The standard brain of *Drosophila melanogaster* and its automatic segmentation

Dissertation zur Erlangung des
naturwissenschaftlichen Doktorgrades
der Bayerischen Julius-Maximilians-Universität Würzburg

vorgelegt von

Johannes Schindelin

Eingereicht am: . . . .

Mitglieder der Promotionskommission:
Vorsitzender: . . . .
Gutachter: . . . .
Gutachter: . . . .

Tag des Promotionskolloquiums: . . . .

Doktorurkunde ausgehändigt am: . . . .

The standard brain of *Drosophila melanogaster* and its automatic segmentation

J. E. Schindelin
Johannes.Schindelin@biozentrum.uni-wuerzburg.de
Lehrstuhl für Genetik und Neurobiologie, Biozentrum,
Universität Würzburg,
Am Hubland, 97074 Würzburg, Germany

Abstract

In this thesis, I introduce the Virtual Brain Protocol, which facilitates applications of the *Standard Brain* of *Drosophila melanogaster*. By providing reliable and extensible tools for the handling of neuroanatomical data, this protocol simplifies and organizes the recurring tasks involved in these applications. It is demonstrated that this protocol can also be used to generate average brains, i.e. to combine recordings of several brains with the same features such that the common features are emphasized.

One of the most important steps of the Virtual Insect Protocol is the aligning of newly recorded data sets with the Standard Brain. After presenting methods commonly applied in a biological or medical context to align two different recordings, it is evaluated to what extent this alignment can be automated. To that end, existing Image Processing techniques are assessed. I demonstrate that these techniques do not satisfy the requirements needed to guarantee sensible alignments between two brains. Then, I analyze what needs to be taken into account in order to formulate an algorithm which satisfies the needs of the Protocol.

In the last chapter, I derive such an algorithm using methods from Information Theory, which bases the technique on a solid mathematical foundation. I show how Bayesian Inference can be applied to enhance the results further. It is demonstrated that this approach yields good results on very noisy images, detecting apparent boundaries between structures. The same approach can be extended to take additional knowledge into account, e.g. the relative position of the anatomical structures and their shape. It is shown how this extension can be utilized to segment a newly recorded brain automatically.

Zusammenfassung

In dieser Arbeit wird das Virtual Brain Protocol vorgestellt, das die Anwendungen rund um das Standardgehirn von *Drosophila melanogaster* erleichtert. Durch das Bereitstellen robuster und erweiterbarer Werkzeuge zum Verarbeiten neuroanatomischer Datensätze ermöglicht es ein strukturiertes Abarbeiten der häufig benötigten Vorgänge im Zusammenhang mit der Arbeit mit dem Standardgehirn. Neben der Einpassung neuer Daten in das Standardgehirn kann dieses Protokoll auch dazu verwendet werden, sogenannte Durchschnittshirne zu erstellen; Aufnahmen mehrerer Hirne mit der gleichen zu zeigenden Eigenschaft können zu einem neuen Datensatz kombiniert werden, der die gemeinsamen Charakteristika hervorhebt.

Einer der wichtigsten Schritte im Virtual Insect Protocol ist die Alignierung neuer Datensätze auf das Standardgehirn. Nachdem Methoden vorgestellt werden, die üblicherweise im biologischen oder medizinischen Umfeld angewendet werden, um Hirne aufeinander zu alignieren, wird evaluiert, inwiefern dieser Prozess automatisierbar ist. In der Folge werden diverse bildverarbeitende Methoden in dieser Hinsicht beurteilt. Es wird demonstriert, dass diese Verfahren den Anforderungen sinnvoller Alignierungen von Hirnen nicht genügen. Infolgedessen wird genauer analysiert, welche Umstände berücksichtigt werden müssen, um einen Algorithmus zu entwerfen, der diesen Anforderungen genügt.

Im letzten Kapitel wird ein solcher Algorithmus mithilfe von Methoden aus der Informationstheorie hergeleitet, deren Verwendung das Verfahren auf eine solide mathematische Basis stellt. Es wird weiterhin gezeigt, wie Bayesische Inferenz angewendet werden kann, um die Ergebnisse darüber hinaus zu verbessern. Sodann wird demonstriert, daß dieser Algorithmus in stark verrauschten Bilddaten ohne zusätzliche Informationen Grenzen zwischen Strukturen erkennen kann, die mit den sichtbaren Grenzen gut übereinstimmen. Das Verfahren kann erweitert werden, um zusätzliche Informationen zu berücksichtigen, wie etwa die relative Position anatomischer Strukturen sowie deren Form. Es wird gezeigt, wie diese Erweiterung zur automatischen Segmentierung eines Hirnes verwendet werden kann.

Contents

List of Figures

Chapter 1

Introduction

The very basis of science is to have a standard, a common ground upon which to place results. Mathematical studies, for example, always begin by defining the terms to be used, and then continue by stating interrelations between the objects described by those terms. Computer science inherits those standards, as do many areas evolved from mathematical roots, but it also adds new types of standards like platforms, languages or network protocols.

Fields like biology or theoretical medicine, which are geared towards explaining results from experiments, also need another type of standard: observations are inevitably imprecise. Thus, a standard is needed to describe differences which can be explained by the process by which the data were obtained.

There exists still another kind of standard: if the observed data are ever changing, but certain aspects remain similar, the standard itself has to be adaptable. For example, a human body normally has one liver, the purpose of which stays the same between two people while the exact form and location does not.

This kind of standard is needed when doing research about brains: evolution being at work, each individual brain has a different shape, but certain areas can be identified by approximate shape and/or consistency. Certain functions of the brain were shown to be performed in very specific areas of the brain, like the memory of shapes or odors. It is important to be able to identify these locations for different brains; So, a standard to do that was created.

What is such a standard? Geographers, who had a similar problem, namely to localize regions, invented maps for that purpose. So, by introducing a coordinate system, a location can be specified consistently by its coordinates. Sometimes it is easier, or just preferable to refer to a location by name (like cities or rivers), which is tantamount to stating the coordinates, as long as the same map is used.

The first attempts to define standard brains imitated the geographers' approach, interpreting brains as three-dimensional maps (also called *atlas*), where locations were named, and identified by arrows pointing to them (see [1, 2, 3]).

However, the geographers' topic is constant, i.e. there are not many similar Earths, but it is one and the same. In contrast, when neurobiologists study the brain, in fact they study many brains of the same species. However a brain is recorded, deviations due to developmental differences and mechanical constraints have to be expected. Corresponding regions of those brains are known to look still similar enough, so that from a brain atlas created by naming landmarks in images of one specific brain, an anatomist can find the same landmarks in images of another brain of the same species. Mathematically speaking, the coordinate system of a brain atlas can not be applied to every brain unalteredly, but has to be adapted slightly to fit that brain. In this context, the term "fitting" does not mean "perfectly matching": the details of any two brains are too different for a perfect match.

An atlas as a reference to describe locations in the brain is one use of the standard brain. At the same time it is important to define the method of determining the mapping of a newly recorded brain onto that atlas. This is usually the same method used to create the standard brain, which involves recordings of several brains and then mapping them onto a common template. Irrespective of the particular algorithm, in so doing the brains are put into a common context, where they can be compared in a meaningful manner. Subsequently, the mean goodness of fit can be calculated, which is often understood as an integral part of the standard brain, because it is one way to measure the expected performance of fitting a newly recorded brain onto the standard[1].

1.1 Overview

1.1.1 Research Goal & Context

The work described here was performed in the Heisenberg laboratory in the context of the *Virtual Neuro Lab* project (see [5]). The principal purpose of this project was to explore and extend the possibilities of computers for neuroanatomical analyses using confocal microscopy. Several laboratories interested in different aspects of brains contributed to that project, studying such diverse subjects as the locations of genetic activity in the brain of *Drosophila melanogaster* (see section 2.2), or the efficient visualization of single neurons in 3d data sets of small brains.

As mentioned earlier, in science a standard is necessary to produce repeatable results. In the context of neuroanatomy, this standard includes the generation of average brains (see chapter 3). In fig. 1.2, an average brain is visualized by volume rendering, as compared to a single brain in fig. 1.1. A special case of an average brain is a standard brain[2], that serves as a common reference. Ideally, only one standard brain exists per species. In some cases, however, several Standard Brains exist, as is the case with the human brain. If several Standard Brains exist for the same species, they typically serve as reference in different research contexts.

For *Drosophila melanogaster*, there is only one Standard Brain, which was introduced by K. Rein in [6]. A part of my work was to implement a software suite to assist the neuroanatomist with the tasks related to the Standard Brain. The most typical task is arguably the generation of average brains, which are registered on the Standard Brain (see section 3.3.1). The main purposes of the software suite are to relieve the user of repetitive tasks, and to provide methods to analyze the results.

To generate an average brain, a reliable method is needed to match corresponding locations in two different brains. In [6], a neuroanatomically sensible method was presented, which was used to generate the Standard Brain of *Drosophila melanogaster*. The most involved step of this method is the segmentation (the labeling of neuropiles), which is necessary to obtain a neuroanatomically sensible matching. In fig. 1.3, some segmented neuropiles of the data set in fig. 1.1 are shown. This particular method of matching two data sets has proven to be of high value for anatomical studies.

In [6], brains were segmented manually, which takes a substantial amount of time: for every second slice of the 3d stack, the outlines of the neuropiles have to be marked. Since the 3d stacks of *Drosophila melanogaster* brains typically consist of $150--200$ slices of 1024×1024 pixels, the segmenter usually spends several

[1]The need to have a robust standard, i.e. an atlas onto which most recordings can be mapped well, was recognized and studied in the context of human brain MRI recordings in [4].

[2]The goodness of fit, which was mentioned earlier, is a byproduct of the average brain process.

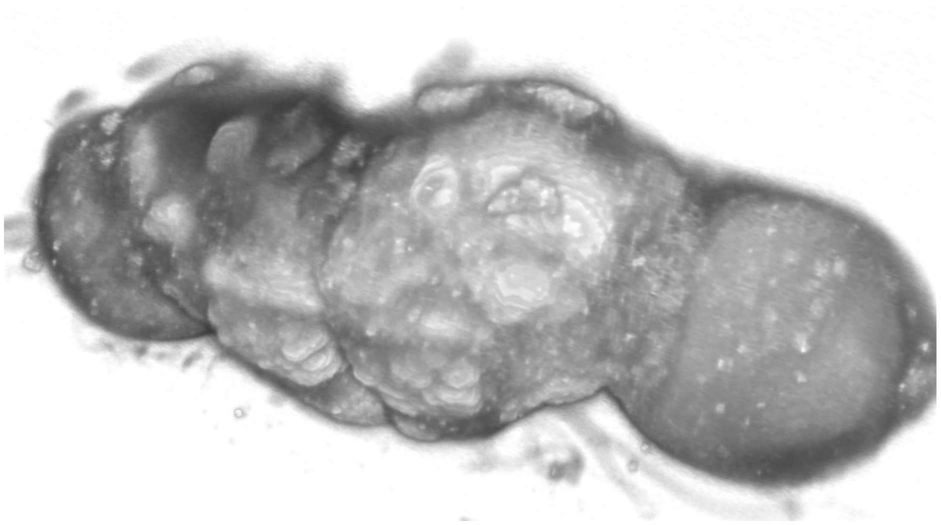


Figure 1.1: A volume rendering of a single brain.

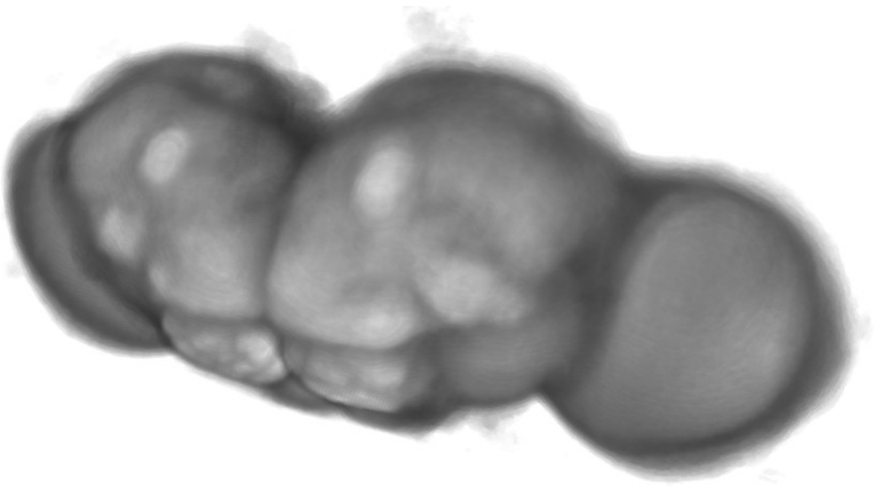


Figure 1.2: A volume rendering of an average brain.

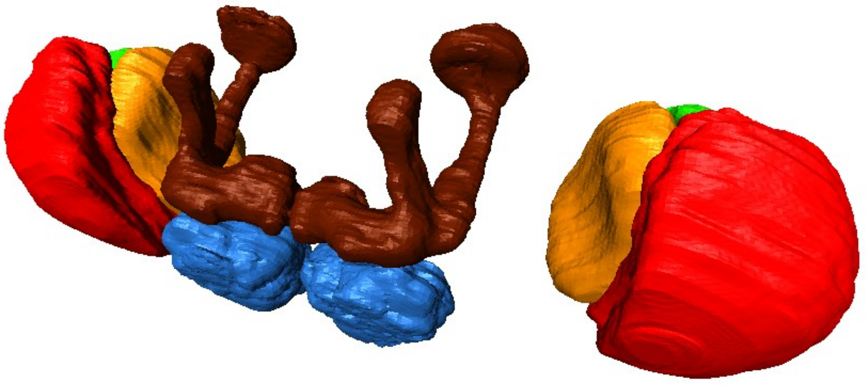


Figure 1.3: Ten segmented neuropiles of the data set shown in fig. 1.1. The neuropiles are color coded: the optic lobes to the right and left are divided in the medulla (red), lobula (yellow) and lobula plate (green). From the central brain, only the mushroom bodies (brown) and the antennal lobes (blue) are shown.

hours on one brain. One of the goals of my work was therefore to automate the segmentation process.

1.1.2 Results

A software suite – the *VIB protocol* – has been set up facilitating the common procedures related to the Standard Brain of *Drosophila melanogaster* (see chapter 3). These include the mapping of newly recorded brains on the standard brain, generation of average brains and calculation of statistics of the recorded brains. The VIB protocol proved to be a helpful tool for these tasks. Furthermore, it can be easily extended to perform other tasks related to the Standard Brain, such as registering single neurons into it.

The VIB protocol is in regular use by the Heisenberg laboratory, and in an ongoing effort, it is being extended to quantify gene expression patterns obtained from Gal4 lines. There exist several thousands of Gal4 lines. In many cases, these lines can be associated with single genes. By studying their expression patterns, changes in the behavior can be linked to genetic activity. Because of the high variability of the anatomy of the brains, the standardization of expression patterns is an important application of the VIB protocol.

Since the methods of the VIB protocol are not restricted to the Standard Brain of *Drosophila melanogaster*, laboratories studying other species have shown interest in the VIB protocol. Due to the fact that the protocol does not expect data from a particular recording method, it is even possible to handle recordings from mammals, such as MRI recordings from rat brains.

Traditional approaches to image segmentation have been analyzed, to determine whether they can be used to segment the neuropiles of *Drosophila melanogaster* automatically. It turned out that they fail, and can not be enhanced to segment a brain (see chapter 4). However, some ideas proved to be useful for my further research.

A new method has been developed, which can segment 2d images by localizing known shapes (see chapter 5). One such segmentation is shown in fig. 1.4.

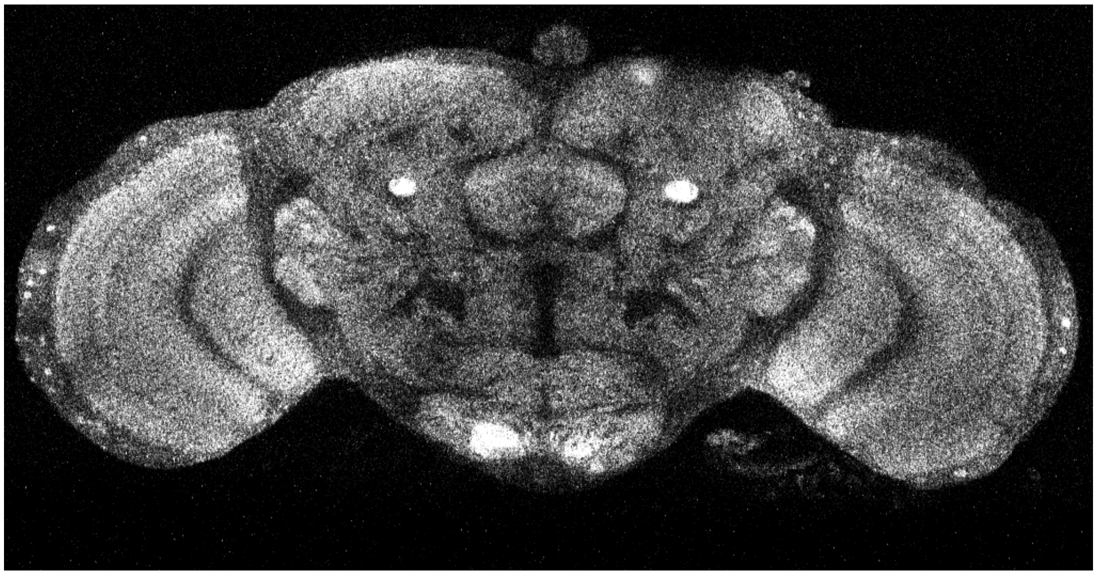
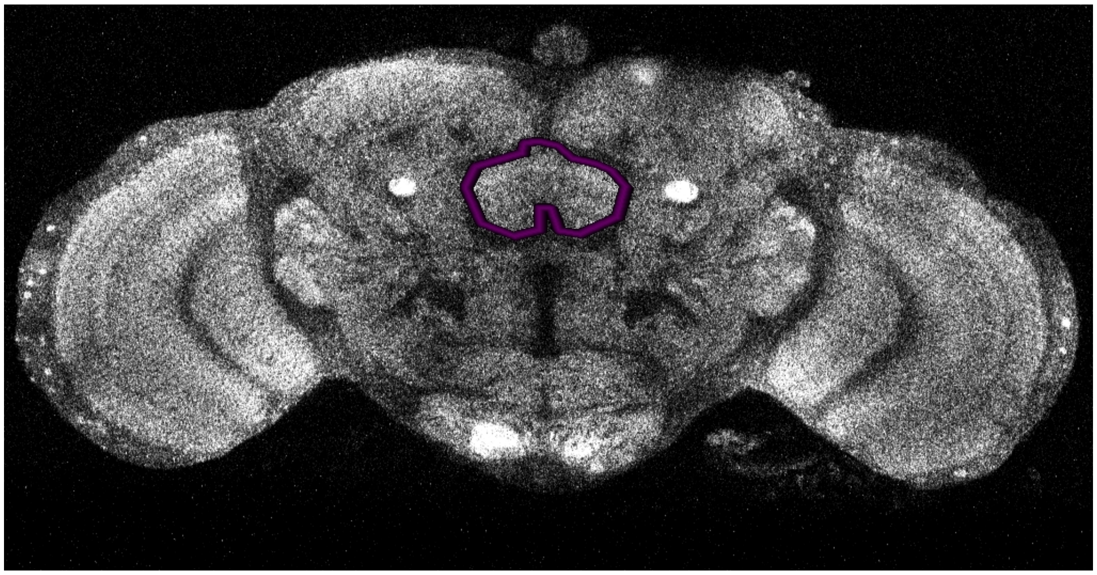



Figure 1.4: Top: a slice through the data set shown in fig. 1.1. Bottom: The automatic segmentation of the fanshaped body. Note that the shape (which is not smooth on purpose to demonstrate that the algorithm can still locate the shape) was obtained manually from a different data set, and not subject to adjustment.

The method as of time of writing only handles rigid 2d shapes (see section 5.2.2). However, the mathematical principles can be applied to deformable 3d shapes as well.

1.1.3 Approach

Using the existing visualization suite Amira (see section 2.3), the VIB protocol has been written in the computer language Tcl. Several methods known from software engineering, such as dependency checking and logging, have been integrated into a library of functions. The tasks of the protocol have been implemented using these functions. Following a few simple rules, the functions can be used to implement new tasks, which integrate seamlessly into the user interface. The VIB protocol is described in detail in section 3.2.

The traditional algorithms to segment images have been implemented in Java as plugins to ImageJ (see [7]), except for the sparse coding algorithm, which includes code written in C kindly provided by D. Endres. Using the scripting facilities of ImageJ and the portability of Java, the algorithms were executed in parallel (on all available computers) with different parameters, so that an evaluation of the algorithms with near optimal[3] parameter settings became feasible, even if the execution sometimes took substantial amounts of time. From the generated images, those best demonstrating the specific features of the algorithm are shown in chapter 4.

In section 5.2.2, a novel algorithm based on Information Theory is presented which can locate a shape in a given slice. It performs well even on recordings with a high noise level, such as recordings obtained using a confocal laser scanning microscope (see section 2.1). The result is shown in fig. 1.4. This algorithm detects 2d shapes, which have to be provided beforehand. In contrast to traditional approaches to object detection, it does not rely on homogeneous gray values of the object to be detected, but rather detects changes of texture at the boundaries of the object. Therefore, it handles recordings from a confocal laser scanning microscope much better than traditional object detection methods.

In an ongoing effort, this algorithm is enhanced to handle 3d shapes, and to refine the segmentation by adjusting the shape of the object according to the recorded data. To reduce distortions, this adjustment is constrained using ideas from the Active Contours algorithm (see section 4.4.3). Estimating from the results of the 2d algorithm, the 3d algorithm can be expected to segment a brain in less than two hours on a conventional personal computer.

1.2 An apology to the reader

This work spans several fields of research: Motivated by genetic studies of *Drosophila melanogaster*, the presented approaches combine ideas of neurobiology, mathematics, information theory and the art of programming. Since it is illusory to assume that each reader is deeply familiar with all of these fields, I decided to explain the methods and ideas behind them in a manner such that the inclined reader need not understand everything in detail, or even the terms, yet can follow my reasoning as to why I preferred some approaches over others.

To make things complicated, each scientific field has invented its own language. Not only is it sometimes difficult to find the right translation, merely the terms' interpretations depend heavily on the context, often leading to confusion between researchers of different areas of expertise. A mathematician, for example, will understand that a tuple of numbers is meant by the term "vector", while the computer

[3]Optimality was determined by visual expection.

scientist believes it stands for a resizable array of items. The geneticist will readily comprehend that the subject is about a mechanism to transmit genes. Therefore, I tried to avoid such terms.

Since it is not always possible to express ideas without using special terminology, this thesis is organized as follows: chapter 3 describes the biological aspects of the standard brain of *Drosophila melanogaster*, chapter 4 discusses its aspects from the angle of computer science, and chapter 5 together with the appendices illustrates an application of information theory.

1.3 Dedications

I dedicate this work to several people. Not so much because I could not think of any single person who I want to thank sincerely, but rather because there are so many. So, here come my dedications in no particular order, and with my sincere apologies to those I forgot.

I want to dedicate this work to Prof. Martin Heisenberg, who gave me the opportunity, and ample leeway, to follow up on my ideas to tackle the subject of this thesis; for introducing me into a part of science I did not know or appreciate before; for showing me that – even in an adverse environment – it is still possible to retain one's own character, not giving in to political games.

To Prof. Frank Puppe, who agreed without hesitation to support my thesis as second mentor, when it became clear that the work relies as much on computer science as on biology.

To Dr. Dominik Endres, who introduced me to the world of Information Theory, patiently explaining and re-explaining time and time again the basic and advanced concepts thereof, seemingly never tiring of my questions.

To Arnim Jenett, who was the best colleague to spend night and day with, who taught me by explaining and asking, and who can do miracles with pasta and tomatoes.

To Conny Grübel, who was always ready to laugh, and always amazed me with the quality of her preparations.

To Jennifer Benson, a true friend and "admirer", who made me laugh and feel at home.

To Marian Endres, who was there when I needed a friend, who is always good for an argument, and never backs down, still being the best boss I ever had.

To my parents, for providing me with the highest good there is: an undestroyable belief in the goodness of humankind, an undying optimism, and being able to laugh about oneself, which are really only different views of the same quality.

To my sisters and my brother, who are the best siblings there are.

To all acquaintances, friends and relatives, who I did not mention yet, who had to put up with me, my humor, and my escapades from time to time, who battled me on a Badminton court, the 8×8 board populated by pawns& friends, or in other circumstances, who partied, talked, sang and enjoyed silence with me.

And finally, but certainly most sincerely, to Anja Stotz, who never fails to amaze me with her *ésprit*, her ability to fix my car or hair, and who is always dragging me back into reality whenever work threatens to swallow me.

Chapter 2

Methods

Traditionally, theses in the context of biology begin by presenting the materials and methods which were used. However, since some special methods were subject to my studies, these are not introduced in this chapter, but instead where they are evaluated.

This chapter is organized in the same way as the whole thesis: first, methods from biology are discussed, then the mathematical concepts needed for chapter 4. After that, the basics of information theory together with an introduction to Bayesian Inference are presented as needed in chapter 5. Since Bayesian Inference is an important, but still too rarely applied, mathematical framework with applications in biology as well as computer science, it is explained in detail.

2.1 The principle of confocal laser microscopy

A setup for confocal laser microscopy is displayed in fig. 2.1. By moving the focus of the laser and measuring the signals at certain intervals, the specimen is optically sliced. The result is a 3D image.

When fluorescence is measured, the error distribution of the signal being recorded by the light detector (usually a photo multiplying receptor) is a normal distribution. Furthermore, when recording 3D images, theoretically one should obtain one value for each coordinate, where in reality the value is measured only at intersections of a regular grid (and the recorded values are called *pixels*).

The physical laws give rise to another problem: The confocal principle (see fig. 2.1) means that not only the light detector introduces errors, but also the refraction and absorption by the recorded specimen itself. The light emitted by the laser has to pass through optically dense matter, there excites fluorescence of a different wavelength, which finds its way back through the same matter. Finally, the few photons which pass through the pinhole are measured by a photon multiplying detector.

There are means to account for those errors (see e.g. [8]). Since known error sources are addressed specifically, this process is also called *restoration*. An important class of restorations assumes that the original distortion can be modeled as a convolution. These restorations are also known as *deconvolutions*. A good review about techniques and error estimates is given in [9].

2.2 The Gal4/UAS system

The Gal4 method presented in [10] can be used to visualize expression patterns of single genes. It works by inserting a specific transposon into an arbitrary genetic

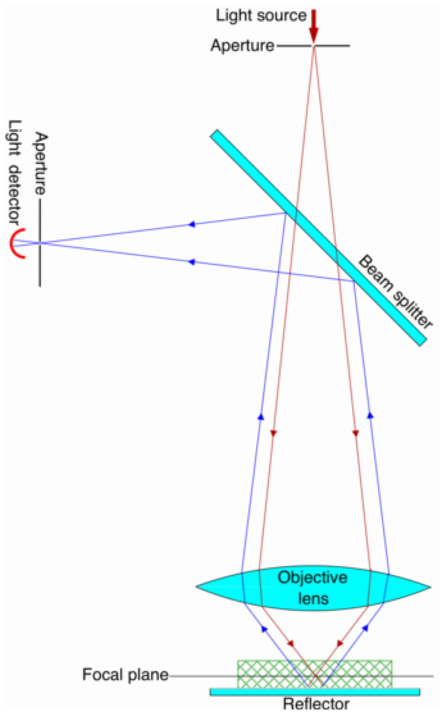
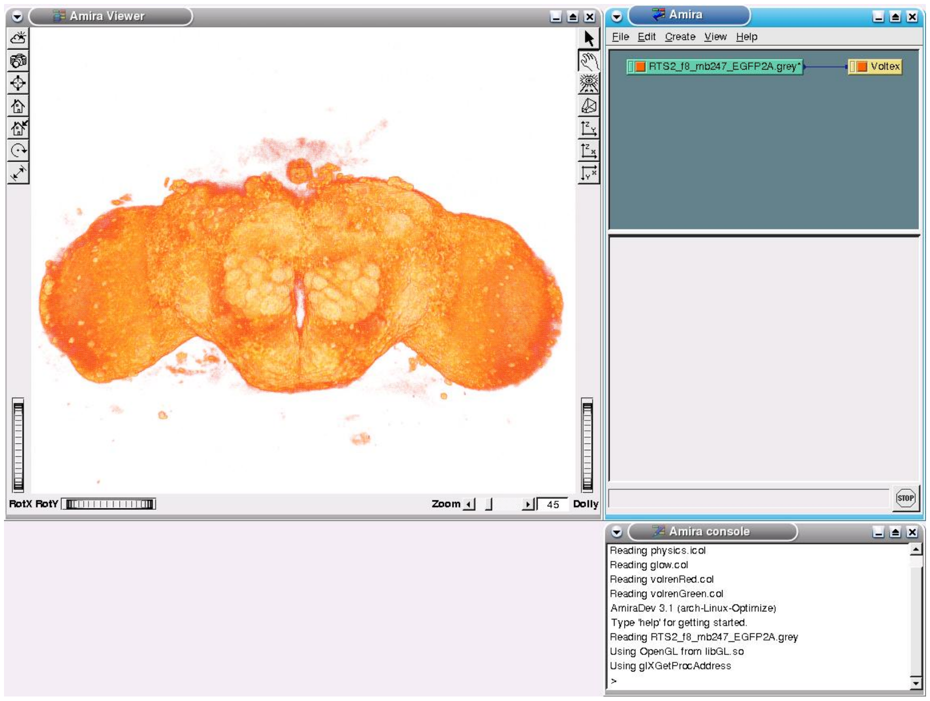


Figure 2.1: The principle of confocal laser scanning microscopy: a light source (laser) emits photons which are focused on one point in space. In our case, this elicits a fluorescent response, which is reflected by a special mirror, the beam splitter, and the resulting signal is recorded using a photo multiplier. The beam is focused using a system of lenses and pinholes with very small apertures. By moving the specimen, the focal point traverses the whole specimen, resulting in a 3D image. *This picture is reproduced from WikiPedia, the free encyclopedia, under the GFD license.*



```
Reading physics.icol
Reading glow.col
Reading volrenRed.col
Reading volrenGreen.col
AmiraDev 3.1 (arch-Linux-Optimize)
Type 'help' for getting started.
Reading RTS2_f8_mb247_EGFP2A.grey
Using OpenGL from libGL.so
Using glXGetProcAddress
>
```

Figure 2.2: The three windows of Amira. Left: visualization window displaying a volume rendering of a *Drosophila* brain. Upper right: the work area, Lower right: the console window

location. This transposon codes for the Gal4 protein. Gal4 is a transcription factor from yeast, which activates transcription by binding to a specific regulatory sequence of a gene, the *Upstream Activating Sequence* ("UAS"). Since wild type *Drosophila melanogaster*, i.e. a fly stock whose DNA was not modified by human intervention, does not contain UAS, the Gal4 has no effect on the living organism. A stock of flies whose genome contains this Gal4 sequence is called a *Gal4 line*. A fly stock is called *UAS reporter line*, when its DNA contains a genomic sequence for an easily detectable protein (such as the *Green Fluorescent Protein*, short GFP) controlled by UAS. Since the reporter line does not include a gene coding for Gal4, no reporter protein is synthesized. Only when a Gal4 line is crossed to a reporter line, the reporter protein is produced at all. It is only synthesized where and when the Gal4 protein is transcribed.

The UAS/Gal4 system may appear overly complicated. However, the separation between the driver strain and the effector strain allows for a greater flexibility: By crossing a driver line to different effector lines, the cells expressing Gal4 can be studied from different angles. For detailed explanations of the method, see [10, 11, 12].

2.3 The visualization suite Amira

Originally designed for medical data, Amira is a versatile tool for investigating 3D data. Amira provides tools to rotate and cut huge data sets easily and to visualize them using different methods, among other manipulations.

As a tool directed at versatility, Amira has a very technical user interface. It consists of three windows: a visualization window, a work area and a console window (see fig. 2.2).

The visualization window shows a 2D projection of the currently loaded data sets according to the chosen transformation and visualization mode. The mode is chosen by creating a visualization module and connecting it to the data set. Examples of such visualization modules are

- *OrthoSlice*, which displays a single slice of the 3D stack,
- *ProjectionView*, which displays cumulative projections parallel to the three axes, and
- *Voltex*, which displays a volume rendering: the intensities are interpreted as gray values as well as optical densities (this mode is shown in fig. 2.2).

Present in the visualization window are a few buttons to modify the display behavior such as background color or gamma value.

The work area contains the menu, a canvas in which the currently loaded data sets and other modules are displayed. The data sets are handled as a special case of a module, and they can be connected to computation or visualization modules, thus serving as input to the latter. At the bottom of the work area is a canvas displaying the options of the currently selected modules, if any.

The console window is used to display messages concerning only advanced usage, and is thus, for the most part, uninteresting to the user. There is one exception, however: the console window not only shows messages, but a command line prompt. By using the script language (see below), commands can be executed in that window.

A very important feature of Amira is its scripting feature: An embedded script language, Tcl (see [13]), allows a user to write small programs, so-called scripts, to facilitate repetitive tasks. For example, if one wants to enhance a set of images equally, such as cropping each image to the same dimensions, then a simple script containing a loop over the list of images can automate the task. It is also used in some modules to allow for complex user input in the form of callback functions.

2.4 Uniform coordinates

Linear algebra provides the mathematical framework to describe linear transformations. These can be represented by a quadratic matrix. If each column vector has the length 1 and the column vectors are pairwise orthogonal, the corresponding transformation is called *rigid*, since it is isogonal and isoscalar.

Linear transformations always map the origin onto itself. Therefore, translations are not linear transformations. Still, linear algebra can be used to express translations, by introducing *uniform coordinates*. A coordinate is transformed into a uniform coordinate by adding another dimension, whose value is always 1, i.e.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

A linear transformation, followed by a translation, now can be expressed by a 3×4 matrix by assigning the translation vector to the fourth column: For

$$\vec{v} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, L = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \vec{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix},$$

the combined transformation is

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & t_1 \\ a_{21} & a_{22} & a_{23} & t_2 \\ a_{31} & a_{32} & a_{33} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = L\vec{x} + \vec{t}$$

2.5 Some concepts of Information Theory

Information Theory is the mathematical field, which is concerned with measurements of information. It is mostly agnostic as to what type of information is being related to (e.g. size or gender), but instead measures the variability of the data. Another way to look at information theory is to ask the question: given certain *a priori* knowledge about a certain subject: How many Yes/No questions have to be answered before the subject is fully described? By counting the average number of questions necessary to achieve that goal, the information can be quantified.

Arguably, the most important quantity in information theory is called the Mutual Information between two random variables. It is a quantification of the term "statistical dependence": if two quantities are statistically independent, the Mutual Information between them is zero, otherwise it is positive. As an example, let's look at a neuro-anatomist who is segmenting a fly brain. Given the knowledge about the brain anatomy, if she already knows where the optical lobes and mushroom bodies are, then she will find the antennal lobes quickly: the Mutual Information between these locations is greater than zero.

The formula to calculate the Mutual Information between the quantities X and Y is

$$I(X;Y) = H(X;Y) - H(X) - H(Y) \tag{2.1}$$

where $X;Y$ means the quantity obtained by pairing X and Y, and H denotes the *entropy*[1]

$$H(X) = -\sum_x P(X=x)\log_2 P(X=x) \tag{2.2}$$

where the sum is carried out over all possible values x of the quantity X.

The Mutual Information and the entropy are measured in *bit*[2], just like the unit used to measure amounts of data in computer files. Indeed, when a text is optimally coded in a computer file using a (prior) letter distribution, the average amount of bits per letter is the entropy of the letter distribution. In fig. 2.3, an optimal code for a simple example is illustrated.

To understand what this could mean in the context of neuroanatomy, let's use the example from before. When encoding the center coordinates[3] of the antennal lobe, and the center coordinates of the optical lobes and mushroom bodies are known, certainly one needs fewer bits than if the other coordinates are not known:

[1]Since only the distribution is needed to calculate the entropy, the entropy can be interpreted as a function which takes distributions as its arguments. However, it is common practice to refer to an "entropy of a quantity".

[2]Some scientists prefer the natural logarithm over the dual logarithm. In this case, the unit is called *nats*.

[3]In this context, it is not important which definition of center coordinates is used. However, in this thesis, the center coordinates of a neuropil always denote the coordinates of the center of gravity when all voxels are treated as equally heavy, i.e. the average coordinate.

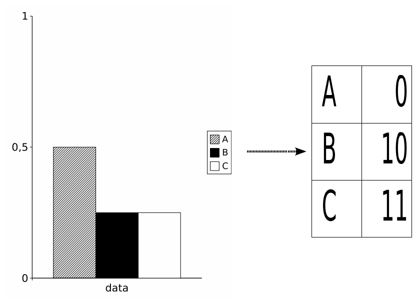


Figure 2.3: Given a probability distribution, the items **A**, **B** and **C** are assigned binary codes, such that the length of a coded message is minimal (for example, **ACBA** is encoded as **011100**): the average length per coded letter is $0.5 \cdot 1 + 0.25 \cdot 2 + 0.25 \cdot 2 = 1.5$ bits, the entropy (see 2.2) of the probability distribution.

There are not only fewer possible locations of the antennal lobe, also some of these locations are much more likely than others. The exact difference – provided that the coding is optimal in information-theoretical terms – is the Mutual Information between the center of the antennal lobe on one hand, and the other centers on the other.

The lowest possible Mutual Information between two quantities is 0. In this case, nothing can be said about one of the two quantities when only knowing the value of the other, i.e. the quantities are statistically independent. The Mutual Information is maximal when one of the quantities is a function of the other. This happens, for example, when calculating the Mutual Information between one quantity and itself, and the result is the entropy (see eq. 2.2). When a joint probability distribution $P(X;Y)$ of two quantities X, Y is given, the probabilities $P(X) = \sum_y P(X;y)$ and $P(Y) = \sum_x P(x;Y)$ can be calculated. These distributions are called *marginal distributions*, and their entropies accordingly *marginal entropies*.

The entropy of the gray value distribution is illustrated in fig. 2.4. In this figure, at each coordinate the entropy in a neighborhood is displayed using a color code. As can be seen, highly variable textures are presented by high entropies.

Sometimes one reads the term *Normalized Mutual Information*: The Mutual Information $I(X;Y)$ can not be greater than the minimum of the marginal entropies $\min\{H(X), H(Y)\}$. Therefore, a (weak) upper bound of the Mutual Information is given by $\frac{H(X)+H(Y)}{2}$. The Normalized Mutual Information is nothing else than the Mutual Information divided by this upper bound.

2.6 A short introduction into Bayesian Inference

A common problem addressed by probability theory is classification. The question one aspires to answer is: How likely do these data come from a particular class[4]? In other words, given the data, calculate the conditional probability that the observed data are from a certain class out of a (usually small) set of classes.

In most cases, it is difficult or not possible at all to calculate this probability directly. Instead, the opposite direction is easier: to estimate or calculate the probability of the data given the class membership. The Bayes' theorem (see [14]) permits the reversal of this direction by a simple division:

[4]Strictly speaking: How likely were the data generated by a process which is described by the class-conditional probability distribution.

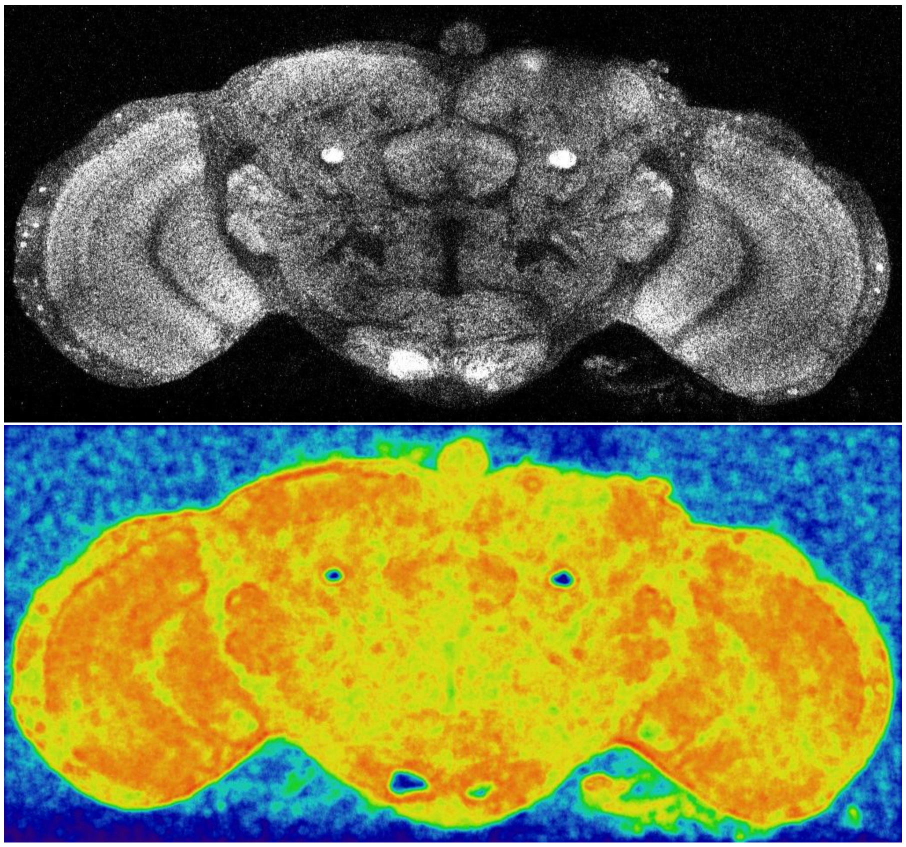


Figure 2.4: Around each pixel of the original image (top), a gray value histogram of a circular neighborhood is created, and its entropy visualized (bottom; blue denotes low entropy and red high entropy).

$$P(C|D) \quad = \quad \frac{P(D|C)P(C)}{P(D)}, \tag{2.3}$$

where D is the tuple of the observed data, C the class membership, and $P(D|C)$ means the conditional probability of the observed data given the class membership[5].

As for the probability of the observed data, note that $P(D)$ can be calculated by a sum:

$$P(D) \quad = \quad \sum_{C_i} P(D|C_i)P(C_i), \tag{2.4}$$

where the sum runs over all[6] possible classes C_i ($P(D)$ is a marginal distribution, see section 2.5). This means that one really only needs the class probabilities and

[5]In many works about Bayesian inference, it is argued that there is no such thing as an unconditional probability. For example, the recording of a sample always depends on the setup, the equipment being used, and of course the observer. Thus, in a strict mathematical sense, I would have to write $P(D|E)$, where E means "everything else". For the sake of clarity, I refrain from that tradition, and humbly request that the reader keep that fact in mind.

[6]For classification tasks, it is assumed that no two classes overlap, and that the union of all classes makes up the total probability space.

the conditional probabilities of the observed data to infer the desired value, the namely conditional class probability.

Let's look at an example application of Bayes' theorem. To make it simpler, I do not yet use quantities like "data" or "class membership". Assume that a medical test for a certain malady has a reliability of 99%, i.e. the result of the test (be that "ill" or "well") is correct in 99 out of 100 cases. Assume further, that 0.1% of the population, which is to be tested, actually is infected by this malady. The somewhat surprising result of Bayesian inference tells us that a person, who was labeled "ill" by that test (denoted by *positive*), is likely in the best of health:

$$\begin{aligned} P(ill|positive) &= \frac{P(positive|ill)P(ill)}{P(positive)} \\ &= \frac{P(positive|ill)P(ill)}{P(positive|ill)P(ill) + P(positive|well)P(well)} \\ &= \frac{0.99 \cdot 0.001}{0.99 \cdot 0.001 + 0.01 \cdot 0.999} \\ &\approx 0.09 \end{aligned}$$

In other words, if that test (which has a reliability of 99%!) says that a person is ill, the probability is actually about 91% that this person is **not** ill!

In the context of Bayesian classification, the quantities are "data" and "class membership", and the probabilities are interpreted as "degrees of belief".

Seemingly, Bayes' theorem is not just a reversal of conditional probabilities, because it needs more than just "the other direction": What are the probabilities of the class membership? In the absence of further information, one should assume that all classes are equally likely, but often more information is available about the classes[7]. In Bayesian terminology, this probability is called the *prior probability*.

The strength of Bayesian inference is its sound mathematical basis: It can be shown that trying to fulfill just five desiderata (see [15]), which match common sense very well, leads to Bayesian inference as the only valid method for plausible reasoning. Every other method fulfilling these desiderata is equivalent to Bayesian inference.

It suggests itself to extend the concept of Bayesian inference: Instead of inferring just the conditional probabilities, one can infer expectations of a scalar quantity E (such as the entropy), by weighting the values with the inferred conditional probabilities. In the following, I refer to this procedure as the "inference of E"[8]. In other words: By averaging $E(C)$ weighted by $P(C|D)$, one obtains the inferred expectation of E given the evidence D.

As an example, let's assume we have a coin which is being tossed, and we do not know the probability assigned to the outcomes "head" and "tail". Suppose that all probabilities between 0 and 1 are a priori equally likely for the event "head", i.e. we have no bias before observing the first coin toss[9]. Let's call this probability q. Since the probability for "tail" is totally defined by q, namely $1 - q$, the true distribution[10] can be parameterized by q.

[7]In the context of segmentation (see section 3.1.2), the classes are the neuropiles, and this information is provided by the standard brain: from the set of brains used to create the standard brain, one can easily estimate these probabilities at each coordinate.

[8]Strictly speaking, it is still the "inference of the expectation of the quantity E". However, since one can not infer E (but only a probability distribution or density thereof), the expression is unambiguous. Think of it as a kind of short hand.

[9]Note that I assign a probability density to a probability. This breaks tradition with the classical probability theory, where probabilities or probability densities are only assigned to events (outcomes of experiments). In the context of Bayesian inference, it is common to attach probabilities to statements or values, which are more properly interpreted as degrees of belief, or plausibility.

[10]The term "true distribution" is debatable. As mentioned earlier, in the context of Bayesian

Now, let's infer the parameter q: $E(q) = q$. To apply Bayesian inference, we need some evidence, so let's toss the coin N times and record the outcomes, writing k for the number of heads. The conditional probability $P(D|q)$ is calculated by $\binom{N}{k}q^k(1-q)^{N-k}$.

Since the parameter q is continuous, and the probability density for q is constant, we have to calculate

$$\begin{aligned} < E(q) > &= \int_0^1 E(q)p(q|D)dq \\ &= \int_0^1 E(q)\frac{P(D|q)p(q)}{\int_0^1 P(D|q)p(q)dq}dq \\ &= \int_0^1 q\frac{\binom{N}{k}q^k(1-q)^{N-k}}{\int_0^1 \binom{N}{k}q^k(1-q)^{N-k}dq}dq \end{aligned} \tag{2.5}$$

Note that by partial integration, for $N-k>0$ it follows that

$$\begin{aligned} \int_0^1 \binom{N}{k}q^k(1-q)^{N-k}dq &= \left[\binom{N}{k}\frac{q^{k+1}}{k+1}(1-q)^{N-k}\right]_0^1 - \\ &\quad -\int_0^1 \binom{N}{k}\frac{q^{k+1}}{k+1}(-1)(N-k)(1-q)^{N-k-1}dq \\ &= 0+\int_0^1 \binom{N}{k+1}q^{k+1}(1-q)^{N-(k+1)}dq \\ &= \dots \\ &= \int_0^1 \binom{N}{N}q^N dq \\ &= \frac{1}{N+1} \end{aligned} \tag{2.6}$$

Continuing 2.5, one obtains

$$\begin{aligned} < E(q) > &= \int_0^1 q\frac{\binom{N}{k}q^k(1-q)^{N-k}}{\frac{1}{N+1}}dq \\ &= (N+1)\int_0^1 \binom{N}{k}q^{k+1}(1-q)^{N-k}dq \\ &= (N+1)\frac{\binom{N}{k}}{\binom{N+1}{k+1}}\int_0^1 \binom{N+1}{k+1}q^{k+1}(1-q)^{N+1-(k+1)}dq \end{aligned}$$

which, once again using 2.6, becomes:

$$< E(q) > = (N+1)\frac{\binom{N}{k}}{\binom{N+1}{k+1}}\frac{1}{N+2}$$

inference it is popular to think of all probabilities as conditional. The more information is available, the better the outcome can be estimated. For example, the outcome of a coin toss can be calculated exactly when knowing enough physical parameters like the applied force, the weight of the coin, the direction, etc. However hard I tried, I could not come up with a better term than "true distribution". In the following, this term means the theoretical distribution when repeating the same experiment over and over again, and assuming that the same prior information about the outcome is available at each trial. In the case of a gray value distribution from confocal microscopy: The "true distribution" of the gray value is the limit, which would be reached if we were to record the gray value at the same coordinate of the same specimen infinitely often.

$$\begin{aligned} &= \frac{(N+1)N!(k+1)!(N-k)!}{k!(N-k)!(N+1)!(N+2)} \\ &= \frac{k+1}{N+2} \end{aligned}$$

In the limit $N \to \infty$ (where $k \to qN$), this result coincides with the naïve estimate $E(q) \approx \frac{k}{N}$. However, the inferred q tells us that we should not trust extreme values of k, i.e. where k is near zero or N: If $k < \frac{1}{2}N$, then $\frac{k+1}{N+2} > \frac{k}{N}$, and vice versa, meaning that the estimate $q \approx \frac{k}{N}$ should always be corrected towards $\frac{1}{2}$. In other words, one should always be cautious when things look extreme!

Chapter 3

The Virtual Insect Brain protocol

The generation of a standard brain typically involves these steps: A set of brains is recorded, a template (usually the most typical of the brains) is picked and the other brains are fit[1] on the template, and then an average is calculated. The result – called *average brain* – is the analogy to a map in geography.

To populate the standard brain, features – such as anatomic structures – are marked in the standard brain. This is normally accomplished by marking these structures in individual brains, and then using the same method as before to fit the individual brains on the standard brain, and thereby the marks.

The averaging of several brains is necessary to factor out deviations of single brains. If just one brain were chosen, it would be uncertain if a certain feature is characteristic of that species in general, or just a peculiarity of the chosen brain. This averaging process is therefore not only useful for the generation of a standard brain, but for neuroanatomical studies in general. In that line of thought, the standard brain can be interpreted as a special case of an average brain.

In this chapter, the ideas used to generate the human standard brain and the standard brain of *Drosophila melanogaster* are presented, setting the historical context for the Virtual Insect Brain protocol, which is subsequently described in detail. Example applications of this protocol and the motivation for the next chapters conclude the chapter.

3.1 The standard brain of *Drosophila melanogaster*

Before generating a standard brain, it is sensible to study earlier endeavors towards that end. One of the first applications of computers to the task of creating a standard brain was presented by Dr. Jean Talairach and Dr. Pierre Tournoux in [16]. They introduced a coordinate system to identify locations in the human brain. By transforming this coordinate system to fit a newly recorded brain, a location can be referenced relative to the standard brain.

Marking distinctive features (commissures and extrema of the cortex), which can be found in most human brains, the transformation of the coordinate system can easily be computed. Since [16], several promising computational methods have come to existence which ease the burden of marking up the feature points manually (see [17]).

[1]As mentioned in chapter 1, such a fit is necessarily imprecise, since no two brains match perfectly.

For studies on *Drosophila melanogaster*, a similar general strategy was applied by [6, 18]. Like in [16], the average of a set of brains has been chosen as the standard brain. Since the *Drosophila* brain's neuropiles and the human cortex differ significantly in shape, the outlines of the neuropiles were used as features instead of certain point landmarks.

Since the optical resolution of MRI (the method applied in [16] to record the brains) is too coarse to visualize the small structures visible in insect brain recordings, such structures are averaged out in MR images. This averaging process leads to more or less three classes of intensities. Background, gray and white matter are easily recognizable, and algorithms using only the intensity to classify the piyels are quite successful. However, it proved impossible to use the same algorithms to mark the features in the brain of *Drosophila melanogaster*.

3.1.1 Registration

The method to fit two brains is to obtain a mapping between them. In mathematical terms, such a mapping is a transformation which maps a coordinate from one brain to the corresponding coordinate in the other brain.The process of finding such a mapping is called *registration*, a term made popular by Ashburner and Friston (see [19]).

The registration methods come in two flavors: rigid and non-rigid. Rigid registrations allow only rigid transformations, i.e. the transformation preserves angles and size. These transformations can be represented by a rotation followed by a translation.

Since any two brains are different, rigid registrations can not optimally map each landmark of one brain to the same landmark of the other brain. Therefore, non-rigid registrations were introduced (see e.g. sections 3.1.2 and 4.4.1). Since the generated mappings are not linear, they are often also referred to as *warpings*. It should be noted that even non-rigid registrations can not cancel out all differences between two brains.

Unintended distortions are a big problem with non-rigid registration methods: without a proper mathematical model of what constitutes a valid mapping, it seems that partially rigid registrations (as used in [6], see section 3.1.2) are superior to any other method. This will be seen when applying elastic transformations (see section 4.4.1), which – lacking proper constraints – warp every location equally well – or equally inadequately. However, if some coordinate which is outside of the brain is wildly displaced, it does not matter as much as if a very fine structure is misplaced by the same margin. Furthermore, comparisons between warped and unwarped neuropiles bear a questionable relevance.

The registration is arguably the most important part of the generation of an average brain (see also [20]).

3.1.2 The standard brain of *Drosophila melanogaster*

In [6], the standard brain of *Drosophila melanogaster* was generated from 28 brains. Sixteen neuropiles were labeled in each data set by hand. Manual labeling works like this: in every slice, the outlines of the neuropiles are marked, and the computer labels the enclosed coordinates accordingly. Since the outlines separate the neuropiles from the background, this marking job is called *segmentation*. Throughout this thesis, the terms *labeling* and *segmentation* are used as if they denoted the same task, because it is computationally easy to transform the result from one to the result of the other.

Obviously, segmentation is a very tedious job. Without losing much precision, every second slice can be interpolated by the enclosing slices, but still the time

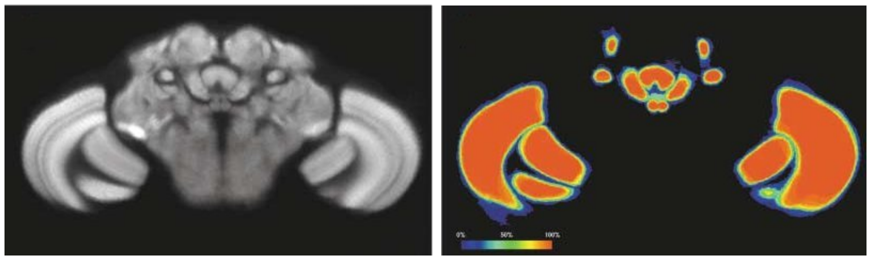


Figure 3.1: The standard brain of *Drosophila melanogaster* from [6]. Left: After mapping each brain of a set of 28, the average intensity is shown. Right: Using the same mapping, the probability is shown that on that particular coordinate, a neuropil is present (red denotes 100%, black 0%).

needed for only one brain easily surpasses a few hours.

After labeling the neuropiles, a template was picked, and the other brains were registered to that template. Then, the average intensity was calculated. To evaluate the quality of the registration, a so-called probability map was calculated from the labelings by counting the number of the brains agreeing on the label at each coordinate. These results are displayed in fig. 3.1.

The registration method chosen in [6] uses the labelings of the neuropiles. Each neuropil was separately aligned rigidly. Between the neuropiles, the mapping was obtained by solving a modified heat diffusion equation iteratively[2]. In this manner, it is assured that the neuropiles are mapped rigidly, thus avoiding distortions which do not make anatomical sense (such as eliminating unstained areas when they are anatomically relevant).

The same approach was also chosen by [21], where an atlas of the honey-bee's antennal lobe was presented. This lobe has distinctive bulge-like structures, referred to as *glomeruli*, playing a role when the bee classifies odors. These glomeruli were labeled according to their location, and the labels were then overlaid on an image of an antennal lobe. In this context, one does not need to specify their boundaries explicitly, because the edges between the glomeruli are clearly visible. This is usually not the case when treating whole insect brains.

A slightly different approach to generate a standard brain was chosen by [22] to create a standard brain of the honeybee: First, the transformation was calculated using the gray values instead of marked anatomical structures, and second, the standard brain was not chosen out of the 20 recorded brains. Instead, a standard brain was computed by iteratively adjusting each brain of the set to more closely match the average of these brains, and after two iterations over the complete set of 20 brains, they were averaged[3]. This approach has the advantage that it is less biased towards one particular brain (the template), however, as will be illustrated in section 4.4.1, the transformation not necessarily keeps anatomical structures intact. The latter problem was overcome by carefully adjusting the transformation algorithm until the resulting standard brain no longer showed obviously unnatural structures. As with the methods for the human brain, the difference of scale[4] made it difficult to use the same technique on *Drosophila melanogaster* brains.

[2]The gray values are treated as denoting temperature at two different points in time, and the force to establish the heat flow is minimized.

[3]While the original plan was to iterate until all transformed brains were identical, in practice it proved to wash out too much detail.

[4]The bee brains were scanned using confocal laser microscopy at a resolution of $3.8\mu\text{m}\times 3.8\mu\text{m}\times 8\mu\text{m}$.

3.1.3 The original set of Amira scripts

The tasks for creating the standard brain in [6] were implemented as a set of Amira scripts that can be downloaded from *http://www.amiravis.com/vib/*. These scripts are meant to be called in a certain order:

1. down-sample the data sets
2. calculate a few statistics from the segmentations
3. calculate the mappings
4. calculate the average brain
5. calculate the average labelings

These scripts expose a user interface in Amira via the module interface, i.e. they are visualized in the work area like the other modules. In order to have a user interface, each script has to follow some relatively weak guide lines, such as a special header, and an interface emulating object oriented methods. Since a script following these rules is presented as a module in Amira, it is also referred to as *script module*[5].

To configure these scripts, i.e. specify the data sets to work on, one has to edit a text file and then run the scripts in the correct order. The scripts are loaded just like images; Amira recognizes them as scripts and executes them.

These scripts expect a set of 3D stacks containing one channel of 8-bit intensities, and a corresponding set of labelings. Both gray values and the labelings have to be in Amira's own file format. It is very important that the neuropil list be identical for all labelings, because these scripts do not check if they are (if the neuropil lists are not identical these scripts will not hesitate to generate a wrong mapping).

The many restrictions, together with a general lack of error handling routines and documentation reduce the usability of these scripts for other laboratories. In all, the scripts were built for the purpose of creating a standard brain, not for working with it. For instance, when studying gene expression patterns, the data sets need to hold an additional channel, which is not supported by the scripts.

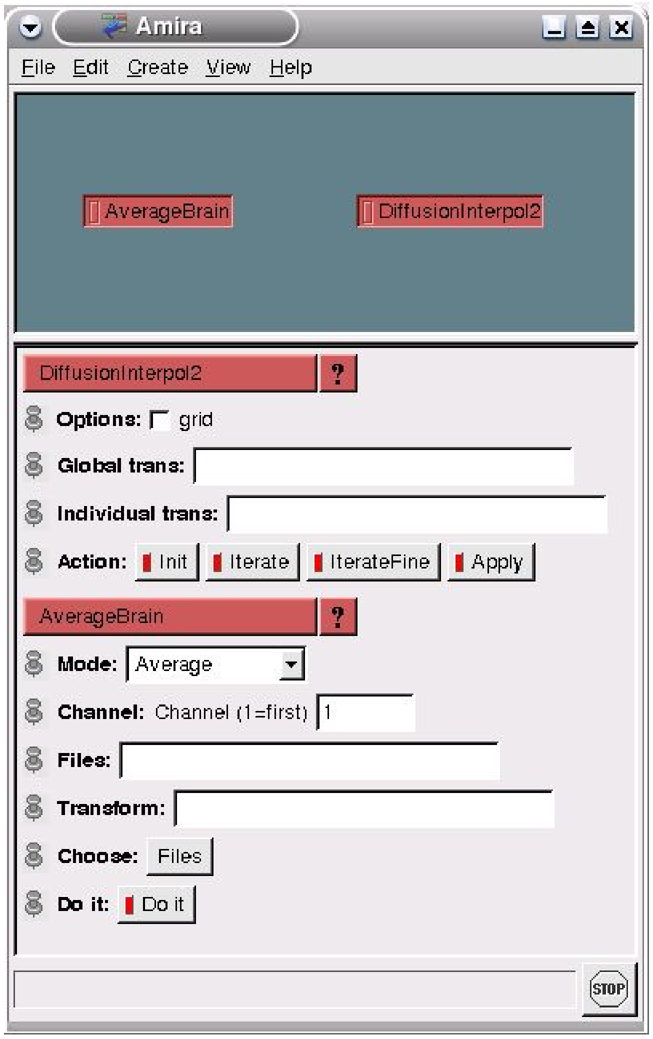
In the course of a joint project[6], the developers of Amira, Indeed 3D, designed two custom extensions implementing mapping functions (see fig. 3.2):

- *AverageBrain* is a module to calculate the mean intensities of a set of brains. The transformations to map those brains onto the standard brain can be specified in the options of the module. AverageBrain only works with linear transformations (including isotropic changes of size) and translation. These transformations are specified as either a uniform matrix (if the transformation is the same for each data set), or a Tcl procedure which returns a uniform matrix for each data set.

  This module is also able to compute a probability map of a set of labelings: When labeled neuropiles are mapped onto a standard, there is no sensible way to calculate an average. Instead, at each coordinate, the agreement of the brain labels is shown, i.e. if all data sets agree on one label at a certain coordinate, the value of the probability map at that coordinate is 100%. This is used to calculate a goodness of fit, which – as mentioned earlier – is an important part of a standard brain.

[5]Amira shows the different module types in different colors: Data modules are displayed in green, compute modules in red, and script modules in blue.

[6]"Virtual Brain supported by Bundesministerium für Bildung und Forschung



Mode: Average

Channel: Channel (1=first) 1

Files:

Transform:

Figure 3.2: The two custom Amira modules created for the VIB protocol. The respective options are shown in the lower part of the work area.

- *DiffusionInterpol2*. This module computes the non-rigid mapping explained in section 3.1.2. The input are two brain recordings, their respective neuropil labelings and transformations for each single neuropil. The input form can be seen in fig. 3.2.

Amira provides a facility ("Save Network") to save a Tcl script, which reinstates the current work area when executed, i.e. all data and script modules as seen in the work area are loaded, the other modules are created and the connections between them established. This feature is important, because Amira's memory handling often leads to memory fragmentation and program terminations.

3.2 The Virtual Insect Brain protocol

As part of this thesis, I revised the scripts of [6] and made them robust, expandable and usable for a wider range of applications. It turned out that these scripts could not be operated without a profound knowledge of Amira and its scripting language.

Moreover, without proper documentation it proved difficult to overcome certain inflexibilities of the original framework.

Since the resulting set of scripts can be used to create a standard brain, to map several newly recorded brains onto the existing standard brain, or to generate an average brain, it was given the name *Virtual Insect Brain* protocol[7] (or, in short, *VIB* protocol). It is capable of handling multiple channels and working on different platforms, and can be downloaded from *http://www.neurofly.de*. Examples of the application of the VIB protocol will be presented in 3.3.3.

In the following, I describe my enhancements.

3.2.1 Error handling

Lacking error handling, the original scripts could return a result which did not make sense, like when half of the data sets were transformed onto another standard brain than the other half. Therefore, error handling and simple consistency checks were introduced, such as checking the files for zero length: due to the size of the data, it can easily happen that the hard disk is full, and in this case new files are truncated. Unfortunately, Amira's save function does not report that. Furthermore, the neuropiles are referred to by name all the time, and not by index, avoiding a common source of problems with the original scripts.

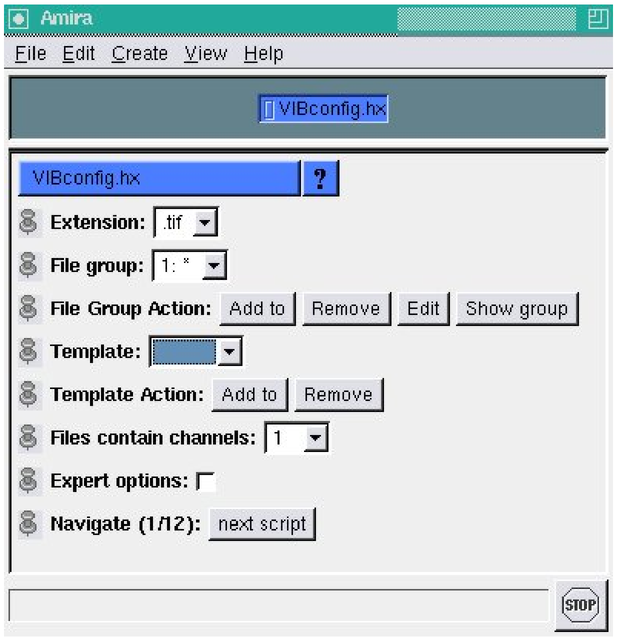
3.2.2 Automated book keeping of the segmentations

When segmenting several brains, it is unlikely one wants to label different neuropiles, or have different colors for the segmented neuropiles. The "label wizard" was introduced to automate this process: When labeling the data sets, it automatically loads the labels for the chosen template, and makes sure that the neuropil list of the new data set matches that of the template. If no template (the specimen to standardize on) was chosen yet, the first in the list is automatically selected. Of course, if that recording seems inadequate, any other can be assigned that role. When a data set is labeled, a click of a button saves that labeling and loads the next data set to be labeled. If the template's labels are modified, it is possible that a new neuropil was added to the list, and consequently this script marks all brains with older label data to be labeled again (retaining the original labeling). If at a later stage, i.e. after the label wizard, a data set has older labels than the template, a warning is issued and a button is shown which takes the user directly to the label wizard with that data set loaded.

3.2.3 Easy configuration and navigation

Also, the way to configure the scripts, namely by editing a file with a very specific syntax, was prone to errors. Therefore, a script was created to handle configuration (see fig. 3.3). Simultaneously, a navigation infrastructure was added to make it more obvious which step to take next, thus eliminating errors stemming from not yet calculated values. The navigation presents itself in the form of three buttons: *previous script*, *next script* and *config*. As seen in fig. 3.3, navigation buttons are hidden when they would not make sense, i.e. when the configuration module is loaded, there is no previous script to run. Depending on the chosen options, some scripts – like the label wizard – are not needed. The navigation buttons automatically skip these scripts.

[7]As is typical in biology, a certain list of instructions to be followed in order to reproduce an experiment is called *protocol*



Extension: .tif
File group: 1: *
File Group Action:
Template:
Template Action:
Files contain channels: 1
Expert options:
Navigate (1/12):

Figure 3.3: The configuration module. All 3D stacks have to be dropped into the *images/* directory. After the module is loaded, it shows the available file extensions, and allows to specify so called *File groups*, containing the data sets which are subsequently registered and averaged. A template (such as the standard from [6]), can be selected as register target. When all options are set, the user proceeds by clicking on *next script*.

3.2.4 Handling of multi channel data sets

As the most important application at hand was the comparison of gene expression patterns, consistent handling of multiple channels was implemented. A side product is a much improved record keeping which allows to add new files or modify existing ones, and not having to perform all calculations again, but instead only those calculations affected by the modifications. Another consequence is the ability to continue easily the calculations after a crash (which happens quite often when working with huge data sets).

3.2.5 Support for different file formats

Amira includes support to read several file formats, notably 3D TIFF, DICOM and raw formats. Which file format is used, usually depends on the context: medical 3D stacks are typically stored in DICOM format, and recordings of a microscope in TIFF. Internally, Amira uses a format called *Amira Mesh*, which is very simple, thus facilitating efficient usage, but it lacks sophisticated features like compression or multi-channel support[8]. While the original set of scripts worked only on Amira Meshes, the VIB protocol supports virtually all file formats Amira can read.

[8]In later versions, the Amira Mesh format can contain several channels, but Amira does not encourage use of this feature.

3.2.6 Alternative registration methods

Newer versions of Amira include a module, named *Registration*[9], which can be used to align two 3D stacks using their gray values. It calculates an affine transformation[10] minimizing a distance measure between the two stacks. When applying this registration to recordings of two brains, provided that the same neuropiles have the same gray levels in both data sets, according to experience, this module achieves a very good alignment. If neuropiles are labeled, the alignment of two brains can be calculated also by using the surfaces of these neuropiles. An experimental module was used in [22] to achieve a non-rigid registration, and other methods will follow. Therefore, the VIB protocol lets the user choose the registration method.

3.2.7 A tool to trim brains virtually

When a registration method is chosen which relies solely on the gray values, and thus does not need segmentation of the individual brains, it is important that the data sets are ”cleaned up”: The process of extracting the brains from the fly heads is a job which requires dexterity, and even then not all superfluous tissues can be removed from the brain without inflicting damage to it.This is not important when the brain is labeled, because an anatomist recognizes these unwanted signals and ignores them. Not so the Registration module, which regards all signals equally important and tries to match them, even if one specimen shows them and the other does not.

Therefore these signals have to be removed from the 3D stacks, when working with mappings calculated from the gray values. Amira offers a module to cut out artifacts, i.e. replace them with black voxels (”volume pixels”). While this tool is versatile, it is also complicated to operate. The VIB protocol includes a script module, called *VIBscissors*, which permits the user to easily take the common action, i.e. cut out certain regions when looking at a volume rendering of the current 3D stack. VIBscissors works transparently, i.e. an expert Amira user can still decide to take advantage of the more sophisticated features of the underlying Amira module.

3.2.8 Logging

A logging facility was built into the scripts, which helps with the book keeping of the different runs of the scripts: when unexpected things happen, the logs help to reproduce, or sometimes even understand right away, the problem. Also, after unexpected program failures it is possible that the last written file contains incomplete data. The logging facilitates finding that file in order to recompute it.

3.2.9 Locking

For each data set, the mapping to the template has to be calculated. As the data sets are independent of each other, they can be easily processed in parallel. An option in the configuration enables locking of the data sets. The processing time can therefore be reduced by using multiple computers on a shared network drive. If this option is set, the logging automatically records the host name with each message for improved book keeping.

[9]This naming is somewhat unfortunate, since the module only implements a particular registration method.

[10]An affine transformation is a linear transformation on the uniform coordinates, thus allowing rotations, scalings and translations.

3.2.10 Visualization and comparison of the calculated transformations

The result of the standard brain process sometimes shows artifacts, which almost certainly originate from a single misaligned data set. In other cases, a certain feature invites investigation of a single brain in comparison to the template. So, a script module was added which facilitates just that: from the current list of specimens the user can choose which one to display together with the template. The visualization method can be easily switched between the two most common modes: a single slice, or volume rendering. It is still possible to deviate from this path by using all functions Amira provides, in order to display the data differently. If multiple registrations were carried out, by specifying another template or registration method, this module lets the user choose which one should be investigated. The data set is then warped accordingly, and the appropriate template data set is automatically loaded and displayed.

3.2.11 Dependency checks

When calculating the average of a set of brains, it could easily happen with the original scripts that a few data sets went unlabeled, and only after running the script, which can easily take a few hours, was the mistake detected. The VIB protocol offers a mechanism known as dependency checking to discover such mistakes earlier. The idea is that certain meta data depend on other meta data (for example, the transformation onto the standard brain depends on the labeling of the neuropiles), and when the latter changes, the former has to be recalculated. Meta data in this context are stored in files, and these files show modification times. In order to check on a dependency, it suffices to ensure that both files exist, and that the modification times maintain the correct order[11]. The dependency checking also guarantees that after adding new data sets, or after a crash, only the necessary calculations are carried out.

3.2.12 Basic plausibility tests

Some mistakes are not as easy to realize, such as misnaming neuropiles. Therefore, a function was included to check for outliers, i.e. labelings which contain neuropil sizes or positions that substantially deviate from the average. By calling that function, it is thus not only possible to find labelings with wrongly named neuropiles, but also to find brains with unusual neuropil dimensions[12].

3.2.13 Documentation

The VIB protocol comes with a proper documentation, which describes the options and the typical usage. Also, a developer's introduction was written, describing how to extend the functionality, which is not only possible with the new version of the scripts, but easy: the integration of a feature like VIBscissors is a matter of less than an hour.

[11] The same principle forms the basis of Unix' make utility (see [23], [24]), and is imitated by every major Integrated Development Environment.

[12] If, for example, the left medulla of one brain shows a high deviation from the average, then it is likely that by some accident the brain was deformed prior to recording it. Also, some specimens show unusual anatomical features, and should therefore not be used to demonstrate a common feature.

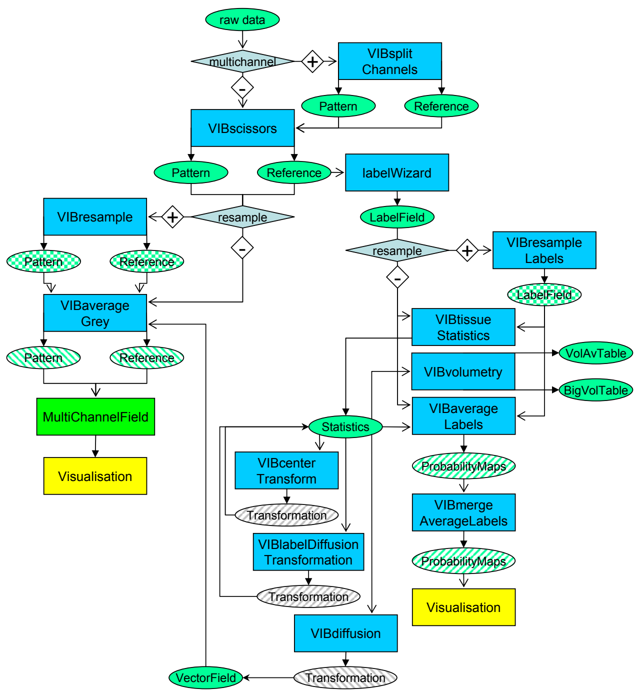


Figure 3.4: The complete data flow of the enhanced set of scripts (A. Jenett, unpublished; with kind permission).

3.2.14 Graphical user interface

The graphical user interface (”GUI”) was enhanced as much as possible within the framework of Amira. Critical errors result in meaningful popup messages. Repetitive tasks are accessible by simply pushing a button. If the user has to choose between several options, a sensible default is preselected. The data flow can be followed easily by using an intuitive navigation framework. Overall, most of the complexity is hidden from the user.

However, Amira was never intended to be enhanced in points of the usability, but rather in technical features such as new or enhanced algorithms for Image Processing and visualization. The user interface exposed by all modules is derived from a basic model which appeals mostly to programmers, because of its structured, modular nature. On the one hand, this means that Amira exposes a neutral, consistent user interface, but on the other it means also that it is hard to use Amira for regular users, i.e. users who are unfamiliar with programming. In any case, the VIB protocol can be exercised easily using the navigation buttons, even if the data flow is complicated (see fig. 3.4). While the original set of scripts allowed only for the execution of a fixed list of tasks, the options in the VIB protocol permit a more elaborate organization.

3.3 Applying the VIB protocol

3.3.1 Typical application of the standard brain

In order to take advantage of genetics in studying how the brain of *Drosophila melanogaster* works, geneticists have developed the Gal4/UAS method (see section 2.2). With a confocal laser scanning microscope (see section 2.1), the resulting brains can be recorded.

Instead of visualizing gene expression patterns by genetic manipulation, it is also possible to stain the brains with antibodies, which recognize certain gene products. Some expression patterns show the synaptic active regions of the brain, i.e. the neuropiles. In the case of *Drosophila melanogaster*, the antibody proposed for the standard brain is *nc82*. This antibody recognizes the *Drosophila* homologue of vertebrate active zone protein ERC/CAST at the pre-synaptic terminals ([25]).

By "double-staining", i.e. by first recording the activity of a gene as made visible by the Gal4/UAS method, and subsequently recording the stained neuropiles, the locations and levels of activity of a certain gene can be found. For consistency reasons, it is important to apply this process to the brains of several flies, and consolidate the results into a standard, because biological systems are highly variable. This has been done by M. Mader in [26], who used an early version of the VIB protocol, in order to study expression patterns in the mushroom body, a neuropil of *Drosophila melanogaster* which plays an important role in learning and memory. The results of this work were subsequently used to understand which genes are involved in the process of forming and retrieving memory.

If many fly lines[13] are available, it becomes important to classify the lines by their expression pattern. A good method to describe these patterns in a meaningful manner is by using a tuple of intensities, which describes how much signal is present in the different neuropiles. This eases the pre-selection for certain applications. The VIB protocol is the tool of choice to create such catalogs of genetic lines.

3.3.2 Generating average brains

Whenever a new gene or expression pattern is investigated, the pattern has to be mapped to a standard.This can be the standard brain from [6], or an average brain using another template can be generated using the VIB protocol. Since individual irregularities are averaged out, this procedure allows to study the stable parts of the expression pattern.

Many mutants of *Drosophila melanogaster* affect brain morphology. For these it is reasonable to generate an average brain in order to put the results of several studies about the same mutant into context. If whole neuropiles are lacking from the mutant, the mapping to the standard brain would lead to an undesirable distortion.

To compare expression patterns or mutants, the corresponding average brains can be compared to the standard brain of [6]. As mentioned earlier, this is normally done by an appropriate mapping function, which transforms the coordinates from one brain to the anatomically corresponding coordinates in the standard brain.

If that transformation can be inverted, locations in the standard brain can be mapped onto the newly recorded data sets. In this manner, corresponding sites can be identified in two different brains by using the standard brain. This is important, as it allows comparison of results from different laboratories.

In order to calculate an appropriate mapping, a function must be found first, which measures the quality of a particular mapping, i.e. how well the mapping performs. This measure should be maximized by the mapping. However, it is difficult at best to formalize what is intuitively clear, namely how good a mapping

[13]A stock or strain or line is a small population of flies with a genetic identity.

is. For example, sometimes the antibodies used to stain the neuropiles do not diffuse well into certain regions. In this case, there are regions with weak signals in one data set, which correspond to regions with a strong signal in the reference. Evidently, measures relying solely of the correspondence on gray values cannot give the desired results in these cases.

From studies of human perception (see [27, 28, 29, 30, 31]), one is tempted to use similarity of shapes as a quality measure. The problem here is that perceived similarity of shapes relies on knowledge which is not easily expressed in mathematical terms: An obvious choice for a similarity measure of shapes is the mean Euclidean distance of points of the two shapes. However, using this measure, a triangle would be more similar to a circle than to a square, which can be perceived as wrong (for example, if the number of vertexes is important).

Another disadvantage of basing the similarity measure on shapes is that the data are normally given as gray values, not shapes. The problem of calculating shapes from gray values will be treated in detail in section 4.4.5.

When deciding upon a quality measure of the mapping between two data sets, not only the agreement with intuition has to be taken into account, but also how fast an algorithm calculating that measure runs on the computer, and how much memory is needed. For all these reasons there exist several different, nevertheless appropriate mappings.

3.3.3 Comparison of registration methods

As described in section 3.2.6, the VIB protocol was extended to support more than one registration method. I used this facility to integrate and study several registration methods. Since one of the intended applications of the VIB protocol is the standardization of Gal4 lines (see section 2.2), I applied the protocol using these registration methods to such a Gal4 line. Several fly brains from that strain were recorded with a confocal microscope. The Gal4 signal and the nc82 signal were recorded in two different channels. Using the protocol, the brains were registered using only the nc82 channel. Applying the result of the registrations, the average signals of both channels were calculated. The results of this study are presented in the following, as well as in figures 3.5 and 3.6.

- The *rigid center transformation* was the only possible choice in the original set of scripts. It is calculated by optimizing the parameters of a rigid transformation, by minimizing the distances of the centers of the labeled neuropiles. Since the optic lobes' location is relatively variable (as compared to the central lobes) due to the preparation process, rigid transformations suffer from the lack of precision at the individual neuropiles' boundaries.

- The *rigid surface transformation* uses the surfaces of the labeled neuropiles instead of the centers to determine a rigid transformation. Somewhat surprising, the results are in general worse than those from the rigid center transformation. In my opinion, this stems from the higher variability of the neuropiles' surfaces as compared to that of their centers.

- The *rigid gray value transformation* uses Amira's Registration module, which correlates the gray values. This method does not depend on segmentation of the brain, and would therefore be the preferred choice of registration. However, as can be seen in fig. 3.6, the resulting goodness-of-fit is not even near the one from the rigid center transformation. Furthermore, instead of labeling the neuropiles, one has to exercise great care when preparing and recording the specimen, and clean up the 3D stack by blackening voxels which show tracheae or other unwanted structures. The effect when not cleaning up the

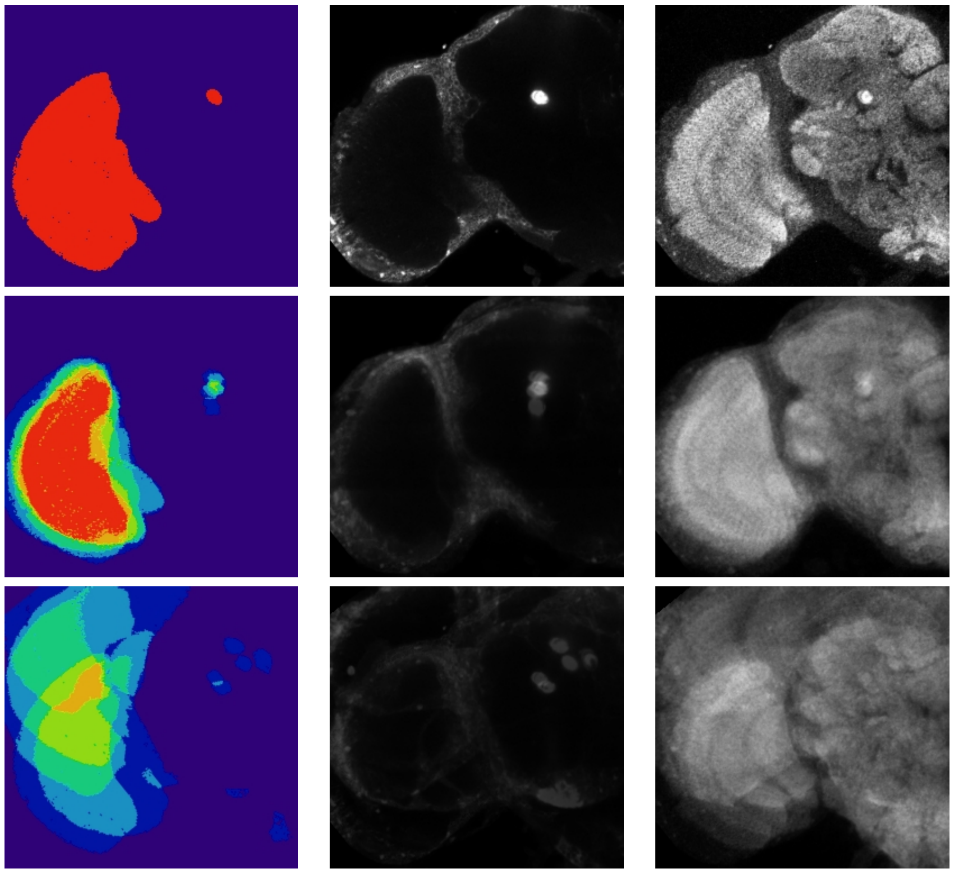


Figure 3.5: Example application of the VIB protocol with comparisons of the registration techniques. For the purpose of this demonstration, only 6 brains were registered and averaged (only the left half of a slice shown). This number is usually too low to produce reliable results, but was chosen on purpose, to demonstrate the qualitative differences of the registrations. The large structure to the left is the optic lobe, the small circle to the right is the peduncle. In the left column, the goodness-of-fit is displayed, i.e. how many brains agree on the neuropil at each coordinate, where blue means no neuropil, and red means that all brains agree. In the middle column, the Gal4 channel is shown. The right column displays the nc82 channel, i.e. the information the registrations are based on. From top to bottom, the rows show: the template onto which the other brains where registered, rigid center transformation and rigid surface transformation

stacks first is illustrated in the figure: one brain is totally misaligned (the light blue spot in the upper right corner is the single optic lobe signal), which leads to the optic lobe of that brain not matching with the others, so that there is nowhere a 100% correspondence of the brains.

- The *non-rigid label diffusion transformation* is the algorithm used in [6], which transforms the labeled neuropiles rigidly, and interpolates between them. The per-neuropil transformations are obtained by using the center transformation as an initial value, and then maximizing the overlap of the volume of that particular neuropil in both 3D stacks. This method is by far the slowest of those presented.

- The *non-rigid Landmark based warping* transforms all centers of the neuropiles to the corresponding centers of the template brain, and interpolates for all other coordinates. This algorithm works very fast, and yields better results than the center transformation. However, it is prone to mapping errors due to the lack of anatomically motivated constraints.

The label diffusion approach turned out to give the most accurate results, as can be seen when looking at the optic lobe: only the label diffusion transformation leaves the lobula plate recognizable (the small structure at the right of the optic lobe). Note that the variability of the neuropiles is much smaller than the variability of the surrounding tissue. Therefore, the neuropiles in the average are about as clearly visible as the neuropiles in a single brain, while the rest is "washed out".

3.4 The need for an automatic segmentation

The VIB protocol made work with the standard brain easier and more efficient. Even so, there remain problems with the labeling process: It needs too much time, and is not reproducible, since it depends on the operator.

For [6], all labelings had to be done by one and the same person, as early experiments had shown a significant deviation between segmentations done by different persons (in fact, it is inevitable that even the same person can not produce exactly the same segmentation twice). These deviations are not errors, they are just consequences of the uncertainties in the data. Being manual recordings, they never are as precise as one wishes, and thus it is often difficult to decide if a pixel at the border of a neuropil belongs to that neuropil, or is outside of it. This fact together with time constraints does not allow for totally consistent labelings.

A less obvious problem is shown in fig. 3.7: Usually, the data set is segmented only in xy slices. Experience showed that this is more efficient than using also yz and xz slices. The reason for this is that whenever the segmented regions are looked at from another direction, there seem to arise inconsistencies. As soon as they are corrected, the segmentation appears to be wrong when seen from another angle. This can be continued ad infinitum, leading to too much time spent adjusting apparent mistakes. The root of this problem lies partly in the noisy nature of the recordings, and partly in the outlines of the neuropiles not being as smooth as one likes to segment them (see for example the lower left slice in fig. 3.7). The segmenter therefore has to compromise between accuracy and time. One could now ask why an automated procedure is expected to be more accurate, but this is the wrong question. In order to find a mapping between two brains – which is the final goal of the VIB protocol, after all – it is not important to segment the neuropiles in a very accurate fashion. Instead, it is sufficient to label identifyable structures both reliably and consistently. These structures need not necessarily coincide with the

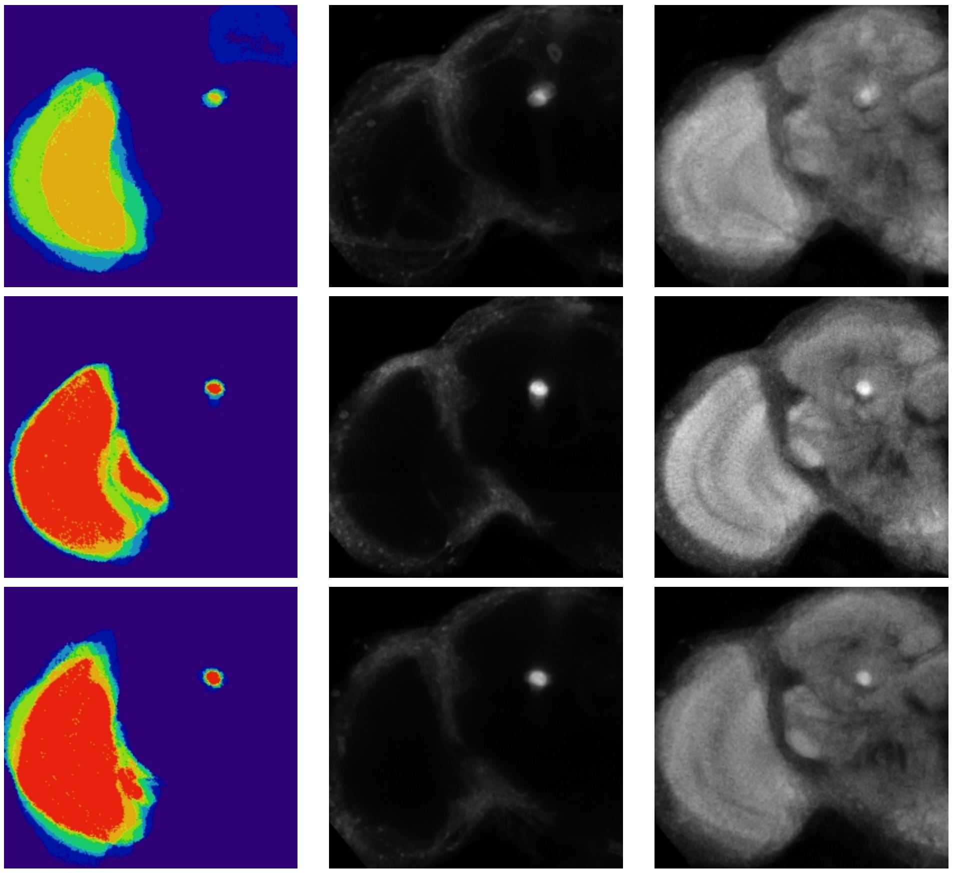
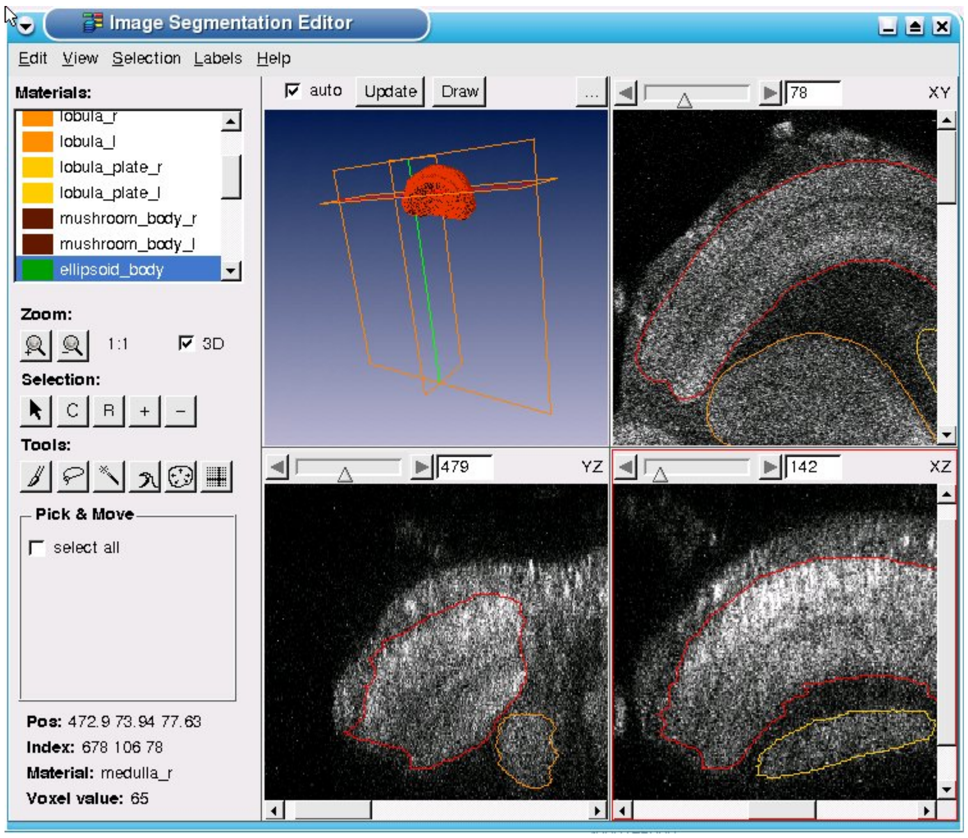


Figure 3.6: The comparison of fig. 3.5 continued. From top to bottom: rigid gray value transformation using Amira's Registration module, non-rigid label diffusion transformation and non-rigid Landmark based warping. Note that the last two transformations are non-rigid, i.e. they warp the data, so that no reliable volumetrical and only limited anatomical studies are possible on the averaged brain. Still, they can match the single neuropiles better than the other methods. Since the best agreement on the neuropiles is obtained by using the label diffusion transformation, this method was chosen as the registration mechanism preselected by default in the VIB protocol.



Pos: 472.9 73.94 77.63
Index: 678 106 78
Material: medulla_r
Voxel value: 65

Figure 3.7: The medulla is shown in 3D (upper left), along with three orthogonal cuts through the data set, where the outline of the medulla is shown in red. The segmentation was done using only the xy slices (upper right), and therefore the other two cuts show irregular outlines. Because of the high noise level, it is difficult to take the yz and xz slices into account. Note for example the lower left peak in the xz slice. The corresponding outlines in the xy slices look correct, though.

neuropiles. If the algorithm yields similar results in two different data sets, a good mapping between them can be calculated.

In addition to said problems, other laboratories can not fall back to the same person who did the original labelings for the standard brain. However, in order to fit results into the standard brain, the standard brain and the newly recorded brains have to be labeled in the same way. This problem can be solved to a certain extent by relabeling the standard brain, but it bears the risk of introducing an inconsistency from the bias of the segmenter.

It is more desirable to have an automatic procedure, which may have a certain bias, but reliably so. All we want is a consistent mapping: possibly unintended deviations in the labeling would be present not only in the newly labeled data set, but also in the standard brain (which is just re-labeled using the automatic procedure). Therefore, the mapping becomes reproducible, and thus robust.

It should be noted that manual labelings are variable, too. If one anatomist labels the same data set twice, differences can be noted. An automated segmentation should therefore be rated by comparing it with multiple manual labelings produced by the same operator. Using the VIB protocol with 10 labelings of the same brain, the standard deviations were calculated, and are shown in fig. 3.8. In this setting, only a few neuropiles were labeled, in particular, only half of the neuropiles which are present both left and right in the brain were segmented. Note that there is a remarkable difference between the results when using registration or not: In one case, it is not assumed that the underlying 3D stacks are identical, and a registration is performed on the labels. The irregularities in the registration are implied by the procedure, and would normally be invisible to the observer. In this special case, though, the optimal mapping is known, since the data sets are identical (but not their labelings). However, one should keep in mind that the same effects are present when registering different brains. This experiment thus gives not only rise to a sensible assessment of automatic procedures, but also cautions against overly broad expectations of the performance of the protocol. Similar studies were done in [32, 33] for MRI images, but they were only used to assess the quality of different working methods.

An automatic segmentation procedure, disposing of volatile errors, would itself become a standard.

3.4.1 Survey of other frameworks to generate average brains

The VIB protocol is not the only implementation of a framework facilitating the work with a standard brain. Especially in medicine, it is important to compare and standardize brains.

To evaluate if these frameworks can be used to generate average **insect** brains, one should first note the differences between human and insect brains in the context of standardization:

The Talairach system as described in [16] needs 12 rectangular regions to be marked in order to calculate a piece-wise linear mapping between two different brains. This approach relies on those regions being very distinctive, so that two different anatomists can reliably identify them. If they were not easy to locate in a precise fashion, this method would not produce robust warpings. When trying to use the same procedure for fly brains, one realizes at once that not only the structure is different between human brains (whose most distinctive features are the sulci) and fly brains (which possess structures with more or less smooth surfaces, i.e. without reliable point landmarks), but also that in the insect brain, the relative position between different neuropiles is more variable than the form of the neuropiles themselves. The latter fact is responsible for imprecise mappings when using piece-wise linear transformations.

I analyzed the applicability of the 4 most known software packages to the task of generating average insect brains:

- A system to facilitate working with the human brain was presented in [34], and is called the *LONI pipeline*. The concept is similar to Amira: work is done on modules which are connected by arrows. However, there are a few important differences: the LONI pipeline was created expressly for the purpose of working with brain data, while Amira was intended as an all purpose visualization software with additional computation facilities. Furthermore, in LONI the connections between modules have a direction, which is more intuitive. In Amira it can get quite complicated to keep track of the meaning of the connections: depending on how this connection was established, it can stand for input or output of a module. Another important distinction is that LONI is free of cost, while Amira is a commercial product. In addition, LONI is written in Java, which means that it runs on any platform supported by Java, such as Windows, Linux, MacIntosh, and most Unix variants. In contrast, Amira runs on Windows, Linux and a few selected Unix platforms. All that said, Amira was chosen for the *Drosophila melanogaster* VIB protocol, because it was ready to be used at the time, and its developers were ready to engage in a research project together with a few work groups in the quest for a common method to create a standard brain.

- Another software package with a similar philosophy is *3D slicer* (see [35]), which is written in C. It is in less wide-spread use than LONI, mostly for two reasons: it is more difficult to obtain (requires personal approval), and it is much more complicated to install. Nevertheless, it provides a viable, if complex user interface to work with brain data, including their visualization and segmentation. Recently, an automatic segmenter was added ([36]), which targets MRI brain images.

- In the context of human brain standardization, a lot of research is currently going on, since ever more precise recordings of invasive and non-invasive electrophysiology become available. While earlier methods often took the whole brain into account, it turned out that for neurophysiological studies, the surface of the cortex is much more relevant. Many standardization techniques therefore target the surface rather than the volume, and consequently the mappings use surface data. To name a few, the *Caret* system ([37, 38]), and the *FreeSurfer* package ([39, 40]) use this approach. These programs are of small value to insect brain researchers, because the functional structure of an insect brain emphasizes volumetric data. Furthermore, the scale of the recordings complicates the reconstruction of surfaces as a reliable means to describe the neuropiles (this will be elaborated in section 4.4.5).

The described programs all contain only rudimentary facilities for automated segmentations, but do provide tools to ease the burden of manual segmentation, thus introducing *semi-automated segmentation*[14]. The idea is to let the anatomist click on a structure, and then execute a detection algorithm which labels the outlines of this structure, subject to correction by the human segmenter. With the exception of Amira, all programs are targeted to MRI or CT images, which explains why experiments with insect brains, using these programs, mostly fail to please.

3.4.2 Evaluating the segmentation facilities of Amira

In chapter 4, I will describe a whole plethora of algorithms which were designed for the purpose of segmentation. But first, let us have a look at how things were done in [6] based on work done in [41, 42], using Amira, and what tools Amira provides to ease the process of segmentation.

The data sets were manually segmented using Amira's *segmentation editor* (see fig. 3.9 top left). It shows one slice through the stack at a time. Confocal recordings usually have a higher xy than z resolution, therefore it is common to work on xy slices. The editor allows to select voxels and assign them to a neuropil. To this end, it provides the following tools:

- The **brush** (see fig. 3.9 top right) is well-known from many painting programs such as Adobe Photo Shop or GIMP: using a circular shape with adjustable diameter, an operator can paint thick lines with the mouse. A nice feature in Amira is that you can mark the outline of the structure, then right click in the middle, and the region inside will be marked, too. It is quite easy to mark up large structures like the optic lobes with this tool, but not finer structures like the mushroom bodies' peduncle.

[14]Recently, there were advances towards automatic segmentations of MR images, which look promising for medical applications (see [32]).

- The **lasso** (see fig. 3.9 middle left) is similar to the brush, but it does not have an adjustable diameter, instead fixing it to one pixel. In contrast to the brush, this tool automatically produces connected lines (which is important when moving the mouse very fast). By right clicking the mouse, the marked line is automatically closed, and the interior is selected. This tool is much more precise than the brush, but also harder to operate: with the brush, a mistake can be corrected at once, either painting the missing part, or unpainting the wrongly labeled area. With the lasso, one needs to complete the curve first, and add or subtract another selection after that.

  An option allows to activate auto tracing, which tries to guess the exact outline of line segments guided by the gray values. A new line segment can be started by pressing the mouse button. With brain images obtained by confocal laser microscopy, this algorithm often fails spectacularly. The grained nature of the images does not allow for smooth color gradients, a feature the auto tracing algorithm relies upon.

- The **magic wand** (see fig. 3.9 middle right) is also called *select contiguous regions* in other applications. By moving the mouse into the desired region, pressing the mouse button down, and then dragging the mouse farther away from the first position, a growing region is selected by merging pixels according to their distance in space and color. It became clear after just a few trials that this tool, if it worked, would speed up the segmentation process dramatically. It also became painfully evident that this tool does not work at all with brain images obtained by confocal laser microscopy. The reason is the same as for the lasso's auto trace option not working: those images do not contain particularly smooth color gradients.

As mentioned earlier, these tools operate in the segmentation editor of Amira. This implies that they work only in two dimensions, because it is not possible to display a truly three-dimensional image on the screen (at most, a 2D projection can be viewed, but this would not help the segmentation process)[15]. As such, the selected items are always pixels. All the same, a slice has a thickness, and the pixels are not really two dimensional, but rather small cuboids[16]. To reflect this, a pixel with a depth is also referred to as *voxel*.

Another method to segment is also included in Amira: segmentation by threshold. By specifying a certain gray level, all voxels whose gray level is below that threshold are assigned to the outside, and all others are assigned inside. The original idea was that once outside and inside were automatically found, connected regions could be separated into the neuropiles. Again, like the problem of the magic wand, the character of the brain images at hand does not permit this method to succeed.

Over the time, Amira, as well as the VIB protocol, evolved. Amira, as of version 3.1, supports these additional segmentation tools:

- The **propagation contour** (see fig. 3.9 bottom left) is an implementation of the *active contour* algorithm, which will be described in detail in 4.4.3. Its idea and its handling are very similar to the magic wand. Two points make this tool less valuable in the context of *Drosophila melanogaster*'s standard brain, however. It is necessary to pre-calculate certain values depending on the starting point, which takes substantially more time than the other tools, and the region growing can not be limited to a certain direction, instead it is

[15]Notice that Amira supports what it calls a "4 viewer layout", displaying a 2D projection of the selected area, and 3 orthogonal slices through the stack (see fig. 3.7). However, the tools can only be used on one slice at a time, effectively restriction operations to two dimensions.

[16]The sides do not necessarily have to have the same length; in fact, confocal recordings rarely have the same resolution laterally as along the z axis

assumed that the texture stays the same along the outlines, which is not the case for many neuropiles.

- The **blow tool** (see fig. 3.9 bottom right) is a generalization of the magic wand. Instead of starting with a single point, a more or less rough outline has to be provided manually by the operator, and the region growing starts with this outline. If this concept were combined with a region growing based on active contours, it would likely be another important tool for the VIB protocol.

- The **wrap tool** truly works in 3D: it uses selections of several slices (possibly orthogonal ones), and fits a 3D surface on these data. The underlying mathematical concept is the optimization of a polynomial function satisfying a constraint implied by the segmented slices, minimizing the degree of the polynomial. In theory, this allows the user to segment only a fraction of the data, ideally using slices in all three orthogonal directions to obtain a smooth surface fitting the data. In practice, the processing time is very high, and the implementation in Amira is unstable, leading to program crashes when the size of the structure to be labeled exceeds a very small threshold. As of the time of writing, these problems could not be resolved by the developers of Amira.

It can be seen in fig. 3.9 that the brush and the lasso yield more accurate results than the other tools. The more sophisticated tools (lasso with auto-tracing, magic wand, propagation contour and blow tool) are too dependent on local gray value extrema, and are not able to extrapolate a sensible neuropil boundary. To a certain extent, this problem can be overcome by smoothing the selections, i.e. fill holes and fit smooth curves to the outlines. However, it turned out in my experiments that no set of parameters (for the tools and the smoothing process) works reliably[17]. For each choice of parameters, there were a few data sets which could be labeled well, but more which could not. Apparently, the structure of the staining is not homogeneous enough, and the stainings of different brains are not similar enough, for the sophisticated tools to work reliably, even after smoothing the results.

Since the wrap tool does not rely on the gray values, but only on selections fed to it, it can be viewed as a smoothing tool. While the result is visually pleasing, often undesired labelings can be found, when looking on slices in which the labeling was interpolated. Furthermore, I could not test this tool as much as I would have liked, because Amira tends to crash when using the wrap tool, especially when large neuropiles are processed. Unfortunately, the labeling of large neuropiles would gain the most from this tool.

As illustrated above, the tools (brush and lasso) are adequate to segment a 3D stack of a brain, but the amount of manual work is still substantial. The number of tools invented for the sole purpose of speeding up the segmentation process, and their respective success, shows how difficult the problem of automatic or semi-automatic segmentation is. Another important point to mention is that however refined these tools became, they never led into a direction making the manual interaction unnecessary. Indeed, for most of these tools the actual times needed to operate them are fairly equal, the only difference being the taste of the user.

This is how far it goes without taking into account the anatomical constraints. As long as the algorithms do not permit the inclusion of knowledge of several segmented brains, they will always fail at the same points. This becomes obvious when

[17]I limited my tests to one neuropil, the fanshaped body, which is displayed in 3.9. After rigid alignment to the standard brain, and using the goodness-of-fit, the center of each neuropil can be estimated reasonably well. Thus one could define a set of parameters for each neuropil and recording method, and still automate the process.

comparing different regions of the brain: some neuropiles are less clearly separated from each other than the substructures of other neuropiles. As an example, the substructures of the medulla are almost visible enough to label them (see fig. 3.10), while the distinction between the ellipsoid body and the fan-shaped body is usually very difficult (as illustrated in fig. 3.11). In the latter case, it is usually impossible to draw the line between the neuropiles without prior knowledge about the anatomy. Even then, one often has to take into account the surrounding sections, i.e. the slices above and beneath the current slice.

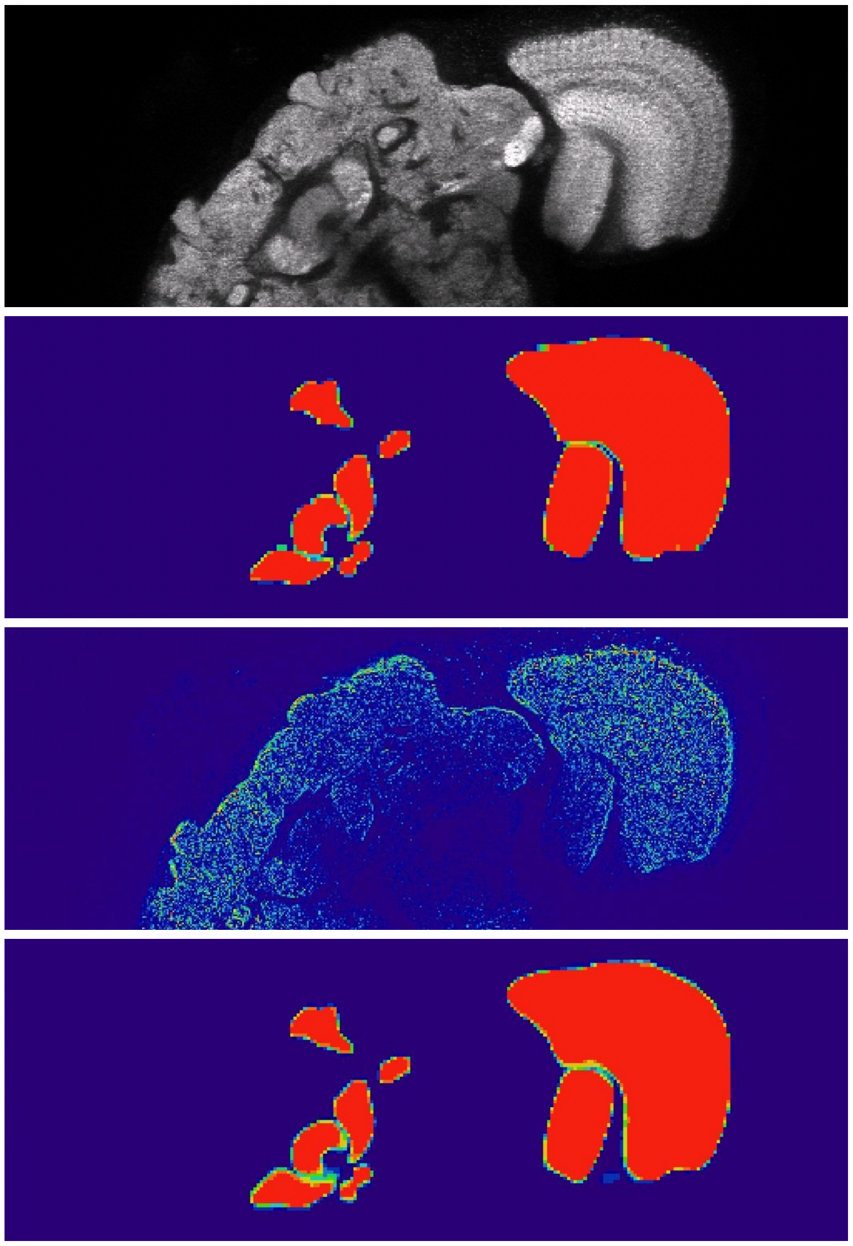
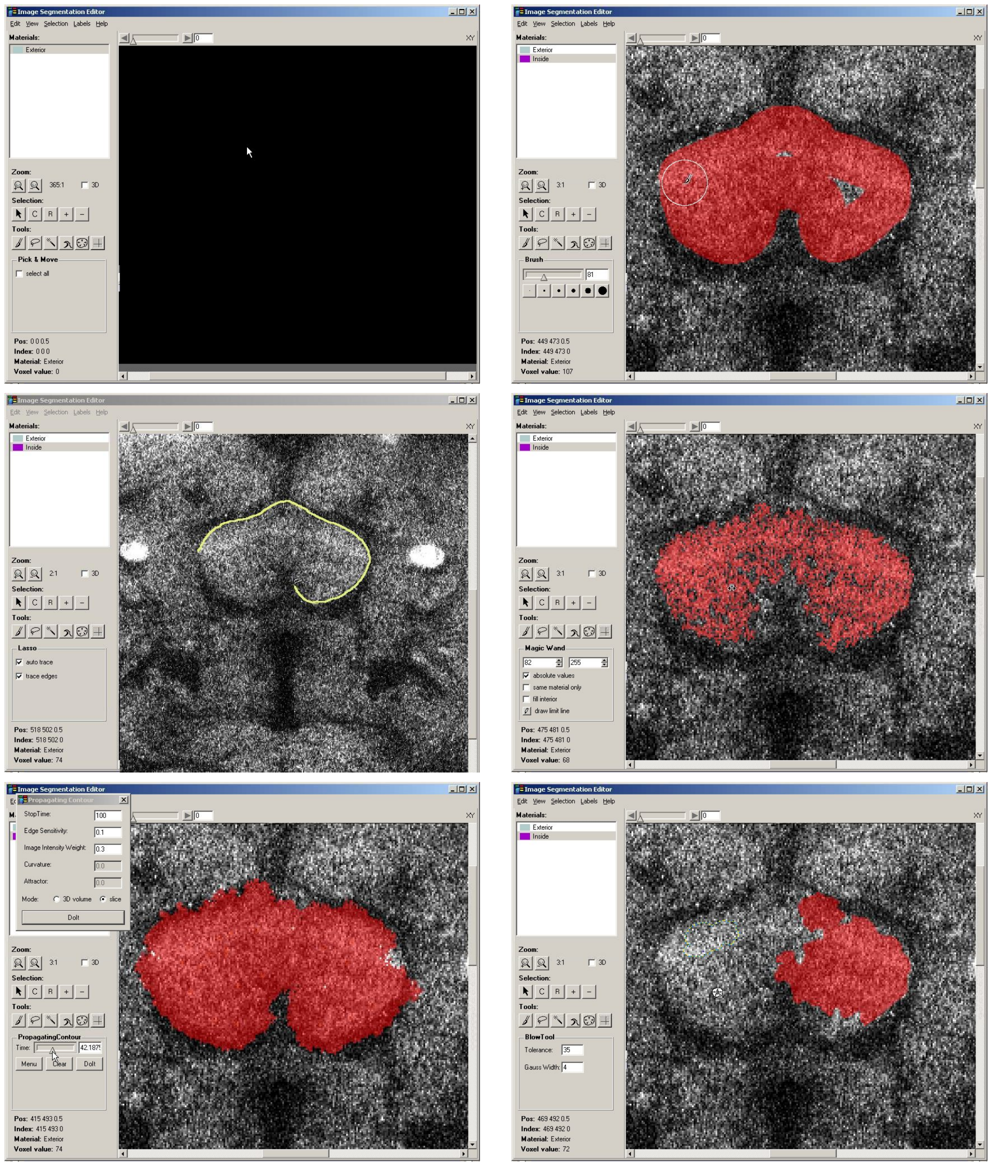


Figure 3.8: From top to bottom: a slice of the original data set, a probability map of ten segmentations of the same brain (without registration; red denotes total congruence of the segmentations, blue means outside), the difference between the original and the average gray image after mapping was calculated from the labeled neuropiles and treating the different segmentations as if they were from different brains (blue means no difference, red is 50 or above, where 256 gray values are available), and a probability map after registration using the information from the labelings.



Zoom: 365:1 ☐ 3D
Pick & Move
☐ select all
Pos: 0 0 0.5
Index: 0 0 0
Material: Exterior
Voxel value: 0

Zoom: 3:1 ☐ 3D
Brush
81
Pos: 449 473 0.5
Index: 449 473 0
Material: Exterior
Voxel value: 107

Zoom: 2:1 ☐ 3D
Lasso
☑ auto trace
☑ trace edges
Pos: 518 502 0.5
Index: 518 502 0
Material: Exterior
Voxel value: 74

Zoom: 3:1 ☐ 3D
Magic Wand
82 255
☑ absolute values
☐ same material only
☐ fill interior
☐ draw limit line
Pos: 475 481 0.5
Index: 475 481 0
Material: Exterior
Voxel value: 68

Propagating Contour
StopTime: 100
Edge Sensitivity: 0.1
Image Intensity Weight: 0.3
Curvature: 0.0
Attractor: 0.0
Mode: ◯ 3D volume [✓ slice]
Zoom: 3:1 ☐ 3D
PropagatingContour
Time: 42.1875
Pos: 415 493 0.5
Index: 415 493 0
Material: Exterior
Voxel value: 74

Zoom: 3:1 ☐ 3D
BlowTool
Tolerance: 35
Gauss Width: 4
Pos: 469 492 0.5
Index: 469 492 0
Material: Exterior
Voxel value: 72

Figure 3.9: Top left: The segmentation editor. On the upper left is a neuropil list, below the tool window, and to the right the image to be segmented. Top right: Using the brush. Middle left: Using the lasso. Middle right: Using the magic wand. Bottom left: Using active contours. Bottom right: Using the blow tool.

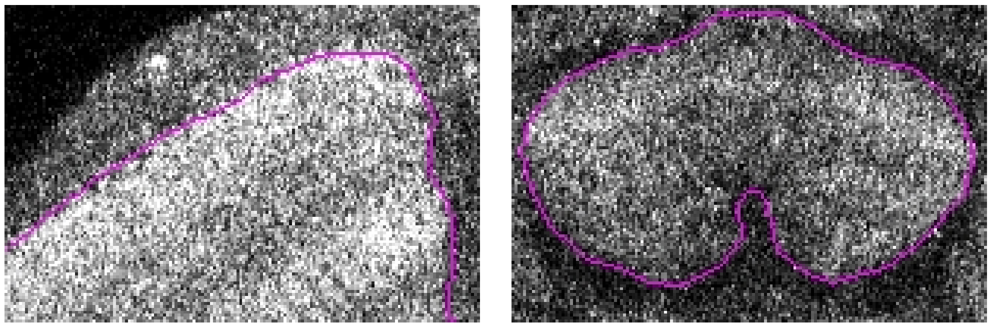


Figure 3.10: Left: a closeup of the medulla. Nicely visible are the dark bands, which are substructures of the medulla. Right: a closeup of the fanshaped body. This time, there is a bright band spanning from the left to right.

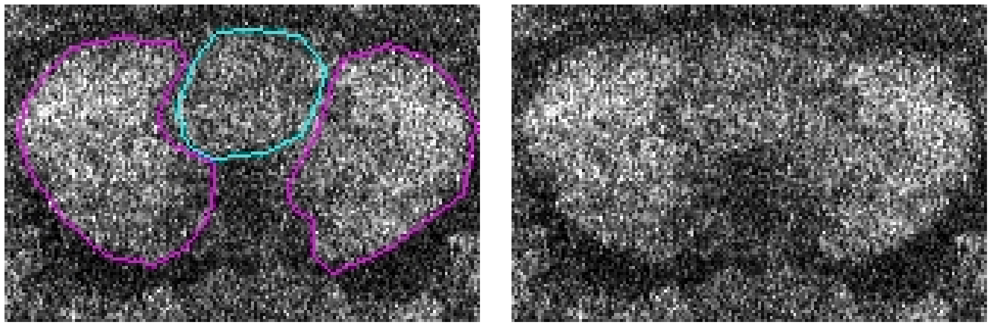


Figure 3.11: Left: a closeup of the ellipsoid body enclosed in the fanshaped body. Right: the same closeup, but without the outlinees of the neuropiles. These two neuropiles are so close that it is often difficult even for experienced neuro-anatomists to distinguish them.

Chapter 4

Examining the limits of traditional approaches to automatic segmentation

Quite a number of strategies have been developed which attempt to solve the problem of segmentation. The common starting point is that the images are treated as intensity functions of the coordinates, i.e. each coordinate is mapped to a gray value. Strictly speaking, it is mapped to a feature vector, whether that be a gray value (1-dimensional), a color (normally represented by a triplet of *red, green and blue*), or any other vector. However, the images presented in chapter 3 contain only gray values[1]. Therefore, this chapter only discusses algorithms for such images.

The gray values are usually given only for discrete Cartesian coordinates, i.e. an image is organized into an orthogonal grid of pixels which have equal rectangular dimensions[2].

It is assumed, however, that the image theoretically is a mostly smooth function of the coordinates, i.e. the derivative of the function is well-defined on the whole plane with the possible exception of a zero set of coordinates. While many algorithms work by assuming that the pixel value is the exact intensity of the center of the pixel, this is not so. Rather, the pixel value is the **mean** intensity of the pixel's area. This fact does not matter much if the resolution is fine compared to the structures one hopes to find. But for images of *Drosophila melanogaster*, where some structures of interest have diameters of about 10 to 20 pixels, it is important to use the precise version.

4.1 Classical Image Processing

As mentioned in the introduction, an image can be looked upon as a function mapping coordinates onto gray values. In the same manner, a "derivative" of the image can be defined: It is a new image of the same dimensions, but each pixel is mapped to the derivative of the intensity function instead of the intensity, sampled in the same way as the original image.

[1]In scientific imaging, a gray value can mean something else than a recorded shade of gray, for example for radiological images, or images from a confocal laser scanning microscope. In the latter case, the gray values denote intensities of fluorescence excited by a laser, which are indicative of the fluophore count at that location.

[2]There are applications where this is not so: Geographers often work on spherical images where the pixels do not have uniform dimensions. For purposes of this work, however, it is irrelevant to treat those cases.

Another popular method to process an image is to convolve it: For each pixel, a square neighborhood is treated as a matrix. The pixel value in the convolved image is the sum of the component-wise products of the convolution matrix's entries with the corresponding pixel values from the original image. Many procedures in Image Processing are implemented as convolutions, because they are very fast due to their linear complexity.

The derivative of an image can be approximated by a convolution matrix. The original data being defined only at discrete locations, the derivative has to be approximated by differences of pixel values, assuming more or less smooth underlying functions. A somewhat surprising consequence of this is that such approximations are *anisotropic*, i.e. the data are not treated equally in each direction. This stems from the fact that adjacent pixels' center points do not have equal distances: Relative to the grid, horizontal and vertical distances of neighbors are shorter than diagonal distances. The approximation of the derivative from a neighborhood therefore introduces different errors depending on the direction of the gradient.

4.1.1 Edge detection filters

One of the first viable methods to detect outlines was based on the observation that borders between objects[3] (called *edges* in Image Processing) usually coincide with a high gray value gradient at that pixel (see [43]). This stems from the fact that many objects bear a homogeneous gray value, and the gray value differs between different objects.

By calculating the local maxima of the absolute value of the derivative of the image, one obtains local edges. Connecting these edges, the outlines of the objects can be found. Or so the theory goes.

In practice, a lot of prerequisites have to be met for this algorithm to work. Among others, these are: the object of interest has to have uniform gray value, the background has to have uniform gray value, these two gray values have to be substantially different, and the physical process of recording the image has to be very precise to minimize measurement errors. In industrial applications, for which the first edge detecting filters were implemented, it is usually possible to adapt the processes to this end.

Instead of using the first order derivative, the zero-crossings of the second order derivative can be used. In theory, this yields better results, but it is even less robust than the first order derivative, because the second derivative is approximated from the approximated first derivative.

4.1.1.1 Experiments

The edge detecting filters I tested are described in the following. Note that each filter consists of two convolution matrices, so as to detect edges in perpendicular directions. The absolute values of the two convolutions are then added, and the result is shown in the left column of fig. 4.2.

- The *Roberts* detector is defined by the matrices

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \text{and} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix}$$

  In contrast to the filters which are described later, the edges are not located on the same grid as the gray values. Instead, they are located on a grid which is translated by half the pixels' width and height: Since the convolution matrix

[3]For purposes of edge detection, the outside is also treated as object.

is 2×2, the corresponding neighborhoods are also 2×2. Therefore, the center of these neighborhoods, which are the coordinates of the approximated derivatives, do not lie on the lattice points, but in-between.

- The *Prewitt* detector consists of the matrices

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

  By approximating the derivative in this manner, the data are treated as if the gray values were constant either horizontally or vertically.

- The *Isotropic* detector consists of the matrices

$$\begin{pmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{pmatrix} \text{ and } \begin{pmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{pmatrix}$$

  This approach emphasizes the center more than the Prewitt filter.

- The *Sobel* detector consists of the matrices

$$\begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \text{ and } \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

  The Sobel filter is probably the best known edge detecting filter, because it has been used by Canny in a more complex edge detection scheme, which will be described in section 4.1.3.

As is shown in the left column of fig. 4.2, the results are less than pleasing. The most common reason for these algorithms to fail is described in the next section.

4.1.2 De-noising

One of the most difficult problems since the beginnings of Computational Image Processing has been the *noise*: The sensors of the camera, together with physical constraints, invariably lead to measurement errors. These are visible as speckles or general inhomogeneity in the image. The trouble is that many algorithms, like estimating the derivative, are highly susceptible to noise.

So, the next step was the invention of *denoising*: Based on the neighborhood of a pixel, which is supposed to have similar intensity and error distribution, an estimate of the true intensity is calculated. The result is an image which is much smoother than the original, but still shows the important features. In particular, the derivative of the result can be estimated much more reliably.

The intrinsic drawback of denoising is that it is often hard to distinguish between noise and signal. Most algorithms therefore provide a parameter to adjust the sensitivity. Setting that parameter is always a trade-off between leaving too much noise and removing details unintentionally.

One such method is also called *despeckling*: For each pixel, the median[4] of the values from a neighborhood is assigned as the new value. This method was motivated by two facts: Often, the recording process implicates an error which is

[4]The *median* of a set of numbers is determined by first ordering the set, and then taking the middle value.

proportional to, or at least increasing with, the true value. Second, the observed value can not be smaller than zero. Thus, the error distribution is skewed towards positive errors. In theory, the true value can be obtained by recording the same pixel over and over again, and taking the peak of the distribution[5]. The despeckling process takes values from the neighborhood instead, thus assuming that the values are close for near by coordinates.

In practice, it is more robust to take the median of the observed values, rather than the average. One reason is that the median is less affected by outliers. Furthermore, the median approximates more reliably the mean of a skewed distribution. In many cases, error distributions are found to be skewed, due to a natural bound on one side only. For example, the intensity of fluorescence has no theoretical upper bound, but it can not be less than zero.

Instead of ignoring the spatial relationship, as is done when computing the median, smooth functions can be fitted to the sampled data (see a review [44]). Even if this algorithm does not strive for a correction of the underlying error by modeling that error, but instead approximates the observed data, this approach has its benefits. The big advantage is that the function to be fitted can be chosen according to the task. For example, when one wants to compute the gradient of an image, one will fit a differentiable function. In [45], for example, splines were used. However, that choice is not always wise: Since splines have a linear second derivative, albeit being easy to work with, the approximation often misrepresents complex scenarios. Furthermore, when it comes to edge detection, the principal problem of such approximations is that edges are **discontinuities** in the derivative. If one now chose to fit a smooth function to the observed image, even if there were visible edges, the fitted function attenuates them.

A related denoising technique is low pass filtering: The signal is transformed onto a (vector[6]) base of functions which can be sorted by spatial frequency. Then, the frequency components, which correspond to higher oscillations, are set to zero, and the reverse transformation is applied. To apply this theory to images, one has to find a base to represent the image. Common choices are wavelets, or windowed sine functions[7]. The sensitivity of this algorithm can be controlled by the cutoff frequency. This frequency need not be global, if the base functions are localized. In the case of wavelets, a viable method to determine the cutoff selectively has been presented in [46], and a slightly different algorithm in [47].

4.1.2.1 Experiments

I tested three different denoising techniques (see fig. 4.1):

- The *despeckling* algorithm works by taking the median of a 3×3 neighborhood of each pixel. Obviously, this algorithm is very fast. Somewhat to my surprise, it nevertheless produces adequate results.

- The *low pass* filtering works by applying a fast Fourier transformation on the image, then setting all factors outside a band (or, in the case of a two-dimensional image, outside a ring) to zero. The denoised image is obtained by the inverse transformation on the modified factors. In my experiments it turned out that a band of 5 to 60 (pronouncing structures with extents between 5 and 60 pixels) gives good results. Since the Fourier transform is

[5]This peak is referred to as the *modal value* in statistics.

[6]The mathematical term is meant here.

[7]A *windowed* function features a finite support, i.e. its values are zero outside a certain interval, which is called *window*.

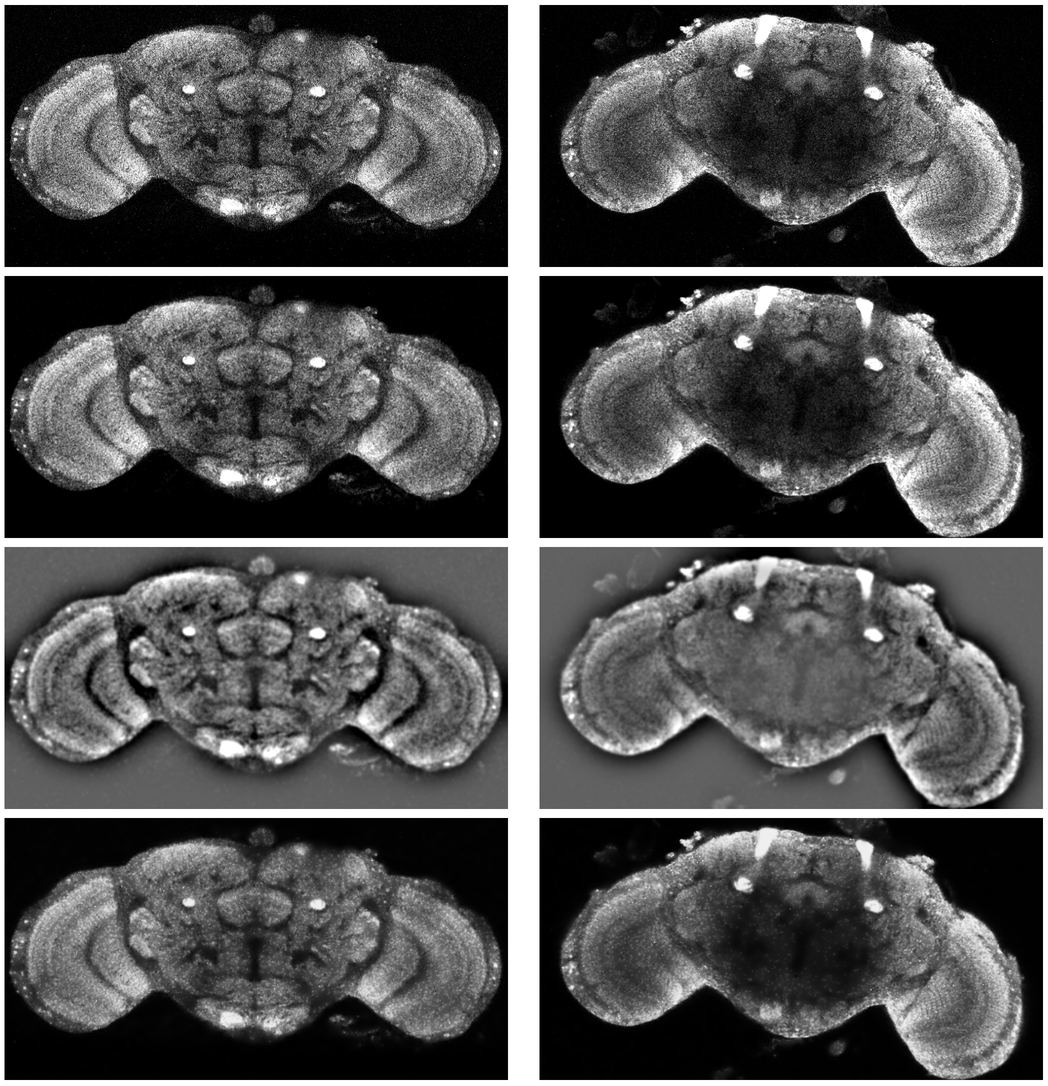


Figure 4.1: Experiments with three different denoising algorithms on two different brains. From top to bottom: original, despeckling, low pass filter using fast Fourier transformations and wavelet based denoising. The right slice was chosen to demonstrate that a denoising algorithm which appears to produce good results on one brain can fail on others and vice versa.

not localized[8], the gray values change drastically, while the edges of the desired structures are still visible. However, for the purpose of edge detection, the difference of the gray values before and after denoising is of little consequence, since the differences between adjacent gray values are smoothed, rather than distorted. Thus, the approximations of the derivatives, upon which the edge detection relies, are straightened out, and not mutilated.

- The *wavelet based* filtering is similar in approach to the low pass filtering. Instead of a Fourier transform, a wavelet transformation is applied. Since it is highly localized, the gray values remain about the same, as one would expect from a denoising algorithm. The price one has to pay is that the resulting image is not smooth. Indeed, in my experiments I had to despeckle the image first, in order to get the result displayed in fig. 4.1. Still, small speckles here and there can be seen. Furthermore, this algorithm takes the longest time of the three investigated denoising schemes.

As demonstrated in fig. 4.1, the wavelet based filter succeeds mostly in reducing the noise, but fails miserably in some cases. The sample shown in the right column indicates a common staining artifact: Even when handled carefully, sometimes the staining does not get through the tissue, resulting in darker areas. Still, the low pass filter appears to cope quite well with that artifact.

Many more strategies and variations have been proposed to remove noise from images (for a survey, see [48]). However, the edge detection filters from section 4.1.1 need a smooth image, not a denoised one. A more directed approach towards that goal is presented in the next section.

4.1.3 Blurring

The denoising process in practice often produces artifacts which are much more visible in the derivative of the image than in the original. However, there is a procedure which smooths both the image and the derivative of the image, and reduces the amount of noise: Gaussian blurring. This means that a Gaussian diffusion[9] is applied to the original image, and it works extremely well when the original noise is normally distributed.

The underlying idea is that the recording process, being a physical process, is necessarily affected by measurement errors, which obey a Gaussian distribution (see section 2.1).

In Image Processing applications, a Gaussian blur is often applied to smooth the data. As discussed before, the gradients calculated from discrete image data, and the second derivatives even more so, suffer from a high error when applied to the original data. When blurring before the derivative is calculated, the mean error is sometimes actually smaller than without blurring, depending on the used discretization method. Often it is more important to have a smooth gradient than a precise approximation of the image: The derivatives can be used to optimize certain functions of the gray values, and in most cases, there are only iterative methods available to find the optimum. In these cases, using unblurred data to obtain the derivatives would lead to an unstable algorithm, i.e. it would not converge towards the global optimum.

Gaussian blurring is used to level out a normally distributed spatial error: Assume that each gray value is spatially dispersed according to a normal distribution.

[8]A transformation is localized if changes in a small area in the original image affect only a small area in the transformed image.

[9]A diffusion of an image means to treat the gray values as heat energy, which disperses in a small number of discrete time steps. It is common to limit this number to 1.

Then, the distribution of the likely origins of the gray value at a certain coordinate is the same normal distribution.

When applying a Gaussian blur, the algorithm has to be provided with the standard deviation (called σ for historical reasons). Usually, the data used to calculate the blurred value at a coordinate are taken only from a circle with radius 2σ. A faster, but less accurate version blurs the image horizontally first (using only a one dimensional blurring), and vertically after that.

The best known all purpose edge detection algorithm was presented in [49]. It consists of a Gaussian blurring, the application of a simple convolution like the Sobel filter (as presented in section 4.1.1), and non-maximum suppression. The suppression consists of a simple local search, i.e. the value is compared to the pixel's neighbors, and set to zero if the neighborhood contains a larger value. In practice, the pixels are only compared to their 4 immediate neighbors (or 6 when working on 3D images).

Based on the work of Canny, Deriche in [50] tried to enhance the localization of the Canny edge detector. It was reasoned that to improve the sharpness of the filter, the blurring process has to take a larger neighborhood into account. Since the Gaussian blurring assumes a normally distributed error, which can be disregarded outside a window four times the standard deviation, it was no longer possible to treat the error as normally distributed. Therefore, this algorithm assumes a different distribution. Since the neighborhood no longer can be restricted to a small window, it was important to implement this filter as a recursive algorithm in order to cut down the execution time.

As the name suggests, blurring an image washes out the detail. To overcome this limitation, and still smooth the image, anisotropic smoothing was invented (see [51]). The idea is to apply a diffusion to the gray values with a locally variable diffusion factor. Usually, the diffusion factor is proportional to the gradient of the image. Of course, here surfaces a chicken-egg problem: In order to calculate the derivative, the image has to be blurred. But to blur it anisotropically, the derivative is needed. In general, this problem is solved by calculating the derivative after a Gaussian blur was applied, and exercise the anisotropic smoothing on the original.

4.1.3.1 Experiments

I applied the same edge filters as in section 4.1.1 after applying a Gaussian blur to the image. The result can be seen in the right column of fig. 4.2. The results are consistent with the image: If only the gray values are taken into account, there are many more edges than the desired boundaries of the neuropiles.

Consequently, an application of Canny's edge detector fails to recognize the complete optical lobe. The results for different parameters for the Gaussian blurring are displayed in fig. 4.3.

Anisotropic diffusion can ameliorate the edge detection, because the edges are still clearly visible after a few iterations. The choice of the diffusion parameter is very delicate: As demonstrated in fig. 4.4, a low value leaves much of the noise, while higher values already wash out details like the boundary between the optical lobes.

The Deriche detector failed in my experiments. The reason is that the approximation of the derivative is too biased to parallels of the axis, as seen in fig. 4.5. This is the consequence of the recursive nature of the implementation. To overcome that problem, the scheme could be applied without recursion, thus avoiding the artifacts. However, the complexity of the algorithm (meaning the execution time) makes this highly unattractive.

In [52], an enhancement of the anisotropic diffusion was proposed. It uses Deriche's method to approximate the derivatives, which was described earlier. As can

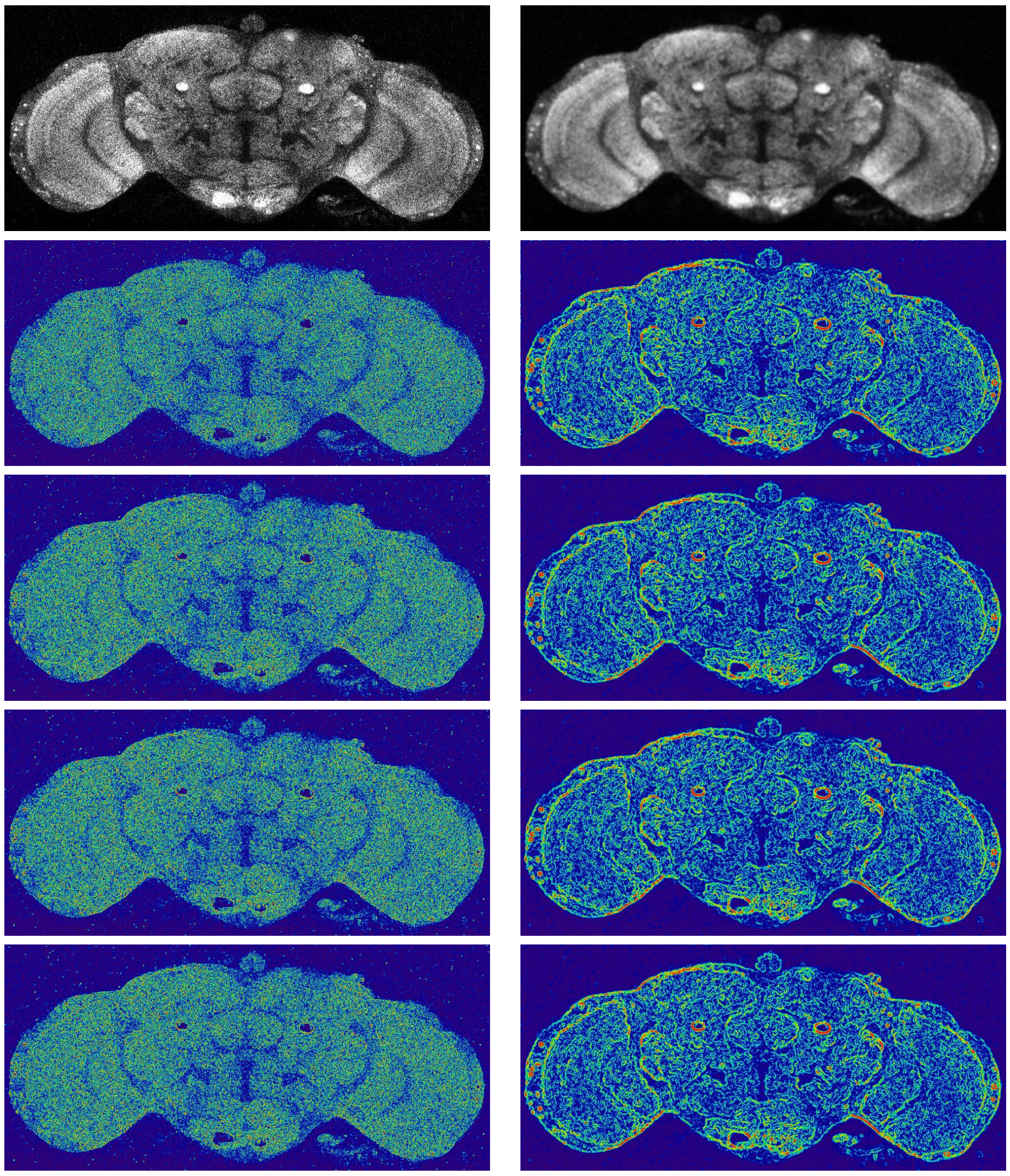


Figure 4.2: Edge Detectors before (left) and after (right) Gaussian blurring (where the standard deviation was assumed to be 3, as was found by experimentation). The rows show from top to bottom: original, Roberts filter, Prewitt filter, Isotropic filter and Sobel filter. To pronounce the (minor) differences between these filters, the images are color coded: blue denotes small derivative, and red means large derivative.

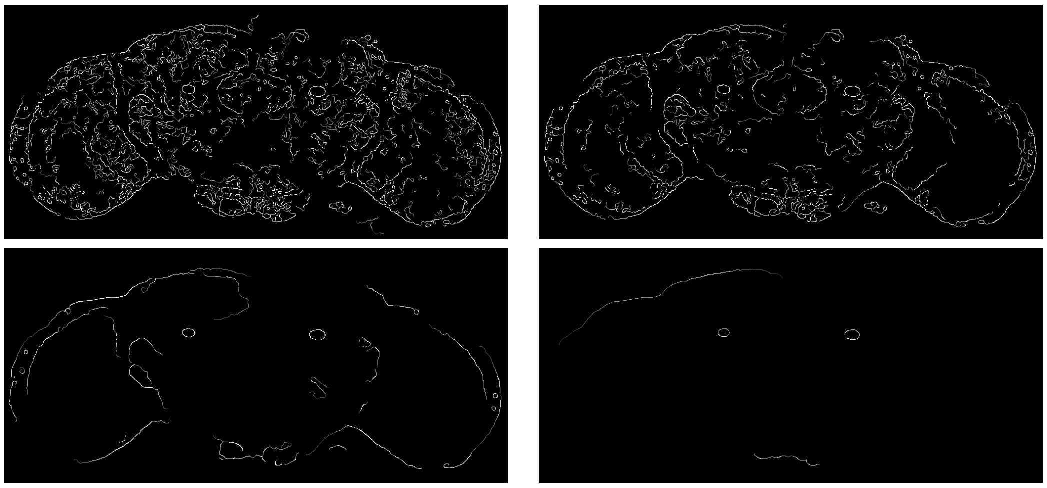


Figure 4.3: Application of Canny's edge detector to the same slice as in fig. 4.2. The parameter to tinker with is the standard deviation of the Gaussian blur. Top left: $\sigma = 1.8$, right: $\sigma = 2$, bottom left: $\sigma = 3$, right: $\sigma = 5$.

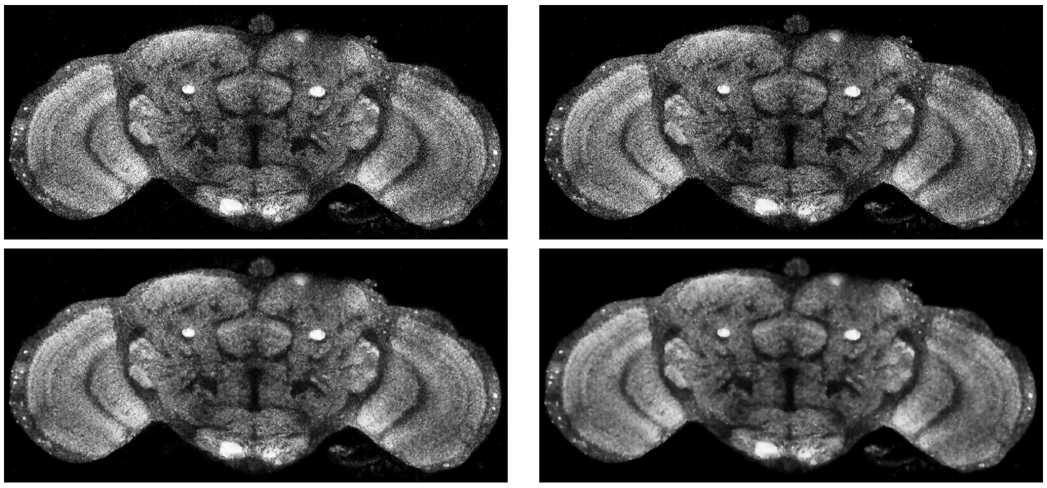


Figure 4.4: Anisotropic diffusion smooths regions with similar gray values, but still pronounces sharp edges. This algorithm heavily depends on the choice of parameters. The displayed images were created using a diffusion parameter of 5 for the top row, and 20 for the lower row. Left: 10 iterations, right: 20 iterations.

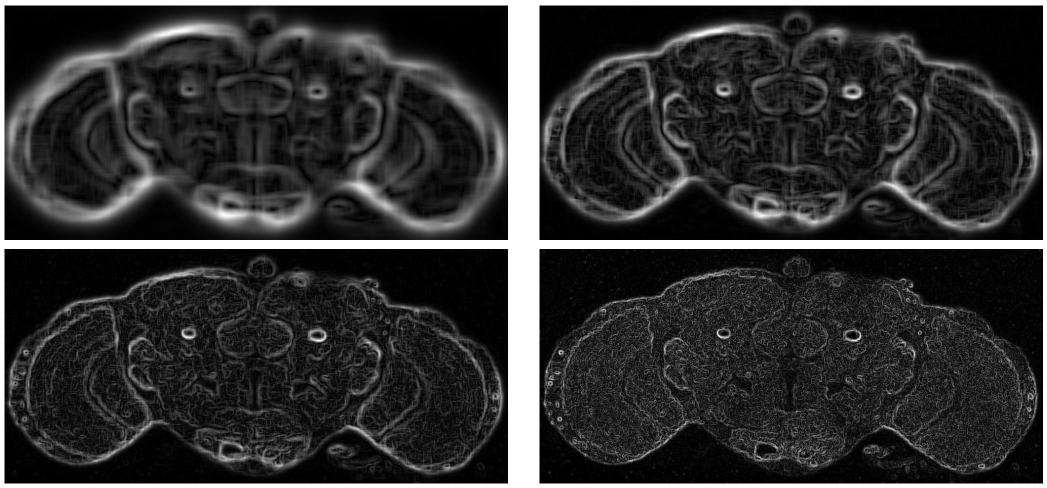


Figure 4.5: Application of Deriche's algorithm to approximate the derivative. Top left: $\alpha = \frac{1}{12}$, right: $\alpha = \frac{1}{6}$, bottom left: $\alpha = \frac{1}{3}$, right: $\alpha = \frac{2}{3}$.

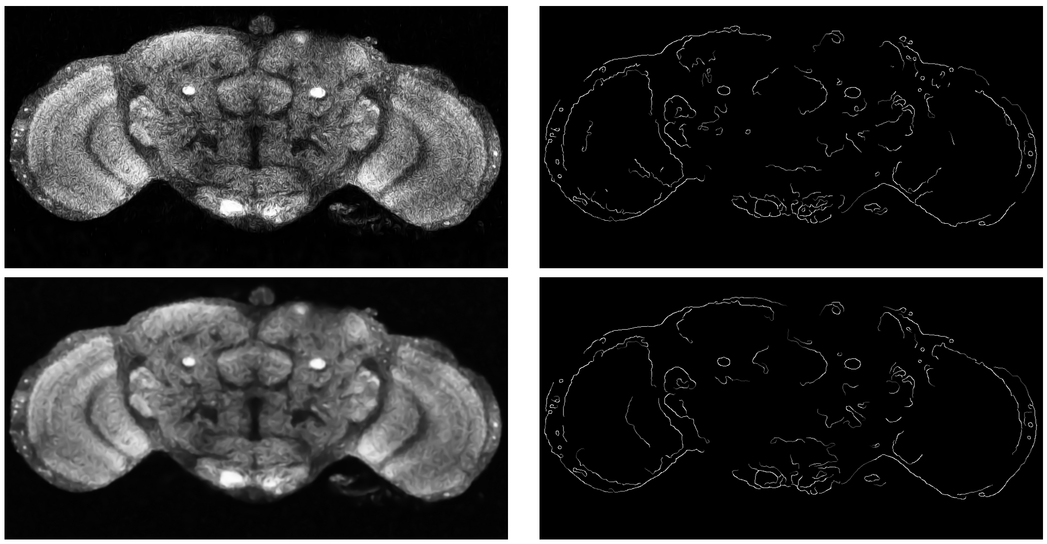


Figure 4.6: Anisotropic diffusion, using Deriche's algorithm to smooth the image before calculating the derivatives instead of Gaussian blurring. Left: anisotropic diffusion applied. Right: Canny edge detection on the diffused image. Bottom row: before anisotropic diffusion, a moderate Gaussian blur ($\sigma = 3$) was applied to despeckle the image. With Gaussian blur pre-applied, the gray image looks much better than without, but the detected edges do not differ significantly.

be seen in fig. 4.6, the method is promising. However, the choice of the parameters still appears to be a trade off between too many structures being detected in the optic lobes, and too few structures in the central brain. There is no clear cut between desired an undesired edges. If the parameter is chosen so as to detect the edges in the central brain sufficiently well, then the optical lobes are cluttered with edges inside the neuropiles.

4.2 Common segmentation techniques

The idea leading to edge detection algorithms was that structures should be bounded by visible edges. Ideally, the set of edges would be the points making up boundaries of structures, and nothing else. In reality, however, one has to isolate meaningful edges and chain them together to obtain a segmentation of the image. Instead of finding edges and then linking them, algorithms were introduced which try to find closed curves right away. The underlying concept is to use a model which only allows for closed boundaries, thus obviating the need to worry about linking pieces together.

4.2.1 Watershed segmentation

The idea behind the Watershed algorithm is that an image, viewed as a mapping of 2D coordinates onto gray values, can also be interpreted as a mapping of 2D coordinates onto a third coordinate. The result is a surface embedded in a 3D space, and it looks more or less like mountains and valleys. The height of a point on the surface thus is proportional to the brightness of the corresponding pixel.

The same view can be taken of the derivative of the image. In this case, the ridges correspond to edges, and the valleys to homogeneous regions in the pictures. Suppose a rain goes down, then the valleys turn into lakes, the lakes join and finally everything is drained in water. Each point on the surface can be associated with the lake it joins first, thus obtaining a segmentation. A less graphic, but mathematically accurate description can be found in [53].

4.2.1.1 Experiments

As displayed in fig. 4.7, most of the boundaries of the neuropiles coincide with detected boundaries. However, if the contrast is low (see bottom row), that method fails.

4.2.2 Level-set segmentation

A common operation on images is *thresholding*: All pixels whose intensity is above a certain threshold are labeled "inside", and the others "outside". It is used to segment a foreground structure, but it relies on the fact that the structures to be labeled have similar gray values, while the background has a high contrast. So, a method has to be found to adapt that threshold locally.

To pursue this idea, the concept underlying the watershed algorithm is used: If an image is treated like a three dimensional surface, it is easy to draw contour lines, i.e. lines on the surface which have the same height. Structures identified by one such contour line are called *level sets*. Thresholding is nothing else than using the same height for all structures. In contrast to threshold segmentation, level-set segmentation typically uses different level lines for different structures. The threshold is adjusted according to some function, which is minimized by the choice of the threshold.

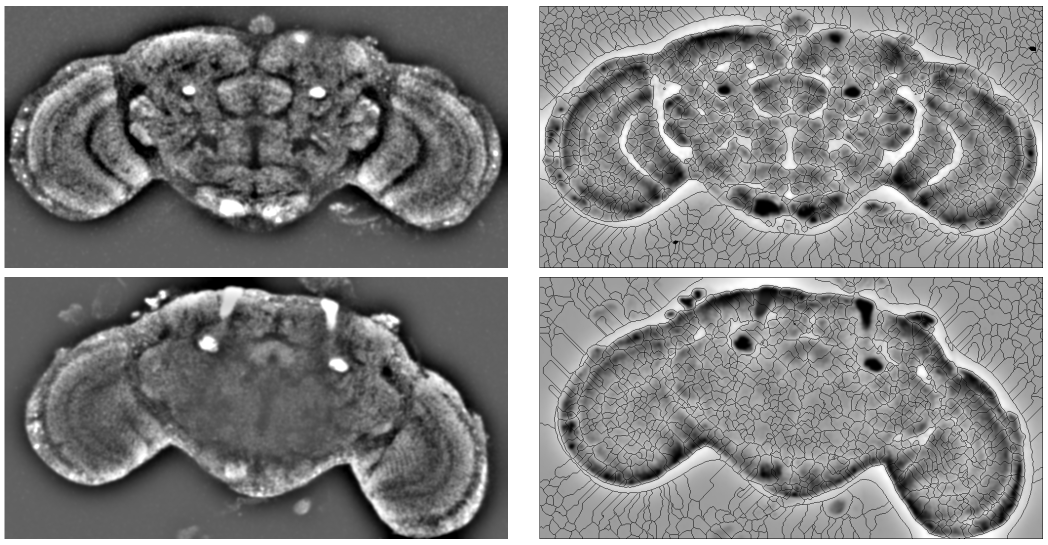


Figure 4.7: Demonstration of the Watershed algorithm. After a vigorous low pass filter from section 4.1.2 (left image), the segmentation is depicted as black lines (right image).

A verbose explanation and discussion of enhancements to this algorithm can be found in [54]. Of main interest for me was the reduction of complexity by using a Fast Marching Method, an adaption of the venerable Marching Cube method from [55].

4.2.2.1 Experiments

In [56], a method was introduced to enhance level-set segmentation by a regularizing component, which in turn is based on the Mumford-Shah cartoon model described in [57, 58]. Basically, it minimizes the length of the boundary along with the variance of the gray values inside the segmented region. This model will be explained in more detail in section 4.4.3. The results of my experiments are shown in fig. 4.8. As can be seen, the algorithm is highly susceptible to different gray levels in one and the same neuropil, and the Mumford-Shah regularization appears to restrict the segmentation to small regions.

4.2.3 k-means segmentation

Given a set of vectors, the idea of k-means clustering is to assign each vector randomly to one of k clusters. After that, the means of these clusters are calculated. Now, every vector is assigned to the cluster whose mean is closest. The last two steps are iterated until either the average error (usually the average distance between the vector and the mean of its cluster) is small enough, or an iteration limit has been reached. This method of clustering along others has been explained in [59].

K-means segmentation is nothing else than treating each pixel as a vector consisting of its coordinates and its gray value. In practice, this algorithm works amazingly well if the structures do not have a texture.

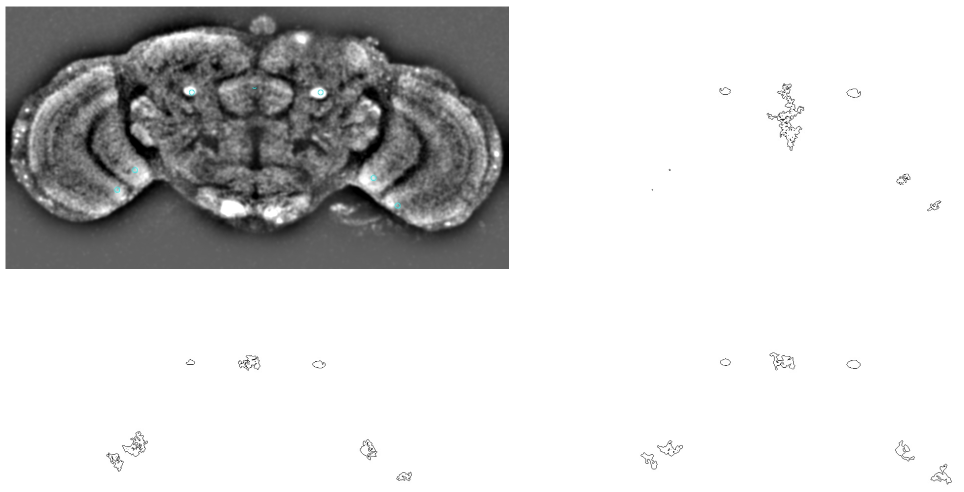


Figure 4.8: Demonstration of the levelset segmentation. Given the seed points (top left), and a despeckled (top right), Gaussian blurred (bottom left) or low pass filtered (bottom right) image respectively, the results of a levelset segmentation are shown.

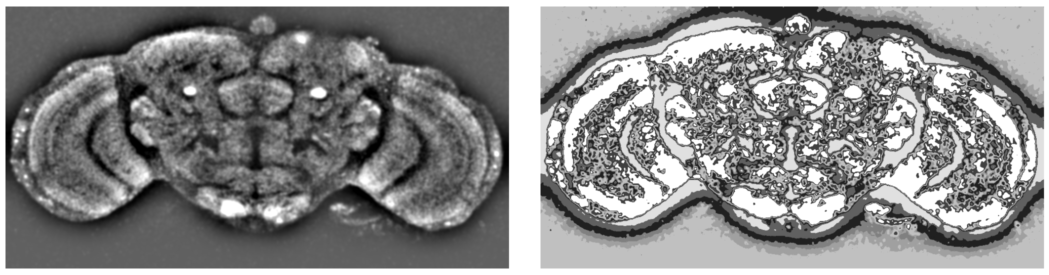


Figure 4.9: Demonstration of the k-means segmentation. After applying a low pass filter (left), k-means segmentation was executed (right).

4.2.3.1 Experiments

As demonstrated in fig. 4.9, the k-means clustering indeed finds clusters. However, they do not coincide with neuropiles. Rather, the regions of similar gray value scattered over the whole image are combined into one cluster.

4.3 Artificial Neural Networks

Artificial Neural Networks (*ANNs*) were first built by Rosenblatt [60], who simulated a single neuron with his *Perceptron* model. It is a simplistic model, which nevertheless achieved good results. Set back by devastating criticism by [61] (the famous *XOR problem*, which is described later on), it took over a decade until the concept was redeemed by the work of Rumelhart in [62].

The fundamental idea is that the activation level of one neuron can be represented by a single number. The neuron is represented by a threshold value: Until the input to the neuron reaches the threshold, the output is zero. When it is greater, the output is calculated by subtracting the threshold from the input.

The key point in Rumelhart's work was to introduce a "hidden" layer of neurons, yielding a three layer model: All input neurons connect to all hidden neurons, and all hidden neurons are connected to all output neurons.To be able to formulate an efficient learning algorithm, the activation is only propagated in one direction, i.e. the hidden neurons do not provide input to the input neurons, and neither do they receive input from the output neurons[10]. Thus, the output connections from one layer are the input connections to the next layer.

4.3.1 Feed forward three-layer networks

When ANNs are displayed, the neurons are usually depicted by circles, whose connections have weights attached (see fig. 4.10). Like the biological neuron, artificial neurons have inbound connections and outbound connections. Continuing the analogy, the signal from the inbound connections is not mapped linearly: Only when a certain threshold is reached, the neuron emits a signal, which depends on its input. It is this feature, which sets the theory of ANNs apart from simple regression analysis, and it is the reason for the robust behavior of ANNs (regression analysis tends to suffer from outliers, which is not the case for neural networks). Since an ANN is meant to calculate values from inputs, there are special "input" and "output" neurons. All other neurons are called "hidden". The neurons are organized into layers, one input layer, one output layer and optionally some hidden layers. For each pair of neurons from adjacent layers, there is a connection with an attached weight.

The process of adjusting these weights so that the network fits a set of input/output examples is called the *learning phase*[11]. Once the learning phase is over, the weights are fixed, and the network is ready to be applied to new input.

The hidden neurons were introduced by [62], because the simple model of [60] proved not to be able to calculate the unlikeliness function (given two inputs with values 0 or 1, the result is 1 if the inputs are different, and 0 if they are equal). This problem is called the *XOR problem*. Experience shows that usually a small number

[10]There are different models with sophisticated algorithms, where feedback loops are allowed. However, they exhibit a substantially higher computational complexity, and are therefore not addressed in this thesis.

[11]This is only true for so-called supervised learning. There are types of neural networks, like the sparse coding networks presented in section 4.3.2, which do not need example outputs. However, these types are not relevant to this section.

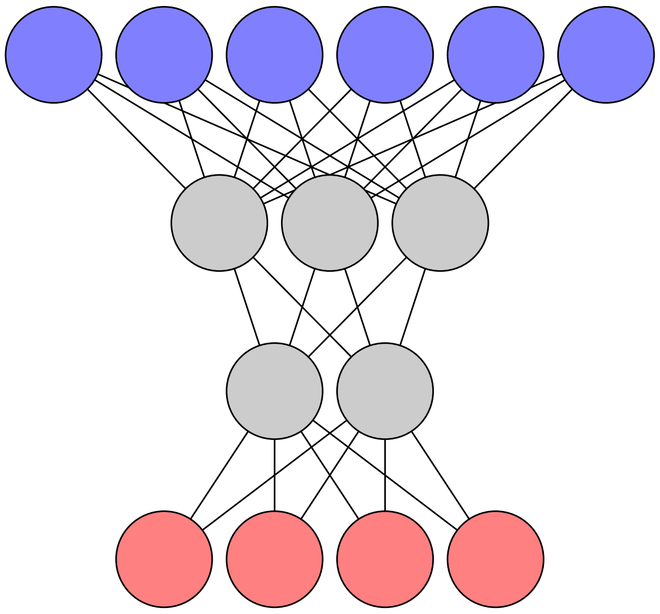


Figure 4.10: The model of an artificial neural network. At the top, the input layer is shown (blue neurons). Two hidden layers process the information, whose neurons are depicted in gray. The output layer is shown at the bottom (red neurons). Usually each neuron has the same activation threshold. All pairs of neurons from adjacent layers have a connection (displayed as a straight line), which has a weight attached to it. Each connection has a direction: The activation level from the input side is propagated to the output side. Thus, the output of the network is obtained by calculating the activation levels of each layer in succession, from input layer to output layer.

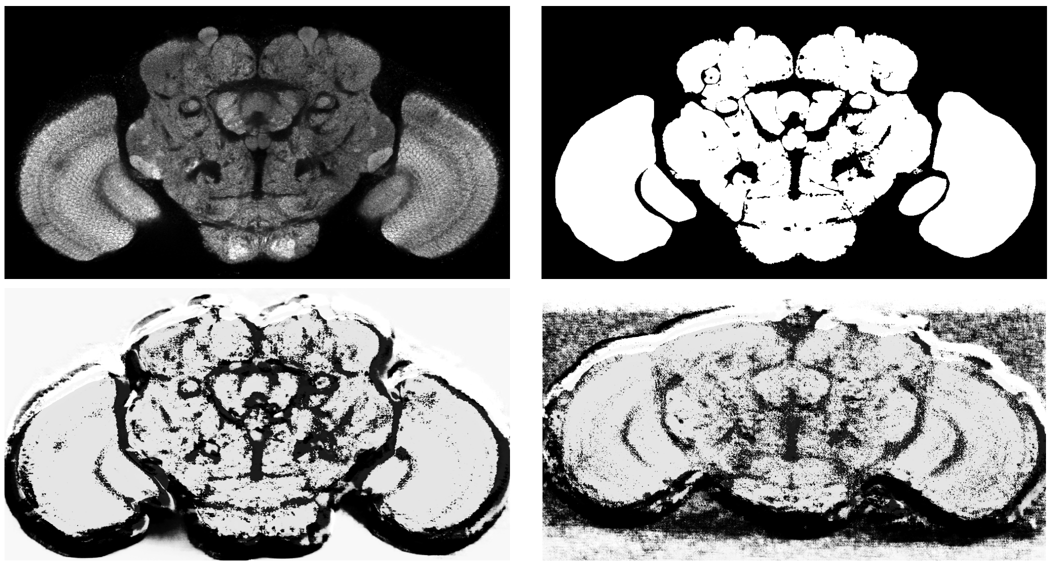


Figure 4.11: Results from training an Artificial Neural Network to recognize neuropil texture. Top left: the original slice. Top right: the neuropil labeling for the original. The ANN was trained on 16104 samples of 61×61 pixels from the original to output 0.9 for neuropil and 0.1 for background. Bottom left: reconstruction of the neuropil label. Bottom right: reconstruction of the neuropil from fig. 4.2 using the same network as for the lower left image.

of neurons in just one hidden layer is often sufficient to learn the desired functions (see [63, 64, 65]).

The most commonly used learning algorithm for networks containing at least one hidden layer is the back propagation algorithm presented in [62]. It works basically like this: Using the current weights and the input example, an output is calculated. From the difference of this output to the desired output, the required change of the weights between the last two layers is calculated. This change is not applied fully, but only multiplied with a factor smaller than one. Then the desired output is back propagated (hence the name of the method) to the last hidden layer using the adjusted weights. Treating the result as desired output of that layer, the same procedure can be applied as if the last hidden layer was the output layer. By iterating over all layers, all weights are "nudged" into the right direction.

4.3.1.1 Experiments

Artificial neural networks have been called "the second best way to do just about anything" (see [66]). When another approach is available, usually it should be preferred. But when one does not get a hold on the problem, it is always possible to train a neural network to do the job. A huge advantage of artificial neural networks is that the same algorithm, the same implementation usually does a good job on a wide range of applications. The downside is that training takes a very long time, and depending on the complexity of the network sometimes also the evaluation of the learned function, i.e. the application of the network to a particular input.

In my experiments I trained an Artificial Neural Network to distinguish neuropil from background. The results are displayed in fig. 4.11. The network was presented patches of 61×61 pixels as input, and as desired output a value telling if the center of that patch is supposed to be neuropil (0.9) or background (0.1). It turned out in my experiments that using a pair 0.1/0.9 resulted in a much faster and more robust learning than the pair 0/1, which seemed more appealing to me at first. The most likely reason is that I use a sigmoid activation function, which never takes on the values 0 or 1, but only approximates those values. By the same token, the gray values were normalized to the interval from 0.1 to 0.9. The bottom row of fig. 4.11 shows the output of the trained network. Centered around each pixel, a 61×61 neighborhood was presented as input, and the output is displayed at that pixel. A bright pixel in the reconstruction means that according to the trained network, the corresponding coordinate is inside a neuropil.

The images demonstrate nicely the biggest advantage of ANNs, which is at the same time their biggest curse: One does not need to specify what the network should learn[12]. For example, the white bands on the top in both reconstructions appear in all reconstructions with that particular network. Likewise, the black bands on the bottom are present in all reconstructions. This anomaly was not a peculiarity of one trained net. When I changed the parameters, or the examples, similar artifacts appeared in the reconstructions.

In summary, it can be ascertained that the trained network mostly succeeds in distinguishing neuropil from background. It should be noted, however, that the misclassifications seem to happen due to either too homogeneous gray value in neuropiles, leading to an output erroneously indicating background, or too variable gray value in the background, which gets labeled as neuropil. This fact gave rise to a few experiments with the mean and variance of the gray values in a small region, which eventually were the motivation for my work presented in chapter 5.

4.3.2 Sparse coding networks

Instead of a matrix (like in section 4.1.1), an image can also be understood as one big vector, whose values are the gray values, and the dimension is the number of pixels of the image. The same principle can still be applied for arbitrary subsets of the pixels. For purposes of image compression, it is common to use small rectangular sections of the image and then make use of similarities between them. Using the vector representation, these similarities can be expressed in terms of linear algebra: Given a (vector) base, the image (or a section thereof) can be represented by a linear combination of this base. Since this mathematical framework deals with vectors, in the following I will call the building blocks *base vectors*, even if they are usually perceived as "patches". The JPEG compression scheme, for example, uses a two-dimensional cosine base. Depending on a quality factor, factors of vectors, which appear to hold less important visual information[13], can be set to zero, thus reducing the file size.

Such codings have more applications than Image Compression: Using the same techniques as before, a priori known patterns can be detected by creating a base consisting of such patterns. When decomposing a section of the image into a linear combination of that base, the larger factors of that linear combination suggest the presence of the corresponding pattern.

When using this approach to detect patterns, it is important that the factors can be interpreted in an easy manner. Nothing is gained by a decomposition which is ambiguous. One method to ensure clear cut detections is to create a so called *sparse base*, i.e. a base where most of the decompositions have only few non-zero factors. That means that the pattern no longer corresponds to one base vector, but to a certain set of base vectors. There are strong indications that the primary visual cortex uses this method to condense visual information prior to processing (see [68, 69]).

[12]According to Murphy's law, the ANN will learn to distinguish features, which one did not expect, but neither cares about (see e.g. [67]).

[13]For example, high frequency signals, which are usually perceived as noise.

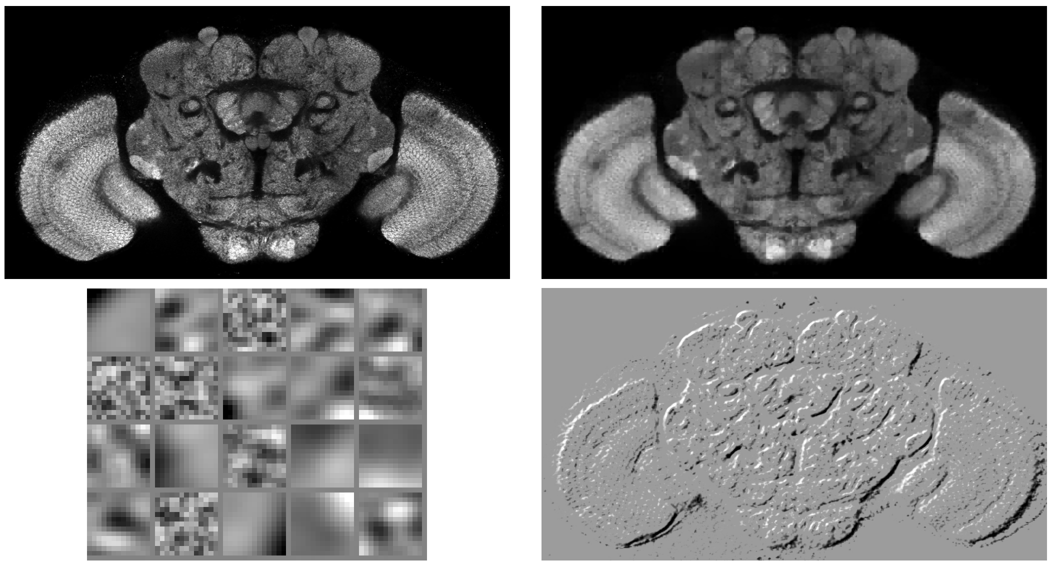


Figure 4.12: Sparse coding of the *Drosophila melanogaster* brain. Random 13×13 samples from the original (top left) were presented during the training phase of a sparse coding network comprising 20 base vectors. The top right image shows a reconstruction using the learned base, which is shown on the bottom left (where the top left patch corresponds to the first base vector, and the lower rightmost patch to the 20th). The factors of the first base vector are shown on the bottom right, where gray denotes 0, dark means negative, and bright positive.

4.3.2.1 Experiments

The approach I used to create such a sparse base was presented in [70]. The basic idea is to introduce a penalty term which punishes a non-sparse linear combination for the current example, i.e. a linear combination where many factors are significantly differing from zero. This penalty term is just the sum of the absolute values of the factors. Since the implementation involves quadratic programming, a rather complicated optimization technique, I refer the interested reader to [70] for details.

My intention was to use the factors to find features: Every base vector represents a visual ”concept” found in the presented examples. More than that, this concept has to be presented a few times, else the training phase will phase that concept out. To find those concepts in the image, one only needs to present each coordinates neighborhood to the sparse coding, and look at the factor of the base vector of the desired feature.

In fig. 4.12, the results are shown from a sparse network, which contains 20 base vectors (patches of 13×13 pixels). During the training, 1.200.000 random patches from the same image were presented to the network. The reconstruction shows that the basic features were indeed learned, and the lower right image shows the locations of the feature represented by the first base vector.

The shown base illustrates how geometric features are recognized: All but 4 base vectors show edge like features, or a dotted pattern. In the reconstruction, which shows how the image looks like according to the (non-complete) base, these features can be recognized. Most of these base vectors no longer appear noisy, even if the training was on the original data, and the base vectors were initialized with white noise[14].

[14]This type of initialization turned out to speed up the convergence to a steady state.

Note that sparse coding networks – just like any Artificial Neural Network approach – can be forced to learn specific features, if this feature can be formalized. For example, if the gradient of the patches should be smooth, this constraint can be put into a term, which favors such patches. To a limited extent, the network can be coaxed towards learning certain features, even if a proper mathematical term is not found. One way is to present cherry picked examples, which display that feature. Another way is to provide additional input in the form of values calculated from the raw input, where the involved functions seem to have a close relationship to the desired feature. In the latter case, the network becomes a so called *functional-link networks* (see [71]). However, both methods introduce a bias towards certain features, which may or may not reflect the data well. Thus, the network is potentially tuned into the wrong direction, not exploiting the full power of ANNs. On the other hand, this supplemental information often can be cast into a straightforward algorithm right from the start. For example, if the goal is segmentation of untextured regions, one should rather make an effort to apply edge detection and tune the parameters, than train an Artificial Neural Network.

4.4 "Atlas-based" methods

An atlas in the context of Image Segmentation consists of a deformable 3D model together with a labeling. When this model is transformed onto a newly recorded data set, the same transformation can be applied to the labeling. The result is a labeling for the data set, and thus it is segmented.

Of course, this only moves the problem of segmentation on to finding sensible mappings between two different data sets based on their gray values.

4.4.1 Elastic registration (based on untransformed gray values)

In section 3.3.3, I presented one method to find a good mapping between two 3D stacks, which is already integrated into the VIB protocol: Amira's Registration module. It optimizes the parameters of a rotation and translation so as to maximize the correspondence between the gray values at each coordinate. Since two different brains have differences in their anatomy, the matching is not optimal, as was seen in fig. 3.5.

Consequently, the idea is to find a mapping which is not rigid, but instead allows minimal deformations. A common method to parameterize such a deformation is to sample the deformation at uniformly distributed points, and interpolate between them. Such an approach, using splines for interpolation, was presented in [72]. Splines are an excellent choice if one expects smooth deformations, i.e. if the local deformations are approximately affine. Also, the computational cost is relatively low as compared to other means, like the diffusion process illustrated in section 3.1.2.

Usually such an elastic registration is accomplished by applying a rigid registration first, and approximating the local deformations from that point on. This is sensible, given that the registration already takes a long time (in the experiments that I performed, the rigid registration took about 12 minutes on a standard PC).

However, there are a few problems with that approach: Since the two images do not exhibit identical gray values, the optimization is highly unstable, often preferring aberrant mappings, because too much importance was ascribed to the gray value, and not so much to the anatomic features. For that reason, regularizing terms were introduced (see for example [73]), which penalize local deviations of the deformation from the average of a larger neighborhood.

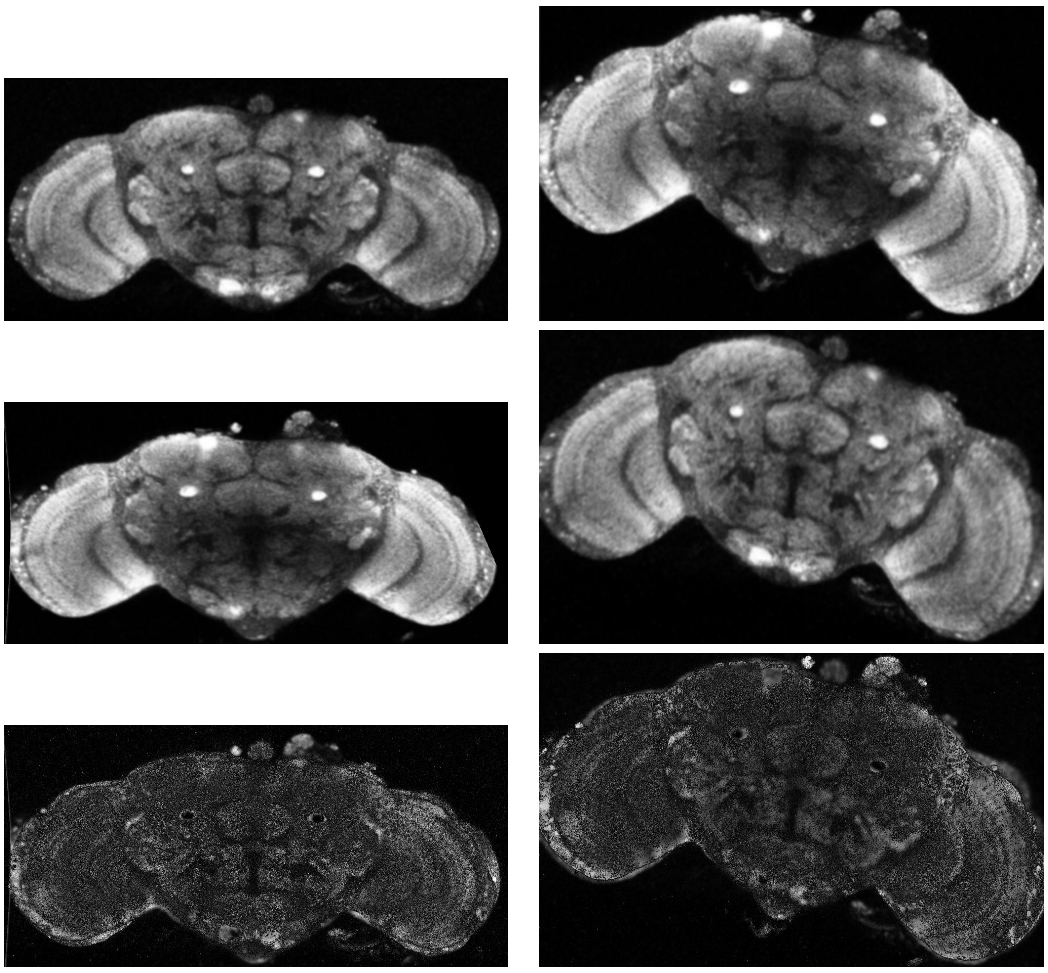


Figure 4.13: Elastic registration (also known as non-rigid registration). Top row: two original slices. Middle row: the results of elastic registration (the left image shows original from top right registered onto top left, and the right image vice versa). Bottom row: differences between the upper two rows.

Other techniques have been proposed for the regularizing: In [74], a soft volume preserving constraint was used, and a generalization of that, using weighted divergence and rotation measures, was implemented in [75].

Especially for confocal recordings, where the noise level is relatively high as compared to e.g. MR images, the choice of the distance between the gray values plays an important role. While in [75], the Euclidean distance of the gray levels was used after a smoothing (see section 4.1.3) was applied, the approach in [74] used a different measure. Instead of using point by point differences of the gray value, the Mutual Information between the gray values and the brain was calculated. The Mutual Information is a powerful measure from Information Theory, which is described in detail in section 2.5.

4.4.1.1 Experiments

In fig. 4.13, the elastic registration is demonstrated. There are obvious differences in the originals, which explain why the match is far from perfect. Still, this is a relatively good match compared to other specimens (compare fig. 4.1). The problem is not the interior of the neuropiles, but their outline. If the outlines are not correct, they have to be relabeled manually (see e.g. the discrepancy along the outlines of the peduncles). The performance of a segmentation algorithm therefore should not be measured in terms of how many voxels were labeled accurately, but instead how much effort has to be put into the manual adjustment of the outlines.

4.4.2 3D Differential Operators

In [76], an overview about 3D differential operators was given. This family of operators are the logical extension of the edge detectors described in section 4.1.1. Instead of using just 2D information, they approximate derivatives in three dimensions, but like the 2D detectors, they use a discrete convolution of neighborhoods for the approximation.

However, these operators are not used to detect edges: Each of the operators described in [76] aspires to detect very specific features. It is the intention to have a massive reduction of brightness to only a few points, which can be reliably detected by the operators. These points can be interpreted as reference points, so called *point landmarks*, which can be matched up in two different recordings. By interpolating a mapping between these landmarks, a complete mapping between the recordings can be constructed. In section 3.3.3, this method of using landmarks for registration was already described (in that section, the landmarks were obtained by calculating the centers of the labeled neuropiles).

4.4.2.1 Discussion of the applicability to confocal recordings and experiments with 2D images

The same unfavorable circumstances as for the edge detectors from section 4.1.1 are still present: If the image is noisy, derivatives can be approximated only in a highly unreliable manner. In particular, the 3D operators do not strive for the detection of edges. Therefore, not just the first derivative has to be approximated, but higher order derivatives. For example, *Rohr3D*, arguably one of the sharpest landmark detectors in [76], uses the determinant of the Hessian matrix. Since approximated derivatives are multiplied, and thus their approximation errors, one can expect bad results from the application to unsmoothed confocal recordings.

As already mentioned, the recording process of confocal microscopy is not isotropic, i.e. the voxels are sampled from a grid with a lateral resolution which is about 5 to 10 times higher than the axial resolution. Therefore, voxels have to be interpreted as average gray values in a cuboid. To calculate a 3D derivative, isotropic samplings of the data are necessary, so the lack of level of detail has to be countered by interpolation. However, the approximation of the 3D derivative does not take the different levels of significance into account. For example, the gradient along an interpolated axis will tell more about the interpolation than about the data.

For that reason, I decided to experiment with 2D images first, and if the results were promising, try to solve the 3D detail problem. The most promising operator from [76] appeared to be Rohr3D, which calculates the determinant of the Hessian matrix, i.e.

$$\begin{vmatrix} \frac{\partial}{\partial x}\frac{\partial}{\partial x} & \frac{\partial}{\partial y}\frac{\partial}{\partial x} & \frac{\partial}{\partial z}\frac{\partial}{\partial x} \\ \frac{\partial}{\partial x}\frac{\partial}{\partial y} & \frac{\partial}{\partial y}\frac{\partial}{\partial y} & \frac{\partial}{\partial z}\frac{\partial}{\partial y} \\ \frac{\partial}{\partial x}\frac{\partial}{\partial z} & \frac{\partial}{\partial y}\frac{\partial}{\partial z} & \frac{\partial}{\partial z}\frac{\partial}{\partial z} \end{vmatrix}$$

The same method can be reduced to 2 dimensions: The Hessian is a 2×2 matrix then, but the principle holds. Since the experiments from section 4.1.1 showed bad results, I applied several denoising techniques first. As can be seen in fig. 4.14,

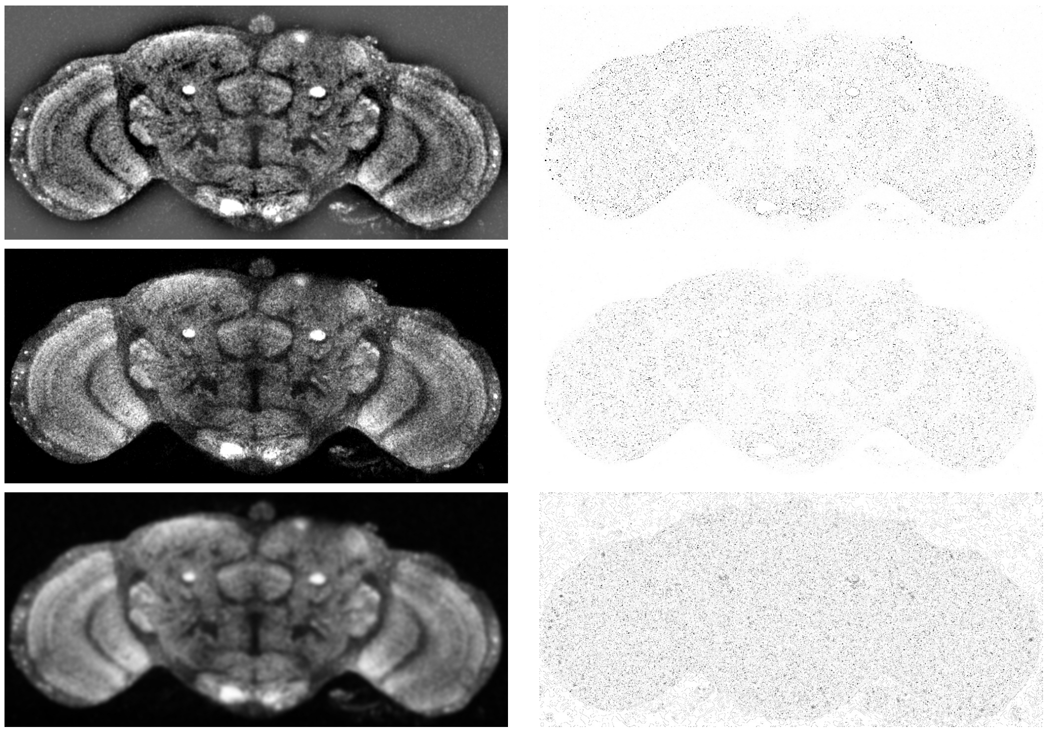


Figure 4.14: The 2D version of the *Rohr3D* differential operator from [76]. It makes no sense to apply such a delicate operator to unsmoothed data, therefore different smoothing algorithms were applied first (left), and then the differential operator (right). Top to bottom: Low pass filter, despeckling and Gaussian blur.

the signal is highly susceptible to differences in the contrast. For example, the two bright regions at the bottom center, which are substructures of the suboesophageal ganglion, are detected nicely (in the form of black regions in the right column). From the view point of anatomy, however, this landmark is a bad choice, because it is highly variable as can be seen from the average image displayed in fig. 3.6.

The results of fig. 4.14 suggest that this approach does not lead to reliable landmarks for the purpose of image registration. There is a lesson to be learned, however. If one aspires to use gray values to calculate a mapping from one brain onto the other, one has to find a way to detect features depending on *a priori* knowledge. A human segmenter, after orienting herself, is able to ignore some distinct structures, which are unimportant for the segmentation. Strictly speaking, the presented edge detection algorithms already use *a priori* knowledge: After all, the basic edge detectors of section 4.1.1 imply the knowledge about typical structures. Nevertheless, for the purpose of segmentation of the brain of *Drosophila melanogaster*, it appears that the algorithm should be improved by providing a means to supply additional a priori knowledge, such as supposed locations of the structures.

4.4.3 Active Contours (”Snakes”)

One problem with the edge detectors described in 4.1.1 was that the detected edges were not closed curves, indeed not necessarily continuous lines. To overcome this, as detailed in 4.2, models were devised, which can only produce closed curves to begin with. However, as shown, in practice these algorithms still detect too many undesired outlines. Therefore, in [57, 58], an algorithm was proposed which takes an initial contour as input in addition to the image, and adapts that contour so that it fits the gray values given by the image.

Similar to anisotropic smoothing (see 4.1.2), instead of the gray values, the contour is set in motion, and large derivatives serve as stop gaps, i.e. the motion decreases with the image gradient. The gradient is now estimated in a different manner, though: The variance of the gray values inside the enclosed region is minimized. Strictly speaking, this is not estimating the gradient, but it is equivalent to it: When the outline moves across a large gradient, the variance of the gray values inside the region increases, and vice versa, provided the image is smooth (if the image is not smooth, it is not sensible to speak of a gradient of the image).

The main improvement in [57] however, is the introduction of a regularizing term. Often, segmentation algorithms like those presented in 4.2 produce very ragged outlines, tribute to the noise. Human segmenters prefer to straighten out the outlines, and so does the regularizing term. It punishes longer outlines by adding the length multiplied with a factor, and the variance of the enclosed gray values to the function which is to be minimized. The resulting function is often referred to as the *Mumford-Shah functional*. By adjusting the factor, it can be controlled how important the smoothness of the curve is in relation to the variance of the gray values.

4.4.3.1 Experiments

In fig. 4.15, this approach is demonstrated. The rough outlines are presented as polygon, and the iterative refinement of the outlines is executed until the Mumford-Shah functional is minimized. The implementation does not use polygons, however, but instead fits a spline curve through the sampling points given by the polygon. Whenever the curve minimizes the functional, new sampling points are introduced by interpolating along the curve, and the process is started again. This is repeated until the resolution of the sampling curve reaches the resolution of the image.

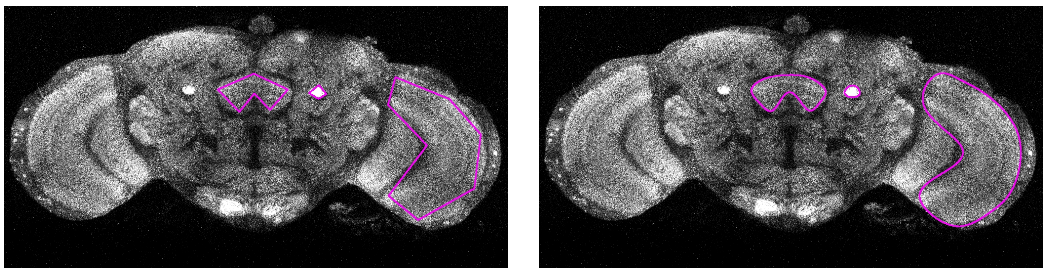


Figure 4.15: The active contours algorithm takes rough outlines of the region to be segmented as input (left) and refines these outlines according to the gray values under a regularizing constraint (right).

This approach can be used to segment newly recorded brains: The idea is to register the newly recorded image rigidly onto the standard brain, and then take the labeled neuropiles of the standard brain to initialize the curves. Thereby, the standard brain serves as atlas. After further adjustment by the active contour algorithm, the outlines are accepted as labelings of the new image.

As is displayed in fig. 4.15, the outline for the peduncle (the circular structure) is accurate. The outlines of the fanshaped body (compare with fig. 3.10) and the medulla are unsatisfactory, though. The reason lies in the functional to be minimized: The structure of these two neuropiles is not uniform. Therefore, dark areas at the boundaries, whose gray value is found inside the structure, too, tend to be merged into the region, especially if the outline becomes shorter by that. This effect is nicely visible at the bottom of the fanshaped body and the left side of the medulla.

4.4.4 Image Understanding

The field of Image Understanding is the logical continuation of Image Processing for the purposes of recognizing objects and their 3D orientations and locations from 2D images. By modeling the 3D objects and the recording process, and optimizing the parameters of these models so that the model fits the recorded image, the most likely orientations and locations can be calculated.

Evidently, the computational cost of this optimization depends on the number of parameters and their respective degrees of freedom: If an object has a texture, the optimization is much more involved than when that object has a uniform color. Because of these complications, researchers in the field of Image Understanding invented several methods to cut down the time needed to calculate the desired parameters.

One recurring concept is that of an *invariant*: When a certain value, calculated from the parameters of an object, does not depend on a certain parameter, it is called invariant to that parameter. For example, if a red ball is depicted on an image, both the outline (a circle) as well as the color (red) are invariant to rotation, but the luminosity is not.

4.4.4.1 Discussion of the applicability to Image Segmentation

While Image Understanding tackles a completely different problem than Image Segmentation, the concepts of one can be put to use in the other. For instance, an edge can be seen as invariant to rotation. The derivative of an image is also invariant to rotation, but not the approximation of it from sampled data.

In this sense, the 3D differential operators introduced in section 4.4.2 detect invariants: The idea is to find landmarks with specificity. Even if I was not able to apply the techniques of Image Understanding to accomplish the task of Image Segmentation, the concept of invariants, together with the ideas from the next section, eventually provided the inspiration for the method I will present in chapter 5.

The main difficulty of deploying the techniques of Image Understanding in the context of the segmentation of the brain of *Drosophila melanogaster*, is that not only the recording process has to be modeled: The staining procedure has a much higher variability. Even though the physical process, namely the diffusion, can be described very well, it is hard to quantify. See fig. 4.1 for differences in staining, which are quite common. Any viable model for the staining process would have to account for those variabilities, which necessarily would further the complexity of the optimization.

4.4.5 Gestalt theory

Ever since psychologists became aware that already at early vision stages, humans are able to distinguish certain shapes better than others, understanding the mechanisms behind that fact has been an active field of research. Commonly referred to as Gestalt theory, this field has applications in a number of areas, including marketing (for example in [77]) and medicine (e.g. [78]).

In Image Processing, Gestalt theory is put to use to compress images: An elementary principle of Gestalt theory is that human vision can selectively ignore noise, i.e. a noisy image of a triangle is still perceived as a triangle. Therefore, images can usually be compressed better after a Fourier transformation was applied, and often the perceived image quality is hardly affected when high-frequency components are filtered (I presented this technique for the purpose of denoising in section 4.1.2).

However, the true power of Gestalt theory in Image Processing unfolds when deducing shapes from sampled data. To get a computational grip on the concept of a Gestalt, i.e. to quantify a shape, in [79], shapes were parameterized by sampling points. Shapes are interpreted as closed curves, and the sampling points are chosen along that curve so that any two neighbors have approximately the same distance. The application of quantitative Gestalt theory permits to reliably compare shapes.

When trying to deduce shapes from 3D stacks, one necessarily introduces yet another layer of quantization errors. As illustrated earlier, image data are never fully specified when working with computers, but rather quantized[15]. When working with recordings from a confocal microscope, in general the resolution is anisotropic, i.e. the quantization is finer in the lateral directions than in the vertical direction. If a surface is fitted to a segmented region, where the region membership is only defined on discrete coordinates on such a grid, it suffers heavily from these anisotropic quantizations, especially when surfaces are to be compared, which stem from specimens recorded in different orientations.

But how to define a 3D shape properly? A shape is commonly perceived as synonymous to a closed surface. A surface can be described by parameterizing it in 2D coordinates, choosing an arbitrary origin and axis. However, the coordinates may not be mistaken for what they are not: It is no longer trivial to calculate the distance between two points on the surface.

Furthermore, when matching shapes, there is the problem of how to define a quality measure of the match. If one were to measure the distance between their surfaces, the calculations needed are complicated, or may be only approximated, depending on the choice of the parameterization of the shapes.

[15]Here, quantization means the process of sampling using a regular grid.

Nevertheless, some studies suggest to create a shape model of the brain, which should then be fitted to a newly recorded brain, thus obviating the segmentation. This task can be accomplished more easily than matching shapes: Testing whether the new (sampled) data have a certain property – like a large gradient – at a certain point on the (deformed) surface is relatively easy. However, one has to keep in mind that to average over points on the surface, one needs to sample these points so that they are equally distributed over the surface. If one can not guarantee at least approximate even distribution, the average will be biased towards the clustered points.

4.4.6 Object detection

Related to Gestalt theory and Image Understanding is the theory of Object detection. However, instead of modeling all aspects from the structure of the specimen to the recording process, Object Detection aspires only to model the outline of certain objects of interest. Thereby the complexity of the processing as well as the complexity of the results is substantially reduced. Gestalt theory comes into play when comparing the model of the object one tries to find with the recognized outlines in the image.

A segmentation method relying on some of these concepts was presented in [80]. There, boundaries of level sets (see section 4.2.2) are filtered, rejecting those with only minimal contrast change perpendicular to their outlines. So far, this approach would rely on the objects having a uniform gray value, at least near their boundaries. However, in [80] the concept of a *local boundary* was introduced. This concept allows the segmentation to coincide with level set outlines only locally, permitting unclosed lines and segmented regions with non-uniform gray values along the outlines. Nevertheless, it is not possible to segment textured regions with this algorithm.

4.4.6.1 Discussion

The application of object detection needs a model of the object to be detected. This may be an exact outline, or a curve (or surface when working on 3D data) with a small set of parameters to be fitted on the image. If the parameters can be further constrained, for example by providing a probability distribution, one can enhance the quality of the object detection by punishing the less likely choices for the parameters, or if an algorithm is used which searches through the parameter space, the values can be ordered according to their probability.

In the context of the VIB protocol, the objects to be detected are the neuropiles of the brain. To find a parameterized model of their outlines, the obvious choice is to find a surface model of the labelings of the standard brain. The goodness-of-fit, while not a completely accurate probability distribution describing the variability of the neuropiles, can still be used to enhance the detection.

Whenever curves or surfaces are to be handled efficiently by a computer program, a common choice is to approximate them by cubic splines. The reason is that splines are piecewise cubic polynomials, thus enabling a speedy execution of the algorithm. Therefore, a spline model for the neuropiles is a good choice. The goodness-of-fit provides an estimate of the probability distribution of the shape, which can be approximated by a normal distribution of the parameters. Since the chosen method is an approximation technique, the parameters are the sampling points of the splines.

4.5 Summary

Looking once again at fig. 4.1, it appears that boundaries of objects have to be detected reliably, before the objects themselves can be detected. All schemes devised to detect known shapes, or shapes which are parameterized, rely in one way or another on recognizing sufficiently many points on the boundary to localize the structures.

Since – as demonstrated in this chapter – none of the tested algorithms appeared to be able to detect reliably edges of structures, I decided to investigate why they fail. Furthermore, I tried to find out why human operators are capable of distinguishing such edges, and to find a model which is adequately reproducing the results, and at the same time can be implemented efficiently as a computer program. The results of this research are presented in the next chapter.

Chapter 5

Information Theory applied to Image Processing

The algorithms described in the previous chapter all failed, for a reason which is obvious when looking at sample images: The provided data differ in significant ways from the images those algorithms were designed for. First, they do not contain homogeneous gray values for pixels belonging to the same neuropil. On the contrary, it is quite common that inside a neuropil, the gray values have a high variance. Second, the gray values obey certain patterns, which are not easily described in a mathematical fashion. Third, the noise observed in our images can not be removed with those algorithms, because it is not pure noise in the classical sense[1]. The ”noise” (noise or features indistinguishable from noise) stems partly from technical sources, like the physical constraints of the method[2], partly from the biological realities[3]. Furthermore, this noise varies with several factors, such as density of the brain.

When analyzing how perception can recognize boundaries between structures, one finds that the repetitive spatial patterns (textures) help to distinguish the neuropiles. If the patterns are regular, i.e. they are repetitious, one can analyze them by Fourier or Wavelet transforms. See for example fig. 3.10 for a semi-regular texture: Even if the bands are overall spanning from left bottom to right top in a smooth slope, the bands consist of structures which are perpendicular to that direction.

Experiments done with monkeys, who seem to perceive visual stimuli similarly to humans (see [81]), suggest that the rotationally invariant classification of textures plays an important role in vision.

The textures found in sections of *Drosophila melanogaster*'s brain have a high variability in size, orientation and regularity. Thus, the search space of texture parameters for an algorithm is huge, and a computational classification would take a long time. A more practical approach is to treat the textures as if only the intensity spectrum mattered, and not also the spatial distribution of the intensity. In practical terms, one can compare textures by comparing intensity histograms of small regions.

The idea to compare histograms leads to Information Theory, the theory which tells how to measure information. For a few years, the idea that statistical methods

[1]The term ”noise” as used in Image Processing usually denotes measurement errors due to the recording process.

[2]For example: The photo-multiplier magnifies single photons' potentials, thus a single photon which misses the pinhole yields a high error in the result.

[3]The optical resolution is not high enough to discern fibers from their environment, resulting in single blips which cannot be distinguished from noise.

might result in good Image Processing techniques, given that the acquisition of images itself can be treated as a statistical process, has become more and more popular. In [82], statistical methods were used to classify textures to segment the images. A more general application of statistics can be found in [83]: Here, statistical methods are used to construct *invariants*, a concept I described in section 4.4.4. See [84] for a good review of such invariants.

A related approach has been used in [85] to classify textures by color distributions. However, the difference between two color distributions was measured by using the *Earth Movers' Distance*. It has the advantage of being easy to calculate, and being well studied. The disadvantage is that nobody can possibly explain how this quantification of the difference should bear any meaning regarding the differences of the textures.

Therefore, in [86] Information Theory was applied to measure the similarity between two probability distributions. Information theory is the right tool in this context, because it is not only important that the measure can tell about good matches. If two pairs of textures appear to be equally different, then the difference measure should reflect that. The Renyi entropy (which was applied in [86]) certainly fulfills this requirement when the pairs match up, but what about (perceived) dissimilar pairs? This requirement can be expressed in mathematical terms, as I will demonstrate later, and Information Theory can be used to find such a measure. In the course of my work, in collaboration with D. Endres (see [87]), I found such a measure which has the nice property of being a metric. In the following sections, also a few shortcomings of the approach in [86] when applied to confocal recordings are addressed.

For the purpose of texture classification, in [88, 89] a method was presented which compares the textures by calculating the χ^2 measure of their gray value distributions. The metric described in [87] has a close relationship to the χ^2 measure, which explains the results of [88].

In this chapter, I apply Information Theory and Bayesian Inference (see chapter 2 to edge detection on confocal recordings of brains of *Drosophila melanogaster*. Using the framework provided by Information Theory, I then discuss the quality of this approach. In the second part, I illustrate how this method can be enhanced by prior knowledge, which is not embedded in the algorithm, but is instead provided as additional data, in order to produce robust segmentations.

5.1 Mutual Information applied as Edge Detection

In order to distinguish textures regardless of their scale and orientation, I chose to compare the distributions of their gray values. For the sake of clarity, let's name the textures: One gets the label "outside", the other "inside". The idea is now to ask the question: How much can one learn about the texture label when only looking at the gray value distribution? The answer is given by the Mutual Information between texture label and gray value. The higher this Mutual information is, the more one can learn about the texture label given the gray value distributions. Conversely, given the texture label, the mutual information indicates how much one knows about the difference between the gray value distributions. It can thus be employed as a measure of dissimilarity between textures.

During my research, I became aware of a work which is similar in motivation, presented in [86]. While – as in my work – information-theoretic ideas were used, I chose to take a slightly different path: It became obvious already at an early stage, that the choice of the information-theoretic measure is critical to the result. If one were to choose just about any differential measure between two distributions, the results would not tell anything about the differences of the textures. For example, the Euclidean distance (which interprets finite distributions as vectors) would punish small differences in the gray value as much as big differences. In [86], the *Renyi entropy* has been used. In mathematical terms, Renyi's entropy is a generalization of Shannon's entropy, because the latter is a special case of the former. However, in statistical terms, Renyi's approach is a restriction, because that entropy measure is only sensible, if the random variables, for which it is computed, are statistically independent. If the textures are similar, then this condition is not fulfilled. If the textures are dissimilar, the Renyi entropy is only sensible in the limit, where it equals the Shannon entropy. Therefore it is not surprising that one of the results of [86] is that to obtain favorable outcomes, the parameter of the Renyi entropy should be chosen so as to equal the Shannon entropy.

The measure D^2_{pq} investigated in [87] is defined by

$$D^2_{pq} \quad = \quad \sum_i \left(p_i \log \frac{2p_i}{p_i + q_i} + q_i \log \frac{2q_i}{p_i + q_i} \right) \tag{5.1}$$

Its motivation has a close relationship to the question of how different the textures are. In fact, it is very similar to the Mutual Information: Suppose that samples are drawn from one of two given distributions, but it is not known which. The measure D^2_{pq} is the result when inferring the average information gain by one sample about which distribution it is. In other words, it answers the question how much one gray value tells about the class it is taken from. N.B.: Why is the measure written as a square? Because $D_{pq} := \sqrt{D^2_{pq}}$ is a metric, its square is not.

Since no Bayesian Inference was applied in [86], that method needs a lot of samples in order to obtain a good estimate of the distributions. Therefore, the whole inside has to be compared to the whole outside[4]. Implicitly, this method assumes that the structure has a homogeneous texture. That may be true for MR images of the human brain, but it is certainly not true for confocal recordings of the brain of *Drosophila melanogaster* (see fig. 3.10).

Instead of the Renyi entropy, I decided to use the Mutual Information, since that measure was successfully used for similar purposes: In [90], Mutual Information between multi spectral images was used to denoise the images. Also, the method described in section 4.4.1 can use the Mutual Information between the gray value and what brain it came from, to measure the agreement of a mapping.

To detect edges, at each coordinate the hypothesis is tested, that the boundary between inside and outside runs through that coordinate. This test needs two regions, the inside and outside. Since the direction of the edge is not known beforehand, I test all directions. At each coordinate and for each direction, I therefore chose to divide a circular region around that coordinate by an axis in that direction into two regions. For each pair of regions obtained in that manner, the Mutual Information between the gray value and the texture label is calculated, and the maximum is calculated. In the following, I refer to this measure as the *edge information* at that coordinate. It denotes how much information in the data support the hypothesis that a textured edge is present at that coordinate. This algorithm is illustrated in fig. 5.1.

As can be seen in this figure, a major complication is the usage of histograms as estimates of probability distributions. There is always a trade-off involved: I am interested in the distribution on that particular coordinate, but only one value is observed at that point. To construct a histogram, it is therefore assumed that in

[4]The Renyi entropy as well as the Shannon entropy allow to compare more than two distributions, and therefore it is possible to segment more than one structure. However, this is not important for the point I am trying to make here.

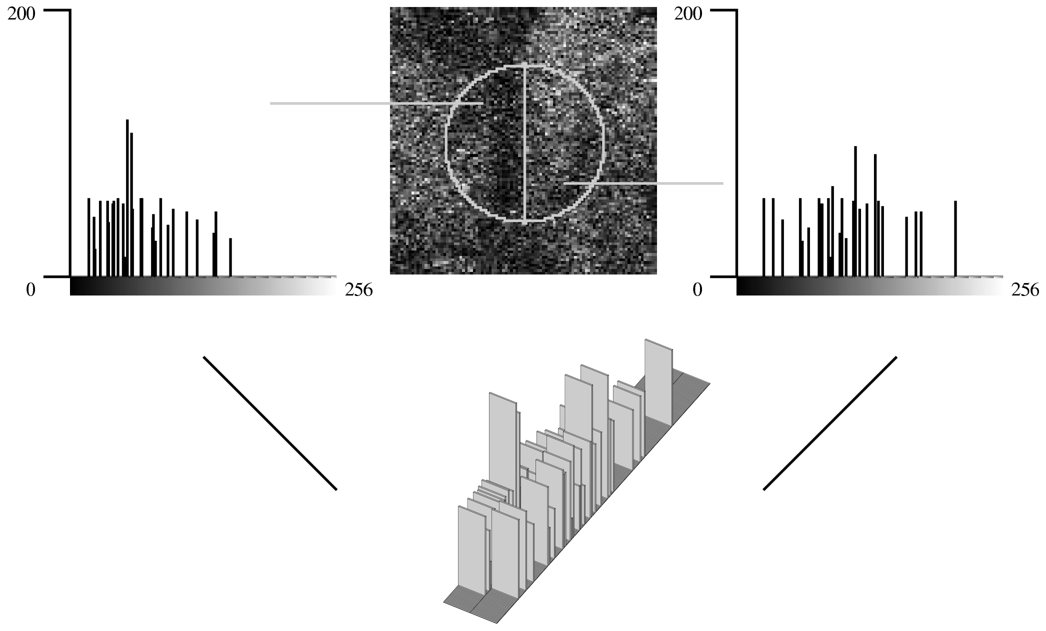


Figure 5.1: The hypothesis, that an edge is visible at a certain point and angle, can be tested by defining half-circular regions around the hypothesized edge, constructing the joint histogram between the gray values and the side relative to that edge, and then inferring the Mutual Information between gray value and region membership. The higher this Mutual Information is, the more visible is the hypothesized edge.

a spatial neighborhood, the distribution is very similar, and values from a region around the coordinate of interest are counted. This elicits three problems:

- A precise localization is not possible when a lot of pixels are taken into account,
- the assumption that the distribution of adjacent pixels is similar can be totally wrong (for example when an edge goes through the region),
- and the estimate of the distribution degrades when only a few values are counted.

As a remedy for the first and the third problem, I chose to use Bayesian inference. Instead of directly using the histograms as if they actually were the underlying distributions, the Mutual Information is inferred. This approach makes it feasible to extract meaningful measures from limited data. (the same observation motivated the use of Information Theory in [70, 91]). As will be shown in section 5.1.3, the application of Bayesian Inference actually makes a huge difference. As for the second problem, the method is to test each direction through that coordinate and take the maximal value of the Mutual Information as edge information. Since the Mutual Information will be low, when the proposed boundary does not coincide with a boundary, that direction will not be considered when another direction is supported by the data.

When two adjacent structures have gray values which are constant inside a structure, then the edges detected by the presented method coincide with those detected by Canny's method (see 4.1.3) regardless of the chosen approximation to the derivative. However, when using the Mutual Information as edge measure, it is not important how different these two gray values are, whereas Canny and related algorithms prefer strong differences in the gray value.

A big advantage of the Mutual Information as edge detector over most detectors presented in the last chapter is that it is quite robust to the addition of noise. The reason is that spatial relationships are ignored to the extent that permutations of the gray values in the neighborhood of a coordinate do not change the Mutual Information. Algorithms depending on an approximation of derivatives of the image do not have this advantage.

Since the inference of the Mutual Information involves some rather complicated mathematics, it is presented in appendix A. The calculation of the inferred Mutual Information is relatively cheap in computational terms: It involves a sum with one operand for each gray value, and partial harmonic series. For practical purposes, the partial harmonic series can be stored in a small lookup table, so that the complexity is linear in the number of gray values.

Originally, I tried to infer the metric presented in [87], because of the nice metric properties as compared to the Mutual Information. The properties of a metric are well suited to formulate an approximation procedure. As will be shown in section 5.2, this is a desirable setting for segmentation: It is easier to approximate a closed contour by adjusting the parameters of a closed curve than to join disjointed edges. However, it proved hard to apply Bayesian Inference to that metric. However, in the limit, i.e. when the histograms approach the true distributions, and assuming that the number of points in the histograms grow equally, i.e. that the two regions being compared have the same size, the inferred Mutual Information becomes

$$< I > \rightarrow \frac{1}{2} D_{pq}^2 .$$

The calculations to prove this are carried out in appendix B. Since each histogram represents a probability distribution (which – in theory – could be the true distribution), the following corollary holds: Given two discrete probability distributions $P_i(X = x), i = 1, 2$ for a random variable X, which are equally likely the true distribution of X (i.e. $P(i = 1) = P(i = 2) = \frac{1}{2}$), the square root of the Mutual Information between X and i is a metric between P_1 and P_2.

5.1.1 Bayesian Rebinning

To further enhance the outcome of the Mutual Information based edge detection, a rebinning process can be applied. The idea of rebinning is that the reliability of the inference of the expectation of the Mutual Information depends highly on the ratio of bins (the number of gray values) to the number of samples. The error decreases as the number of samples increases in relation to the bin number. Luckily, there is a sensible way to reduce the number of bins, because noise and sampling errors of a gray value histogram are not uniform. The gray values are located on a natural scale, and deviations are much more likely be local on that scale. Therefore, it can be reasonable to join adjacent gray values into bins.

Indeed, a very simple rebinning into just two bins is quite popular: the *thresholding*. By splitting the pixels at a certain gray value threshold, one obtains a classification in dark and bright pixels. For a survey on finding an optimal threshold value, see [92].

For indexed color images, i.e. images containing only a very limited number of colors (typically at most 256 colors), algorithms were invented to reduce arbitrary images to a small number of colors. These algorithms are called *quantization* in Image Processing (see e.g. [93]). If the original image contains only gray values, then this quantization achieves a rebinning of the image.

In [94], however, a fast algorithm was presented for rebinning any histogram, applying Bayesian inference with a uniform prior. This algorithm uses a clever scheme to reuse precomputed values to reduce the exponential complexity to cubic complexity. While it still is slower than the classical quantization algorithms, its outcome is more accurate.

Furthermore, in [94], again applying Bayesian inference, an algorithm is given to infer the optimal number of bins, which still has cubic complexity. These two algorithms[5] constitute a parameter free rebinning algorithm.

5.1.2 Calculating a quality measure

Whenever a new algorithm is proposed, the authors claim that it is better than all algorithms hitherto invented. While claiming the same, I go one step further: The tools which were used to formulate the edge detection can be used to calculate a goodness of these results: Not only the Mutual Information, but also its standard deviation is inferred. In the latter case, there is no easily computable closed form, but instead a good approximation for the variance was found in [95], which I present in appendix C.

When the standard deviation of the Mutual Information is small compared to its value, it can reasonably be assumed that different values of the inferred Mutual Information indeed contain information about the differences in textures present in the image. Further, it gives an idea how much better the results could possibly be: When the standard deviation is relatively high, so is the uncertainty about the true Mutual Information. In that case one can only hope to find out more prior information, which then can lead to a better approximation of the true value.

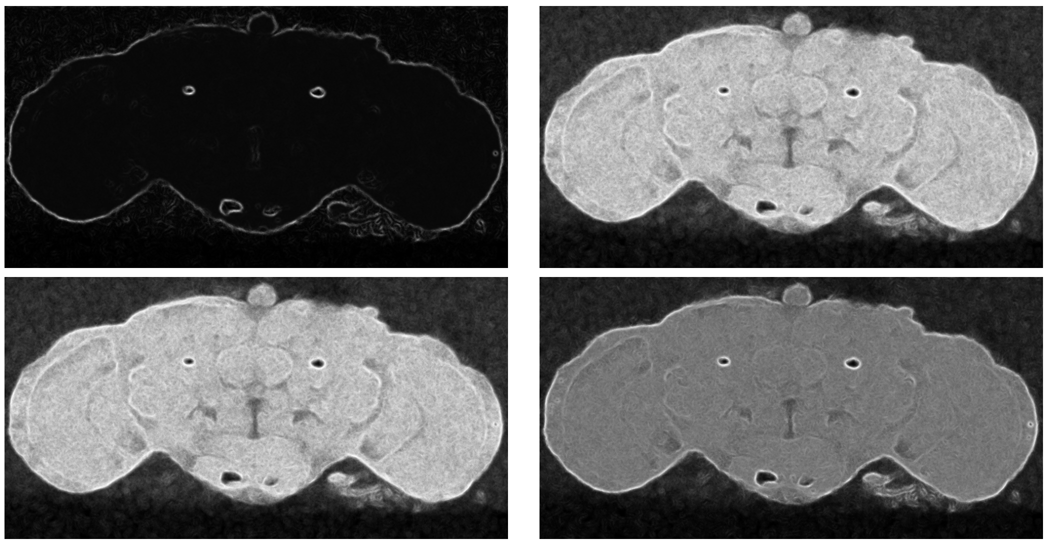


Figure 5.2: A comparison between four histogram difference measures for edge detection. The histograms are constructed as described in fig. 5.1, and the difference measure is depicted as a gray level at the corresponding coordinate. Top left: the Euclidean distance. Top right: the metric presented in [87]. Bottom left: the Mutual Information. Bottom right: the inferred expectation of the Mutual Information.

5.1.3 Experiments

In fig. 5.2, four measures are compared:

- the Euclidean distance, $|p - q| = \sqrt{\sum_i (p_i - q_i)^2}$,
- the metric from [87], $D_{pq} = \sqrt{\sum_i \left(p_i \log \frac{2p_i}{p_i+q_i} + q_i \log \frac{2q_i}{p_i+q_i} \right)}$,
- the Mutual Information, $I = \frac{1}{2} \sum_{j=1}^{N} \left(p_{1j} \log \frac{2p_{1j}}{p_{1j}+p_{2j}} + p_{2j} \log \frac{2p_{2j}}{p_{1j}+p_{2j}} \right)$,
- and the inferred expectation of the Mutual Information (see appendix A).

It turns out that the Euclidean distance produces very sharp indications of edges, but at the same time looses most visually perceivable boundaries. The metric and the Mutual Information, which are closely related (see appendix B), yield comparable results, while the Bayesian inference boosts the performance of the latter.

In fig. 5.3, the results of the edge detection presented in fig. 5.2 (lower left) are repeated after Bayesian rebinning (see section 5.1.1). The improvement over the edge detection without rebinning is substantial.

Note that Bayesian rebinning can be understood as a Bayesian optimal pixel wise denoising. This is somewhat counterintuitive when looking at the left column of fig. 5.3. However, from a pixel wise instead of a spatial perspective, there are now less errors to be expected: When adding (or multiplying) noise to a gray value, the original value and the noisy value are more likely to be in the same bin than not. Still, information (in terms of entropy) is reduced by the rebinning. Nevertheless, one can expect that the entropy stemming from the original data is much larger than the entropy of the noise, else a human operator could not make sense of the image

[5]In fact, these two algorithms are just variations of the same underlying scheme.

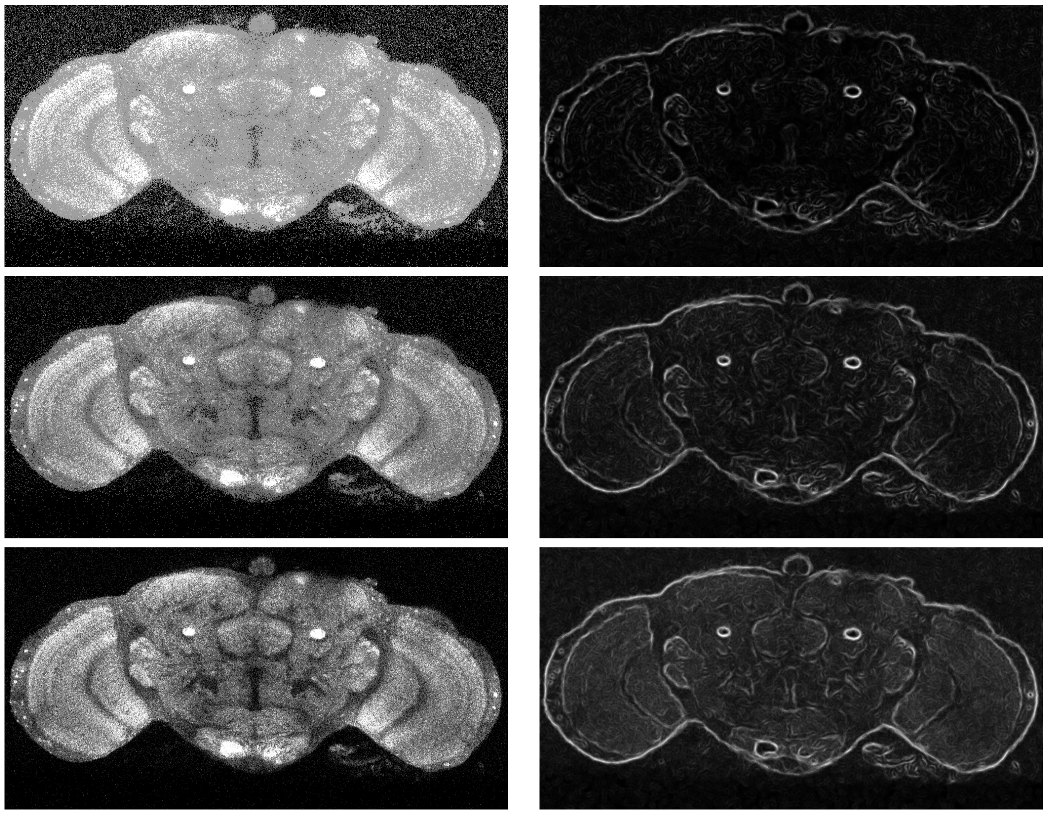


Figure 5.3: Demonstration of Bayesian rebinning. In the left column, the rebinned images are displayed. For the sake of clarity, the maximal gray value of the bin is displayed at each pixel. In the right column, the Mutual Information based edge detection is displayed. From top to bottom: 4 bins, 12 bins and 30 bins.

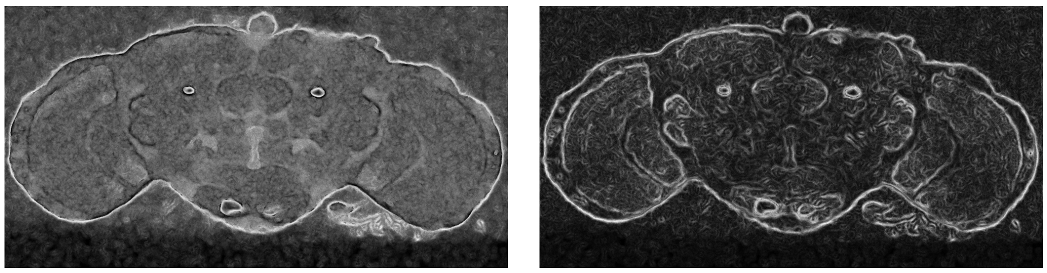


Figure 5.4: The inferred variance of the expectation of the Mutual Information. Left: this image corresponds to the lower left image in fig. 5.2, which was not rebinned. Right: this image corresponds to the middle row of fig. 5.3, which was rebinned to 12 gray levels before the edge detection was applied.

either. Therefore, the rebinning process typically results in a significant reduction of noise, while retaining most of the desired signal.

The quality measure described in section 5.1.2 is demonstrated in fig. 5.4. Dark gray values correspond to a low variance of the expectation, while bright gray values denote a high variance. Note that the standard deviation is well below 10% of the inferred expectation of the Mutual Information at each pixel. This means that the error involved in the inference is noticable (as one should expect by the small number of samples.), albeit small enough that the results can be deemed robust.

It should also be noted that the estimated variance is uniformly lower after Bayesian rebinning than before. The relevant edges are still detected – a strong hint that the Bayesian rebinning indeed succeeded to reduce the per-coordinate noise.

5.2 Towards automatic segmentation

The method described in the last section was shown to detect edges in the absence of smooth gradients. Nevertheless, this approach still uses the paradigm of edge detectors, which still has the problems described in section 4.4.2: There may be structures present, which are highly distinguishable, but they are nevertheless unimportant for the segmentation of the brain.

As solution to this problem, in the last chapter I proposed to use the information already available in the form of the standard brain. It consists not only of recordings of particular high quality, it includes also a complete segmentation thereof, and – last, but certainly not least – it features an estimate of the variability of the neuropiles. As mentioned in section 4.4.6, this information can be transformed into a shape model of the neuropiles of the standard brain. Furthermore, a deformation model can be conceived, which takes the expected variability into account.

A shape model, as illustrated in section 4.4.5, has severe shortcomings if used for the purpose of registration, and both brains are represented as shapes rather than 3D stacks. However, if only one of them is parameterized as a shape model, but the other still remains in the original form, one can bypass these problems. Even better, the idea to match a shape model onto a 3D stack has one big advantage over the registration of two 3D stacks: A shape model, which is given as a smooth surface, does not have the transformation error attached, that haunts all mappings of 3D stacks. There are no average gray values to be sampled, but instead coordinates. Therefore, no sampling errors occur, and the interpolation method does not have a bias towards one rotation or another. Therefore, rigid transformations do not affect the accuracy of a shape model. Since the optimal mappings between two

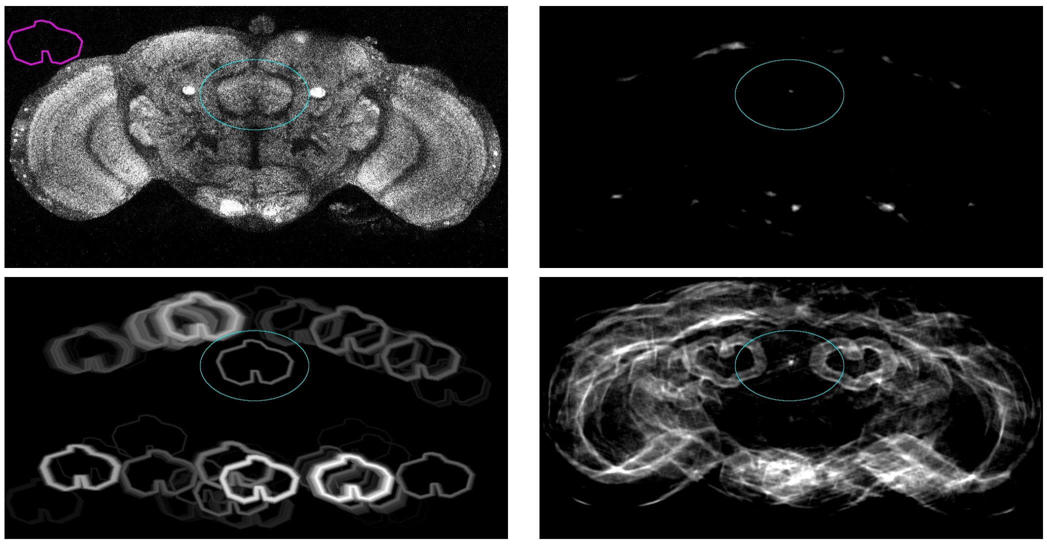


Figure 5.5: Applying the edge detection to object detection. Top left: the original slice, and the object to be found (a very rough outline of the fanshaped body). Top right: the edge information for each coordinate to be the center of the shape (brighter is more likely than darker). Bottom left: the same information, visualized as the shape. Bottom right: the edge information for each coordinate to be the center, when using the isotropic edge detector. The ellipse designates the location of the fanshaped body.

brains ideally do not imply large aberrations from the rigid transformation, the same advantage holds to a certain extent for non-rigid transformations.

5.2.1 Using Mutual Information to localize objects

When a shape is given a priori in the form of a surface model, the following idea can be applied to find out if this shape is present at a certain location: Given an approximately uniform sampling of the coordinates on the surface, at each of these coordinates it can be tested, to what extent the data supports the hypothesis that an edge is present at that point. The outcome of this test can be interpreted as edge information, i.e. the higher the edge information is, the more does the data suggest the presence of an edge. By summing[6] these edge informations over all sampling points, one obtains the edge information for the whole shape. Thus, by maximizing the edge information through adjustments to the location at first, and then the shape parameters, the shape can be localized in the 3D stack.

To improve that scheme in terms of accuracy as well as speed, the surface gradients can be sampled as well as the surface points. The edge detection by means of the Mutual Information is just a hypothesis testing algorithm, therefore the information about the gradient speeds up the algorithm. Furthermore, by requiring the edge to be detected in that angle, the fidelity of the match should improve (this was tested and found to be true in the context of Elastic Registration using manually specified landmarks with attributes in [96]).

5.2.2 Experiments

In fig. 5.5, results from a 2D experiment with the fanshaped body are shown. First, a rough outline was defined. A not completely matching outline of a shape which is hard to detect in the gray image was chosen on purpose, to demonstrate the strengths and weaknesses of my approach.

As described, for each coordinate, the rough shape was centered on that coordinate, along the outlines the edge information of an edge in the direction of the outline at that point were calculated, and the sum of these values was the result for that coordinate.

In the upper right image, the edge information is displayed for each coordinate, that the given outline centered on that coordinate would be justified by the data in the original image. Note that there is a single dot right on target, indicating that the algorithm had no problem identifying the fanshaped body.

However, the algorithm tries to tell us that there are hints for more than one fanshaped body. This is visualized in the lower left image, which contains the same information as the upper right image, but instead of a dot, whose intensity corresponds to the edge information of the shape, the shape itself is displayed with that intensity. Again, it can be seen that the fanshaped body is nicely detected at the correct place. But it becomes evident that the object localization fails at the outlines of the whole brain, detecting the shape where it is not. However, by comparing the misdetections with the original image, one finds that parts of the shape are indeed present. For example, roof-like outlines – just like the upper part of the fanshaped body – are in plain view where the optic lobes, the central brain and the outside touch. As explained, the algorithm does not care, if the neuropil is on the wrong side of the outline. Instead, boundaries between different textures are sought.

The lower right image displays the result when applying the same calculation as for the upper right image, only that this time, the edges are not tested with a specific direction. In other words: From the middle right image of fig. 5.3, at each coordinate the intensities along the shape centered around this coordinate where accumulated, and visualized. As one would expect with this approach, it is very sensitive to differences in the edge signal. As demonstrated in fig. 5.3, while still being highly distinguishable, the detected edges have by no means the same edge information. This leads to the somewhat funny heart-like structure, which really is just the input shape turned by 180 degrees. This is due to the strong signal of the peduncles (the small, bright circular structures in fig. 5.3), which single-handedly outweighs all other signals: Even just a few values from the peduncle along the shape are as bright as the edge signal from the fanshaped body. Still, in a neighborhood of the fanshaped body, the detection is accurate.

As demonstrated by the isotropic approach, and less so by the anisotropic approach, the edge detection still does not rely only on visible differences of texture, but is highly affected by brightness differences. Along the outline of the whole brain, this difference is very large, and therefore, misdetections occur mostly there. However, as I showed, in a quite large neighborhood of the true location the detection achieves a high degree of reliability.

These results are promising, as they show that objects can be detected reliably, just by restricting the search to a sensible area. Nevertheless, a few simple extensions to the algorithm should be investigated, such as ways to incorporate the variance of the edge information along the outline, multi resolution approaches, and methods to take intensities or textures into account, too.

[6]Since it is not known *a priori* if the shape is present at that location, each sampling point denotes an independent test.

5.2.3 Future plans: Deformation using thin-plate splines

To give a parameterized model of a neuropile's surface, it could be specified using thin-plate splines. These were used successfully for the purpose of elastic registration (see e.g. [96, 74, 75, 72]), so it seemed the appropriate choice. Thin-plate splines are the two-dimensional continuation of one dimensional cubic splines. Both types of splines are motivated by the same setup: An elastic material (like splines in shipbuilding) is clamped tautly on a set of support points. Since the material is elastic, the tension of the material is distributed linearly. Therefore, the points along the spline can be given as a piecewise cubic polynomial.

Now, the parameters of a spline approximation of a surface are the coordinates of the sampling points. Using these sampling points, the deformation can be parameterized by the deformation vectors which begin at these sampling points. In less mathematical terms, the deformation is done by moving each sample point a little bit, and interpolating the motion between the sample points using a two dimensional cubic spline. The effect is the same as modifying the coordinates of each sample point directly. However, it is much easier to assess the amount of deformation, and to regularize it, when the deformation is itself modeled as a vector field which is added to the vector field of the surface model. For example, a good regularization term might be the sum of the lengths of the deformation vectors at the sampling points. A perfect match would then correspond to no deformation, i.e. the regularization term would be zero.

This concept can be turned into a segmentation algorithm, which automatically segments a newly recorded *Drosophila melanogaster* brain. To this end, the parameters of the surface model described above have to be calculated from the existing standard brain. The implementation of the algorithm used in section 5.2.2 has to be extended to work with 3D data. The algorithm from section 5.1.2 has to be adapted, so that it assesses the quality of the segmentation. This information can be used to review the areas potentially needing manual relabeling. Finally, these algorithms have to be integrated into, and called from the VIB protocol.

Chapter 6

Conclusions and Outlook

In this thesis, I presented the Virtual Insect Brain protocol, a framework which facilitates the use and application of the standard brain of *Drosophila melanogaster*. It was shown to provide a consistent user interface on top of Amira. The protocol relieves the anatomist of many recurring tasks, such as keeping track of the different files depending on each other, and processing steps still necessary to finalize the processing.

It is easily extensible, by making a library of Tcl procedures available to Amira scripts, which then integrate seamlessly into the protocol. This was demonstrated by extending it to operate on multichannel data, and by offering new methods for registration.

Despite the name, this protocol is by no means limited to insect brains. It can be effortlessly applied to any type of brain, or even other anatomical structures. Wherever a standardization process similar to the creation of a standard brain is needed, the VIB protocol is likely to be of value.

After the VIB protocol, I presented a few approaches to enhance the registration process by automating parts of the segmentation, or by avoiding it altogether. I demonstrated, that these approaches do not live up to their claims, often requiring a complete manual relabeling.

I introduced Information Theoretical considerations to improve edge detection techniques and Atlas-based segmentation methods. The shown examples demonstrate that indeed, this algorithm finds edges which coincide with the boundaries a neuro-anatomist would agree to. Furthermore, the algorithm was enhanced to provide an estimate of its performance.

The proposed edge detection algorithm itself represents a novel way to cope with noisy images and textured regions. Indeed, it is one of the first of a whole family of robust Image Processing algorithms, which employ the natural concept of Information Theory.

By the results, one can expect that the extension to 3D data exceeds the quality of the results in 2D. This opens the door for massive parallel processing of recordings. Since a performance measure can be calculated at the same time, imperfect labelings can be flagged for inspection.

With this powerful tool, it becomes feasible to build a huge catalog of existing Gal4 lines along with quantified expression patterns, permitting fast querying of said lines by neuro-anatomical features. For instance, a researcher could look up all Gal4 lines whose expression patterns show a high signal in the antennal lobe, but a negligible signal in the peduncle.

This catalog could further be integrated with the standard brain of *Drosophila melanogaster*, by automatically localizing the expression patterns.

Appendix A

Bayesian inference of the Mutual Information from experimental data

Let M be the number of gray values (for 8-bit gray images, $M = 256$), $l_1, .., l_M$ and $m_1, .., m_M$ the histograms of inside and outside respectively. Further, let both regions contain the same number N of samples, i.e. $\sum_{i=1}^{M} l_i = \sum_{i=1}^{M} m_i = N$.

These two histograms can be merged into a joint histogram $J_{c,i}$ between the gray value and the class membership: $J_{inside,i} := l_i$ and $J_{outside,i} := m_i$.

Of course, this histogram, when normalized, is only an estimate of the probability distribution. The true distribution is of the form $\hat{p}_{side,i} \in [0;1], side = inside, outside, i = 1, .., M$, where $\sum_{side,i} p_{side,i} = 1$. To enhance the readability, in the following I write $p_j := \begin{cases} p_{inside,j} & \text{for } j \leq M \\ p_{outside,j-M} & \text{for } M < j \leq 2M \end{cases}$ and similarly J_j instead of $J_{side,i}$.

I now use the Bayesian approach to infer the probability $p(\{p_j\}|D)$ of one such model $\{p_j\}$:

$$\begin{aligned} p(\{p_j\}|D) &= \frac{P(D|\{p_j\})p(\{p_j\})}{P(D)} \\ &= \frac{P(D|\{p_j\})p(\{p_j\})}{\iint_{\{p_j\}} P(D|\{p_j\})p(\{p_j\})} \end{aligned} \tag{A.1}$$

The integral $\iint_{\{p_j\}}$ is meant to be carried out over all possible models $\{p_j\}$, i.e. all tuples of values $p_j \in [0;1]$ where $\sum_j p_j = 1$.

Since I do not want to put an emphasis on any particular distribution, I treat them as equally likely. Thus the term $p(\{p_j\})$ is constant. Consequently this factor can be canceled out.

The probability for the data given the model is:

$$P(D|\{p_j\}) = (J_1, J_2, ..., J_{2N})! \prod_j p_j^{J_j} \tag{A.2}$$

where $(J_1, J_2, ..., J_{2N})!$ is the multinomial coefficient, telling the number of permutations of the samples, which still show the same histogram $\{J_j\}$. Putting A.1 and A.2 together, one obtains

$$p(\{p_j\}|D) \quad = \quad \frac{(J_1, J_2, ..., J_{2N})! \prod_j p_j^{J_j}}{\iint_{\{p_j\}} (J_1, J_2, ..., J_{2N})! \prod_j p_j^{J_j}} \tag{A.3}$$

The multinomial coefficient does not depend on the values $\{p_j\}$, so it can be canceled out. I now carry out the integration in the denominator of A.3:

$$\begin{aligned}\iint_{\{p_j\}} \prod_j p_j^{J_j} \quad &= \quad \int_0^1 p_1^{J_1} \int_0^{1-p_1} p_2^{J_2} \cdots \\ &\int_0^{1-\sum_{j<2M-1} p_j} p_{2M-1}^{J_{2M-1}} \cdot p_{2M}^{J_{2M}} \; dp_{2M-1} \cdots dp_2 dp_1 \end{aligned} \tag{A.4}$$

By substituting $s := 1 - \sum_{j<2M-1} p_j$, $x := p_{2M-1}$, $a := J_{2M-1}$ and $b := J_{2M}$, the probability p_{2M} is equal to $s - x$, and the innermost integral becomes

$$\int_0^{1-\sum_{j<2M-1} p_j} p_{2M-1}^{J_{2M-1}} \cdot p_{2M}^{J_{2M}} \; dp_{2M-1} \quad = \quad \int_0^s x^a (s-x)^b \, dx$$

Further substitution of $x := sy$ reveals the equality

$$\begin{aligned}\int_0^s x^a (s-x)^b \, dx \quad &= \quad \int_0^1 s^{a+b+1} y^a (1-y)^b dy \\ &= \quad s^{a+b+1} B(a+1, b+1)\end{aligned} \tag{A.5}$$

where $B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ is the *Beta function*, and the *Gamma function* is defined by $\Gamma(x) = \int_0^1 e^y y^{x-1} dy$. A nice feature of the Gamma function is that it is an extension of the factorial $n! := \prod_{i=1}^n i$ to the whole complex plane[1](see [97]):

$$\Gamma(n+1) = n!$$

Therefore, at positive integer values of x, the Gamma function can be evaluated easily. Furthermore, the Gamma and Beta functions have been studied extensively, so that calculations may be based on well known results of those studies.

Putting back A.5 into A.3 yields a multiple integral of the same form as in A.3, reduced by one integral, and the term $p_{2M}^{J_{2M}}$ is replaced by the factor $s^{a+b+1} = p_{2M-1}^{J_{2M-1}+J_{2M}+1}$. Just the factor $B(a+1, b+1) = \frac{J_{2M-1}! J_{2M}!}{(J_{2M-1}+J_{2M}+1)!}$ is new, and can be put in front of the integral.

By iterating this procedure, all integrals in A.4 can be evaluated. The result is:

$$\begin{aligned}\iint_{\{p_j\}} \prod_j p_j^{J_j} \quad &= \quad \frac{J_1!(2M-2+\sum_{j>1} J_j)!}{(2M-1+\sum_j J_j)!} \cdots \\ &\cdots \frac{(J_{2M-2})!(J_{2M-1}+J_{2M}+1)!}{(J_{2M-2}+J_{2M-1}+J_{2M}+2)!} \cdot \frac{(J_{2M-1})! J_{2M}!}{(J_{2M-1}+J_{2M}+1)!} \\ &= \quad \frac{\prod_j (J_j)!}{(2M-1+\sum_j J_j)!}\end{aligned} \tag{A.6}$$

[1]Indeed, the Gamma function is completely characterized by the equation $\Gamma(x+1) = x\Gamma(x)$ and the convexity on R^+.

Applying this result to A.1 yields:

$$p(\{p_j\}|D) = \frac{(2M-1+\sum_j J_j)!}{\prod_j (J_j)!} \prod_j p_j^{J_j} \tag{A.7}$$

I want to infer the expectation of the Mutual Information $I(side, i) = H(side, i) - H(side) - H(i)$, where H is the entropy, i.e.

$$I(side, i) = -\sum_j p_j \log p_j + \sum_{j \le M} \hat{p}_j \log \hat{p}_j + \sum_{side} \hat{p}_{side} \log \hat{p}_{side}$$

where $\hat{p}_j = p_j + p_{M+j}$ is the marginal probability for the gray value j, and likewise for $\hat{p}_{side}$. To calculate this expectation, the following integral has to be calculated:

$$\begin{aligned} < I(side, j) > &= \iint_{\{p_j\}} I(side, j) p(\{p_j\}|D) \\ &= \iint_{\{p_j\}} \left(-\sum_j p_j \log p_j + \sum_{j \le M} \hat{p}_j \log \hat{p}_j - \sum_{side} \hat{p}_{side} \log \hat{p}_{side} \right) \\ &\quad \cdot \frac{(2M-1+\sum_j J_j)!}{\prod_j (J_j)!} \prod_j p_j^{J_j} \end{aligned} \tag{A.8}$$

This integral can be written as a sum of integrals in the form of A.4 and integrals which contain an additional factor $\log p_{\hat{j}}$ for a particular $\hat{j} \in 1, .., 2M$.

I now calculate the latter integrals: Using the same technique as before, the problem is reduced to the integral

$$\begin{aligned} \int_0^1 \log x \cdot x^a (1-x)^b dx &= \int_0^1 \frac{\partial}{\partial a} x^a (1-x)^b dx \\ &= \frac{\partial}{\partial a} \int_0^1 x^a (1-x)^b dx \\ &= \frac{\partial}{\partial a} B(a+1, b+1) \\ &= \frac{\partial}{\partial a} \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \\ &= \frac{\Gamma'(a+b+2)\Gamma(a+1)\Gamma(b+1) - \Gamma(a+b+2)\Gamma'(a+1)\Gamma(b+1)}{\Gamma^2(a+b+2)} \\ &= \frac{\Gamma(a+1)\Gamma(b+1)}{\Gamma(a+b+2)} \left(\frac{\Gamma'(a+1)}{\Gamma(a+1)} - \frac{\Gamma'(a+b+2)}{\Gamma(a+b+2)} \right) \end{aligned} \tag{A.9}$$

Note that the Weierstraß product form (see [97]) of the Gamma function is

$$\Gamma(x) = \frac{1}{x} e^{-\gamma x} \prod_{i=1}^{\infty} \frac{e^{\frac{x}{i}}}{1 + \frac{x}{i}}$$

By differentiating the logarithm of both sides, one obtains

$$\frac{\Gamma'(x)}{\Gamma(x)} = -\gamma - \frac{1}{x} + \sum_{i=1}^{\infty} \left(\frac{1}{i} - \frac{1}{x+i} \right)$$

Using this result in A.9 leads to

$$\begin{aligned}\int_0^1 \log x \cdot x^a(1-x)^b dx &= \frac{a!b!}{(a+b+1)!}\left(-\gamma - \frac{1}{a+1} + \sum_{i=1}^{\infty}\left(\frac{1}{i} - \frac{1}{a+i+1}\right) + \right. \\ &\quad \left. +\gamma + \frac{1}{a+b+2} - \sum_{i=1}^{\infty}\left(\frac{1}{i} - \frac{1}{a+b+i+2}\right)\right) \\ &= \frac{a!b!}{(a+b+1)!}\left(-\frac{1}{a+1} + \frac{1}{a+b+2} - \sum_{i=a+2}^{a+b+2}\frac{1}{i}\right)\end{aligned}$$

Note that the integrals including terms of the form $\log(p_{j_1} + p_{j_2} + .. + p_{j_K})$ can be reordered in such a manner that the innermost integral looks like this:

$$\iint_{\{p_{j_k}\}, \sum_j p_j = 1} \log\left(\sum_k p_{j_k}\right) \cdot \prod_k p_{j_k}^{J_{j_k}}$$

The argument of the logarithm is therefore $\sum_k p_{j_k} = 1 - \sum_{j \notin \{j_1,\ldots,j_K\}} p_j$, thus constant, and the logarithm can be factored out from the integrals. The integration then goes forward as in A.4.

After a lengthy, nevertheless straightforward summation of these integrals, one obtains the following formula for the inferred expectation of the Mutual Information:

$$\begin{aligned}< I(side, j) > &= G_{2(M+N)} - G_{M+N} - \frac{1}{4N}\sum_{j=1}^{N}\big((n_{1j}+1)(G_{n_{1j}+n_{2j}+2} - G_{n_{1j}+1}) + \\ &\quad +(n_{2j}+1)(G_{n_{1j}+n_{2j}+2} - G_{n_{2j}+1})\big) \qquad \text{(A.10)}\end{aligned}$$

where

$$G_n := \sum_{i=1}^{n}\frac{1}{i} \qquad \text{(A.11)}$$

Appendix B

The connection between the inferred Mutual Information and the metric D_{pq}

The Mutual Information can be inferred for joint histograms of any size, i.e. the same calculation as in appendix A can be carried out for a histogram given by the values $n_{ij} \in \{0, 1, 2, ...\}$, where $i = 1, .., M, j = 1, .., N$. This is in contrast to what was assumed so far, namely that one quantity denotes the two classes *inside* and *outside*, which are equally likely, or in mathematical terms: $M = 2$ and $P(i = 1) = P(i = 2) = \frac{1}{2}$. For this special case, the Mutual Information can be viewed as a distance measure between the probability distributions of the classes $i = 1$ and $i = 2$.

Indeed, the inferred Mutual Information approximates half the square of the metric presented in [87]. This is demonstrated in the following:

When $N \to \infty$, the bin counts divided by the total counts approximate the true probabilities ($\frac{n_{1j}}{N} \to p_{1j}$ and $\frac{n_{2j}}{N} \to p_{2j}$). Further, using G_n as defined in A.11 note that

$$\lim_{n\to\infty} G_n - log(n) = \gamma$$

where $\gamma \approx 0.5772156$ is the Euler-Mascheroni constant, and therefore

$$\lim_{n\to\infty} G_{mn} - G_n = \log m$$

The Mutual Information then becomes:

$$\begin{aligned}
<I> &= G_{2(M+N)} - G_{M+N} - \frac{1}{4N}\sum_{j=1}^{N}\big((n_{1j}+1)(G_{n_{1j}+n_{2j}+2} - G_{n_{1j}+1}) + \\
&\qquad +(n_{2j}+1)(G_{n_{1j}+n_{2j}+2} - G_{n_{2j}+1})\big) \\
&\to \log(2) - \frac{1}{2}\sum_{j=1}^{N}\left(p_{1j}\log\frac{p_{1j}+p_{2j}}{p_{1j}} + p_{2j}\log\frac{p_{1j}+p_{2j}}{p_{2j}}\right) \\
&= \frac{1}{2}\sum_{j=1}^{N}\left(p_{1j}\log\frac{2p_{1j}}{p_{1j}+p_{2j}} + p_{2j}\log\frac{2p_{2j}}{p_{1j}+p_{2j}}\right) \\
&= \frac{1}{2}D^2_{p_1p_2}
\end{aligned}$$

This relationship holds only in the limit $N \to \infty$, however. In order to infer the metric itself, the integral

$$< D_{p_1 p_2} > \quad = \quad \iint_{\{p_j\}} \sqrt{\sum_j \left(p_{1j} \log \frac{2p_{1j}}{p_{1j} + p_{2j}} + p_{2j} \log 2p_{2j} p_{1j} + p_{2j} \right)} \prod_j p_{1j}^{n_{1j}} p_{2j}^{n_{2j}}$$

would have to be carried out, under the further constraint that $\sum_j p_{1j} = \sum_j p_{2j} = \frac{1}{2}$. To my knowledge, this integral has not yet been successfully computed.

Appendix C

Inference of the variance of the Mutual Information

In this appendix, I assume a general joint histogram, namely $n_{ij} \in \{0, 1, 2, ...\}$ for $j = 1, .., M$ and $i = 1, .., N$. In the following, the naïve estimate of the probability is denoted by $\hat{p}_{ij} := \frac{n_{ij}}{n}$, where $n := \sum_{ij} n_i j$. As in appendix A, I write "the inference" when I mean the inference given a uniform prior, i.e. $p(\{p_{ij}\})$ is assumed to be constant.

As shown in [95], the variance of the inferred Mutual Information can be approximated by

$$\frac{1}{n} \sum_{ij} \frac{n_{ij}}{n} \left(\log \frac{n_{ij} n}{n_{i \cdot} n_{\cdot j}} \right)^2 - \frac{1}{n} \left(\sum_{ij} \frac{n_{ij}}{n} \log \frac{n_{ij} n}{n_{i \cdot} n_{\cdot j}} \right)^2 + O(n^{-2}) \tag{C.1}$$

where $n_{i \cdot} := \sum_j n_{ij}$, $n_{\cdot j} := \sum_i n_{ij}$, and $O(n^{-2})$ denotes the *Landau symbol*, meaning that the error of this approximation is bounded by a constant times n^{-2}. The error term decreases with the second power of the number of samples.

The method used in [95] to derive this formula is elegant, and I will outline it here: The Mutual Information is interpreted as a function which maps the vector $\{p_{ij}\}$ to its corresponding Mutual Information. By expanding this function into a Taylor series around the expected distribution $< \{p_{ij}\} >= \{\hat{p}_{ij}\}$, the following result is obtained:

$$\begin{aligned} \mathrm{Var} I(i,j) &= < (I(i,j) - < I(i,j) >)^2 > \\ &= \underbrace{< \left(\sum_{ij} \log \left(\frac{\hat{p}_{ij}}{\hat{p}_{i \cdot} p_{\cdot j}} \right) \Delta_{ij} \right)^2 >}_{=:F_{ij}} + O(n^{-2}) \end{aligned}$$

where $\Delta_{ij} := p_{ij} - \hat{p}_{ij}$. The first summand can be evaluated further:

$$F_{ij} = \sum_{ijkl} \log \frac{\hat{p}_{ij}}{\hat{p}_{i \cdot} p_{\cdot j}} \log \frac{\hat{p}_{kl}}{\hat{p}_{k \cdot} \hat{p}_{\cdot j}} \mathrm{Cov}(p_{ij}, p_{kl})$$

To infer the variance of the Mutual Information, the following integral has to be carried out:

$$< \mathrm{Var} I(i,j) > \quad = \quad \iint_{\{p_j\}} \mathrm{Var} I(i,j) p(\{p_{ij}\}|D) \tag{C.2}$$

Since the term F_{ij} approximates the variance of the Mutual Information, one can substitute $F_{ij}+O(n^{-2})$ for $\mathrm{Var}I(i,j)$ in the integral to approximate the inferred variance of the Mutual Information. The error term is to be treated as a constant, which can be factored out. The integral then looks similar enough to the integral in A.8, that the same techniques can be applied as in appendix A. The final result is stated above.

In [98, 99] an exact formula was given for the inferred Mutual Information. Since it uses the confluent hyper-geometric function of the first order, which is best avoided when it comes to computational cost, I did not use it in my research. Due to the fact that the definition of the confluent hyper-geometric function alone would go beyond the scope of the present work, I will not even repeat the result of [99].

Appendix D

Lebenslauf

Persönliche Daten:	Johannes E. Schindelin Hans-Löffler-Str. 20a 97074 Würzburg geb. 15.01.1973
Schulausbildung:	1979-1983 Grundschule Zellingen
	1983-1988 Mozartgymnasium Würzburg
	1988-1992 Röntgengymnasium Würzburg Abschluss: Abitur
Zivildienst:	07/1992-11/1992 Rudolf-Alexander-Schröder-Haus Würzburg
	12/1992-09/1993 Arbeiter-Samariter-Bund in Würzburg
Studium:	10/1993-07/1999 Universität zu Würzburg Hauptfach Mathematik, Nebenfach Informatik Abschluss: Diplom
Promotion:	seit 04/2001 bei Prof. Dr. M. Heisenberg am Lehrstuhl für Genetik und Neurobiologie Biozentrum Würzburg
	Wissenschaftlicher Mitarbeiter der Universität Würzburg im Zeitraum 04/2001-03/2005 im Rahmen des BMBF Forschungsverbundes "Virtual Brain"

Bibliography

[1] Nick Strausfeld. *Atlas of an Insect Brain.* Springer-Verlag, Berlin, Heidelberg, New York, 1976.

[2] Armstrong, J. D. and Kaiser, K. and Müller, A. and Fischbach, K.-F. and Merchant, N. and Strausfeld, N. J. Flybrain, an on-line atlas and database of the drosophila nervous system. *Neuron*, 15:17–20, 1995.

[3] Heisenberg, M. and Kaiser, K. The Flybrain Project. *Trends in Neurosciences*, 18:418–483, 1995.

[4] Guimond, A. and Meunier, J. and Thirion, J. P. Average brain models: A convergence study. *Computer Vision and Image Understanding*, 77:192–210, 2000.

[5] Virtual Neuro Lab. http://www.virtual-neurolab.org/.

[6] Karlheinz Rein, Malte Zöckler, Michael T Mader, Cornelia Grübel, and Martin Heisenberg. The Drosophila standard brain. *Curr Biol*, 12(3):227–231, Feb 2002.

[7] W.S. Rasband. ImageJ. http://rsb.info.nih.gov/ij/, 1997–2005.

[8] G. M. P. van Kempen, H. T. M. van der Voort, and van Vliet L. J. A quantative comparison of two restoration methods as applied to confocal microscopy. In *ASCI'96, Proceedings of the second Annual Conference of the Advanced School for Computing and Imaging*, pages 196–201. Advanced School for Computing and Imaging, June 1996.

[9] J. B. (editor) Pawley. *Handbook of Biological Confocal Microscopy 2nd ed.* Plenum Press, New York, 1995.

[10] AH Brand and N Perrimon. Targeted gene expression as a means of altering cell fates and generating dominant phenotypes. *Development*, 118(2):401–415, 1993.

[11] Tzumin Lee and Liqun Luo. Mosaic analysis with a repressible cell marker for studies of gene function in neuronal morphogenesis. *Neuron*, 22(3):451–461, 1999. TY - JOUR.

[12] Tzumin Lee and Liqun Luo. Mosaic analysis with a repressible cell marker (marcm) for drosophila neural development. *Trends in Neurosciences*, 24(5):251–254, 2001. TY - JOUR.

[13] John K. Ousterhout. Scripting: Higher-level programming for the 21st century. *Computer*, 31(3):23–30, 1998.

[14] T. Bayes. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 1763.

[15] E. T. Jaynes. *Probability theory: the logic of science.* Cambridge University Press, New York, 2003.

[16] J. Talairach and P. Tournoux. *Co-planar stereotaxic atlas of the human brain: 3-dimensional proportional system: An approach to cerebral imaging.* Thieme, New York, 1988.

[17] A W Toga and P M Thompson. Maps of the brain. *Anat Rec*, 265(2):37–53, Apr 2001.

[18] Zöckler, M. and Rein, K. and Stalling, D. and Brandt, R. and Hege, H.-C. Creating Virtual Insect Brains with Amira. *Report of Zuse Institut Berlin*, 2001.

[19] J. Ashburner and K.J. Friston. Spatial normalization. In A.W. Toga, editor, *Brain Warping*, pages 27–44. Academic Press, 1999.

[20] A. W. Toga and P. Thompson. The role of registration in brain mapping. *Image Vision Comput.*, 19:3–24, 2001.

[21] C. G. Galizia, S. L. McIlwrath, and R. Menzel. A digital three-dimensional atlas of the honeybee antennal lobe glomeruli based on optical sections acquired using confocal microscopy. *Cell and Tissue Research*, 295:383–394, 1999.

[22] Robert Brandt, Torsten Rohlfing, Jürgen Rybak, Sabine Krofczik, Alexander Maye, Malte Westerhoff, Hans-Christian Hege, and Randolf Menzel. A three-dimensional average-shape atlas of the honeybee brain. *Journal of Comparative Neurology*, .(.):(in press), 2005.

[23] Stuart Feldman. Make–a computer program for maintaining computer programs. *Software-Practice and Experience*, 9(4):255–265, 4 1979.

[24] Evan L. Ivie. The programmer's workbench – a machine for software development. *Commun. ACM*, 20(10):746–753, 1977.

[25] D. Wagh. personal communication, 2002.

[26] M. T. Mader. Analyse von Expressionsmustern in den Pilzkörpern von Drosophila melanogaster. Diplomarbeit, University Würzburg, 2001.

[27] J. E. Hochberg and E. McAllister. A quantitative approach to figural 'goodness'. *J. Experimental Psychol.*, 46:361–364, 1953.

[28] J. E. Hochberg. Effects of the Gestalt Revolution: The Cornell Symposium on Perception. *Psychological Review*, 64(2):73–84, 1957.

[29] D. G. Kendall. A Survey of the Statistical Theory of Shape. *Statistical Science*, 4(2):87–120, 1989.

[30] F. L. Bookstein. Size and Shape Spaces for Landmark Data in Two Dimensions. *Statistical Science*, 1(2):181–242, 1986.

[31] T. R. Reed and H. Wechsler. Segmentation of Textured Images and Gestalt Organization Using Spatial/Spatial-Frequency Representations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(1):1–12, 1990.

[32] M R Kaus, S K Warfield, A Nabavi, P M Black, F A Jolesz, and R Kikinis. Automated segmentation of MR images of brain tumors. *Radiology*, 218(2):586–591, Feb 2001.

[33] Edward A Ashton, Chihiro Takahashi, Michel J Berg, Andrew Goodman, Saara Totterman, and Sven Ekholm. Accuracy and reproducibility of manual and semiautomated quantification of MS lesions by MRI. *J Magn Reson Imaging*, 17(3):300–308, Mar 2003.

[34] David E Rex, Jeffrey Q Ma, and Arthur W Toga. The LONI Pipeline Processing Environment. *Neuroimage*, 19(3):1033–1048, Jul 2003.

[35] David T. Gering, Arya Nabavi, Ron Kikinis, W. Eric L. Grimson, Nobuhiko Hata, Peter Everett, Ferenc A. Jolesz, and III William M. Wells. An integrated visualization system for surgical planning and guidance using image fusion and interventional imaging. In *MICCAI '99: Proceedings of the Second International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 809–819. Springer-Verlag, 1999.

[36] David T. Gering. Automatic segmentation of cardiac mri. In *MICCAI (1)*, pages 524–532, 2003.

[37] H A Drury, D C Van Essen, C H Anderson, C W Lee, T A Coogan, and J W Lewis. Computerized mappings of the cerebral cortex: a multiresolution flattening method and a surface-based coordinate system. *J Cogn Neurosci*, 8(1):1–28, 1996.

[38] D C Van Essen, H A Drury, J Dickson, J Harwell, D Hanlon, and C H Anderson. An integrated software suite for surface-based analyses of cerebral cortex. *J Am Med Inform Assoc*, 8(5):443–459, Sep 2001.

[39] A. M. Dale, B. Fischl, and M. I. Sereno. Cortical surface-based analysis. i: Segmentation and surface reconstruction. *NeuroImage*, 9(2):179–194, 1999.

[40] E. Busa. *FreeSurfer Manual.* Massachusetts General Hospital, Boston, Massachusetts, 2002.

[41] Rein, K. and Zöckler, M. and Heisenberg, M. A quantitative three-dimensional model of the *Drosophila* optic lobes. *Curr. Biol.*, 9:93–96, 1999.

[42] Rein, K. and Hiesinger, P. and Zöckler, M. and Kirsten, J. and Fischbach, K.-F. and Heisenberg, M. Three-dimensional reconstruction of the *Drosophila* larval and adult brain. Flybrain (http://www.flybrain.org), 2000.

[43] E. Argyle. Techniques for edge detection. *Proc. IEEE*, 59:285–286, 1971.

[44] Thomas Martin Lehmann, Claudia Gönner, and Klaus Spitzer. Survey: Interpolation methods in medical image processing. *IEEE Trans. Med. Imaging*, 18(11):1049–1075, 1999.

[45] M. Unser. Splines: A perfect fit for signal and image processing. *IEEE Signal Processing Magazine*, 16(6):22–38, November 1999. IEEE Signal Processing Society's 2000 Magazine Award.

[46] A. Bijaoui, J.-L. Starck, and F. Murtagh. Restauration des images multi-échelles par l'algorithme à trous. *Traitement du Signal*, 11:229–243, 1994.

[47] E P Simoncelli. Bayesian denoising of visual images in the wavelet domain. In P Müller and B Vidakovic, editors, *Bayesian Inference in Wavelet Based Models*, chapter 18, pages 291–308. Springer-Verlag, New York, 1999. Lecture Notes in Statistics, vol. 141.

[48] M. Motwani, M. Gadiya, R. Motwani, and F. Harris. A Survey of Image Denoising Techniques. In *Proceedings of GSPx 2004*, pages 27–30, Santa Clara, CA, September 2004. Santa Clara Convention Center.

[49] J Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[50] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *The International Journal of Computer Vision*, 1(2):167–187, May 1987.

[51] J. Weickert. *Anisotropic Diffusion in Image Processing*. Dissertation, University of Kaiserslautern, Faculty of Mathematics, 1996.

[52] D. Tschumperle and R. Deriche. Vector-valued image regularization with pde's: A common framework for different applications. In *In IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[53] Lee Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. *IEEE PAMI, 1991*, 13(6):583–598, 1991.

[54] James Albert Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge Monograph on Applied and Computational Mathematics. Cambridge University Press, 1999.

[55] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[56] L. Guigues, H. Le Men, and J. Cocquerez. Scale-sets image analysis. In *ICIP03*, pages II: 45–48, 2003.

[57] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *Proc. IEEE Conf. Comp. Vis. Pattern Recognition*, 1985.

[58] D. Mumford and J. Shah. Optimal approximations by piecewise smooth functions and associated variational problems. *Comm. Pure Appl. Math.*, 42:577–685, 1989.

[59] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[60] F Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[61] M Minsky and S Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, Massachussets, 1969.

[62] D Rumelhart and J McClelland. *Parallel Distributed Processing*. MIT Press, Cambridge, Massachussets, 1986.

[63] Y. Ito. Approximation of Continuous Functions on R^n by Linear Combinations of Shifted Rotations of a Sigmoid Function with and without Scaling. *Neural Networks*, 5(1):105–115, 1992.

[64] V. Kurkova. Kolmogorov's Theorem and Multilayer Neural Networks. *Neural Networks*, 5(3):501–506, 1992.

[65] H. J. Sussmann. Uniqueness of the Weights for Minimal Feedforward Nets with a Given Input-Output Map. *Neural Networks*, 5(4):589–593, 1992.

[66] John Hertz, Richard G. Palmer, and Anders S. Krogh. *Introduction to the Theory of Neural Computation.* Perseus Publishing, 1991.

[67] E.B. Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1(1):151–160, 1989.

[68] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[69] William E. Vinje and Jack L. Gallant. Sparse coding and decorrelation in primary visual cortex during natural vision. *Science*, 287(5456):1273–1276, February 2000.

[70] D. Endres. *Bayesian and Information-theoretic Tools for Neuroscience.* PhD thesis, School of Psychology, University of St Andrews, St Andrews, UK, 2005.

[71] Yoh-Han Pao and Yoshiyasu Takefuji. Functional-link net computing: Theory, system architecture, and functionalities. *IEEE Computer*, 25(5):76–79, 1992.

[72] Richard Szeliski and James Coughlan. Spline-Based Image Registration. *International Journal of Computer Vision*, 22(3):199–218, March/April 1997.

[73] Karl Rohr, Mike Fornefett, and H. Siegfried Stiehl. Approximating thin-plate splines for elastic registration: Integration of landmark errors and orientation attributes. In *IPMI '99: Proceedings of the 16th International Conference on Information Processing in Medical Imaging*, pages 252–265. Springer-Verlag, 1999.

[74] Torsten Rohlfing, Robert Brandt, Calvin R. Maurer, Jr., and Randolf Menzel. Bee brains, B-splines and computational democracy: Generating an average shape atlas. In Lawrence Staib, editor, *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, pages 187–194, Kauai, HI, 2001. IEEE Computer Society, Los Alamitos, CA.

[75] C.Ó Sánchez Sorzano, M. Blagov, P. Thévenaz, E. Myasnikova, M. Samsonova, and M. Unser. Algorithm for spline-based elastic registration in application to confocal images of gene expression. In *Proceedings of the Seventh International Conference on Pattern Recognition and Image Analysis: New Information Technologies (PRIA-7'04)*, volume 3, pages 928–931, St. Petersburg, Russian Federation, October 18-23, 2004.

[76] Thomas Hartkens, Karl Rohr, and H. Siegfried Stiehl. Evaluierung von differentialoperatoren zur detektion charakteristischer punkte in tomographischen bildern. In *DAGM-Symposium*, Informatik Aktuell, pages 637–644. Springer, 1996.

[77] Jan Borchers, Oliver Deussen, and Clemens Knörzer. Getting it across: layout issues for kiosk systems. *SIGCHI Bull.*, 27(4):68–74, 1995.

[78] C Piotrowski. A review of the clinical and research use of the Bender-Gestalt Test. *Percept Mot Skills*, 81(3 Pt 2):1272–1274, Dec 1995.

[79] Song-Chun Zhu. Embedding Gestalt Laws in Markov Random Fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(11):1170–1187, 1999.

[80] F. Cao, P. Musé, and F. Sur. Extracting meaningful curves from images. *Journal of Mathematical Imaging and Vision*, 22(2–3):159–181, 2005.

[81] C. Keysers, D. Xiao, P. Földiák, and D. I. Perrett. The speed of sight. *Journal of Cognitive Neuroscience*, 13(1):90–101, 2001.

[82] M. Heiler and C. Schnorr. Natural image statistics for natural image segmentation. In *ICCV03*, pages 1259–1266, 2003.

[83] Terry S. Yoo. Multiscale statistical image invariants. In *7th International Conference on Computer Vision*, 1999.

[84] A Srivastava, A B Lee, E P Simoncelli, and S-C Zhu. On advances in statistical modeling of natural images. *J. Math. Imaging and Vision*, 18(1):17–33, January 2003.

[85] Mark A. Ruzon and Carlo Tomasi. Edge, junction, and corner detection using color distributions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1281–1295, 2001.

[86] Lyndon S. Hibbard. Region segmentation using information divergence measures. In *MICCAI (2)*, pages 554–561, 2003.

[87] D. Endres and J.E. Schindelin. A new metric for probability distributions. *IEEE Transactions on Information Theory*, 49(7):1858–1860, 2002.

[88] M. Varma and A. Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1–2):61–81, April 2005.

[89] M. Varma and A. Zisserman. Unifying statistical texture classification frameworks. *Image and Vision Computing*, 22(14):1175–1183, 2005.

[90] H.Z. Rafi and H. Soltanian-Zadeh. Mutual Information Restoration of Multispectral Images. In *10th International Workshop on Systems, Signals and Image Processing (IWSSIP'03)*, Prague, Czech Republic, Sept. 2003.

[91] I. Nemenman, W. Bialek, and R.R. van Steveninck. Entropy and information in neural spike trains: Progress on the sampling problem. *Physical Review E*, 69(5), 2004.

[92] P.K. Sahoo and S. Soltani and K.C. Wong and Y.C. Chen. A Survey of Thresholding Techniques. *Computer Vision, Graphics, and Image Processing*, 41:233–260, 1988.

[93] P. S. Heckbert. Color Image Quantization for Frame Buffer Display. *ACM Computer Graphics (ACM SIGGRAPH '82 Proceedings)*, 16(3):297–307, 1982.

[94] D. Endres and P. Földiák. Baysian bin distribution inference and mutual information. *IEEE Transactions on Information Theory*, 2005, in press.

[95] M. Hutter. Distribution of mutual information. In *Advances in Neural Information Processing Systems 14*, pages 339–406, Cambridge, MA, 2002. MIT Press.

[96] K. Rohr and M. Fornefett and H.S. Stiehl. Spline-Based Elastic Image Registration: Integration of Landmark Errors and Orientation Attributes. *Computer Vision and Image Understanding*, 90(2):153–168, May 2003.

[97] I. N. Bronstein and K. A. Semendyayev. *Handbook of mathematics.* Harri Deutsch, Frankfurt/Main, 24th edition, 1989.

[98] D. Wolpert and D. Wolf. Estimating functions of probability distributions from a finite set of samples, part 1: Bayes estimators and the shannon entropy.

[99] D. Wolpert and D. Wolf. Estimating functions of probability distributions from a finite set of samples, part 2: Bayes estimators for mutual information, chi-squared, covariance, and other statistics.

Hiermit erkläre ich, Johannes E. Schindelin, dass ich

1. die Dissertation mit dem Titel

   The standard brain of *Drosophila melanogaster* and its automatic segmentation

   selbständig und nur unter Zuhilfenahme der in der Arbeit selbst angegebenen Mittel und Quellen erstellt habe,

2. die Dissertation mit dem Titel

   The standard brain of *Drosophila melanogaster* and its automatic segmentation

   zum ersten mal in einem Prüfungsverfahren vorlege, und dass ich

3. ausser dem Diplom in Mathematik bisher weder einen akademischen Grad erworben, noch einen solchen zu erwerben versucht habe.

Johannes E. Schindelin