3) with optional user interaction. Compared to purely manual tracing, processing time is sped up by a factor of about 50. As images are segmented sequentially, rather than the entire stack at once, no storage intensive operations are necessary. Ultimately, algorithms that do not require any or very little user interaction are desirable and the segmentation algorithms described here present a first step in this direction.

A possible extension for the segmentation algorithms is the integration of flexible priors. So far the prior for the segmentation ϕ_n of image n is peaked at the segmentation ϕ_{n-1} of the previous image, independent of the segmentation of slices $n-k$, with $k > 1$. Given sufficient training data, one could drop this rather restrictive assumption and use more than one image in the calculation of the prior (Cremers, 2006). For example, one could assume ϕ_n to be similar to $2\phi_{n-1} - \phi_{n-2}$, which is the linear prediction based on the two previous images. More generally, one could learn the best prediction of ϕ_n (given previous slices) and use both the prediction and the estimated uncertainty of the prediction for the segmentation process. Operations such as image preprocessing (section 3.2.2) and registration (section 3.2.1) do not require user interaction after the parameters are adjusted and therefore huge image stacks can also be selected for filtering or alignment.

Neuron3D allows the subsequent 3D reconstruction of neuronal structures based on

XML-files generated by Neuron2D. Neuron3D provides standard rendering techniques to display natural 3D scenes. Operations, such as shadows, which have a high computation time (section 3.3.2.2) can be switched off until the final scene is reconstructed. Additionally the integration of single SBFSEM image stacks or an image cube (section 3.3.2.3) gives the user the opportunity to correlate the reconstructed 3D structure with the original data. Furthermore XML-files can be read by visualization softwares, such as Amira, that offers e.g. stereo 3D projection. The surface reconstruction is based on a given collection of planar contours representing cross-sections through the neurons and is implemented as a DTW approach. Therefore 3D objects are represented as tube-like structures. In addition the implemented algorithm allows, parallel to the segmentation step of Neuron2D, the reconstruction of neuronal structures, as only the consecutive planar contours of an object are considered for the triangulation of the surface. As it may be desirable at times to visualize complete reconstructed neurons as isosurfaces, the additional implementation of the marching cubes algorithm (Lorensen and Cline, 1987) is useful. The basic principle behind the marching cubes algorithm is to subdivide space into a series of small cubes. The algorithm "marches" through each of the cubes testing the corner points and replacing the cube with an appropriate set of polygons. The total sum of all generated polygons will be a surface that approximates the one the data set describes.

Both programs are stand-alone applications and therefore are mainly used in single mode. However, as it is sometimes desirable and helpful to visualize the growing 3D structures during the segmentation process, it is also possible two run both programs in parallel. Hereby errors during the segmentation step are identified at an early reconstruction stage and the user can interact. In addition both programs are executable under 32 and 64 bit operating systems.

So far the main implementation of Neuron2D and Neuron3D took into consideration the special requirements for the image analysis and reconstruction of SBFSEM data. Now the software can be extended further e.g. with the usage of plug-ins. One possible extension might be to enlarge the image viewer of Neuron2D. At the moment the image viewer functionality displays images in a sequential order. To allow an interactive and more user-friendly preview of neuronal structures, spanning more than one image stack, it is required to implement an image viewer functionality where the user can navigate through image stacks with the mouse pointer. Consequently the user will have the possibility to preselect neuronal structures across multiple image stacks obtained by tiling (section 3.2.1.1) and generate a sub image stack around the specified region of interest. The final reconstruction

would be less time-intensive, as only a part of the entire data set is considered.

To sum up, the described segmentation algorithms in combination with the developed software packages, Neuron2D and Neuron3D, offer a convenient method of SBFSEM image stack analysis. Furthermore the labeled data-set obtained with Neuron2D and Neuron3D could be used as a training set for more elaborated segmentation algorithms based on artificial neuronal networks.

4.2 Biological Application of the Software

So far axonal tracts of the fly outer chiasm were reconstructed. The next step will be to obtain SBFSEM image stacks from the lobula plate, one of the neuropils of the fly visual system, to analyze the neural circuitry of motion-sensitive LPTCs. To analyze these motion-sensitive neurons it is necessary to first obtain the cell skeletal data such as axons, dendrites and cell bodies. And secondly to acquire information about the anatomical distribution of synapses in the neuronal circuit. Denk and Horstmann (2004) showed that it is possible to visualize and analyze chemical synapses with the current resolution limit (10 x 10 x 30 nm^3) of SBFSEM (fig. 4.1).

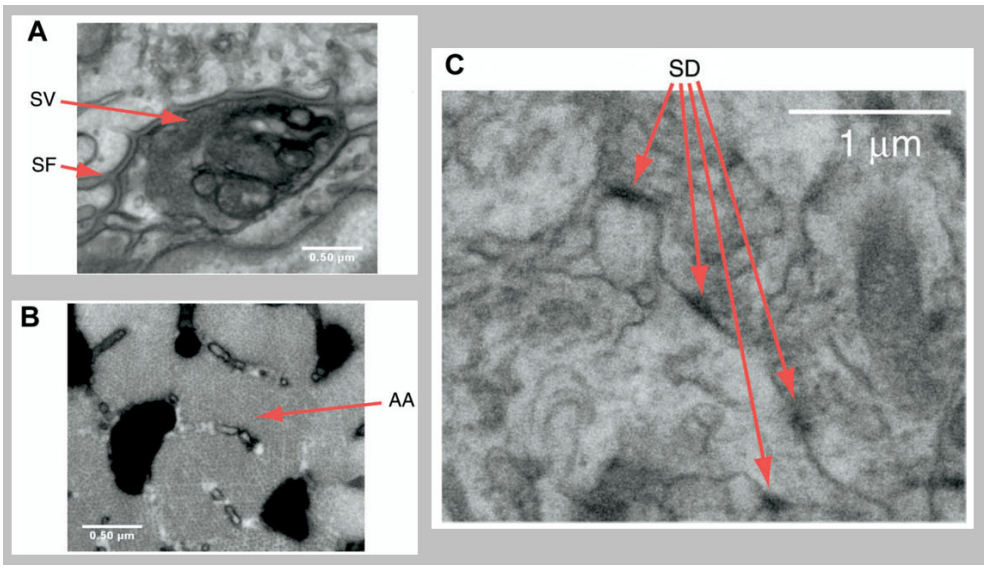


Figure 4.1: Visualization of synapses with SBFSEM
(A) Presynaptic vesicles (SV) and postsynaptic folds (SF) are clearly visible in a motor endplate preparation (Denk and Horstmann, 2004). (B) Similarly, the hexagonal array of action filaments (AA) can be resolved (Denk and Horstmann, 2004). (C) Cortical tissue, pre- and postsynaptic densities (SD) are clearly discernable (Denk and Horstmann, 2004).

However for the reconstruction of complete circuit diagrams of LPTCs it is necessary to obtain information about electrical synapses, as these cells are electrically coupled to their direct neighbors (Haag and Borst 2002, Haag and Borst 2004). To prove the electrical coupling of LPTCs dye coupling was used in addition to electrophysiological recordings. Hereby the injection of Neurobiotin in a single vertical system (VS) cell led to a co-staining of the neighbors of the cell with fainter staining of cells further away from the injected cell (fig. 4.2, Haag and Borst 2005).

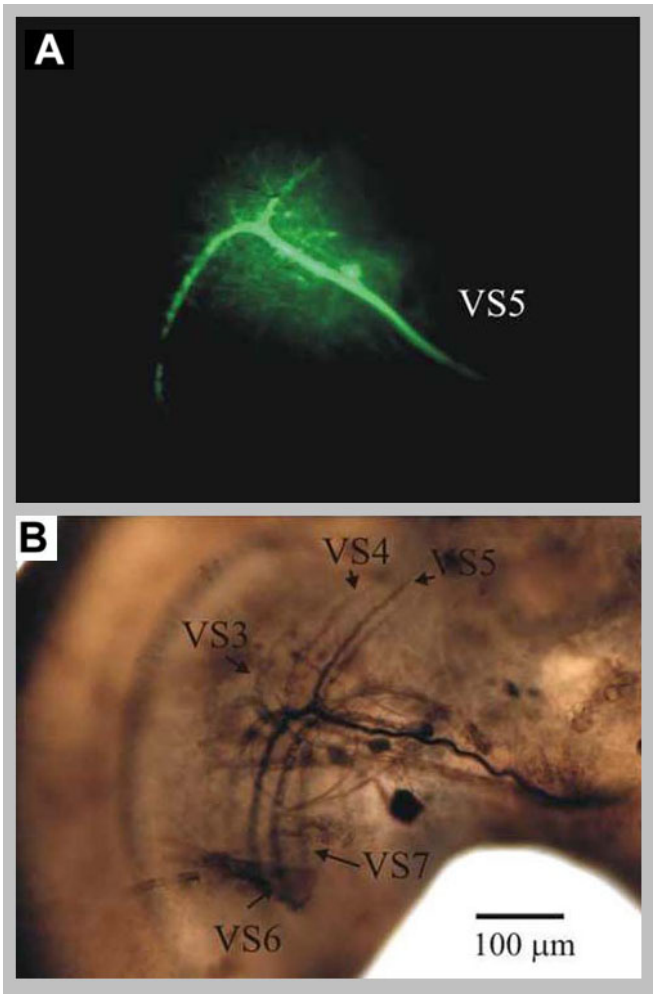


Figure 4.2: Fluorescein and neurobiotin staining of VS-cells
(A) The dye-mixture was injected into a VS5-cell, which is the only cell that is visible under fluorescent light (Haag and Borst, 2005). (B) In the neurobiotin staining, several adjacent VS-cells can be seen. Besides the VS5 cell, which showed the strongest staining, VS3, VS4, VS6 and VS7 were costained. This demonstrates that these cells are connected via electrical synapses (Haag and Borst, 2005).

With the current lateral resolution limit of SBFSEM, 10 nm, it has not been possible

to identify electrical synapses which range from 1 to 7 nm in diameter. Therefore the main structural components of invertebrate gap junction channels, the protein innexins (invertebrate connexin analogues, fig. 4.3), need to be labeled with electron dense material to increase the backscattered coefficient, η. Traditional immunohistochemical techniques e.g. with gold particles are not possible, as the tissue of the specimen block is too compact and consequently the gold labeled antibodies cannot diffuse into the tissue. Using the fruitfly *Drosophila melanogaster* and its powerful genetic toolbox, it might be possible to endogenous enrich innexins with Zinc (using Zinc-fingers) or Selene (using Selenocysteine).

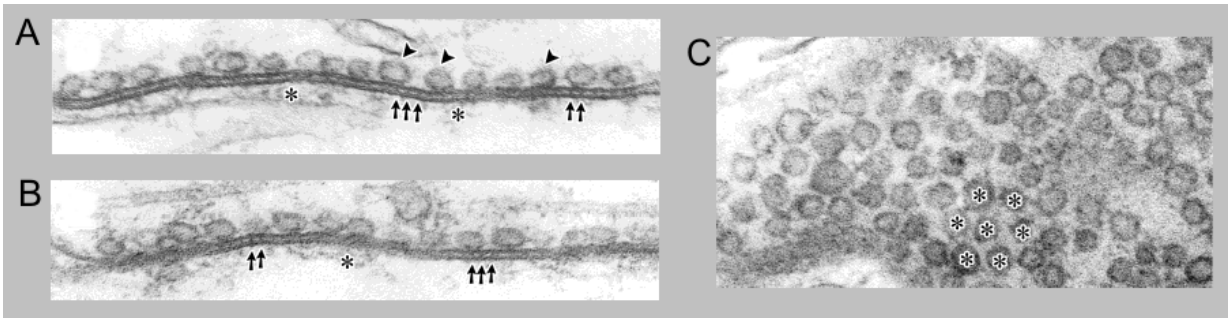


Figure 4.3: EM images of gap junctions
(A) High power view of a region of an electrical-type synapse between the giant fiber (GF, upper) and the tergotrochanteral motorneuron (TTMn, lower) in an Oregon-R animal. The junction has a 2-4 nm intercellular space, with cross striations spaced approximately 18 nm apart (arrows). There is a single row of vesicles in the GF (arrowheads) and some dense material under the TTMn membrane (asterisks) (Blagburn et al., 1999). (B) Neighboring section to that shown in A (Blagburn et al., 1999). (C) Electrical-type synapse sectioned in parallel to the plane of the membranes, showing the regular array of vesicles in GF (asterisks) (Blagburn et al., 1999).

In addition to the detection problem of electrical synapses with SBFSEM the segmentation algorithms of the software package Neuron2D must be adjusted. Generally the junctional region is arranged in a hexagonal array of subunits and appear as simple black dots in EM images (fig. 4.3C). For the development of segmentation algorithms to analyze gap junctions it might be useful to incorporate texture information in addition to pixel intensities. Using texture information allows to look at the neighborhood, $\mathcal{N}(x)$, of each pixel x, and estimate the distributions of these patches. The neighborhood could either be taken to be two-dimensional, i.e. by looking at nearby pixels, or three-dimensional by also including pixels from adjacent images. A classification algorithm could be trained on a set of patches sampled from labeled images, and $L(x)$ is set to be the output of the classifier to the input $\mathcal{N}(x)$. The value of $L(x)$ will then be positive if the neighborhood $\mathcal{N}(x)$ is more likely to belong to the foreground than to the background, and negative otherwise. This

newly constructed image, $L(x)$, can be segmented using the algorithms described above. This approach can be implemented with any classifier that produces real-valued rather than binary output, such as support-vector machines (Kim et al., 2002).

4.3 Outlook

Extending the software package with the described approaches, e.g. image viewer or texture information for the analysis of electrical synapses, will provide a powerful tool for the analysis of SBFSEM data sets. Finally the integration of the obtained anatomical knowledge about the neuronal circuitry of the fly visual system will answer one of the classic questions in computational neuroscience, i.e. how neurons compute the direction of motion.

5 References

Al-Kofahi KA, Lasek S, Szarowski DH, Pace CJ, Nagy G, Turner JN, Roysam B (2002) Rapid automated three-dimensional tracing of neurons from confocal image stacks. *IEEE Transactions on Information Technology in Biomedicine* 6:171–187.

Amira (2006) Amira - Advanced 3D visualization and volume modeling. *http://www.amiravis.com/* .

Barequet G, Shapiro D, Tal A (2000) Multilevel sensitive reconstruction of polyhedral surfaces from parallel slices. *The Visual Computer* 16:116–133.

Blagburn JM, Alexopoulus H, Davies JA, Bacon JP (1999) Null mutation in shaking-B eliminates electrical, but not chemical, synapses in the Drosophila Giant Fiber System: A structural study. *Journal of Comparative Neurology* 404:449–458.

Blinn J (1988) Jim Blinn's Corner: Me and my (fake) shadow. *IEEE Computer Graphics and Applications* 8:82–86.

Boissonnat JD (1988) Shape reconstruction from planar cross sections. *Computer Vision, Graphics and Image Processing* 44:1–29.

Borst A, Haag J (1996) The intrinsic electophysiological characteristics of the fly lobula plate tangential cells. I. Passive membrane properties. *Journal of Computational Neuroscience* 3:313–336.

Borst A, Haag J (2002) Neural networks in the cockpit of the fly. *Journal of Comparative Physiology* 188:419–37.

Bracewell RN (1965) *The Fourier Transform and its applications.* McGraw-Hill Science Engineering.

Briggman K, Denk W (2006) Towards neural circuit reconstruction with volume electron microscopy techniques. *Current Opinion in Neurobiology* 16:562–570.

Brotz TM, Egelhaaf M, Borst A (1995) A preparation of the blowfly (Calliphora erythrocephala) brain for in vitro electrophysiological and pharmacological studies. *Journal of Neuroscience Methods* 57:37–46.

Brown L (1992) A survey of image registration techniques. *ACM Computing Surveys* 24:326–376.

Buzsaki G (2004) Large-scale recording of neuron ensembles. *Nature Neuroscience* 7:446–451.

Canny JF (1986) A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8:679–698.

Carlbom I, Klinker G, Terzopoulus D (1995) Soft tissue segmentation from medical images using interactive deformable contours. *Proceedings of the 9th Scandinavian Conference on Image Analysis, Uppsala, Sweden,* pp. 905–912.

Chan TF, Vese L (2001) Active contours without edges. *IEEE Transactions on Image Processing* 10:266–277.

Conchello JA, Lichtman JW (2005) Optical sectioning microscopy. *Nature Methods* 2:920–931.

Cremers D (2006) Dynamical statistical shape priors for level set based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28:1262–1273.

Cremers D, Rousson M (2006) Efficient kernel density estimation of shape and intensity priors for level set segmentation In Suri JS, Farag A, editors, *Parametric and Geometric Deformable Models: An application in Biomaterials and Medical Imagery*. Springer.

Cremers D, Rousson M, Deriche R (2006) A review of statistical approaches to level set segmentation: integrating color, texture, motion and shape. *International Journal of Computer Vision* pp. 67–81.

D'Almeida F (2000) Nonlinear Diffusion Toolbox for Matlab. *http://www.mathworks.com/matlabcentral/fileexchange/loadAuthor.do?objectType=author&objectId=938351* .

de Boor C (1978) *A practical guide to splines* Springer Verlag.

Denk W, Horstmann H (2004) Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology* 2:e329.

Denk W, Strickler JH, Webb WW (1990) 2-photon laser scanning fluorescence microscopy. *Science* 248:73–76.

Donald A (2003) The use of an environmental scanning electron microscopy for imaging wet and insulating materials. *Nature Materials* 2:511–516.

Ekoule AB, Peyrin FC, Odet CL (1991) A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics* 10:165–172.

Fonseca L, Manjunath B (1996) Registration techniques for multisensory remotely sensed imagery. *Photogrammetric Engineering and Remote Sensing.* 62:1049–1056.

Frey TG, Perkins GA, Ellisman MH (2006) Electron tomography of membrane-bound cellular organelles. *Annual Review of Biophysics & Biomolecular Structure* 35:199–224.

Ganapathy S, Dennehy TG (1978) A new general triangulation method for planar contours. *In Proceedings SIGGRAPH* pp. 69–75.

Goldstein J, Newbury D, Joy D, Lyman C, Echlin P, Lifshin E, Sawyer L, Michael J (2003) *Scanning Electron Microscopy and X-Ray Microanalysis* Springer Science and Media.

Gonzalez R, Woods R (2002) *Digital Image Processing* Prentice Hall.

Gouraud H (1971) Continuous shading of curved surfaced. *IEEE Transactions on Computers* 20:623–628.

Haag J, Borst A (2004) Neural mechanism underlying complex receptive field properties of motion-sensitive interneurons. *Nature Neuroscience* 7:628–34.

Haag J, Borst A (2005) Dye-coupling visualizes networks of large-field motion-sensitive neurons in the fly. *Journal of Comparative Physiology* 191:445–454.

Haag J, Borst A (2002) Dendro-dendritic interactions between motion-sensitive large-field neurons in the fly. *Journal of Neuroscience* 22:3227–33.

Hayat M (2000) *Principles and techniques of electron microscopy, biological applications.* Cambridge University Press.

Heckbert P (1991) Simulating global illumination using adaptive meshing. *University of California at Berkeley* .

Hell SW (2003) Toward fluorescene nanoscopy. *Nature Biotechnology* 21:1347–1355.

Hirschberg D (1975) A linear space algorithm for computing maximal common subsequences. *Communications of the ACM* 18:341–343.

Hosmer D, Lemeshow S (2000) *Applied logistic regression* Wiley-Interscience Publication.

Huang B, Li H, Huang X (2005) A level set method for oil slick segmentation in SAR images. *International Journal of Remote Sensing* 26:1145–1156.

Jurrus E, Tasdizen T, Koshevoy P, Fletcher PT, Hardy M, Chien CB, Denk W, Whitaker R (2006) Axon Tracking in Serial Block-Face Scanning Electron Microscopy In *Workshop on Microscopic Image Analysis with Applications in Biology*.

Kass M, Witkin A, Terzopoulus D (1988) Snakes: Active contour models. *International Journal of Computer Vision* 1:321–331.

Keppel E (1975) Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development* 19:2–11.

Kim K, Jung K, Park S, Kim H (2002) Support Vector Machines for texture classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24:1542–1550.

Koh I, Lindquist W, Zito K, Nimchinsky EA, Svoboda K (2002) An image analysis algorithm for dendritic spines. *Neural Computation* 14:1283–310.

Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* 7:48–50.

Kuglin C, Hines D (1975) The phase correlation image alignment method. *Proceedings of IEEE International Conference on Cybernetics and Society* pp. 163–165.

Kuzirian AM, Leighton SB (1983) Oxygen plasma etching of entire block face improves the resolution and usefulness of serial scanning electron microscopic images. *Scanning Electron Microscopy* pp. 1877–1885.

Leighton SB (1981) SEM images of block faces, cut by a miniature microtome within the SEM - A technical note. *Scanning Electron Microscopy* 2:73–76.

Lorensen WE, Cline HE (1987) Marching Cubes: A high resolution 3D surface construction algorithms. *Computer Graphics* 21:163–169.

Lowe DG (1999) Object Recognition from Scale-Invariant Features. *Proceedings of the International Conference on Computer Vision* pp. 1150–1157.

Luebke DP (2001) A developer's survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications* pp. 24–35.

Meinertzhagen I, O'Neil S (1991) Synaptic organization of columnar elements in the lamina of the wild type in Drosphila melanogaster. *Journal of Comparative Neurology* 305:232–263.

Meyers D, Skinner S, Sloan K (1992) Surfaces from contours. *ACM Transactions on Graphics* 11:228–258.

Moeller T, Haines E (2002) *Real-Time Rendering* AK Peters, Ltd.; 2 edition.

Mumford D, Shah J (1989) Optimal approximations by piecewise smooth functions and associated variational problems. *Communications on Pure and Applied Mathematics* 42:577–684.

Myers C, Rabiner L (1981) A comparative study of several dynamic time-warping algorithms for connected word recognition. *The Bell System Technical Journal of Comparative Neurology* 60:1389–1409.

Olivia JM, Perrin M, Coquillart S (1996) 3D reconstruction of complex polyhedral shapes from contours using simplified generalized Voronoi diagram. *Computer Graphics Forum* 15:397–408.

Osher S, Fedkiw R (2003) *Level Set Methods and Dynamic Implicit Surfaces*, Vol. 153 of *Applied Mathematical Sciences* Springer Verlag, New York.

Osher S, Sethian J (1988) Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics* 79:12–49.

Perona P, Malik J (1990) Scale-Space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12:629–639.

Phong BT (1975) Illumination for computer generated pictures. *Communications of the ACM* 18:311–317.

Pratt WK (1974) Correlation techniques of image registration. *IEEE Transaction on Aerospace and Electronic Systems* 10:353–358.

Reimer L (1993) *Transmission Electron Microscopy: Physics of image formation and microanalysis* Springer Verlag.

Schmitt S, Evers J, Duch C, Scholz M, Obermayer K (2004) New methods for the computer-assisted 3-D reconstruction of neurons from confocal image stacks. *Neuroimage* 23:1283–98.

Schoenberg I (1946) Contributions to the problem of approximation of equidistant data by analytic functions. *Quarterly of Applied Mathematics* 4:45–99,112–141.

Sethian JA (1999) *Level Set Methods and Fast Marching Methods* Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press.

Shapiro L, Stockman G (2001) *Computer Vision* Prentice Hall.

Shreiner D, Woo M, Neider J, Davis T (2005) *OpenGL Programming Guide* Addison-Wesley.

Sielhorst T, Blum T, N. N (2005) Synchronizing 3D movements for quantitative comparison and simultaneous visualization of actions. *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality* pp. 38–47.

Soto GE, Young SJ, Martone ME, Deerinck TJ, Lamont S, Carragher BO, Hama K, Ellisman MH (1994) Serial section electron tomograph: a method for three-dimensional reconstruction of large structures. *Neuroimage* 1:230–243.

Stempak J, Ward R (1964) Improved staining method for electron microscopy. *Journal of Cell Biology* 22:697–701.

Stevens JK, Davis TL, Sterling P (1980) A systematic approach to reconstructing microcircuitry by electron microscopy of serial sections. *Brain Research* 2:265–293.

Strausfeld NJ (1984) Functional neuroanatomy of the blowfly's visual system. In Ali MA, editor, *Photoreception and vision in invertebrates*, pp. 483–522. Plenum Publishing Corporation.

Strausfeld NJ, Douglass J (2003) Anatomical organization of retinotopic motion-sensitive pathways in the optic lobes of flies. *Microscopy Research and Technique* 62:132–150.

Viola P, Wells W (1997) Alignment of maximization of mutual information. *International Journal of Computer Vision* 24:137–154.

Vollgraf R, Scholz M, Meinertzhagen I, Obermayer K (2004) Nonlinear filtering of electron micrographs by means of Support Vector Regression In Thrun S, Saul L, Scholkopf B, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Whitaker R, Breen D, Museth K, Soni N (2001) Segmentation of biological volume datasets using a level set framework. *Volume Graphics* pp. 249–263.

White JG, Southgate E, Thomson JN, Brenner S (1986) The structure of the nervous system of the nematode Caenorhabditis elegans. *Philosophical Transaction Royal Society Publishing London* A 314:1–340.

Woo A, Poulin P, Fournier A (1990) A survey of shadow algorithms. *IEEE Computer Graphics and Applications* 10:13–32.

Yin L, Yang M, M. G, Y. N (1996) Weighted median filters: A tutorial. *IEEE Transactions on Circuits and Systems* 43(3):157–192.

Zitova B, Flusser J (2003) Image registration methods: a survey. *Image and Vision Computing* 21:977–1000.

Acknowledgements

I want to thank all the people for supporting me during the past years and in particular during the development of this thesis, even if I cannot mention all of their names here.

First of all, I want to thank my supervisor Alexander Borst a lot, not only for making my dissertation possible, but also for giving me helpful advice, guidance, and support during the last years. In addition for providing a great working environment in his department. Christoph Kapfer I want to thank for many interesting discussions and very helpful suggestions on my work. Gudrun Klinker for being part of my thesis committee and for her theoretical and practical support.

I also want to thank the whole 'Borst Department' for having a wonderful time in the laboratory and all the PhD students who came along with me. Especially Yong Choe for helpful suggestions on the manuscript. Friedrich Förstner for Amira support and computer science discussions. Bettina Schnell and Thomas Hendel for having lots of fun and discussions in our room. And of course the members of the 'coffee-lunch group': Adrian Wertz, Marco Mank, Stephan Direnberger, Johannes Plett, Alex Ihring, Nicola Heim for sharing a lot of time with me and having many discussions about science and life.

In addition I want to thank my family, especially my parents who always supported me, and friends for being with me. Finally, I want to thank David for always encouraging me and his ability to make me laugh and feel good at any time.

Curriculum Vitae

Nina Maack

Education

since 06/2005	**Ph.D. Thesis:** Max-Planck-Institute of Neurobiology, Martinsried, Germany. Department of Systems and Computational Neurobiology. Supervised by Prof. Dr. Alexander Borst. "3D Reconstruction of Neural Circuits from Serial EM Images".
08/2007	**PENS Neuroscience School:** Arcachon, France. Course: Advanced Course in Computational Neuroscience Project: Integration of flow-field signals by descending neurons in the fly visual system.
08/2004 - 02/2005	**Diploma Thesis:** Max-Planck-Institute of Neurobiology, Martinsried, Germany. Department of Systems and Computational Neurobiology. Supervised by Prof. Dr. Alexander Borst. "Software Development for 3D Reconstruction of Image Stacks"
04/2004 - 07/2004	**Student Assistant:** GSF-Institute of Bioinformatics, Munich, Germany. Installation of a Distributed Sequence Annotation System (DAS) Server.

08/2003	**Internship:** GSF-Institute of Molecular Biology and Tumor Genetics, Munich, Germany. Practical Experience in the field of molecular biology.
10/2001 - 07/2003	**Student Assistant:** Dept. of Computer Science, Ludwig Maximilian University, Munich, Germany. Tutor for Computer Science Courses.
10/2000 - 04/2005	**Undergraduate studies in Bioinformatics:** Ludwig Maximilian University and Technical University of Munich, Germany.
05/2000	**Graduation from High School:** Theresia Gerhardinger Gymnasium, Munich, Germany.

Publications (Parts of this thesis have previously been published)

Macke J, Maack N, Gupta R, Denk W, Schoelkopf B, Borst A (2008): **Contour-propagation algorithms for semi-automated reconstruction of neural processes.** J Neurosci Methods 167/2: 349-357.

Ehrenwörtliche Versicherung

Ich versichere hiermit ehrenwörtlich, dass die vorgelegte Dissertation von mir selbständig und ohne unerlaubte Beihilfe angefertigt ist.

Martinsried, den .....

Erklärung

Hiermit erkläre ich, dass ich mich anderweitig einer Doktorprüfung ohne Erfolg **nicht** unterzogen habe und die Dissertation nicht ganz oder in wesentlichen Teilen einer anderen Prüfungskommission vorgelegt worden ist.

Martinsried, den .....