A Framework for the Visualization of Brain Structures

Sebastian Thelen
s_thelen@informatik.uni-kl.de
University of Kaiserslautern, Germany

Torsten Bierz
bierz@informatik.uni-kl.de
International Research Training Group
University of Kaiserslautern, Germany

Britta Müller
bmuelle@rhrk.uni-kl.de
University of Kaiserslautern, Germany

Hans Hagen
hagen@informatik.uni-kl.de
University of Kaiserslautern, Germany

Achim Ebert
ebert@informatik.uni-kl.de
University of Kaiserslautern, Germany

Eckhard Friauf
eckhard.friauf@biologie.uni-kl.de
University of Kaiserslautern, Germany

Jörg Meyer
jmeyer@uci.edu
University of California, Irvine, USA

Abstract: Nowadays biologists investigate different causes for deafness. One reason is a damage in a particular region of the auditory brain stem. Anatomical differences were discovered when investigating brain slices of different laboratory mice. However, these slices are only a two dimensional representation of a part of the brain. The arising question was how these differences of structure affect the three dimensional representation of this region. Therefore, an interdisciplinary framework was developed, which allows even unexperienced users to investigate and compare these regions.

1 Introduction

The *Superior Olivary Complex* (SOC) is a part of the auditory brain stem playing an important role in hearing. The SOC consists of several *cores*, each responsible for a particular aspect of sound processing. It was discovered that genetic suppression of a certain calcium channel in mouse brains results, among other things, in deafness by birth. Cross-sections of the SOC revealed anatomic differences in a core called *Lateral Superior Olivary* (LSO). LSOs of mice with genetic suppression (knock-out mice) differ in form and size, respectively number and density of cells from the LSOs of healthy mice. Further studies of these differences should be supported by a framework, that can visualize the cores of the SOC three dimensionally (especially the LSO) and determine characteristic parameters, such as volumes and surface areas of the cores. Of course professional applications used in *Magnetic Resonance Imaging* (MRI) or *Computed Tomography* (CT) can accomplish this task. Unfortunately their acquisition goes far beyond the financial scope of most research groups. Therefore this paper focuses on a low cost alternative, that does not require expen-

sive hardware. Instead the framework processes grayscale images of cross-sections, that can easily be obtained by using a camera and a microscope.
Section 2 provides an overview of related work in the field of brain visualization. In Section 3 the reconstruction of cores, which is based on contour information taken from digital grayscale images of the brain slices is presented. Section 4 describes methods for volume- and surface estimation used in the analysis of the cores. Finally Section 5 presents implementational aspects of the framework. The paper ends with a discussion and perspective on future work.

2 Related work

Nowadays numerous different applications deal with the visualization and analysis of cerebral data sets for various purposes. Some representative examples are given:
The *Allen Brain Explorer* [Al07], a product of the *Allen Institute for Brain Science*, provides a detailed cellular-resolution, genome-wide map of gene expression in the mouse brain. Because there is a high degree of similarity between the human genome and that of a mouse ($\sim 90\%$), the project offers the opportunity to further understanding of human disorders and diseases (e.g. Alzheimer's, Parkison's, epilepsy etc). In 2006, the explorer was able to visualize the expression of approximately 20.000 genes in a three dimensional model of the brain.
AnatQuest [An07] is based on the *Visual Human Project* [NLM07], a collection of digital images of complete human male and female cadavers in MRI, CT and anatomical modes. The images of the Visual Human data set are used by AnatQuest to create three dimensional models of anatomical objects within these slices. The *Insight Toolkit* (ITK) [ITK07] also supports the Visual Human Project and employs leading-edge segmentation and registration algorithms in two, three, and more dimensions. Developing these algorithms is of great importance in many branches of medicine, e.g. in the automatic identification of tumors.
Nearly all applications dealing with the visualization of biological data sets rely on a method to reconstruct the models to display. In the current approach this is done based on contour information. Besides the approach applied in this paper, which was first presented in [EPO91], various other authors dedicated their work to this topic. A detailed summary of previous work can be found in [MS92].
In order to achieve good results in the reconstruction process, it is advisable to align all slices in a preceding step. For cross-sectional data sets, pin holes in the preparations can simplify this task [SM01]. Unfortunately this method was not applicable in the current case, because the preparations were too fragile.

3 Core reconstruction

A typical data set to be processed consists of 6-10 grayscale images of anatomical cuts of the SOC, sliced at approximately 30 μm intervals. All images were segmented by manually marking the contours of the cores to be reconstructed (e.g. the LSO) with unique colors. Although it would be of great benefit to segment all pictures automatically, the manual marking is the only appropriate way at the moment. Most regions are so diffuse that it takes an experienced person with biological background to identify them correctly. All steps of the reconstruction process are described subsequently. After the extraction of contour information from the input files, an alignment procedure compensates transformations caused by the manual treatment of preparations. The preprocessed contours can then be used to identify correspondences between points of consecutive contours, which will provide the information needed to construct a geometric model of a core.

3.1 Contour extraction

First all core contours are extracted from the grayscale images. In a mathematical sense, a contour is an ordered sequence of two dimensional points. Obviously some caution is needed to identify this sequence correctly. Traversing the image pixelwise row by row will ignore the predecessor-successor order between contour pixels. *Contour Tracing Algorithms*, like the one of *Pavlidis* [Pa82], provide a solution to this problem. Pavlidis' algorithm extracts the point set of a contour in counterclockwise direction by always searching for the next rightmost contour pixel in moving direction. The idea is illustrated in Figure 1, where the contour pixels of a gray object are identified one after another in correct order. Solid arrows indicate directions in which a pixel is searched for. Directions with dashed arrows do not have to be checked anymore, because a suited pixel was already found before.

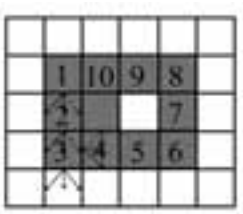
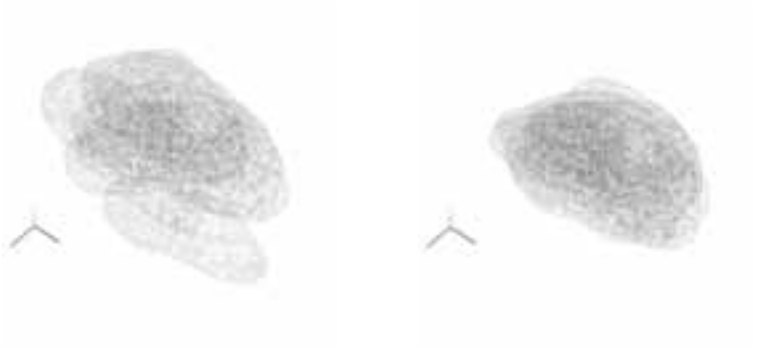


Figure 1: Contour extraction of a gray object by Pavlidis' algorithm. Arrows indicate directions where to search for the next pixel. Directions with dashed arrows do not have to be checked anymore. All pixels belonging to the contour are identified in correct order.



(a) (b)

Figure 2: (a) Situation before the alignment with barycenters mutually displaced an eigenvector crosses twisted against each other. (b) Situation after the alignment. Transformations due to manual placement are compensated.

3.2 Alignment

The next steps in the reconstruction process are based on the extracted contours. However, a misalignment of the slices due to manual placement onto glass object carriers prevents a direct use. Slices can be displaced, twisted and even distorted. The former two result from an imprecise placement, whereas the latter one is due to the varying pressure of a brush, used to place the preparations. Therefore, contours must be aligned in order to compensate these transformations. By performing a *Principal Component Analysis* (PCA) it is possible to treat translations and rotations between successive contours [La06]. Identifying the principal components of a two dimensional contour is done by calculating the eigenvectors and eigenvalues of the points' covariance matrix. This yields to two perpendicular vectors in the barycenter of each contour, i.e. the principal components. Contours are aligned by making the crosses formed by the eigenvectors coincident through mutual shifting and twisting, as shown in Figure 2. Provided that contours are quite similar and not too much transformed, this technique produces useful results in most situations.
However, results should be interpreted with some care, for it is possible that the structure of a core gets lost during the alignment. One can imagine a core where the centers of its contours are mutually displaced. By making these centers coincident, the natural structure of the core will be destroyed. The reason is the impossible determination whether the displacement of barycenters is due to the core's natural structure or to the manual positioning onto object carriers. Finding an appropriate alternative will be subject of future work.

3.3 Linking

The aligned contours are used to identify corresponding points in successive contours. Information gained in this step will be used to form the triangle meshes of a core's wire

frame model. The applied method was first presented by Ekoule et al. [EPO91]. A *correspondence*, or *linkage*, can be expressed by a function $Z : \mathcal{P} \rightarrow \mathcal{Q}$ from the set of points of the first contour into the set of points of the following one. The mapping of Z should be "natural", as illustrated in Figure 3. Here the point p is mapped to $Z(p)$ which seems

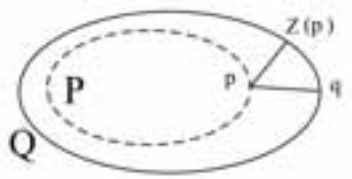


Figure 3: Defining a linkage between points of consecutive contours. Due to the geometry of both contours, a link between p and q would seem most "natural".

quite unnatural. Regarding the geometry, q would be a more suitable candidate. Ekoule et al. determine corresponding points by the use of a *local distance criterion* for *convex* contours. If a pair of points $(p_i, Z(p_i))$ is selected, $Z(p_{i+1})$ is searched for among the successors of $Z(p_i)$ within a local neighborhood, so that the distance between p_{i+1} and $Z(p_{i+1})$ becomes minimal.
Nonconvex contours are treated by projecting them onto their convex hulls and determining a linkage between the projected contours. The projection procedure must take into account the local deformation of a contour. For this purpose, contours are subdivided recursively into concavities which are projected onto the convex hull of the superior concavity.
At the moment the framework is able to identify *single branching* correspondences. For each core there is exactly one contour in every slice of the data set. *Multiple branching* situations arise when a core splits up. Implementing a correct multiple branching detection and treatment will enhance the frameworks functionality in future versions. Approaches how to do this can be found in [EPO91].

3.4 Triangulation

After linking points of consecutive contours, the triangle meshes of a core's model are constructed. The order of pixels within a contour and the mapping of $Z : \mathcal{P} \rightarrow \mathcal{Q}$ already define some triangle edges. However, assuming without loss of generality $|\mathcal{P}| \leq |\mathcal{Q}|$ holds (otherwise they are switched), there can be points in $\mathcal{Q}$ that are not linked to any point in $\mathcal{P}$. These must be taken into account when constructing the meshes. The method used is described by Linsen [Li97]. Unlinked points are assigned to points in $\mathcal{P}$ and possible gaps are closed. It is verified that the orientation of points is consistent for all triangles of a model. This is important for a correct visualization with OpenGL. As an example of the whole reconstruction process, the wire frame model of a Lateral Superior Olivary is shown in Figure 4.

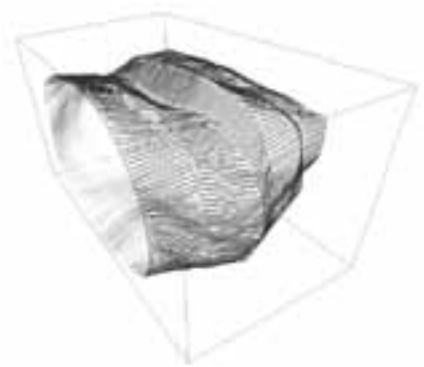


Figure 4: Wire frame model of a LSO. The model was reconstructed from 7 slices of a healthy mouse brain.

4 Analysis

As stated in the introduction, the LSOs of knock-out mice and wild-type mice differ from each other. Several parameters characterize these differences. The framework supports the comparison of the two mouse groups by providing an automated estimation of the volume- and surface values for the reconstructed cores.

4.1 Volume estimation

The volume of a core can be calculated by figuring out the integral $\int_Z A(z)dz$, where $A(z) =$ *contour area at depth* z [RH90]. This integral is approximated by

$$\sum_{i=0}^{n-2}(z_{i+1} - z_i)\left[\frac{A(z_i) + A(z_{i+1})}{2}\right],$$

a discrete sum of trapezoidal areas beneath the graph of A. The parameter n describes the number of contours, z_i their depths in z-direction and $A(z_i)$ the contour area at position z_i. A comparison of methodologies shows a high degree of similarity between the results of automatic estimations and the ones of conventional manual calculations. The resemblance underlines the correctness of this approach.
An interesting fact is that the volumes are independent from the method of alignment (as far as only translations and rotations are treated). This is guaranteed by *Cavalieri's Principle*, which claims that "*if, in two solids of equal altitude, the sections made by planes parallel to and at the same distance from their respective bases are always equal, then the volumes of the two solids are equal*" [We03]. Hence, possible variations in the method of alignment will not affect volume estimation.

4.2 Surface estimation

Surface estimation of a core is straight forward compared to its volume estimation. It is done by summing up all triangle areas of a core's model. Obviously this estimation is strongly affected by the method of alignment, so alterations in future versions will lead to different results.

5 Implementation

Implementing a prototypic software application requires deliberate decisions regarding the tools to be used. The framework is designed for WindowsXP and was implemented choosing C++ as a programming language. C++ is widely spread and comes along with robust and efficient compilers for nearly every platform. Microsoft's Windows is the most convenient operating system and is best known by most users. It provides various tools for handling a development environment (e.g. Microsoft Visual Studio 2005). In order to create an appealing visualization and comfortable Graphical User Interface (GUI), OpenGL and Trolltech's Qt4 were used, which are freely available for a multitude of systems. The framework was tested on an Intel Centrino 1.7 GHz system with 1024 MB RAM and a 128 MB ATI 9700 graphic card. At least 50 MB of free disk space are required.

5.1 Basic architecture

The framework is built of four modules which implement the methods described in the previous sections. The modules depend on each other an are organized in a pipeline as follows:

ContourMaker This component extracts contours from image files in the *Portable Network Graphics* format (PNG) by executing an implementation of Pavlidis' algorithm. All contour pixels within a slice are identified and saved in *xls*-files.

SliceConstructor The *xls*-files produced by the ContourMaker are processed by the SliceConstructor, which performs a principal component analysis on the point sets and identifies translation vectors and rotation angles for the following alignment. All alignment information is stored in *xls*-files for further use.

VolumeConstructor This component is responsible for the reconstruction of core models. Points in consecutive contours are linked as described as described above triangle meshes are created. Results are saved as *Obj/Wavefront*-files, which can be displayed by various viewers.

Viewer3D Finally a three dimensional representation of the SOC and its cores is created and displayed to the user. This module offers the possibility to examine data sets

in a virtual environment and to interact with them by adjusting the perspective as required (through translation, rotation and zooming).

All components are controlled by *configuration files* which mainly contain information about the input files to process and the output files to create.

5.2 Graphical User Interface

When starting the framework, the application's main window, shown in Figure 5, is presented to the user. From here it is possible to perform all provided operations. It mainly consists of two graphic windows (instances of Viewer3D) which can display different data sets simultaneously. Thus it is possible to directly compare knock-out and wild-type mice. Beneath each window several control elements allow the user to take influence on the pre-

Figure 5: The framework's main window consists of two graphic windows. The user intuitively interacts with data sets by using the mouse to adjust his view. The design facilitates the direct comparison of wild-type and knock-out mice.

sentation of data sets (distance between slices, current time step in a series of data sets, parameters for artificial end pieces). Further operations are supported by elements of the main menu. Results of the automated analysis are presented in form of histograms. A histogram always displays parameter estimations for the cores in the left and right graphic window, which simplifies the task of recognizing regularities and differences in data sets. Samples of scenes showing a reconstructed Lateral Superior Olivary are presented in Figure 6. Figure 6(a) illustrates the model of a LSO, reconstructed from seven cross-sections which were sliced at intervals of 30 μm. In order to investigate the core together with the embedding tissue, the original data set (grayscale images of the preparations) has been added to the scene in Figure 6(b). By rotating, translating and zooming the user can adjust

the point of view as required. A color-mapped version of Figure 6(b) can be found in the colorplate on p. 191. Color-mapping is achieved by applying a transfer function to the

(a) (b)

Figure 6: A sequence of scenes generated by the framework. (a) Surface model of a LSO. (b) Surface model embedded into the surrounding tissue.

original data set. They are a common technique in scientific visualization and are used to map scalar values to optical features. The framework uses transfer functions to map grayscale values to RGB-colors. Thus this it gets possible to emphasize certain features of a data set, like darker regions with lots of cells, by highlighting them with different colors. Transfer functions are defined in a separate dialog (see colorplate) that allows the declaration of up to three disjoint intervals in the set of grayscale values over which RGB-colors are interpolated linearly.

6 Conclusion and Future Work

In this paper the development of an interdisciplinary framework for the visualization of brain structures was described. The framework can visualize a particular region of the auditory brain stem of a mouse. Data sets consist of cross-sections containing contour information about the cores to be reconstructed. The reconstruction process was discussed in detail and difficulties with the alignment were pointed out. Different opportunities to analyze the structures were described.
Considering the specific needs of the involved biologists, the framework is used in the validation of a deafness model by providing ways to visualize and quantify differences between two mouse groups.
Future work will focus on enhancing the alignment of contours and increasing the framework's flexibility by providing ways to handle branching structures. Another extension is the automatic estimation and calculation of the amount of cells and cell cores, because biologists still try to count their occurrences by hand.

References

[Al07] Allen Brain Atlas - Neuroscience Gateway. http://www.brainatlas.org/aba/. 2007.

[An07] Anatquest - Anatomic Images Online. http://anatquest.nlm.nih.gov/. 2007.

[CS78] Christiansen, H. und Sederberg, T.: Conversion of complex contour line definitions into polygonal element mosaics. In: *Computer Graphics*. volume 12. pp. 187–192. ACM. 1978.

[EPO91] Ekoule, A., Peyrin, F., und Odet, C.: A triangulation algorithm from arbitrary shaped multiple planar contours. In: *Transactions on Graphics*. volume 10. pp. 182–199. ACM. Juli 1991.

[Fr07] Friauf, E. Vom GehËr zum Gehirn: die Neurobiologie des HËrens. http://www.uni-kl.de/wcms/agf_hoeren.html. 2007.

[ITK07] Insight Toolkit (itk). http://www.itk.org/index.htm. 2007.

[Ka99] Kaballo, W.: *EinfÂhrung in die Analysis III*. Spektrum. 1999.

[La06] Lacour, F.: Large-scale biomedical image registration using the principal component analysis. Project thesis. University of Kaiserslautern. 2006.

[Li97] Linsen, L.: Schnitt und Vereinigung von Kontrollnetzen. Diploma thesis. University of Karlsruhe. 1997.

[MS92] Meyers, D. und Skinner, S.: Surfaces from contours. In: *Transactions on Graphics*. volume 11. pp. 228–258. ACM. Juli 1992.

[NLM07] The Visible Human Project. http://www.nlm.nih.gov/research/visible/visible_human.html. 2007.

[Pa82] Pavlidis, T.: *Algorithms for Graphics and Image Processing*. Computer Science Press. Maryland. 1982.

[RH90] Rosen, G. und Harry, J.: Brain volume estimation from serial section measurements: A comparison of methodologies. In: *Journal of Neuroscience Methods*. volume 35. pp. 115–124. 1990.

[SM01] Shulga, D. und Meyer, J.: Aligning large-scale medical and biological data sets: Exploring a monkey brain. In: *Visualization, Imaging and Image Processing (VIIP 2001)*. volume 3. pp. 434–439. The International Association of Science and Technology for Development (IASTED). ACTA Press. September 2001.

[Th07] Thelen, S.: Entwicklung eines Frameworks zur Visualisierung, Exploration und Komparation biologischer Datensâtze. Diploma thesis. University of Kaiserslautern. 2007.

[We03] Weisstein, E. Cavalieri's Principle. From MathWorld - A Wolfram Web Resource. http://mathworld.wolfram.com/CavalierisPrinciple.html. 2003.