

Amira 5

User's Guide



Amira[®] 5

Amira User's Guide

Intended Use

Amira[®] is intended for research use only. It is not a medical device.

Copyright Information

©1995-2009 Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany

©1999-2009 Visage Imaging

All rights reserved.

Trademark Information:

Amira is being jointly developed by Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB) and Visage Imaging.

Amira[®] is a registered trademark of Konrad-Zuse-Zentrum für Informationstechnik Berlin and Visage Imaging.

HardCopy, MeshViz, VolumeViz, TerrainViz, ScaleViz are trademarks of Mercury Computer Systems S.A.S.

Mercury Computer Systems S.A.S. is a source licensee of OpenGL[®], Open Inventor[®] from Silicon Graphics, Inc.

OpenGL[®] and Open Inventor[®] are registered trademarks of Silicon Graphics, Inc.

All other products and company names are trademarks or registered trademarks of their respective companies.

This manual has been prepared for Visage Imaging licensees solely for use in connection with software supplied by Visage Imaging and is furnished under a written license agreement. This material may not be used, reproduced or disclosed, in whole or in part, except as permitted in the license agreement or by prior written authorization of Visage Imaging. Users are cautioned that Visage Imaging reserves the right to make changes without notice to the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including but not limited to typographical, arithmetic or listing errors.

Contents

I	Amira User's Guide	1
1	Introduction	3
1.1	Overview	4
1.2	Features overview	4
1.2.1	Data import	5
1.2.2	Viewing, navigation, interactivity	6
1.2.3	Visualization of 3D Image Data	6
1.2.4	Image processing	7
1.2.5	Model reconstruction	8
1.2.6	Visualization of 3D models and numerical data	9
1.2.7	General Data Processing and Data Analysis	10
1.2.8	Matlab integration	12
1.2.9	High Performance Visualization	12
1.2.10	Automation, Customization, Extensibility	12
1.3	Application Areas	12
1.4	Options	13
2	First steps in Amira	17
2.1	Getting Started	18
2.1.1	Loading Data	19
2.1.2	Invoking Editors	21
2.1.3	Visualizing Data	22
2.1.4	Interaction with the Viewer	23
2.2	How to load image data	26
2.2.1	The Amira File Browser	26
2.2.2	Reading 3D Image Data from Multiple 2D Slices	27
2.2.3	Setting the Bounding Box	28
2.2.4	The Stacked Slices file format	29

2.2.5	Working with Large Disk Data	30
2.3	Visualizing 3D Images	32
2.3.1	Orthogonal Slices	33
2.3.2	Simple Data Analysis	33
2.3.3	Resampling the Data	34
2.3.4	Displaying an Isosurface	36
2.3.5	Cropping the Data	36
2.3.6	Volume Rendering	37
2.4	Segmentation of 3D Images	40
2.4.1	Interactive Image Segmentation	40
2.4.2	Volume Measurement	42
2.4.3	Threshold Segmentation	43
2.4.4	Refining Threshold Segmentation Results	43
2.5	Surface Reconstruction from 3D Images	45
2.5.1	Extracting Surfaces from Segmentation Results	45
2.5.2	Simplifying the Surface	45
2.6	Creating a Tetrahedral Grid from a Triangular Surface	47
2.6.1	Simplifying the Surface	47
2.6.2	Editing the Surface	49
2.6.3	Generation of a Tetrahedral Grid	51
2.7	Warping and Registration Using Landmarks	53
2.7.1	Displaying Data Sets in Two Viewers	54
2.7.2	Creating a Landmark Set	54
2.7.3	Registration via a Rigid Transformation	57
2.7.4	Warping Two Image Volumes	57
2.8	Registration of 3D image data sets	58
2.8.1	Basic Manual Registration	59
2.8.2	Automatic Registration	60
2.8.3	Image Fusion	61
2.9	Alignment of 2D Physical Cross-sections	62
2.9.1	Basic Manual Alignment	63
2.9.2	Alignment Via Landmarks	65
2.9.3	Optimizing the Quality Function	67
2.9.4	Resampling the Input Data	67
2.9.5	Using a Reference Image	68
2.10	Visualization of Vector Fields	69
2.10.1	Simple Vector Representation	69
2.10.2	Illuminated Stream Lines	71

2.10.3	Animated Particle Plot	72
2.11	Creating animated demonstrations	73
2.11.1	Creating a Network	74
2.11.2	Animating an OrthoSlice Module	74
2.11.3	Activating a Module in the Viewer Window	77
2.11.4	Using a Camera Rotation	78
2.11.5	Editing or Removing an Already Defined Event	79
2.11.6	Overlaying the Bone with Skin	79
2.11.7	Using Clipping to Add the Skin Gradually	80
2.11.8	More Comments on Clipping	83
2.11.9	Breaks and Function Keys	84
2.11.10	Loops and Go-to	85
2.11.11	Storing and Replaying the Animation Sequence	86
2.12	Creating movie files	86
2.12.1	Attaching MovieMaker to a Camera Path	87
2.12.2	Attaching MovieMaker to DemoMaker	89
2.13	Using MATLAB Scripts	90
2.13.1	Lowpass Filtering on Images	90
2.13.2	Thresholding on a Volume	93
2.14	Introduction to the Filament Editor	93
2.14.1	Exploration of the volume data	94
2.14.2	Automatic extraction of the dendritic tree	94
2.14.3	Filament Tracing	97
2.14.4	Visualize the network	98
2.14.5	Alternative way to extract a network using the Segmentation Editor	99
2.15	Introduction to the Multi-planar Viewer	100
2.15.1	Exploration of the volume data	101
2.15.2	Explore two data sets using fusion mode	102
2.15.3	Manually register two data sets	104
3	Program Description	107
3.1	Interface Components	107
3.1.1	File Menu	107
3.1.2	Edit Menu	110
3.1.3	Pool Menu	112
3.1.4	Explorer Menu	114
3.1.5	Create Menu	114

3.1.6	View Menu	115
3.1.7	Online Help	118
3.1.8	Main Window	121
3.1.9	Viewer Window	131
3.1.10	Console Window	136
3.1.11	File Dialog	137
3.1.12	Job Dialog	139
3.1.13	Preferences Dialog	141
3.1.14	Snapshot Dialog	146
3.1.15	System Information Dialog	147
3.2	General Concepts	148
3.2.1	Class Structure	148
3.2.2	Scalar Field and Vector Fields	150
3.2.3	Coordinates and Grids	151
3.2.4	Surface Data	152
3.2.5	Vertex Set	153
3.2.6	Transformations	153
3.2.7	Parameters	154
3.2.8	Shadowing	154
3.2.9	Sub-application Concept	155
4	Technical Information	157
4.1	Data Import	157
4.2	Command Line Options	158
4.3	Environment Variables	159
4.4	User-defined start-up script	161
4.5	System Requirements	162
4.5.1	Troubleshooting	163
4.5.2	Microsoft Windows	164
4.5.3	Linux	164
4.5.4	Mac OS	165
4.6	Amira License Manager	166
4.6.1	Contents	166
4.6.2	About Password Protection	166
4.6.3	About the Amira License Manager	166
4.6.4	Troubleshooting	167
4.6.5	Using TGS_LICENSE_DEBUG	168
4.6.6	Contacting the License Administrator	170

4.6.7	Contacting Technical Support	170
4.7	Notes for Mac users	171
4.8	Amira and the /3GB switch	171
4.8.1	The Problem	172
4.8.2	How to activate the /3GB switch	172
5	Scripting	175
5.1	Introduction	175
5.2	Introduction to Tcl	176
5.2.1	Tcl Lists, Commands, Comments	176
5.2.2	Tcl Variables	177
5.2.3	Tcl Command Substitution	178
5.2.4	Tcl Control Structures	178
5.2.5	User-Defined Tcl Procedures	180
5.2.6	List and String Manipulation	181
5.3	Amira Script Interface	182
5.3.1	Predefined Variables	184
5.3.2	Object commands	185
5.3.3	Global Commands	185
5.4	Amira Script File	199
5.5	Configuring Popup Menus	200
5.6	Registering pick callbacks	203
6	Demo Framework	207
6.1	Directory Structure and Files	207
6.1.1	Directories and Files in src/	208
6.1.2	Data Storage	214
6.2	Selecting Demos	215
6.3	Prerequisites	218
II	Molecular Option User's Guide	219
7	Molecular Option Introduction	221
7.1	First Steps with Molecular Visualization in Amira	221
7.1.1	Getting Started with Molecular Visualization	222
7.1.2	Selection, Labeling, and Masking	225
7.1.3	Alignment of Molecules	232

7.1.4	Molecular Surfaces	237
7.1.5	Sequential and Structural Alignment	240
7.1.6	Editing of molecules	242
7.1.7	Molecular Interfaces	245
7.1.8	Measurement	248
7.2	Molecular Data Structures	249
7.2.1	Internal Structure of Molecules	249
7.3	Displaying Molecules	249
7.3.1	Coloring Molecules	250
7.3.2	Selecting and Filtering atoms	252
7.4	Aligning Molecules	254
7.4.1	Alignment of Trajectories	254
7.4.2	Mean Distance Alignment	256
7.4.3	Sequence alignment	256
7.5	Visualizing Molecular Trajectories and Metastable Conformations	257
7.6	Atom Expressions	257
7.6.1	Overview	257
7.6.2	Grammar	258
7.6.3	Literals	258
7.6.4	Operators	259
7.6.5	Shortcuts	261
7.6.6	Further Examples	261
III	Virtual Reality Option User's Guide	263
8	Virtual Reality Option User's Guide	265
8.1	Virtual Reality Option Essentials	266
8.1.1	Virtual Reality Option configurations	266
8.1.2	Immersive interaction and the trackd daemon	266
8.1.3	Multiple displays and parallel rendering	267
8.2	Using Virtual Reality Option on a multi-pipe system	267
8.2.1	Configuration	267
8.2.2	Starting Amira	268
8.3	Using Virtual Reality Option on a cluster	268
8.3.1	Overview	268
8.3.2	Requirements	269
8.3.3	Preparing cluster slave nodes	269

8.3.4	Running Virtual Reality Option on cluster	272
8.3.5	Limitations	273
8.3.6	Troubleshooting cluster configurations	274
8.4	Flat Screen Configurations	274
8.4.1	Example: A 2-channel Passive Stereo Configuration . . .	275
8.4.2	Example: A Super-wide Configuration with Soft-edge Blending	279
8.4.3	Example: A Tiled 4-channel 2x2 Monitor Configuration .	282
8.5	Immersive Configurations	284
8.5.1	Example: A Workbench Configuration	285
8.5.2	Example: A Holobench Configuration	287
8.5.3	Example: A 4-side CAVE Configuration	290
8.6	Calibrating the Tracking System	293
8.6.1	Overview	293
8.6.2	Tracking system essentials	294
8.6.3	Calibration step by step	298
8.7	3D User Interaction	301
8.7.1	The 3D Menu	302
8.7.2	User-defined 3D Menu Items	303
8.7.3	3D Module Controls	307
8.7.4	Navigation Modes	307
8.7.5	Tcl Event Procedures	308
8.7.6	The 2D Mouse mode	309
8.8	Writing Virtual Reality Option Custom Modules	310
8.9	Config File Reference	313
8.9.1	Tracker Emulator	322

IV Very Large Data Option User's Guide 325

9 Very Large Data Option User's Guide 327

9.1	Working with out-of-core data files (LDA)	327
9.1.1	Tune the Size Threshold to Allow Conversion	327
9.1.2	Load the Out-of-core Data	328
9.1.3	Raw Data Parameters	329
9.1.4	Out-of-core Conversion	329
9.1.5	Display an Orthoslice, an Oblique Slice, and a 3D Volume	332

V	Quantification+ Option User's Guide	335
VI	Microscopy Option User's Guide	339
10	Deconvolution	341
10.1	General remarks about image deconvolution	342
10.2	Data acquisition and sampling rates	343
10.3	Standard Deconvolution Tutorial	345
10.4	Blind Deconvolution Tutorial	350
10.5	Bead Extraction Tutorial	354
10.6	Performance issues and multi-processing	359
11	Working with Multi-Channel Images	365
11.1	Loading Multi-Channel Images into Amira	365
11.2	Using OrthoSlice with a MultiChannelField	367
11.3	Using ProjectionView with a MultiChannelField	368
11.4	Using Voltex with a MultiChannelField	368
11.5	Saving a MultiChannelField in a Single AmiraMesh File	370
VII	Skeleton Option User's Guide	371
12	Skeleton Option User's Guide	373
12.1	Importing your Image Data	373
12.2	Arranging the Bricks	374
12.3	Aligning Bricks	375
12.4	Filtering, Correcting Z-Drop, Resampling	375
12.5	Creating the Large Disk Data	377
12.6	Accessing the Large Disk Data	378
12.7	Computing directly on the Large Disk Data	379
12.8	Region of Interest	381
12.9	Check Network	383
12.10	Coloring a Lineset According to its Depth Value	383

Part I

Amira User's Guide

1 Introduction

Amira is a 3D data visualization, analysis and modelling system. It allows you to visualize scientific data sets from various application areas, e.g. medicine, biology, bio-chemistry, microscopy, biomed, bioengineering. 3D data can be quickly explored, analyzed, compared, and quantified. 3D objects can be represented as image volumes or geometrical surfaces and grids suitable for numerical simulations, notably as triangular surface and volumetric tetrahedral grids. Amira provides methods to generate such grids from voxel data representing an image volume, and it includes a general-purpose interactive 3D viewer.

Amira is a powerful, multifaceted software platform for visualizing, manipulating, and understanding Life Science and bio-medical data coming from all types of sources and modalities. Initially known and widely used as the 3D visualization tool of choice in microscopy and biomed research, Amira has become a more and more sophisticated product, delivering powerful visualization and analysis capabilities in all visualization and simulation fields in Life Science.

- Multi purpose - One platform for visualizing, analyzing and presenting
- Flexible - Option packages to configure Amira to your needs
- Efficient - Takes advantage of the latest graphics cards and processors
- Easy to use - Intuitive user interface and great documentation
- Cost effective - Multiple options and flexible license models
- Handling large data - Very large data sets are easily accessible with specific readers
- Extensible - C++ coding wizard for technical extension and customization
- Support - Direct customer support with high level of interaction
- Innovative - Technology always updated to the latest innovation

Section [1.1](#) (Overview) provides a short overview of the fundamentals of Amira, i.e. its object-oriented design and the concept of data objects and modules.

Section [1.2](#) (Features) summarizes key features of Amira, for example direct volume rendering, image processing, and surface simplification.

Section 1.3 (Application Areas) illustrates some typical application areas of Amira and shows what kinds of problems can be solved using the system.

Section 1.4 (Options) shortly describes optional extensions available for Amira and what they can be used for.

1.1 Overview

Amira is a modular and object-oriented software system. Its basic system components are modules and data objects. Modules are used to visualize data objects or to perform some computational operations on them. The components are represented by little icons in the *Pool*. Icons are connected by lines indicating processing dependencies between the components, i.e., which modules are to be applied to which data objects. Alternatively, modules and data objects can be displayed in a *tree view Explorer*. Modules from data objects of specific types are created automatically from file input data when reading or as output of module computations. Modules matching an existing data object are created as instances of particular module types via a context-sensitive popup menu. Networks can be created with a minimal amount of user interaction. Parameters of data objects and modules can be modified in Amira's interaction area.

For some data objects such as surfaces or colormaps there exist special-purpose interactive editors that allow the user to modify the objects. All Amira components can be controlled via a Tcl command interface. Commands can be read from a script file or issued manually in a separate console window.

The biggest part of the screen is occupied by a 3D graphics window. Additional 3D views can be created if necessary. Amira is based on the latest release of Open Inventor from Mercury. In addition, several modules apply direct OpenGL rendering to achieve special rendering effects or to maximize performance. In total, there are more than 270 data object and module types. They allow the system to be used for a broad range of applications. Scripting can be used for customization and automation. User-defined extensions are facilitated by the Amira developer version.

1.2 Features overview

Amira provides a large number of data and module types allowing you to visualize, analyze and model various kinds of 3D data. The Amira framework is ideal to

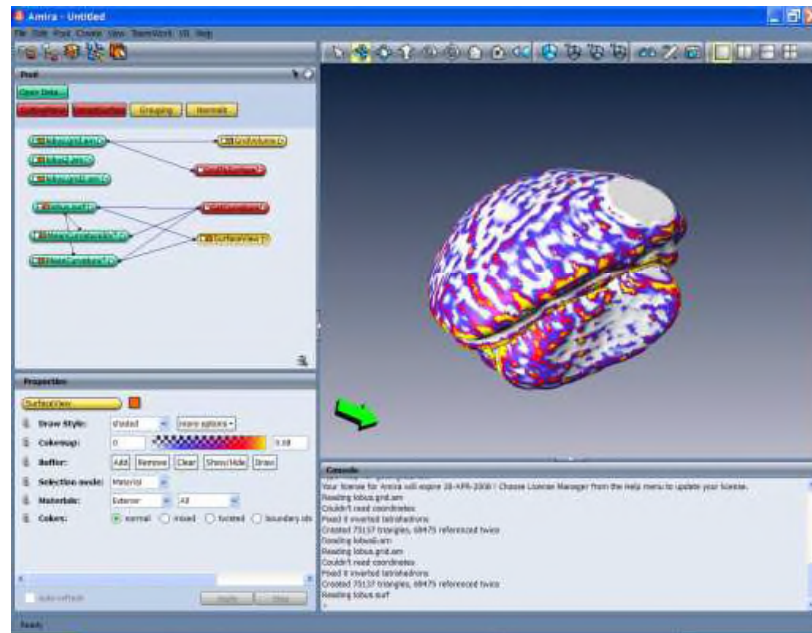


Figure 1.1: Data objects and modules are represented as little icons in the Pool (top left). In the 3D graphic window a surface colored according to its curvature is shown. Curvature information has been computed by a computational module and is stored as a separate data object. In the left window the parameters of selected modules are shown. The lower right pane provides a Tcl-command shell as well as access to the help browser.

integrate the data from multiple sources into a single environment.

This section summarizes the main features of Amira software suite. For more complete information you may browse indexes for data types, file formats and modules in the Reference Guide. This is accessible from the home page of the online help browser.

Section 1.4 describes the Amira optional extensions.

1.2.1 Data import

Amira can load directly different types of data, including:

- 2D and 3D image and volume data
- Geometric models such as point sets, line sets, surfaces, grids
- Numerical simulation data

- Time series and animations

A large number of file formats are supported in the standard edition or through specific optional readers. For an introduction to data import, see Chapter 2. For more details, see section 4.1.

1.2.2 Viewing, navigation, interactivity

All visualization techniques can be arbitrarily combined to produce a single scene. Moreover, multiple data sets can be visualized simultaneously, either in several viewer windows or in a common one. Thus you can display single or multiple data sets in a single or multiple viewer windows, and navigate freely around or through those objects.

A built-in spatial *transformation editor* makes it easy to register data sets with respect to each other or to deal with different coordinate systems. Automatic registration of volume or geometric data is also possible.

Direct interaction with the 3D scene allows you to quickly control regions of interest, slices, probes and more.

Combinations of data sets, representation and processing features can be defined with minimal user interaction for simple or complex tasks. See section 1.1.

1.2.3 Visualization of 3D Image Data

1.2.3.1 Slicing and Clipping

You can quickly explore 3D images looking at single or multiple orthographic or oblique sections. Multiple data sets can be superimposed on slices, or displayed as height fields. You can cut away parts of your data to uncover hidden regions. Curved or cylinder slices are also available.

1.2.3.2 Volume Rendering

One of the most intuitive and most powerful techniques for visualizing 3D image data is *direct volume rendering*. Light emission and light absorption parameters are assigned to each point of the volume. Simulating the transmission of light through the volume makes it possible to display your data from any view direction without constructing intermediate polygonal models. By exploiting modern graphics hardware, Amira is able to perform direct volume rendering in real time, even

on very large data when using the Very Large Data Option. Thus volume rendering can instantly highlight relevant features of your data. Volume rendered images can be combined with any type of polygonal display. This improves the usefulness of this technique significantly. Moreover, multiple data sets can be volume rendered simultaneously – a unique feature of Amira. Transfer functions with different characteristics required for direct volume rendering can be either generated automatically or edited interactively using an intuitive colormap editor.

1.2.3.3 Isosurfaces

Isosurfaces are most commonly used for analyzing arbitrary scalar fields sampled on discrete grids. Applied to 3D images, the method provides a very quick, yet sometimes sufficient method for reconstructing polygonal surface models. Beside standard algorithms, Amira provides an improved method, which generates significantly fewer triangles with very little computational overhead. In this way, large 3D data sets can be displayed interactively even on smaller desktop graphics computers.

1.2.3.4 Large Volume Data

With the Very Large Data Option, even very large data sets that cannot be fully loaded in memory can be manipulated at interactive speed. Multi-resolution techniques can manage and visualize extremely large amounts of volume data of up to hundreds of gigabytes. You can then, for instance, quickly select a region of interest and extract down-sampled or partial data for further processing.

1.2.4 Image processing

1.2.4.1 Alignment of image slices

The image *slice aligner* enables you to build a consistent stack of images with manual or automatic tools, if, for instance, physical cross-sections have been shifted during image acquisition.

1.2.4.2 Image filters

Image features can be enhanced by applying a wide range of filters for controlling contrast, smoothing, noise reduction and much more. See *Image Filters* and *Quantification+* Option.

1.2.4.3 Image segmentation

Segmentation means assigning labels to image voxels that identify and separate objects in a 3D image. Amira offers a large set of segmentation tools, ranging from purely manual to fully automatic: brush (painting), lasso (contouring), magic wand (region growing), thresholding, intelligent scissors, contour fitting (snakes), contour interpolation and extrapolation, wrapping, smoothing and de-noising filters, morphological filters for erosion, dilation, opening and closing operations, connected component analysis, images correlation, objects separation and filtering (see Quantification+ Option), etc. See section 2.4 for a tutorial about image segmentation.

1.2.5 Model reconstruction

1.2.5.1 Surface generation

Once the interesting features in a 3D image volume have been segmented, Amira is able to create a corresponding polygonal surface model. The surface may have non-manifold topology if there are locations where three or more regions join. Even in this case the polygonal surface model is guaranteed to be topologically correct, i.e. free of self-intersections. Fractional weights that are automatically generated during segmentation allow the system to produce optionally smooth boundary interfaces. This way realistic high-quality models can be obtained, even if the underlying image data are of low resolution or contain severe noise artifacts. Making use of innovative acceleration techniques, surface reconstruction can be performed very quickly. Moreover, the algorithm is robust and fail-safe.

1.2.5.2 Surface Simplification and Editing

Surface simplification is another prominent feature of Amira. It can be used to reduce the number of triangles in an arbitrary surface model according to a user-defined value. Thus, models of finite-element grids, suitable for being processed on low-end machines, can be generated. The underlying simplification algorithm is one of the most elaborate ones available. It is able to preserve topological correctness, i.e., self-intersections commonly produced by other methods are avoided. In addition, the quality of the resulting mesh, according to measures common in finite element analysis, can be controlled. For example, triangles with long edges or triangles with bad aspect ratio can be suppressed.

A surface editor is also available for smoothing or refining surface in whole or part, cutting and copying parts of surfaces, defining boundary conditions for further numerical simulation, checking and modifying surface triangles.

1.2.5.3 Generation of Tetrahedral Grids

Amira allows you not only to generate surface models from your data but also to create true volumetric tetrahedral grids suitable for advanced 3D finite-element simulations. These grids are constructed using a flexible advancing-front algorithm. Again, special care is taken to obtain meshes of high quality, i.e., tetrahedra with bad aspect ratio are avoided. Several different file formats are supported, so that the grid can be exported to many standard simulation packages.

1.2.5.4 Point Clouds/Scattered Data

Amira can also reconstruct surfaces from scattered points (see *Delaunay* and *PointWrap* modules).

1.2.5.5 Skeletonization

A set of tools is included for reconstructing and analyzing a dendritic, porous or fracture network from 3D image data.

1.2.6 Visualization of 3D models and numerical data

1.2.6.1 Point sets, line sets

Amira can visualize arbitrary functional data given on 3D *point sets* or *line sets*.

1.2.6.2 Polygonal models

A number of drawing styles and coloring schemes help to yield meaningful and informative visualizations of polygonal models, whether generated from image data or imported from CAD or simulation package. Surface and 3D grid meshes can be colored or textured in order to visualize a second independent data set.

Another Amira feature comprises the realistic view-dependent way of rendering semi-transparent surfaces. By correlating transparency with local orientation of the surface relative to the viewing direction, complex spatial structures can be understood much more easily.

1.2.6.3 Numerical data post-processing

Amira allows you to analyze results of numerical simulations. It supports polygonal surfaces such as triangular meshes, 3D lattices with uniform, rectilinear or curvilinear coordinates, and polyhedral 3D grids such as tetrahedral or hexahedral grids. Most general purpose image visualization techniques and analysis tools can be applied, e.g.: slice extraction, computation of isolines or isosurfaces, data probing and histograms. In addition, scalar quantities can be visualized with pseudocolors on the grid itself.

Beside visualization, data representations such as isosurfaces, grid cuts or contour lines can be extracted as first class data objects.

Displacement vectors can be visualized on grids or applied as grid deformation that can be animated.

1.2.6.4 Flow Visualization

Amira provides many advanced tools for vector fields and flow visualization. Vector arrows can be drawn on a slice, within a volume, or upon a surface. The flow structure may be better revealed by representations such as fast Line Integral Convolution on slices or arbitrary surfaces, illuminated and animated streamlines, stream ribbons, stream surfaces, particle animations, synthetic *vector probe*... All of these stream visualization techniques are highly interactive. While seedpoint distributions can be automatically calculated, you can also select and interactively manipulate seed points and structures, thus supporting the investigation the flow field and highlighting of different features. Amira can also support six-component complex vector fields and phase visualization, e.g. electromagnetic fields.

See tutorial section [2.10](#) (visualization of vector fields).

1.2.6.5 Tensor Data

Amira has support for iconic visualization of tensor field, extraction of eigenvalues, computation of rate of strain tensor, gradient tensor.

1.2.7 General Data Processing and Data Analysis

1.2.7.1 3D registration of multiple data sets

Multiple data sets can be combined to compare images of different objects, or images of an object recorded at different times or with different imaging modalities

such as X-ray CT and MRI. In addition, fusion of multi-modal data by arbitrary arithmetic operations can be performed to increase the amount of information and accuracy in the models. Amira allows manual registration through interactive manipulators, automatic rigid or non-rigid registration through landmarks, and automatic registration using iterative optimization algorithms (see *AffineRegistration* module).

Surfaces can also be registered using rigid or non-rigid transformations, based on landmarks sets warping, alignment of centers or principal axes, or distance minimization algorithms.

1.2.7.2 Operating on 3D data

Many utilities are available for data processing. Here are some important ones. Re-sampling can reduce or enlarge the resolution of a 3D image or data sets defined on regular grids, and different sampling kernels are supported. Data can be cropped or regions of interest can be defined. Data can be converted to any supported primitive type, from byte to 64-bits floating point numbers. Multi-component data such as multi-channel images or vector data can be composed or decomposed. Standard 3D field operators such as scalar field gradient or vector field curl are available. Surface curvatures and distances between surfaces can also be computed, as scalar or vector information. The powerful *Arithmetic* module allows to perform calculations on data sets with user-defined expression, and can be used to interpolate data between regular grids and polyhedral grids. Data sets can also be created from arithmetic expressions.

1.2.7.3 Measurements, quantification

You can query the exact values of your data sets at arbitrary locations specified interactively by a mouse click, or along user-defined lines and spline curves. Probe points can serve to set interactively isosurfaces. You can plot or export the data for further processing with spreadsheet or plotting applications, with probing, measuring, counting, and other statistical modules quantify densities, distances, areas, volumes, mean value and standard deviation, ...

Histograms of values can be computed and plotted, possibly restricted to a region of interest. *Volume/value* diagrams can be also plotted.

The Quantification+ Option provides an extensive set of intensity and geometrical measurements on image or label data, either for individual labelled particles or as statistics. See the section dedicated to Quantification+ Option.

1.2.8 Matlab integration

You can integrate complex calculus using Matlab software from The Mathworks, Inc. by means of the *CalculusMatlab* module. This module allows for a connection to your Matlab server from your Amira session, and executes Matlab computations directly on your Amira data. It is also possible to import and export Matlab matrices to and from Amira, and export Amira surfaces to Matlab surfaces. See section [2.13](#).

1.2.9 High Performance Visualization

Amira makes extensive use of graphics hardware for optimal performance and rendering quality on your system. Moreover, the Virtual Reality Option allows for combining multiple graphics engines for high-performance requirements.

1.2.10 Automation, Customization, Extensibility

Tcl scripting

All Amira components can be controlled via a Tcl command language interface. Tcl scripts are used for saving your work session. Tcl scripts also allow the advanced user to automate or customize tasks with Amira for routine workflows, without the need for C++ programming. Custom Amira modules with user interface can even be created as Tcl scripts. Amira module behaviour and 3D interaction can be customized by using Tcl. Amira can also be used for batch processing. See Chapter [5](#), including a short introduction to the Tcl scripting language.

C++ programming

With the Virtual Reality Option, Amira can also be extended by programmers. The Developer Option allows for the creation of new custom components for Amira such as file readers and writers, computation modules, and even new visualization modules, using the C++ programming language. New modules and new data classes can be defined as subclasses of existing ones. In order to simplify the creation of new custom extensions, a development wizard is included. See the Developer Option User's Guide for detailed information.

1.3 Application Areas

Amira is successfully being used in a number of different application areas. Among these are:

- Medicine
- Biology
- Microscopy
- Computational Fluid Dynamics
- Neuroscience

Examples from these disciplines are illustrated by several *demo scripts* contained in the online version of the user's guide.

1.4 Options

Amira Options are additional sets of modules providing solutions for dedicated application areas. Options can be added to a standard Amira installation at any time. For each option a separate license is required. Currently, the following options are available for Amira:

- **Amira Developer Option** allows you to develop your own custom modules, file readers, and file writers using the C++ programming language.
- **Amira Molecular Option** adds advanced tools for the visualization of molecules. It combines Amira's strong capabilities for 3D data visualization such as hardware-accelerated volume rendering, with specific tools for molecular visualization and data analysis, such as molecular surfaces, sequence alignment, configuration density computation, and molecule trajectories. Amira Molecular Option includes a very powerful molecule editor.
- **Amira Mesh Option** supports mesh generation for flow, stress, and thermal analysis; for export of surface or 3D meshes to solvers; and for post-processing of data coming back from these solvers, providing very powerful visualization on scalar, vector, and tensor fields.
- **Amira DICOM reader** allows for import/export of DICOM data in Amira, making Amira the perfect application for medical data analysis.
- **Amira Microscopy Option** includes readers for most important microscopy file formats, 3D image deconvolution and a dedicated filament editor.

- **Amira Skeleton Option** supports reconstruction and analysis of porous, vascular and dendritic networks.
- **Amira Virtual Reality Option** is designed to enable the use of Amira's advanced data visualization and analysis features on immersive VR systems and tiled screen configurations. It has built-in support for efficient multi-threaded rendering on multipipe systems and for distributed rendering on cluster systems using application-level distribution. This approach leads to optimal performance with minimal bandwidth requirements. Tracking capabilities allow for a true immersive experience as well as interaction with the visualization.
- **Amira Very Large Data Option** manages and visualizes very large amounts of volume data, up to hundreds of gigabytes. The multi-resolution technique used in this option allows for interactive visualization and navigation through vast amounts of data.
- **Amira Quantification+ Option** includes new image processing capability as well as image analysis and quantification tools. Very powerful for material sciences, this option allows for efficient statistical analysis of data in this application area.
- **Specialized Readers** For the following file formats / applications specialized readers are available: **SEG-Y, CATIA 4, CATIA 5, IGES, STEP, TNO Madymo, and Mecalog Radioss**

Table 1.1 shows the license keyword associated with each of the Amira Options. For additional information about Amira and its options, please refer to the Amira web site, <http://www.amira.com/>.

Amira Option	License keyword
Developer Option	AmiraDev
Molecular Option	AmiraMol
Mesh Option	AmiraMesh
DICOM reader	AmiraDicomReader
Microscopy Option	ResolveRT
Skeleton Option	AmiraSkel
Virtual Reality Option	AmiraVR
Very Large Data Option	AmiraVLD
Quantification+ Option	AmiraQuant
Specialized readers:	
SEGY reader	AmiraSegyReader
Catia5 reader	AmiraCatia5Reader
Catia4 reader	AmiraCatia4Reader
IGES reader	AmiraIgesReader
STEP reader	AmiraStepReader
Madymo reader (TNO)	AmiraMadymoReader
Mecalog Radioss reader	AmiraRadiossReader

Table 1.1: Amira Options and their associated license keywords

2 First steps in Amira

This chapter contains step-by-step tutorials illustrating the use of Amira. The tutorials are almost independent of each other, so after reading the basics in the Getting Started section it is possible to follow each tutorial without knowing the others. If you go through all tutorials you will get a good survey of Amira's basic features. In particular, these topics will be covered:

- *Getting started* - the basics of Amira
- *Reading images* - how to read images
- *Visualizing 3D images* - slices, isosurfaces, volume rendering
- *Image segmentation* - segmentation of 3D image data
- *Surface reconstruction* - surface reconstruction from 3D images
- *Grid generation* - creating a tetrahedral grid from a triangular surface
- *Warping* - how to work with landmark sets
- *3D image registration* - how to register 3D image data sets
- *Alignment of 2D Physical Cross-Sections* - how to reconstruct a 3D model
- *Vector fields* - stream lines and other techniques
- *Filament Editor* - filament tracing for neurons and vessels images
- *The DemoMaker module* - creating animations with the DemoMaker module
- *Creating movie files* - how to use the MovieMaker module
- *Using MATLAB* - how to use the CalculusMatlab module
- *Trace Filaments* - how to extract a network model from a neuron 3D image data
- *Multi-planar Viewer* - visualize and register multiple volumes

In all tutorials the steps to be performed by the user are marked by a dot. If you only want to get a quick idea how to work with Amira you may skip the explanations between successive steps and just follow the instructions. But in order to get a deeper understanding you should refer to the text.

Note: If you want to visualize your own data, please first refer to Section 4.1. This section contains some general hints on how to import data sets into Amira.

Note for Mac users: Please read Section 4.7 before starting the tutorials.

2.1 Getting Started

In this section you will learn how to

1. start the program
2. load a demo data set into the system
3. invoke editors for editing the data
4. connect visualization modules to the data
5. interact with the 3D viewer.

The following text has the form of a short step-by-step tutorial. Each step builds on the steps described before. We recommend that you read the text online and carry out the instructions directly on the computer. Instructions are indicated by a dot so you can execute them quickly without reading the explanations between the instructions.

- Windows, select the Amira icon from the start menu.
- Mac, select the Amira icon from the Application menu.
- Unix-Linux, start Amira by entering Amira in a shell window.

If there is no such command, the software has not been properly installed. In this case try to execute the script `bin/start` located in the `AMIRA_ROOT` directory. When Amira is running, a window like the one shown in Figure 2.1 appears on the screen. The user interface is divided into four major regions. The 3D viewer window displays visualization results, e.g., slices or isosurfaces. The Pool will contain small icons representing data objects and modules. The Properties Area displays interface elements (*ports*) associated with Amira objects. Finally, the lower left pane is shared by the console and Amira's integrated hypertext help browser. Click on the Console or Help tab to select which window you want to view. The console prints system messages and lets you enter Amira commands. You can use the help browser to read the user's guide online.

You can also activate the help browser by pressing F1, selecting *User's Guide* from the *Help* menu of Amira's main window, or by typing `help` in the console window.

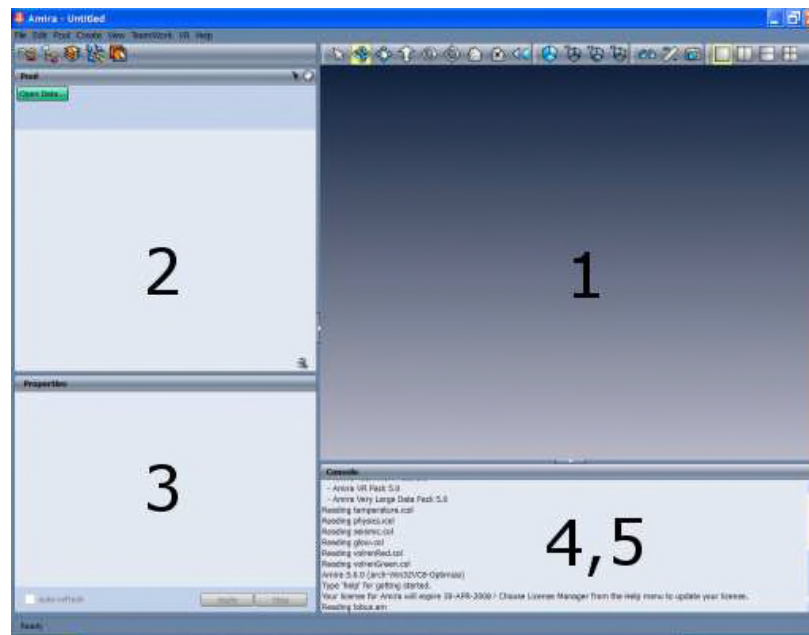


Figure 2.1: The Amira user interface consists of five major parts: the 3D viewer (1), the Pool (2), the Properties Area (3), the console window (4), and the help browser (5).

2.1.1 Loading Data

Usually, the first thing you will do after starting Amira is to load a data set. Let's see how this can be done:

- Choose *Open Data ...* from the *File* menu.

After selecting this menu item, the file dialog appears (see Figure 2.2). By default the dialog displays the contents of the first directory defined in the environment variable `AMIRA_DATADIR`. If no such variable exists the contents of Amira's demo data directory are displayed. You can quickly switch to other directories, e.g., to the current working directory, using the directory list located in the upper part of the dialog window.

At the top of the Pool is an Open Data button which is a shortcut to the File/Open Data dialog. You may use it for opening data files in the tutorials that follow. However, the tutorials will instruct you to use the File/Open Data command.

Amira is able to determine many file formats automatically, either by analyzing the file header or the file name suffix. The format of a particular file will be printed in

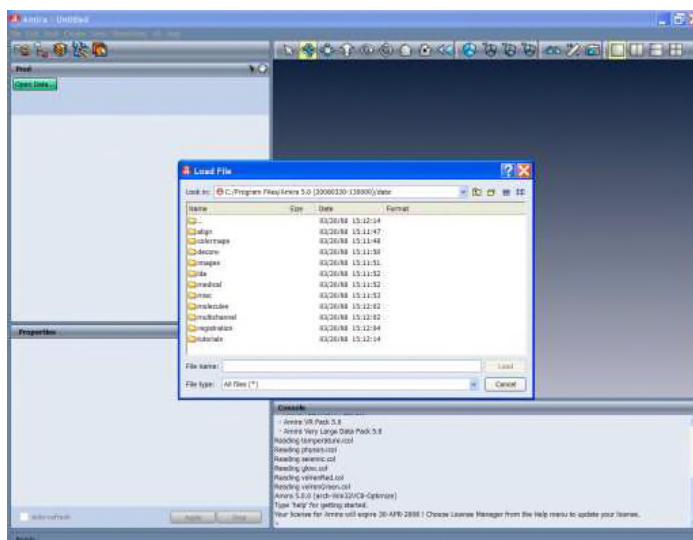


Figure 2.2: Data sets can be loaded into Amira using the file browser. In most cases, the file format can be determined automatically. This is done by either analyzing the file header or the file name suffix.

the file dialog right beside the file name.

Now, we would like to load a scalar field from one of the demo data directories contained in the Amira distribution.

- Change to the directory `data/tutorials`, select the file `lobus.am` and press *OK*.

The data will be loaded into the system. Depending on its size this may take a few seconds. The file is stored in Amira's native *AmiraMesh* format. The file `lobus.am` contains 3D image data of a part of a fruit fly's brain, namely an optical lobe, obtained by confocal microscopy. This means the data represents a series of parallel 2D image slices across a 3D volume. Once it has been loaded, the data set appears as a green icon in the Pool. In the following we call this data set "lobus data set".

- Click on the green data icon with the left mouse button to select it.

This causes some information about the data record to be displayed in the Properties Area (Figure 2.3). In our case we can read off the dimensions of the data set, the primitive data type, the coordinate type, as well as the voxel size. To deselect

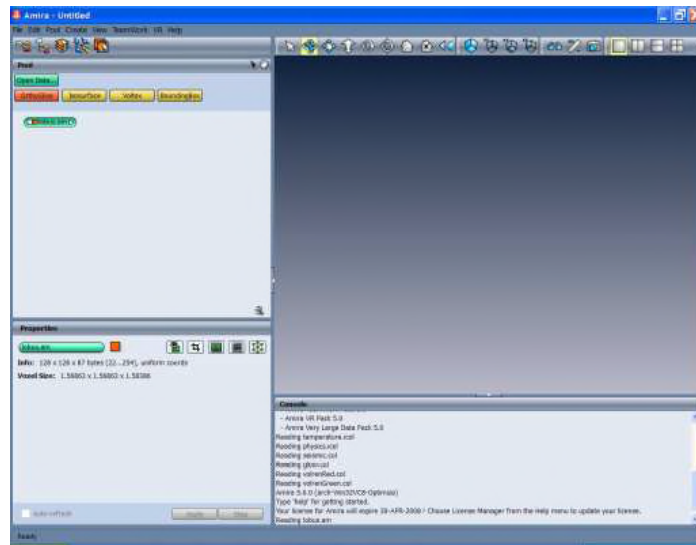


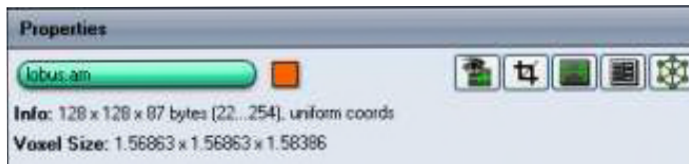
Figure 2.3: Data objects are represented by green icons in the Pool. Once an icon has been selected information about the data set such as its size or its coordinate type is displayed in the Properties Area.

the icon, click on an empty area in the Pool window. You may also pick the icon with the left mouse button and drag it around in the Pool.

Clicking on an object typically causes additional buttons to be displayed in the button area at the top of the Pool. These buttons are convenience buttons allowing easy one-click access to the modules most frequently used by the selected object. The tutorials, however, will have you to access modules via the menu interface to help familiarize you with the organization of modules within Amira.

2.1.2 Invoking Editors

After selecting an object, in addition to the textual information, some buttons appear in the Properties Area, to the far right of the data object's name. These buttons represent *editors* which can be used to interactively manipulate the data object in some way. For example, all data objects provide a *parameter editor*. This editor can be used to edit arbitrary attributes associated with the data set, e.g. filename, original size, or bounding box. Another example is the *transform editor* which can be used to translate or rotate the data in world coordinates. However, at this point we don't want to go into details. We just want to learn how to create and delete an editor:



- Invoke one of the editors by clicking on an editor icon.
- Close the editor by clicking again on the editor icon.

Further information about particular editors is provided in the user's reference manual.

2.1.3 Visualizing Data

Data objects like the lobus data can be visualized by attaching *display modules* to them. Each icon in the Pool provides a popup menu from which matching modules, i.e., modules that can operate on this specific kind of data, can be selected. To activate the popup menu

- click with the right mouse button on the green data icon. Choose the entry called *BoundingBox*.

After you release the mouse button, a new *BoundingBox* module is created and is automatically connected to the data object. The *Bounding Box* object is represented by a yellow icon in the Pool and the connection is indicated by a blue line connecting the icons. At the same time, the graphics output generated by the *BoundingBox* module becomes visible in the 3D viewer. Since the output is not very interesting, in this case we will connect a second display module to the data set:

- Choose the entry called *OrthoSlice* from the popup menu of the lobus data set.

Now a 2D slice through the optical lobe is shown in the viewer window. Initially, a slice oriented perpendicular to the z-direction and centered inside the image volume is displayed. Slices are numbered 0, 1, 2, and so on. The slice number as well as the orientation are parameters of the *OrthoSlice* module. In order to change these parameters, you must select the module. As for the green data icon, this is done by clicking on the *OrthoSlice* icon with the left mouse button. By the way, in contrast to the *BoundingBox*, the *OrthoSlice* icon is orange, indicating that this module can be used for clipping.

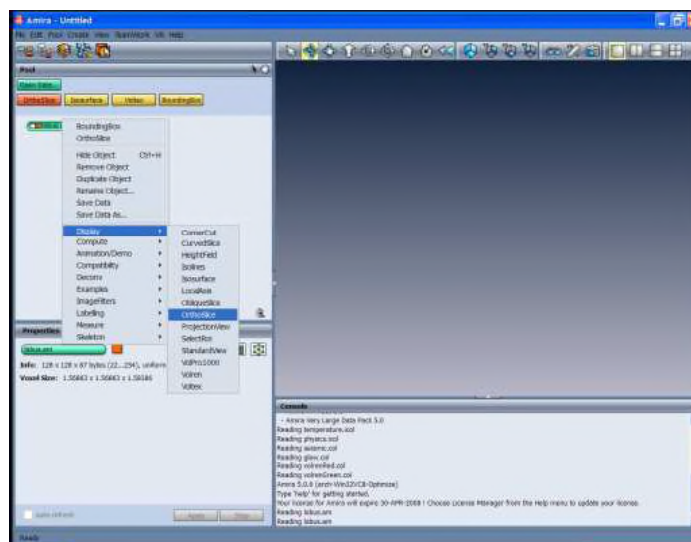


Figure 2.4: In order to attach a module to a data set, click on the green icon using the right mouse button. A popup menu appears containing all modules which can be used to process this particular type of data.

- Select the *OrthoSlice* module.

Now you should see various buttons and sliders in the Properties Area, ordered in rows. Each row represents a *port* allowing you to adjust one particular control parameter. Usually, the name of a port is printed at the beginning of a row. For example, the port labeled *Slice Number* allows you to change the slice number via a slider.

- Select different slices using the *Slice Number* port.

By default, *OrthoSlice* displays slices with axial orientation, i.e., perpendicular to the z-direction. However, the module can also extract slices from the image volume perpendicular to x- and y-direction. These two alternate orientations are sometimes referred to as *sagittal* and *coronal* (these are standard phrases used in radiology).

2.1.4 Interaction with the Viewer

The 3D viewer lets you look at the model from different positions. If you click on the *Trackball* button in the viewer toolbar, moving the mouse inside the viewer

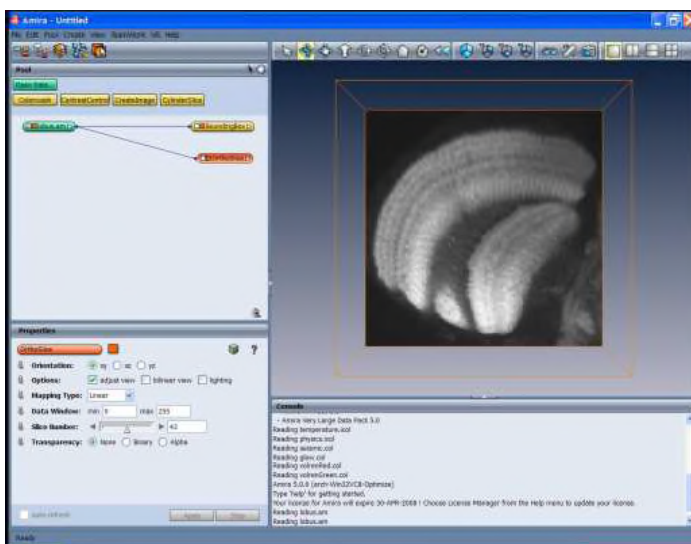


Figure 2.5: Visualization results are displayed in the 3D viewer window. Parameters or ports of a module are displayed in the Properties Area after you select the module.

window with the left mouse button pressed lets you rotate the object. If you click on the *Translate* or the *Zoom* buttons, you can translate or zoom the object. (For zoom, move the mouse up and down.)

Alternatively, with the middle mouse button pressed you can translate the object. For zooming press both the left and the middle mouse buttons at the same time and move the mouse up or down.

Notice that the mouse cursor has the shape of a little hand inside the viewer window. This indicates that the viewer is in viewing mode. By pressing the **ESC** key you can switch the viewer into interaction mode. In this mode, interaction with the geometry displayed in the viewer is possible by mouse operations. For example, when using *OrthoSlice* you can change the slice number by clicking on the slice and dragging it.

- Select different buttons of the *Orientation* port of the *OrthoSlice* module.
- Rotate the object in a more general position.
- Disable the *adjust view* toggle in the *Options* port.
- Change the orientation using the *Orientation* port again.
- Choose different slices using the *Slice Number* port or directly in the viewer

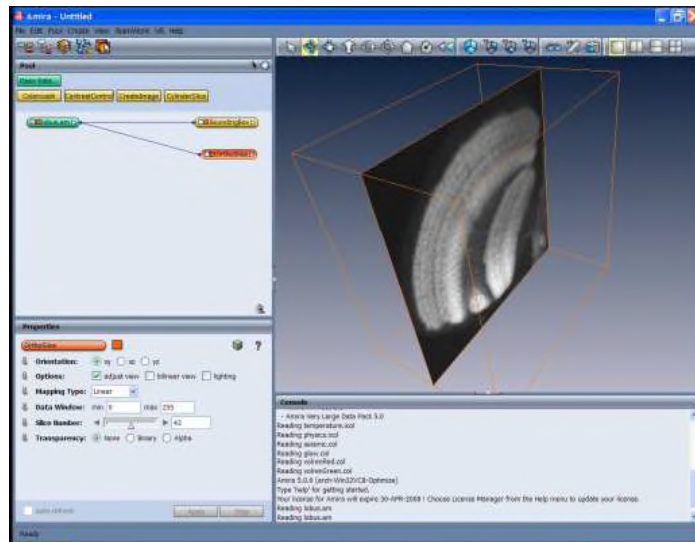
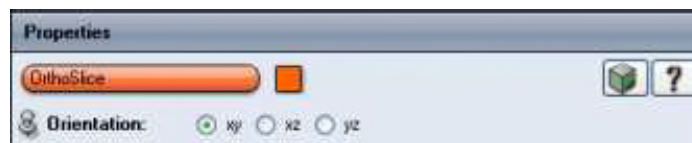


Figure 2.6: The *OrthoSlice* module is able to extract arbitrary orthogonal slices from a regular 3D scalar field or image volume.

with the interaction mode described above.

Each display module has a *viewer toggle* by which you can switch off the display without removing the module. This button is just to the right of the colored bar where the module name is shown as illustrated below.

- Deactivate and activate the display of the *OrthoSlice* or *BoundingBox* module using the *viewer toggle*.



If you want to remove a module permanently, select it and choose *Remove Object* from the *Pool* menu. Choose *Remove All* from the same menu to remove all modules.

- Remove the *BoundingBox* module by selecting its icon and choosing *Remove Object* from the *Pool* menu.
- Remove all remaining modules by choosing *Remove All Objects* from the same menu.

Now the Pool should be empty again. You may continue with the next tutorial, i.e., the one on *scalar field visualization*.

2.2 How to load image data

Loading image data is one of the most basic operations in Amira. Other than with 2D images, there are not many standardized file formats containing 3D images. This tutorial guides you by means of examples on how to load the different kinds of 3D images into Amira. In particular this tutorial covers the following topics:

1. Using the *File/Open Data...* browser and setting the file format.
2. Reading 3D image data from multiple 2D slices.
3. Setting the bounding box or voxel size of 3D images.
4. The *Stacked Slices* file format.
5. Working with LargeDiskData.

2.2.1 The Amira File Browser

Image data is loaded in Amira with the *File/Open Data...* dialog. All *file formats* supported by Amira are recognized automatically either by a data header or by the file name suffix. What follows is only of concern in these cases:

- The automatic file format detection fails.
- 3D image data is stored in several 2D files.
- The data is larger than the available main memory.

Setting the file format

In most cases the format of a file is determined automatically, either by checking the file header or by comparing the file name suffix with a list of known suffixes. In the load dialog the file format is displayed in a separate column in detail view.

Example:

- Files containing the string `AmiraMesh` in the first line are considered *AmiraMesh* files.
- Files with the suffix `.stl` are considered STL files.

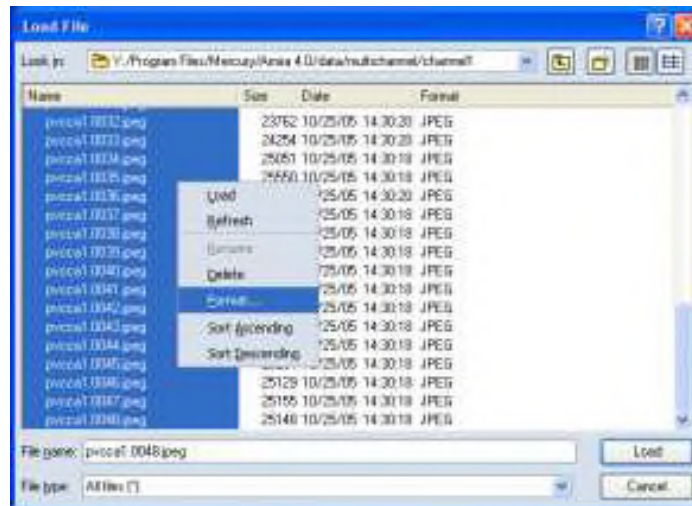


Figure 2.7: The *Format* option of the file browser

If automatic file format detection fails, e.g. because some non-standard suffix has been used, the format may be set manually using the *Format* entry in the pop-up menu of the *Load* dialog (right mouse button).

2.2.2 Reading 3D Image Data from Multiple 2D Slices

A common way to store 3D image data is to write a separate 2D image file for each slice. The 2D images may be written in TIFF, BMP, JPEG, or any other supported image *file format*. In order to load such data in Amira, all 2D slices must be selected simultaneously in the file browser. This can be done by clicking the first file and shift clicking the last one.

- Open the *File/Open Data...* dialog.
- Browse to the `/Amira-5/data/multichannel1/channel/` directory.
- Select the first file `pvcca1.0001.jpeg`
- Shift-click the last file (`pvcca1.0048.jpeg`).
- Click *Load*.



Figure 2.8: Loading multiple 2D images

2.2.3 Setting the Bounding Box

When loading a series of bitmap images, usually the physical dimensions of the images are not known to Amira. Therefore an *Image Read Parameters* dialog appears that prompts you for entering the physical extent of the *bounding box*. Alternatively, the size of a single voxel can be set. In Amira the bounding box of an object is the smallest rectangular, axis-aligned volume in 3D space that encompasses the object. *Note that in Amira the bounding box of a uniform data set extends from the center of the first voxel to the center of the last one. For example, if you have 256 voxels and you know the voxel size to be 1 mm, the bounding box should be set to 0 - 255 (or to some shifted range).*

- Enter 0.85 in the first and second text fields and 3.5 in third text field of the *Voxel Size* port.
- Click *OK*.

This method will always create a data set with uniform coordinates, i.e., uniform slice distance. In case of variable slice distances, the *StackedSlices* format should be used.

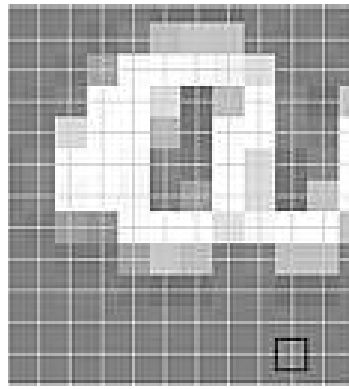


Figure 2.9: The definition of the bounding box in Amira. Different gray shades depict the intensity values defined on the regular grid (white lines). The black square depicts the extent of one voxel. The outer frame depicts the extent of the bounding box.

2.2.4 The Stacked Slices file format

Especially with histological serial sections it often happens that slices are lost during preparation. To handle such cases, Amira provides a special data type corresponding to a file format, called *Stacked Slices*. This file format allows a stack of individual image files to be read with optional z- values for each slice. The slice distance is not required to be constant. The images must be one-channel or RGBA images in an image format supported by Amira (e.g. TIFF). The reader operates on an ASCII description file, which can be written with any editor. Here is an example of a description file:

```
# Amira Stacked Slices
# Directory where image files reside
pathname C:/data/pictures
# Pixel size in x- and y-direction
pixelsize 0.1 0.1
# Image list with z-positions
picture1.tif 10.0
picture7.tif 30.0
picture13.tif 60.0
colstars.jpg 330.0
```

end

Some remarks on the syntax:

- `# Amira Stacked Slices` is an optional header that allows Amira to automatically determine the file format.
- `pathname` is optional and can be included if the pictures are not in the same directory as the description file. A space separates the tag "pathname" from the actual pathname.
- `pixelsize` is optional, too. The statement specifies the pixel size in x- and y-directions. The bounding box of the resulting 3D image is set from 0 to `pixelsize*(number_of_pixels-1)`.
- `picture1.tif 10.0` is the name of the slice and its z-value, separated by a space character.
- `end` indicates the end of the description file.
- Comments are indicated by a hash-mark character (#).

2.2.5 Working with Large Disk Data

Sometimes image data are so large that they do not fit into the main memory of the computer. Since the Amira visualization modules rely on the fact that data are in physical memory, this would mean that such data cannot be displayed in Amira. To overcome this, a special purpose module is provided that leaves most of the data on disk and retrieves only a user-specified subvolume. This subvolume can then be visualized with the standard visualization modules in Amira.

- Use the *File/Open Data...* dialog and go to `c:/Program Files/Amira-5/data/medical/`
- Right-click on the `reg005.ctdata.am` and select the *Format* entry from the pop-up dialog
- Select *AmiraMesh as LargeDiskData* as format and confirm your choice with *OK*.
- Press the *Load* button.

The data will be displayed in the Pool as a regular green data icon. The info line indicates that it belongs to the data class *HxRawAsExternalData*.

- Right mouse click, attach a *BoundingBox* module.

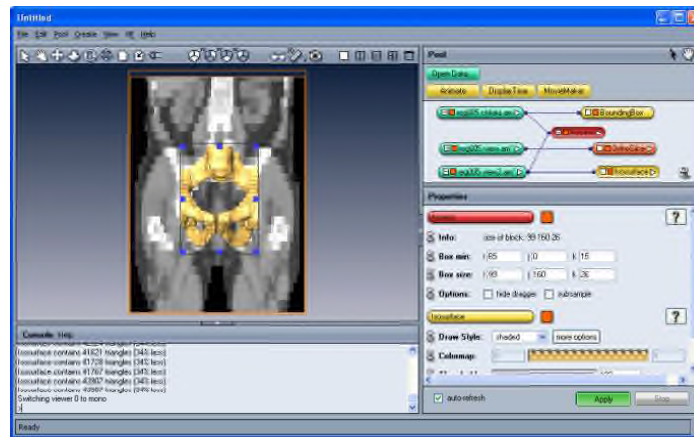


Figure 2.10: The usage of *AmiraMesh* as *LargeDiskData*. For instantaneous update, the *auto-refresh* check boxes of the *Access* and *Isosurface* modules have been checked

- Right mouse click, attach an *Access* module.
- Select the *Access* module in the Pool and enter 224, 161, and 59 into the *BoxSize* text fields.
- Check *Subsample* and enter 4 4 2 into the *Subsample* fields and press the *Apply* button.

This retrieves a down-sampled version of the data. Disconnect the *reg005.view.am* data icon from the *Access* module and use it as an overview (e.g. with *OrthoSlice*).

- Select the *Access* module in the Pool and deselect the *subsample* check box.
- Use the dragger box in the viewer to resize the subvolume.
- Press the *Apply* button.
- Attach an *Isosurface* module to the *reg005.view2.am* (set threshold set to 100).

Tip: To browse the data, check the *auto-refresh* check box for the *Access* and *Isosurface* modules. Now each time the blue subvolume dragger is repositioned, the visualization is updated automatically.

Loading *AmiraMesh*, *StackedSlices*, and *Raw* "asLargeDiskData" is a convenient and fast way of exploring data that exceed the size of system memory. However, especially with *StackedSlices*, it is not always the most efficient way of doing this. Amira can store the image data in a special format that facilitates the random

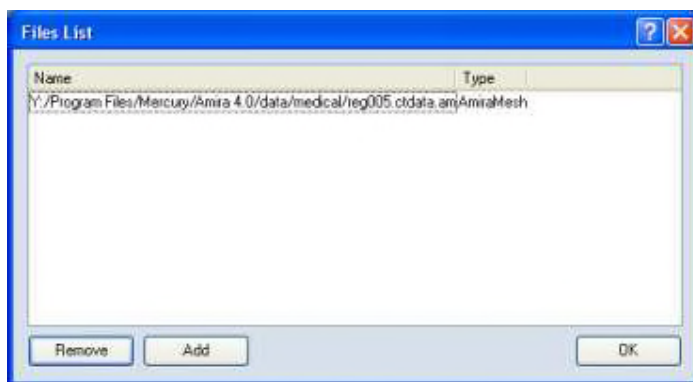


Figure 2.11: The *Input* dialog of the *ConvertToLargeDiskData* module.

retrieval of data from disk.

- Choose from the *Create/Data* menu *ConvertToLargeDiskData*
- Click *Browse* from the *Input* port.
- Click *Add*, go to `/Amira-5/data/medical/` and select `reg005.ctdata.am`, then click *Load*.
- Click *Browse* from the *Output* port.
- Go to `C:/tmp/` and enter a filename of your choice.
- Press the *Apply* button.

Although you will most likely not notice any difference with the small image data used in this tutorial, this method for retrieving large data significantly accelerates the *Apply* operation.

2.3 Visualizing 3D Images

This section provides a step-by-step introduction to the visualization of regular scalar fields, e.g., 3D image data. Amira is able to visualize more complex data sets, such as scalar fields defined on curvilinear or tetrahedral grids. Nevertheless, in this section we consider the simplest case, namely scalar fields with regular structure. Each step builds on the step before. In particular, the following topics will be discussed:

1. orthogonal slices

2. simple threshold segmentation
3. resampling the data
4. displaying an isosurface
5. cropping the data
6. volume rendering

We start by loading the data you already know from Section 2.1 (Getting Started): a 3D image data set of a part of a fruit fly's brain. The data set has been recorded with a confocal laser scanning microscope at the University of Wuerzburg.

- Load the file `lobus.am` located in subdirectory `data/tutorials`.

2.3.1 Orthogonal Slices

The fastest and in many cases most "standard" way of visualizing 3D image data is by extracting orthogonal slices from the 3D data set. Amira allows you to display multiple slices with different orientations simultaneously within a single viewer.

- Connect a *BoundingBox* module to the data (use right mouse on `lobus.am`).
- Connect an *OrthoSlice* module to the data.
- Connect a second and third *OrthoSlice* module to the data.
- Select *OrthoSlice2* and press *xz* or *coronal* in the *Orientation* port.
- Similarly, for *OrthoSlice3* choose *yz* or *sagittal* orientation.
- Rotate the object in the viewer to a more general position.
- Change the slice numbers of the three *OrthoSlice* modules in the respective ports or directly in the viewer as described in section Getting Started.

In addition to the *OrthoSlice* module, which allows you to extract slices orthogonal to the coordinate axes, Amira also provides a module for slicing in arbitrary orientations. This more general module is called *ObliqueSlice*. You might want to try it by selecting it from the Display submenu of the `lobus` data popup menu.

2.3.2 Simple Data Analysis

The values of the *data window* port of the *OrthoSlice* module determine which scalar values are mapped to black or white, respectively. If you choose a range of e.g., 30...100, any value smaller or equal to 30 will become black, and all pixels with an associated value of more than 100 will become white. Try modifying the

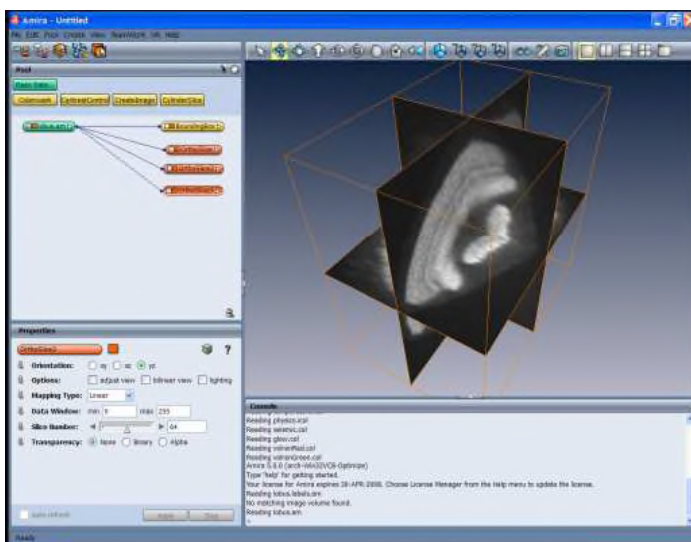


Figure 2.12: Lobus data set visualized using three orthogonal slices.

range. This port provides a simple way of determining a threshold, which later can be used for segmentation, e.g., in biology or medicine to separate background pixels from anatomical structures. This can be most easily done by making the minimum and maximum values coincide.

- Remove two of the *OrthoSlice* modules.
- Select the remaining *OrthoSlice* module.
- Make sure that the *mapping type* is set to *linear*.
- Change the minimum and maximum values of the data window port until these values are the same and a suitable segmentation result is obtained. For this data set 85 should be a good threshold value.

A more powerful way of quantitatively examining intensity values of a data set is to use a data probing module *PointProbe* or *LineProbe*. However, we will not discuss these modules in this introductory tutorial.

2.3.3 Resampling the Data

Now we are going to compute and display an *isosurface*. Before doing so, we will resample the data. The resampling process will produce a data set with a coarser resolution. Although this is not necessary for the isosurface tool to work,

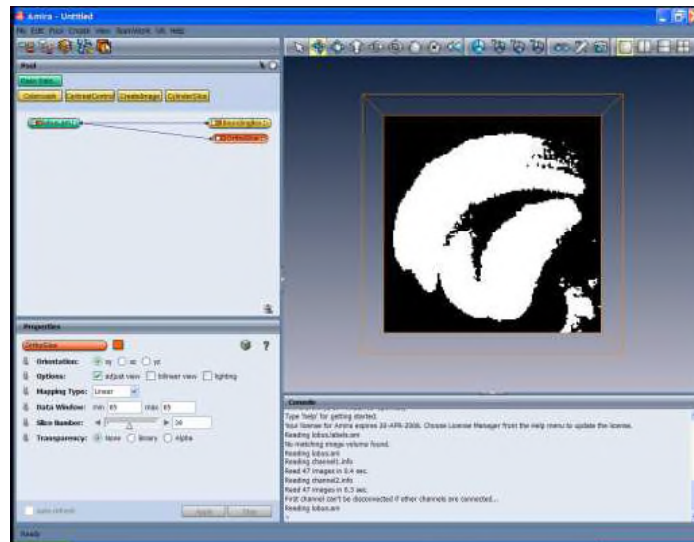


Figure 2.13: By adjusting the data window of the *OrthoSlice* module a suitable value for threshold segmentation can be found. Intensity values smaller than the min value will be mapped to black, intensity values bigger than the max value will be mapped to white.

it decreases computation time and improves rendering performance. In addition, you will get acquainted with another type of module. The *Resample* module is a computational module. Computational modules are represented by red icons. Typically you must press the green *Apply* button at the bottom of the Properties Area to start the computation. After you press this button they produce a new data object containing the result.

- Connect a *Resample* module to the data and select it.
- Enter values for a coarser resolution, e.g., $x=64$, $y=64$, $z=43$.
- Press the *Apply* button.

A new green data icon representing the output of the resample computation named *lobus.Resampled* is created. You can treat this new data set like the original lobus data. In the popup menu of the resampled lobus you will find exactly the same attachable modules and you can save and load it like the original data.

You may want to compare the resampled data set with the original one using the *OrthoSlice* module. You can simply pick the blue line indicating the data connection and drag it to a different data source. Whenever the mouse pointer is over a

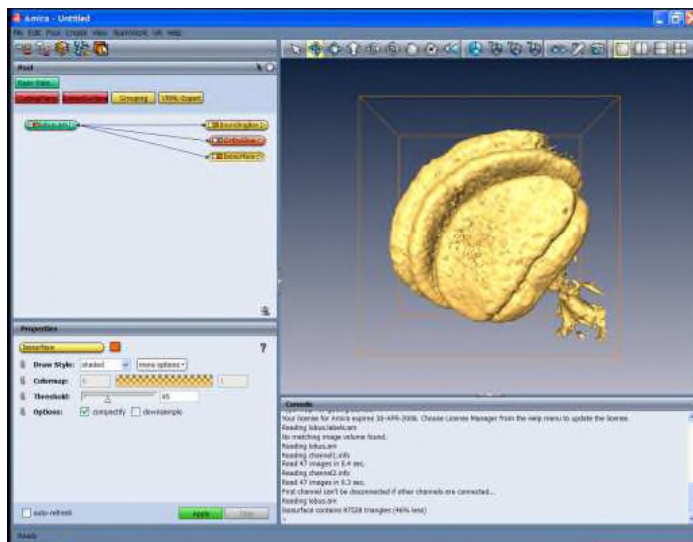


Figure 2.14: Lobus data set visualized in 3D using an isosurface.

valid source, the connection line appears highlighted in lighter blue.

2.3.4 Displaying an Isosurface

For 3D image data sets, isosurfaces are useful for providing an impression of the 3D shape of an object. An isosurface encloses all parts of a volume that are brighter than some user-defined threshold.

- Turn off the viewer toggle of the *OrthoSlice* module.
- Connect an *Isosurface* module to the resampled data record and select it.
- Adjust the threshold port to 85 or a similar value.
- Press the *Apply* button.

2.3.5 Cropping the Data

Cropping the data is useful if you are interested in only a part of the field. A crop editor is provided for this purpose. Its use is described below:

- Remove the resampled data *lobus.Resampled*.
- Activate the display of the *OrthoSlice* module.

- Select the *lobus.am* data icon.
- Click on the Crop Editor button in the Properties Area.

A new window pops up. There are two ways to crop the data set. You can either type the desired ranges of x, y, and z coordinates into the crop editor's window or put the viewer into interaction mode and adjust the crop box using the green handles directly in the viewer window.

- Put the viewer into interaction mode.
- With the left mouse button, pick one of the green handles attached to the crop volume. Drag and transform the volume until the part of the data you are interested in is included.
- Press *OK* in the crop editor's dialog window.

The new dimensions of the data set are given in the Properties Area. If you want to work with this cropped data record in later sessions you should save it by choosing *Save Data As ...* from the *File* menu.

As you already might have noticed, the crop editor also allows you to rescale the bounding box of the data set. By changing the bounding box alone, no voxels will be cropped. You may also use the crop editor to enlarge the data set, e.g., by entering a negative value for the *k min* number. In this case the first slice of the data set will be duplicated as many times as necessary. Also, the bounding box will be updated automatically.

2.3.6 Volume Rendering

Volume Rendering is a visualization technique that gives a 3D impression of the whole data set without segmentation. The underlying model is based on the emission and absorption of light that pertains to every voxel of the view volume. The algorithm simulates the casting of light rays through the volume from pre-set sources. It determines how much light reaches each voxel on the ray and is emitted or absorbed by the voxel. Then it computes what can be seen from the current viewing point as implied by the current placement of the volume relative to the viewing plane, simulating the casting of sight rays through the volume from the viewing point.

You can choose between two different methods for these computations: *maximum intensity* projections or an ordinary emission-absorption model.

- Remove all objects in the Pool other than the *lobus.am* data record.

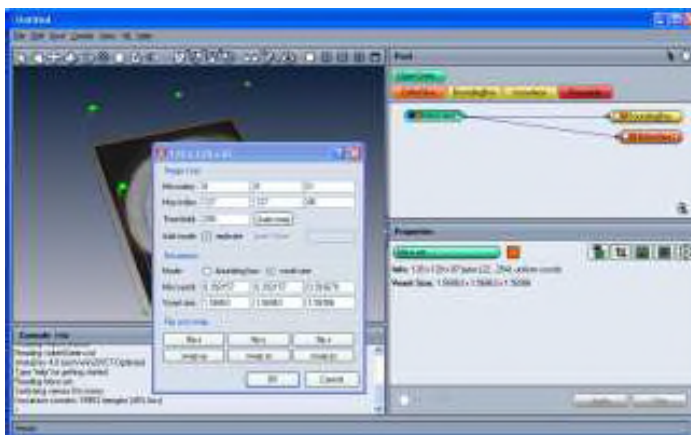


Figure 2.15: The crop editor works on uniform scalar fields. It allows you to crop a data set, to enlarge it by replicating boundary voxels, or to modify its coordinates, i.e. to scale or shift its bounding box.

- Connect a *Vortex* module to the data.
- Select the data icon and read off the range of data values printed on the first info line (22...254).
- Select the *Vortex* module and enter the range in the *Colormap* port.
- Press the *Apply* button in order to perform some texture preprocessing which is necessary for visualizing the data.

By default, emission-absorption volume rendering is shown. The amount of light being emitted and absorbed by a voxel is taken from the color and alpha values of the colormap connected to the *Vortex* module. In our example the colormap is less opaque for smaller values. You may try to set the lower bound of the colormap to 40 or 60 in order to get a better feeling for the influence of the *transfer function*. In order to compute *maximum intensity* projections, choose the *mip* option of port *Options*.

Internally, the *vortex* module makes heavy use of OpenGL texture mapping. Both texture modes, 2D and 3D, are implemented. 3D textures yield slightly better results. However, this mode is not supported by all graphics boards. The 3D texture mode requires you to adjust the number of slices cut through the image volume. The higher this number the better the results are.

Alternatively, 2D textures can be used for volume rendering. In this case, slices perpendicular to the major axes are used. You may observe how the slice orienta-

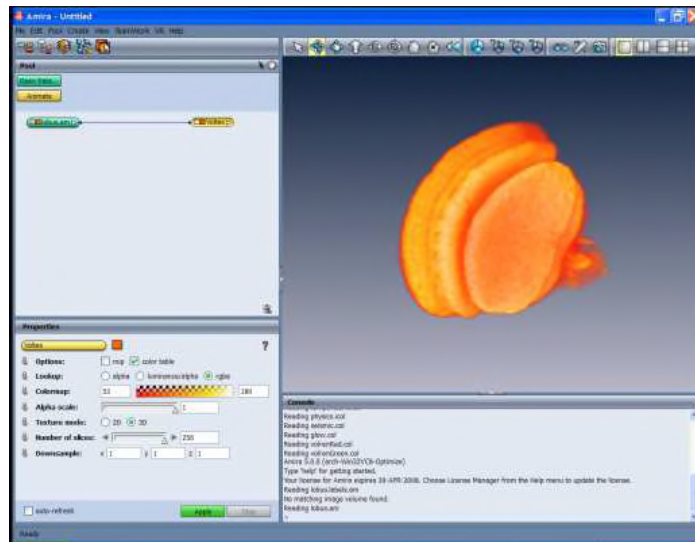


Figure 2.16: The *Voltex* module can be used to generate maximum intensity projections as well as volume renderings based on an emission-absorption model. In both cases, 2D or 3D texture mapping techniques can be applied.

tion changes if you slowly rotate the data set. The 2D texture mode is well suited for mid-range graphics workstations with hardware accelerated texture mapping. If your computer does not support hardware texture mapping at all, you should use visualization techniques other than volume rendering.

- Make sure the *mip* button of port *Options* is unchecked.
- If 3D texture mode is enabled, choose about 200 slices.
- Click with the right mouse button on port *Colormap* and choose *vol-renRed.icol*.
- Set *Lookup* to RGBA and change the min and max values of the colormap to 40 and 150.
- Finally, press *Apply* in order to initialize the *Voltex* textures.

Whenever you choose a different colormap or change the min and max values of the colormap, you must press the *Apply* button again. This causes the internal texture maps to be recomputed. An exception are SGI systems with Infinite Reality graphics. On these platforms a hardware-specific OpenGL extension is exploited, causing colormap changes to take effect immediately.

2.4 Segmentation of 3D Images

By following this step-by-step tutorial you will learn how to interactively create a segmentation of a 3D image. A segmentation assigns to each pixel of the image a label describing to which region or material the pixel belongs, e.g., bone or the kidney. The segmentation is stored in a separate data object called a *LabelField*. A segmentation is the prerequisite for surface model generation or accurate volume measurement.

This tutorial comprises the following steps:

1. Creation of an empty *LabelField*.
2. Interactive editing of the labels in the *Image Segmentation Editor*.
3. Measuring the volume of the segmented structures.
4. An alternative segmentation method: Threshold segmentation.

2.4.1 Interactive Image Segmentation

- Load the *lobus.am* data file from the directory *data/tutorials*.
- Right click on the green icon and choose *LabelField* from the Labelling section.

A new green icon appears, the *LabelField* that will hold the segmentation results. Simultaneously, the image segmentation editor is displayed in the 3D viewer pane. By default, the segmentation editor operates in 2-viewer mode. In this mode, one 2D viewers with different orientations and an additional 3D viewer are displayed. The tools of the segmentation editor are displayed where the Pool and Properties Area are normally displayed. For this tutorial, you'll want to display the segmentation tools.

- Use the mouse to expand the viewer so that you have more room to maneuver in.
- In the lower right view, use the slider on the bottom to scroll through the slices. Go to slice 20. You see two bigger structures and one structure just appearing on the top.
- If necessary, click on the second button under the label *Zoom and Data Window* to zoom out the data so that you have a view of the entire slice.
- Click on the second button under the label *Tools*, the brush.

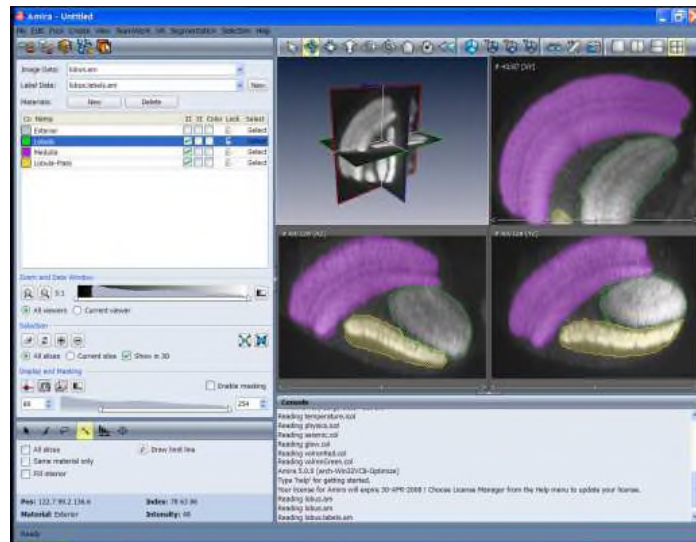


Figure 2.17: Image segmentation editor after selecting and assigning pixels for two structures in one slice

- Mark the rightmost structure with the mouse. Hold down the control button to unselect wrongly selected pixels if necessary.
- When done, select the entry *Inside* in the *Materials* list. Then click the + button under the *Selection* label.

The previously selected pixels are now assigned to the material *Inside*. You can right click on the entry *Inside* in the *Materials* list and choose a different draw style (for example, dotted).

- Click into the material list and choose *New Material* from the right button menu.
- Mark the middle structure using the brush, select the new material in the *Materials* list and assign the pixels to that structure.
- Go to slice 21 and practice by segmenting the two structures.

Hint: If you prefer to work with one larger view rather than four smaller views, click on the *Single Layout* button in the viewer toolbar. To cycle through each of the four views, press the *Single Layout* button repeatedly. To return to 4-viewer mode, press the *Layout4* button.

If a structure does not change a lot from slice to slice, you can use interpolation.

- Go to slice 22 and mark the right structure using the brush. Go to slice 31 and mark the same structure.
- Choose from the menu bar: Selection/Interpolate.
- Scroll through the data set. You should see that the in between slices 23 to 30 are selected too.
- In order to assign the selected pixels in all slices to the Inside material, select the Inside material in the list, then click the + button.
- Repeat the procedure between slice 32 and 50.
- Repeat the procedure for the middle structure.

Hints:

- It is highly recommended to frequently save the segmentation results while working. In order to do so, select the label field in the Amira main window and choose *Save* or *Save As...* from the Amira File menu.
- The brush is only the most basic segmentation tool. The segmentation editor provides many more functions that are described on its *reference page*.
- There are many useful key bindings, including SPACE and BACKSPACE to change the slice number or "d" to toggle the draw style.
- Of course you can give the materials more meaningful names or colors using the context menu (right mouse button in the list).

At this point save the label field and leave the Editor. In the next tutorial you will learn how to create a 3D surface model from the segmentation results.

2.4.2 Volume Measurement

Once a structure is segmented, you can easily measure its volume:

- Right click on the LabelField's green icon. Choose *Measure/MaterialStatistics*.
- Press the *Apply* button. A new icon appears.
- Select this icon and press the *Show* button.

The units in the volume column depend on the units you have specified the voxel size. In case of the *lobus.am*, the voxel size is in μm , therefore the volume is in μm^3 .

2.4.3 Threshold Segmentation

We now describe an alternative way of segmentation, which can require less manual interaction, but only works for images with good quality.

In some cases a satisfying segmentation can be achieved automatically based solely on the gray values of the image data set.

The first step is to separate the object from the background. This is done by segmenting the volume into exterior and interior regions on the basis of the voxel values.

- Load the *lobus.am* data record from the directory *data/tutorials*.
- Attach a *LabelVoxel* module to the data icon and select it.
- Type 85 into the text field of port *Exterior-Inside*. You may also determine some other threshold that separates exterior and interior as described in the tutorial on Image Data Visualization.
- Press the *Apply* button.

By this procedure each voxel having a value lower than the threshold is assigned to *Exterior* and each voxel whose value is greater than or equal to the threshold is assigned to *Interior*. This may, however, cause artifacts that are not part of the object but which have voxel values above the threshold to be assigned to the interior. This can be suppressed by setting the *remove couch* option which assures that only the biggest coherent area will be labeled as the interior and all other voxels are assigned to the exterior.

After you press the *Apply* button, a new data object is computed and its icon appears in the Pool. The data object is denoted *lobus.Labels*. It is of type *LabelField*, represents a cubic grid with the same dimensions as *lobus.am*, and contains an interior or exterior label for each voxel according to the segmentation result.

2.4.4 Refining Threshold Segmentation Results

You can visualize and manually modify a *LabelField* by using Amira's *image segmentation editor*. A more detailed description of this tool is contained in the User's Reference guide. Here, we use the image segmentation editor to smooth the data in order to get a nicer looking surface of the object.

- Select the *lobus.Labels* icon and click on the Image Segmentation icon in the green title bar in the Properties Area.

In response the image segmentation editor is popped up.

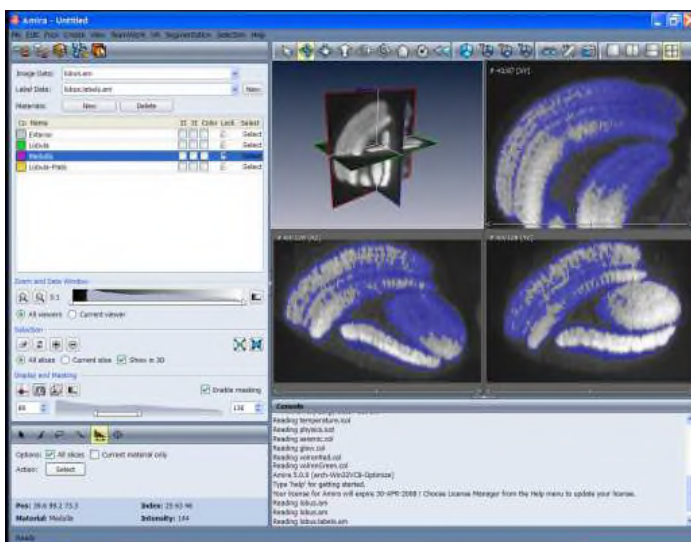


Figure 2.18: Data from confocal microscopy is segmented using Amira's image segmentation editor.

- In the lower right view, use the slider on the bottom to select slice 39.
- Choose a magnification ratio of 2:1 by pressing the zoom-up button in the Zoom and Data region of the editor.

The image segmentation editor shows the image data to be segmented (*lobus.am*) as well as brownish contours representing the borders between interior and exterior regions as contained in the *lobus.Labels* data object. As you can see, the borders are not so smooth and there are many little islands, bordered by brownish contours. This is what we want to improve now.

- Choose *Remove Islands* from the editor's *Segmentation* menu. In response, a little dialog window appears.
- In the dialog window select the *all slices* mode. Then press *Remove* in order to apply the filter in all slices. Note how the segmentation results become less noisy.
- To further clean up the image, choose *Smooth Labels* from the editor's *Segmentation* menu. Another dialog box appears.
- Select the *3D volume* mode and press the *Apply* button in order to execute the smoothing operation.
- To examine the results of the filter operations, browse through the label field

slice by slice. In addition to the slice slider you may also use the cursor-up and cursor-down keys for this.

- Leave the Segmentation Editor.

2.5 Surface Reconstruction from 3D Images

By following this step-by-step tutorial, you will learn how to generate a triangular surface grid for an object embedded in a voxel data set. A surface grid allows for producing a 3D view of the object's surface and can be used for numerical simulations.

The generation process consists of these steps:

1. Extracting surfaces from segmentation results
2. Simplifying the surface

As a prerequisite for the following steps, you need a label field, which holds the result of a previous image segmentation. You can either use the label field which you created in the previous tutorial or load the provided *lobus.labels* data set from the *data/tutorials* directory.

2.5.1 Extracting Surfaces from Segmentation Results

Now we let Amira construct a triangular surface of the segmented object.

- Connect a *SurfaceGen* module to the *lobus.Labels* data.
- Press the *Apply* button.

The option *add border* ensures that the created surface be closed. A new data object *lobus.surf* is generated. Again, it is represented by a green icon in the Pool.

2.5.2 Simplifying the Surface

Usually the number of triangles created by the *SurfaceGen* module is far too large for subsequent operations. Thus, the number of triangles must be reduced in a *surface simplification* step. In Amira a *Surface Simplification Editor* is provided for this purpose.

- Select the surface *lobus.surf*.
- Click on the Simplifier button (triangle mesh icon) in the Properties Area.

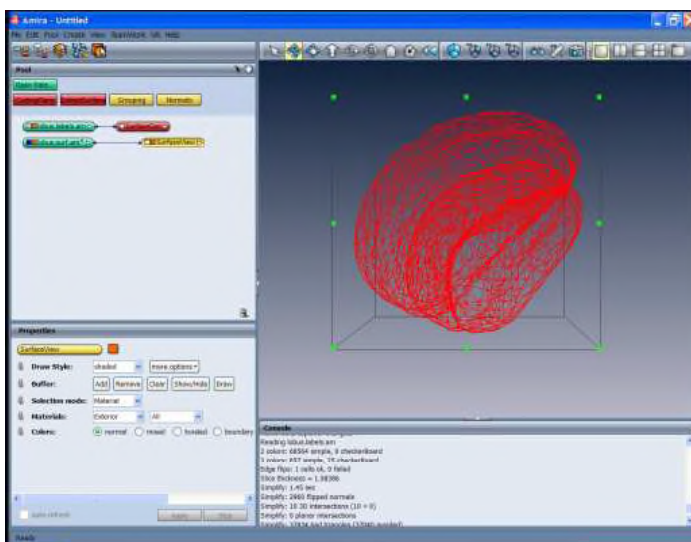


Figure 2.19: Surface representation of optical lobus as triangular grid

- Set the desired number of faces to 3500 in the *Simplify* port.
- Turn on the *fast* toggle in the *Options* port. This option disables some time-consuming intersection tests.
- Push the *Simplify now* button in the *Action* port.

The number of triangles is reduced to about 3500 now. The progress bar tells you how much of the simplification task has already been done.

To examine the simplified surface, attach a *SurfaceView* module to the *lobus.surf* data object.

The *SurfaceView* module maintains an internal buffer and displays all triangles stored in this buffer. By default the buffer shows all triangles forming the boundary to the exterior. If you change the selection at the *Materials* port, the newly selected triangles are highlighted, i.e., they are displayed using a red wireframe representation. The *Add* and *Remove* buttons cause the highlighted triangles to be added to or removed from the buffer, respectively. You may easily visualize a subset of all triangles using a 3D selection box or by drawing contours in the 3D viewer. Press the *Clear* button of the *Buffer* port to see the display shown in Figure 2.19.

2.6 Creating a Tetrahedral Grid from a Triangular Surface

By following this step-by-step tutorial, you will learn how to generate a volumetric tetrahedral grid from a triangular surface as created in the previous tutorial. A tetrahedral grid is the basis for producing various views of inner parts of the object, e.g., cuts through it, and is frequently used for numerical simulations.

The generation process consists of these steps:

1. Simplifying the surface
2. Editing the surface
3. Generating a tetrahedral grid

As a prerequisite for the following steps, you need a triangular surface, which is usually the result of a previous surface reconstruction. Load the supplied *lobus.surf* data set from the *data/tutorials* directory.

2.6.1 Simplifying the Surface

Usually the number of triangles created by the *SurfaceGen* module is far too large for subsequent operations, e.g., for a numerical simulation. Thus, the number of triangles should be reduced in a *surface simplification* step. In Amira a *Surface Simplification Editor* is provided for this purpose. There may be different goals for the simplification:

- In *computer graphics*, one wants to prescribe just the number of faces, because this determines the rendering speed.
- For a *numerical simulation*, one often wants to specify the maximum edge length occurring in the grid model.

This tutorial shows how the maximum edge length can be controlled during simplification.

- Select the surface *lobus.surf*.
- Click on the Simplifier (triangle mesh icon) in the Properties Area.
- Set the desired number of faces to 1000 and the desired maximal distance (i.e. edge length) to 10 in the *Simplify* port.
- Leave the *fast* toggle turned off in the *Options* port. This will cause intersection tests to be performed during simplification, which will considerably

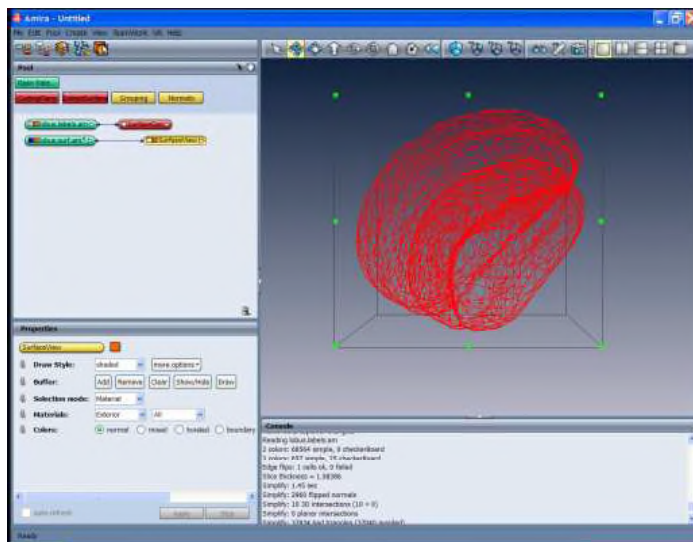


Figure 2.20: Surface representation of optical lobus as triangular grid

reduce the probability that the simplified surface contains self intersections.

- Press the *Simplify now* button in the *Action* port.

Simplification terminates when either of the limits given by the number of faces or the maximum distance is reached. The progress bar tells you how much of the simplification task has already been done. In this example the maximum distance will be the limiting factor, and the resulting surface will contain about 6000 faces.

Besides the maximum edge length, the minimum edge length occurring in the surface should also be controlled, because the ratio of maximum and minimum edge length will influence the quality of the resulting tetrahedral grid. This ratio should not be much larger than 10. If edges that are too short occur in the simplified surface, they can be removed as follows.

- Set the desired minimum distance to 2 in the *Simplify* port.
- Observe the number of faces as shown at the *Surface* port, and press the *Contract edges* button in port *Action*. All edges shorter than 2 will be contracted. In this example about 30 small edges will be detected. You will observe that the number of faces slightly decreases.

2.6.2 Editing the Surface

As a second step of preparation for tetrahedral grid generation, invoke the *Surface Editor*.

- Select the surface *lobus.surf*.
- Leave the *Surface Simplification Editor* (Simplifier) by again clicking on the triangle mesh icon.
- Enter the *Surface Editor* by clicking on the Surface Editor button in the Properties Area.

Automatically, a *SurfaceView* module will be attached to the *lobus.surf* surface. For details about that module see its description.

When the *Surface Editor* is invoked, the *Surface* menu is added to Amira's menu bar and a new toolbar is placed just below Amira's viewer toolbar. The *Surface/Tests* menu contains 6 specific tests which are useful for preparing a tetrahedral grid generation. Each of the tests creates a buffer of triangles which can be cycled through using the back and forward buttons.

- Select *Intersection test* from the *Surface/Tests* menu. The total number of intersecting triangles is printed in the console window. Intersections shouldn't occur too often if toggle *fast* was switched off during surface simplification. In case they occur, the first of the intersecting triangles and its neighbors are shown in the viewer window.
- You can manually repair intersections using four basic operations: *Edge Flip*, *Edge Collapse*, *Edge Bisection*, and *Vertex Translation*. See the description of the *Surface Editor* for details.
- After repairing, invoke the intersection test again by selecting it from the *Surface/Tests* menu or by pressing the *Compute* button.
- When the intersection test has been successfully passed, select the *Orientation test* from the *Surface/Tests* menu. After surface simplification, the orientation of a small number of triangles may be inconsistent, resulting in a partial overlap of the materials bounded by the triangles. In case of such incorrect orientations, which should occur quite rarely, there is an automatic repair. If this fails, the detected triangles will be shown, and you can use the above mentioned manual operations for repair. **Note:** There are two prerequisites for the orientation test: the surface must be free of intersections, and the outer triangles of the surface must be assigned to material *Exterior*. If the surface does not contain such a material or if the assignment to *Exterior*

is not correct, the test will falsely report orientation errors.

A successful pass of the intersection and orientation test is mandatory for tetrahedral grid generation. These tests are automatically performed at the beginning of grid generation. So you can directly enter the *TetraGen* module (see below) and try to create a grid. If one of the tests fails, an error message will be issued in the console window. You can then go back to the *Surface Editor* and start editing.

The remaining three tests analyze the surface mesh with respect to different quality measures. These tests only need to be performed if the tetrahedral quality of the volumetric grid plays an important role, e.g., if the grid will be used for a numerical simulation.

- Select *Aspect ratio* from the *Surface/Tests* menu. This computes the ratio of the radii of the circumcircle and the incircle for each triangle. The triangle with the worst (i.e. largest) value is shown first, and the actual value is printed in the console window. The largest aspect ratio should be below 20 (better below 10). Fortunately there is an automatic tool for improving the aspect ratio included in the *Surface Editor*.
- Select *Flip edges* from the *Edit* menu. A small dialog window appears. In the Radius Ratio area, set the value of the "Try to flip a triangle..." field to 10. Select mode *operate on whole surface*. Press button *Flip*. All triangles with an aspect ratio larger than 10 will be inspected; if the aspect ratio can be improved via an edge flip, this will be done automatically. The console window will tell you the total number of bad triangles and how many of them could be repaired. Press the *Close* button to leave the *Flip edges* tool.
- Select again *Aspect ratio* from the *Surface/Tests* menu. Only a small number of triangles with large aspect ratio should remain after applying the *Flip edges* tool.
- Select *Dihedral angle* from the *Surface/Tests* menu. For each pair of adjacent triangles, the angle between them at their common edge will be computed. The triangle pair including the worst (i.e. smallest) angle is shown in the viewer, and the actual value is printed in the console window. The smallest dihedral angle should be larger than 5 degrees (better larger than 10).
- For a manual repair of a small dihedral angle proceed as follows: select the third points of both triangles (i.e. the points opposite to the common edge) and move them away from each other. For moving vertices you must enter *Vertex Translation* mode by clicking on the first icon from the right on the

top of the viewer window or by pressing the "t" key. If the viewer is in viewing mode, switch it into interaction mode by pressing the ESC key or by clicking on the arrow icon (the first icon from the top) on the right of the viewer window. Click on the vertex to be moved. At the picked vertex a point dragger will be shown. Pick and translate the dragger for moving the vertex.

- In some cases an edge flip might also improve the situation. Enter *Edge Flip* mode by clicking on the third icon from the right on the top of the viewer window or by pressing the "f" key. Switch the viewer into interaction mode. Click on the edge to be flipped.
- Select *Tetra quality* from the *Surface/Tests* menu. For each surface triangle the aspect ratio of the tetrahedron which would probably be created for that triangle will be calculated. The aspect ratio for a tetrahedron is defined as the ratio of the radii of the circumsphere and the inscribed sphere. The triangle with the worst (i.e. largest) value is shown in the viewer, and the actual value is printed in the console window. The largest tetrahedral aspect ratio should be below 50 (better below 25). If all small dihedral angles have already been repaired, the tetra quality test will mainly detect configurations where the normal distance between two triangles is small compared to their edge lengths. Again, the *vertex translation* and the *edge flip* operation are best suited for a manual repair of large tetrahedron aspect ratios.
- Leave the *Surface Editor* by again clicking on the Surface Editor button in the Properties Area.

Hint: In order to see the entire surface again, select the SurfaceView icon, then press its *Show/Hide* button, then press the *ViewAll* button in the viewer toolbar.

2.6.3 Generation of a Tetrahedral Grid

The last step is the generation of a *volumetric tetrahedral grid* from the surface. This means that the volume enclosed by the surface is filled with tetrahedra. Because the computation of the tetrahedral grid may be time consuming it can be performed as a *batch job*. You can then continue working with Amira while the job is running. However, for demonstration purposes we want to compute the grid right inside Amira.

- Connect a *TetraGen* module to the *lobus.surf* surface by choosing *Compute TetraGen* from the popup menu over the *lobus.surf* icon.

- Leave toggle *improve grid* switched on and toggle *save grid* switched off at the *Options* port. The *improve grid* option will invoke an automatic post-processing of the generated grid, which improves tetrahedral quality by some iterations that move inner vertices and flip inner edges and faces. See the description of the *Grid Editor* for details.

If toggle *save grid* is selected, an additional port *Grid* appears, where you can enter a filename. The resulting tetrahedral grid will be stored automatically under that name. If you want to run grid generation as a batch job, you must select the *save grid* option.

- Press the *Meshsize* button of the *Action* port. An editor window will appear. It allows you to define a desired mesh size, i.e., mean length of the inner edges to be created, for each region. For this you must enter the bundle of that region, and select parameter *MeshSize*. Then you can change the value in the text field at the lower border of the editor. There are some predefined region names in Amira for which a default mesh size will be automatically set. Make sure that the default values are suitable for your application. If you are not sure about a suitable value, set the desired mesh size to 0. In this case the mean edge length of the surface triangles will be used.
- Press the *Run now* button at port *Action*. A pop-up dialog appears asking you whether you really want to start the grid generation. Click *Continue* in order to proceed.

Once grid generation is running, the progress bar informs you about the number of tetrahedra which already have been created. In some situations grid generation may fail, for example, if the input surface intersects itself. Then an error message will occur at the *Console Window*. In this case go back to the *Surface Editor* to interactively fix any intersections.

After the tetrahedral grid has been successfully created, a new icon called *lobus.grid* will be put in the Pool. You can select this icon in order to see how many tetrahedra the created grid contains. If grid generation takes too long, you may also load the pre-computed grid *lobus.grid* from the *data/tutorials* directory. As the very last step you may want to have a look at the fruits of your work:

- Attach a *GridVolume* module to the *lobus.grid*.
- Select the *GridVolume* icon and press the *Add to* button of the *Buffer* port.

The *GridVolume* module maintains an internal buffer and displays all tetrahedra stored in this buffer. By default the buffer is empty, but all tetrahedra are high-

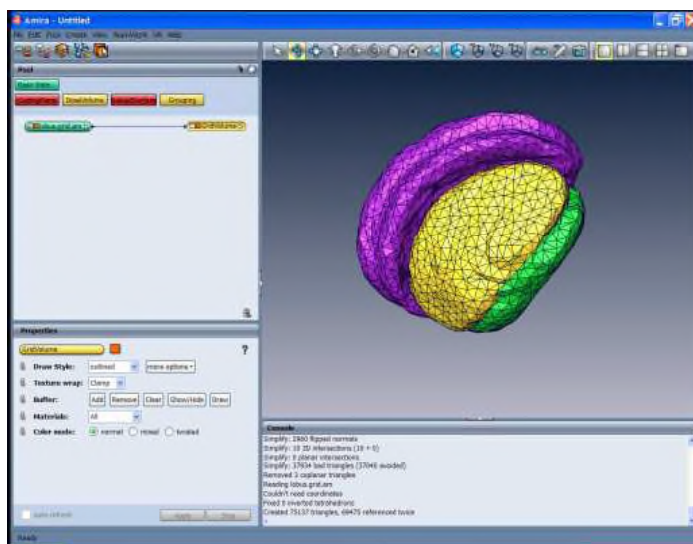


Figure 2.21: Volumetric representation of optical lobe as tetrahedral grid

lighted, i.e., they are displayed using a red wireframe representation. The *Add to* button causes the highlighted tetrahedra to be added to the buffer. You may easily visualize a subset of all tetrahedra using a 3D selection box or by drawing contours in the 3D viewer.

Similar to the *Surface Editor*, there is a *Grid Editor* which can be invoked by selecting the green icon of the tetrahedral grid and clicking on the pencil icon (first from the right in the title bar) in the Properties Area. The editor allows for selecting tetrahedra with respect to different quality measures, e.g., aspect ratio, dihedral angles at tetrahedron edges, solid angles at tetrahedron vertices, and edge length. The editor contains several modifiers that can be applied for improving mesh quality.

2.7 Warping and Registration Using Landmarks

This is an advanced tutorial. You should be able to load files, interact with the 3D viewer, and be familiar with the 2-viewer layout and the viewer toggles.

We will transform two 3D objects into each other by first setting landmarks on their surfaces and then defining a mapping between the landmark sets. As a result we shall see a rigid transformation and a warping which deforms one of the objects to match it with the other. The steps are:

1. Displaying data sets in two viewers.
2. Creating a landmark set.
3. Alignment via a rigid transformation.
4. Warping two image volumes.

2.7.1 Displaying Data Sets in Two Viewers

The data we will be working with in this tutorial are of the same kind you have already seen before: Two optical lobes of a drosophila's brain.

- Load the two lobes by executing the script `share/examples/landmark.hx`.

This script will load two data sets called *lobus.am* and *lobus2.am*. In addition, two isosurface modules connected to each of the data sets will be created. In the viewer the two lobes are visualized by isosurfaces, the first in yellow and the second in blue. As we can see, the lobes are oriented differently. We want to look at each lobe in its own viewer.

- Choose *2 Viewers horizontal* from the *View Layout* menu.

You can see the two lobes in both viewers.

- Visualize the first lobe (blue) in the upper viewer and the second lobe (yellow) in the lower viewer. This is done by deactivating the lower viewer toggle (orange buttons in the icons) of the *Isosurface* module and by deactivating the upper viewer toggle in the *Isosurface2* module.

2.7.2 Creating a Landmark Set

Now, let us create a landmark set object.

- From the main window's *Create* menu select *Data/Landmarks (2 sets)*

in order to create an empty landmark object. A new green icon will show up in the Pool. Since we are going to match two objects by means of corresponding landmarks we had to select the landmark objects containing 2 sets of landmarks (*Landmarks (2 sets)*).

- Select object *Landmarks*.
- Launch the *Landmark Editor* by clicking on the Landmark Editor button in the Properties Area.

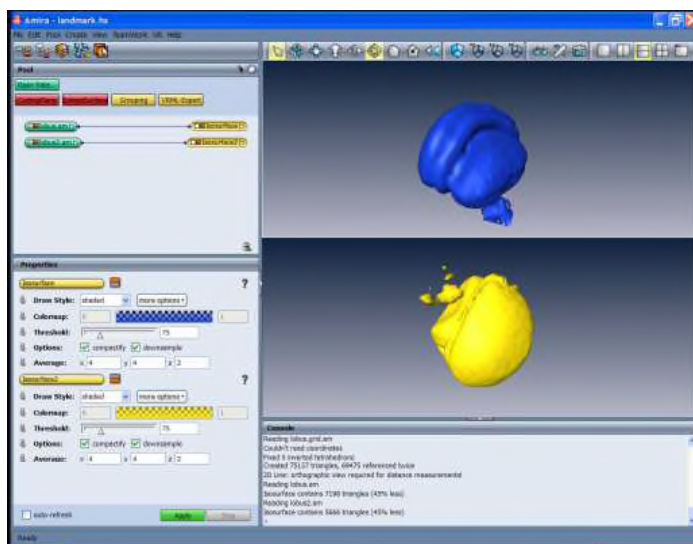


Figure 2.22: Two lobes visualized with isosurfaces in 2-viewer layout.

When starting the editor, a *LandmarkView* module is automatically created and connected to the *Landmarks* data object. As indicated on the info line, two empty landmark sets are available now. We use the editor to define some markers in both objects. For the following, it is useful to push-pin the three ports on the landmark editor. In order to do so, select the gray push-pin toggles to the left of the port labels.

- Connect a second *LandmarkView* module to the *Landmarks* object.
- Select the first *LandmarkView* module and choose *Point Set: Point Set 1*.
- Shift-select the second *LandmarkView* module and choose *Point Set: Point Set 2*.
- Set the viewer toggles of the two *LandmarkView* modules in the same manner as described for the *SurfaceView* modules previously. *LandmarkView* should be visible in the upper viewer and *LandmarkView2* in the lower viewer.

Before starting to set landmarks it is helpful to rotate the two lobes in their viewers such that they are approximately aligned. This will make it easier to locate corresponding features in the two objects and to select reasonable positions for landmarks.



Figure 2.23: The image shows how the viewer toggles and *Point Set* ports should be set.

- Rotate the two lobes to align them roughly.

Now, we are ready to define and set corresponding landmarks. Select the *Landmarks* object if necessary and choose the option

Edit mode: Add.

To set corresponding landmarks simply click with the left mouse button anywhere on the surface in the first viewer first and click on the surface in the second viewer subsequently. The landmarks are visualized as small spheres, the first landmark in yellow and the corresponding landmark in blue. Make sure to always set the first landmark on the first (yellow) surface and the second landmark on the second (blue) surface!!

If you want to change the position of an existing landmark set

Edit mode: Move

and select the respective landmark (blue or yellow) by clicking on it with the left mouse button. Then just click at the desired position.

You can also delete existing landmarks by setting

Edit mode: Remove

and clicking on the respective landmark. Both corresponding landmarks (blue and yellow) will be removed, no matter which one was selected.

You should now be able to create several landmarks. You may want to change the view of the objects to set landmarks on the back. In case you have problems defining landmarks, you may use an existing set of landmarks by loading the file `landmarkSet` from the directory `data/tutorials`. Once landmarks have been created, the next step is to transform the two objects into each other.

2.7.3 Registration via a Rigid Transformation

To register one object to the other, connect a *LandmarkWarp* module to the *Landmarks* object by clicking with the right mouse button on the *Landmarks* icon in the Pool and selecting *Compute LandmarkWarp*.

We want to perform an alignment of the first lobe to the second. Therefore the *LandmarkWarp* module must be connected to the image data of the first lobe (use the right mouse button in the white square of the *LandmarkWarp* icon).

The *LandmarkWarp* module is able to perform several transformations. We start with a purely rigid transformation to match the corresponding landmarks as well as possible by performing only rotations and translations of the first object. To do that choose

Method: Rigid

in the *LandmarkWarp* module and make sure that *Direction* is set to

Direction: 1 → 2.

Then press *Apply* to start the computation. The module creates a new data object named *lobus.Warped*. To visualize the result, connect the isosurface that was initially connected to the data of the first lobe to the result, select it and press the *Apply* button. In order to compare the result with the second lobe, adjust the viewer toggle of its *Isosurface* module to display it in the first viewer. You should see that the result of the transformation fits the second object quite well.

2.7.4 Warping Two Image Volumes

Using the rigid transformation the object will not be deformed. To perform a deformation and obtain a better fit we can use another transformation method of the *LandmarkWarp* module. Select the latter and choose

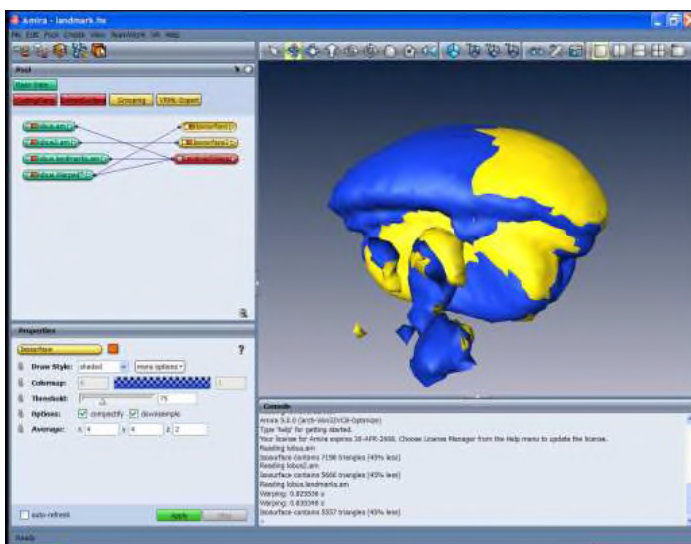


Figure 2.24: Result of landmark-based elastic warping using the Bookstein method.

Method: Bookstein and press the DoIt button.

To visualize the result, the isosurface has to be recomputed. Having done that, you can see both the deformed and the second lobe in the first viewer. To merely see the resulting deformation in the first viewer, switch off the viewer toggle of the second lobe's *Isosurface* module. Only a little deformation will be seen because the two original objects were rather similar. Using more different data sets results in larger deformations.

2.8 Registration of 3D image data sets

In medical imaging a frequent task has become the registration of images from a subject taken with different imaging modalities, where the term *modalities* here refers to imaging techniques such as Computed Tomography (CT), Magnetic Resonance Tomography (MRT) and Positron Emission Tomography (PET). The challenge in inter-modality registration lies in the fact that e.g in CT images "bright" regions are not necessarily bright regions in MRT images of the same subject. In registration typically one of the data sets is taken as the *reference*, and the other one is transformed until both data sets match. Amira's *AffineRegistration* module

provides an affine registration, i.e. it determines an optimal transformation with respect to translation, rotation, anisotrope scaling, and shearing.

Closely related to registration is the task of *image fusion*, i.e., the simultaneous visualization of two registered image data sets.

This tutorial shows how a registration can be performed and how to visualize the results. The following issues will be discussed:

1. Basic manual registration using the *Transform Editor*
2. Automatic registration
3. Image fusion

2.8.1 Basic Manual Registration

In this tutorial we want to register a CT and an MRT data set of a patient, showing the pelvic region. The images are located in the Amira data directory in the subdirectory *registration*.

- Load the files `data/registration/CT-data.am` and `data/registration/MRT-data.am` into Amira.
- Attach an *OrthoSlice* module to each of the data sets.
- Select *Coronal* (or *xz*) at the *Orientation* port of the *OrthoSlice* module connected to the MRT data.
- Select a camera position for the 3D viewer where you can see both the axial slices of the CT data and the coronal slices of the MRT data.
- Select slice 31 at the *Slice Number* port of the *OrthoSlice* module connected to the CT data.

Now one *OrthoSlice* module should show an axial slice through the hip joints. Move the coronal slice through the MRT data. You will observe that the two data sets are not correctly aligned.

- Select the green icon of the MRT data set. Invoke the *Transform Editor* by pressing the Transform Editor button in the Properties Area. The *Transform Editor* enables you to specify an affine transformation, including translation, rotation, and scaling. This transformation will be applied to the corresponding 3D data set. You can edit the transformation interactively in the 3D viewer using different Open Inventor draggers. You can also enter transformations numerically.

- Press the *Dialog* button. A dialog window will pop up.
- Enter -2 at the third text field of the *Translation* port of the dialog window. This means a translation of -2 cm in the z direction.
- Enter 5 at the first text field at the *Rotation* port. This means a rotation of 5 degrees. The axis of rotation is defined at the next ports, here it is the z-axis.
- Press the *Close* button of the dialog window. Leave the Transform Editor by pressing again the Transform Editor button.

Inspect some coronal slices through the MRT data set. Now there is a better alignment of the CT and MRT data, but it's still not perfect.

2.8.2 Automatic Registration

The *Registration* module provides an automatic registration via optimization of a quality function. For registration of data sets from different imaging modalities, the *Normalized Mutual Information* is the best suited quality function. In short, it favors an alignment which "maps similar gray value structures to similar gray value structures". A hierarchical strategy is applied, starting at a coarse resampling of the data sets, and proceeding to finer resolutions later on.

- Attach an *Registration* module to the MRT data set by choosing *Compute/AffineRegistration* from the popup menu over the *MRT-data.am* icon.
- Connect the second input port *Reference* of module *Registration* to the CT data set. For this click with the right mouse button on the white square at the left hand side of the module's icon.
- Select toggle *Extended options*. More ports of the *Registration* module will become visible.

The first three ports of the *Registration* module define the optimization strategy. The default settings mean that an *ExtensiveDirection* optimizer is used for the coarse levels and a *QuasiNewton* optimizer for the finest two levels of the resampling hierarchy. At the *CoarsestResampling* port you can select the resampling rate for the coarsest resolution level. The default resampling rate is smaller in the z direction because the reference data set has a finer resolution in the x and y direction (0.17 cm) than in the z direction (0.5 cm). For the default settings (8,8,3), the resampling hierarchy will consist of four levels: (8,8,3), (4,4,2), (2,2,1), and the original resolution, (1,1,1).

The *Normalized Mutual Information* is calculated from gray value histograms. The selected histogram ranges should enclose the essential information of each data set. Normally you can choose the same range as for visualization via an *OrthoSlice* module.

- Set -200 and 200 at the two text fields of the *Histogram range reference* port.

At the *Transform* port you can specify the type of affine transformation. The default settings mean that only rigid body motions will be applied, i.e. translations and rotations.

Option *Ignore finest resolution* means that optimization is done on all but the finest level of the resampling hierarchy. This will slightly reduce the accuracy, but save a large amount of computing time.

Automatic registration may take some time depending on the resolution of the images and the quality of the pre-alignment. You can interrupt automatic registration at any time using the stop button. Interruption may take some seconds. The progress bar shows the current hierarchy level and the progress at that level.

- Start automatic registration by pressing the *Register* button at the *Action* port.

2.8.3 Image Fusion

The task of image fusion is the simultaneous visualization of two data sets. To that end Amira offers for all types of slicing modules (*Orthoslice* and *ObliqueSlice*) the *Colorwash* module. Using *Colorwash*, the images from one data set can be overlayed over that of another taking into account their orientation in space.

- Remove the *OrthoSlice* module connected to the MRT data set.
- Select the green icon of the MRT data set.
- Select a *Colorwash* module from the popup menu over the icon of the *OrthoSlice* module connected to the CT data set.
- Select the yellow icon of the *Colorwash* module.
- Select the *physics.icol* colormap at port *Colormap*.
- Set 70 as the upper bound for the colormap range.
- Select the icon of the *OrthoSlice* module.
- Inspect axial slices with slice numbers between 15 and 45 .

You will observe a good alignment of the pelvic bone from both data sets. The soft tissue contours are not perfectly aligned because there was some soft tissue deformation between both scans. This cannot be described by a rigid transformation. In image fusion it is sometimes necessary to observe all three orthogonal directions simultaneously. For that the *StandardView* module can be used for image fusion. The *StandardView* module opens a separate window with four viewers, three of them showing the three orthogonal slices of the image data and a fourth being a new instance of the 3D viewer.

- Attach a *StandardView* module to the CT data set by choosing *Display StandardView* from the popup menu over the *CT-data.am* icon. Amira's *Viewer* window will now be split into four parts showing three orthogonal slices through the CT data, and the 3D Viewer in the upper left part.
- Connect the second input port *OverlayData* of the *StandardView* module to the MRT data set. For this, click with the right mouse button on the white square at the left hand side of the module's icon.
- Select slice numbers 179, 149, and 31 at ports *Slice x*, *Slice y*, and *Slice z*, respectively. The three orthogonal slices will show the hip joints now.
- Increase the zoom-factor by clicking twice on button *>* at the *Zoom* port.
- Select *checkerboard* at the *Overlay mode* port.
- Vary the size of the checkerboard tiles by moving the slider at the *Pattern size* port. In this way you can again check the alignment of the CT and MRT data sets.

The bone contours around the hip joints show a good match. Note that bone is represented by white (i.e high intensity) voxels in the CT data, but may occur as both white and black voxels in the MRT data. In the axial slice you can observe larger deviations of the outer body contour between the CT and MRT data.

2.9 Alignment of 2D Physical Cross-sections

Many microscopic techniques require the sample to be physically cut into slices. Then images are taken from each cross-section separately. Usually the images will be misaligned relative to each other. Before a 3-dimensional model of the sample can be reconstructed the images have to be aligned taking into account translation and rotation. This tutorial shows how this task can be performed using the *AlignSlices* module.

The following issues will be discussed:

1. Basic manual alignment
2. Alignment via landmarks
3. Optimizing the quality function
4. Resampling the input data
5. Other alignment options

2.9.1 Basic Manual Alignment

In this tutorial we want to align 10 microscopic cross-sections of a leaf showing a stomatal pore. The images are located in the data directory in the subdirectory *align*. Each slice is stored as a separate JPEG image. The file *leaf.info* defines a 3D image stack consisting of the 10 individual slices. It is a simple ASCII file as described in the *stacked slices* file format section.

- Load the file `data/align/leaf.info`.
- Create an align module by choosing *Compute AlignSlices* from the popup menu over the *leaf.info* icon.
- Press the *Edit* button of *AlignSlices*.

A new graphics window is popped up allowing you to interactively align the slices of the 3D image stack. To facilitate this task, usually two consecutive slices are displayed simultaneously. One of the two slices is *editable*, i.e., it can be translated and rotated using the mouse. By default the upper slice is editable. This is indicated in the tool bar of the align window (the "upper slice" button is selected).

- If necessary, press the zoom out button ("-" magnifying glass) to allow the entire slice to be visible in viewer.
- Translate the upper slice by moving the mouse with the left mouse button pressed down.
- Rotate the upper slice by moving the mouse with the left mouse button and the Ctrl key pressed down. Alternatively, slices can be rotated using the middle mouse button.
- Make the lower slice editable by selecting the "lower slice" tool button. Translate and rotate the lower slice.
- Hold down key number 1. While this key is hold down only the lower slice is displayed.

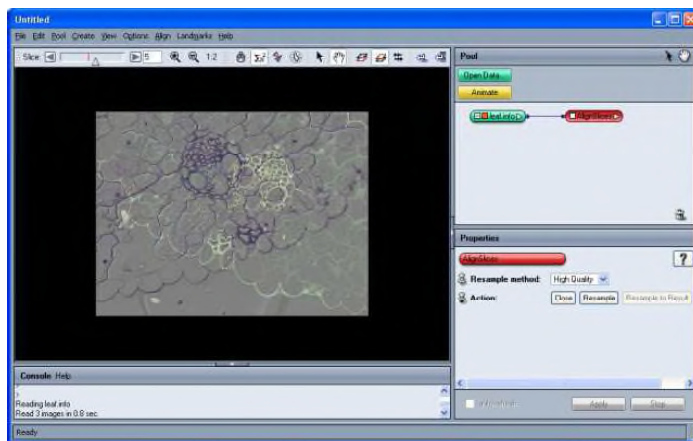


Figure 2.25: User-interface of the align tool.

- Hold down key number 2. While this key is hold down only the upper slice is displayed.
- Pressing key number 1 and 2 also changes the editable slice. Note how the slice tool buttons change their state.

Other pairs of slices can be selected using the slider in the upper left part of the align window. Note that the number displayed in the text field at the right side of the slider always refers to the editable slice. The next or the previous pair of slices can also be selected using the space bar or using the backspace key, respectively. The cursors keys are used to translate the current slice by one pixel in each direction.

- Browse through all slices using the space bar and the backspace key. Translate and rotate some slices in an arbitrary way.
- Translate all slices at once by moving the mouse with the left mouse button and the Shift key pressed down.
- Rotate all slices at once by moving the mouse with the left mouse button and the Shift and Ctrl key pressed down.

Transforming all slices at once can be useful in order to move the region of interest into the center of the image.

2.9.2 Alignment Via Landmarks

Besides manual alignment, four automatic alignment options are supported, namely alignment using a principal axes transformation, automatic optimization of a quality function, edge detection-based alignment and alignment via user-defined landmarks. The principal axes method and the edge detection method are only suitable for images showing an object which clearly separates from the background. The optimization method requires that the images be already roughly aligned. Often such a pre-alignment can be achieved using the landmarks method.

Alignment via landmarks first requires you to interactively define the positions of the landmarks. This can be done in *landmark edit* mode.

- Activate *landmark edit* mode by pressing the arrow-shape tool button located between the hand-shape button and the edge detection button.

In *landmark edit* mode only one slice is displayed instead of two. Two default landmarks are defined in every slice.

- Click on one of the default landmarks. The landmark gets selected and is drawn with a red border.
- Click somewhere into the image in order to reposition the selected landmark.
- Click somewhere into the image while no landmark is selected. This causes the next landmark to be selected automatically.
- Click at the same position again in order to reposition the next landmark.

The double-click method makes it very easy to define landmark positions. Of course, additional landmarks can be defined as well. Landmarks can also be deleted, but the minimum number of landmarks is two.

- Choose *Add* from the *Landmarks* menu.
- Click anywhere into the image in order to define the position of the new landmark.
- Select the yellow landmark by clicking on it.
- Choose *Remove* from the *Landmarks* menu in order to delete the selected landmark again.

Two landmarks should be visible now, a red one, and a blue one. Next, let us move these landmarks to some reasonable positions so that we can perform an alignment.

- Select slice number 0.

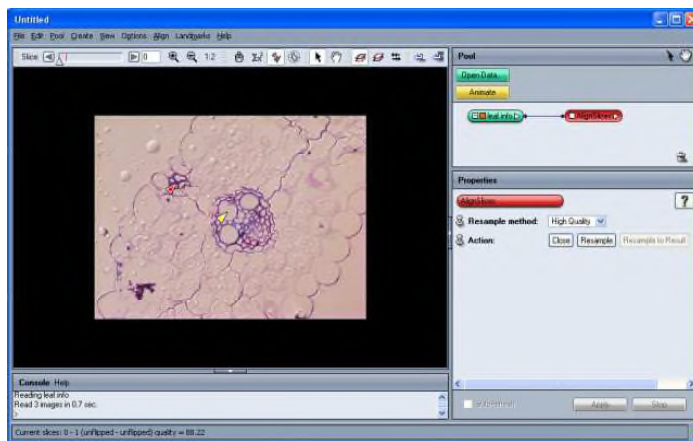


Figure 2.26: The figure shows how the landmarks should be set in the tutorial.

- Place the landmarks as shown in Figure 2.26. Make use of the double-click method.
- In all other slices place the landmarks at the same positions.

Once all landmarks have been set, we can align the slices. It is possible to align only the current pair of slices, or to align all slices at once. Note that all alignment actions as well as landmark movements can be undone by pressing Ctrl-Z.

- Switch back to *transform* mode by pressing the hand-shape tool button. Two slices should be displayed again.
- Align the current pair of slices by pressing the second tool button from the right (the one with only two lines).
- Align all slices by pressing the first tool button from the right (the one with many lines).
- Move and rotate the whole object into the center of the image using the mouse with the Shift key hold down.

In most slices the alignment now should be quite good. However, looking at the pairs 3-4 and 4-5 (displayed in the lower left corner of the align window) you'll notice that there is something wrong. In fact, slice number 4 has been accidentally inverted when taking the microscopic images. Fortunately, we can compensate for this error as follows:

- Select slice pair 3-4 and make sure that the upper slice, i.e., slice number 4,

is editable.

- Invert the upper slice by pressing the invert button (third one from the right).
- Realign the current pair of slices by pressing the second button from the right).
- Select slice pair 4-5 and realign this pair of slices as well.

Alternatively, you could have aligned all slices from scratch by pressing the first button from the right.

2.9.3 Optimizing the Quality Function

Once all slices are roughly aligned, we can further improve the alignment using the automatic optimization method. At the bottom of the align window the quality of the current alignment is displayed. This is a number between 0 and 100, where 100 indicates a perfect match. The quality function is computed from the squared gray value differences of the two slices. The optimization method tries to maximize the quality function. Since only local maxima are found, it is required that the slices be reasonably well aligned in advance.

- Click on the slice in the viewer. The quality of the alignment is displayed in the status bar at the bottom of the window. Remember the current quality measure.
- Activate the optimization mode by pressing the tool button with the sum x squared symbol.
- Align the current pair of slices by pressing the second button from the right. Observe how the quality is improved.

Automatic alignment is an iterative process. It may take quite a long time depending on the resolution of the images and of the quality of the pre-alignment. You can interrupt automatic alignment at any time using the *Stop* button.

- Automatically align all slices by pressing the first button from the right.

2.9.4 Resampling the Input Data

If you are satisfied with the alignment, you can resample the input data set in order to create a new aligned 3D image. This is done using the *Resample* button of the *AlignSlices* module.

- Press the *Resample* button of the *AlignSlices* module.
- Attach an *OrthoSlice* module to the resulting object *leaf.align* and verify that the slices are aligned.

Sometimes you may want to improve an alignment later on. In this case it is a bad idea to align the resampled data set a second time, since this would require a second resampling operation. Instead, you could write the transformation data into the original image object and store this object in *AmiraMesh* format. After reloading the *AmiraMesh* file you can attach a new *AlignSlices* module and continue with the stored transformations.

- Choose *Save transformation* from the *Options* menu of the align tool. This will store the transformation data in the parameter section of the input object *leaf.info*.
- Delete the *AlignSlices* module.
- Save *leaf.info* in *AmiraMesh* format.
- Reload the saved object *leaf.am*.
- Attach a new *AlignSlices* module to *leaf.am* and click the *Edit* button. Note that the original alignment is restored.

2.9.5 Using a Reference Image

In some cases you might want to correct the alignment after image segmentation has been performed. In order to avoid segmenting the newly resampled image from scratch, you can apply the same transformations to a label field using a reference image.

- Delete any existing align tool.
- Load the file `data/align/leaf-unaligned.labels`.
- Attach a new *AlignSlices* module to the label field.

In the label field the guard cells of the stomatal pore are marked. Segmentation has been performed before the images were aligned. Now we want to apply the same transformation defined for the image data to the labels.

- Connect the *Reference* port of *AlignSlices* to *leaf.am* (this is done by activating the popup menu over the small white square at the left side of the *leaf.am* icon). Observe how the transformations are applied to the label field.
- Export an aligned label field by pressing the *Resample* button.

The image volume used in this tutorial is an RGBA color field. However, the image segmentation editor only supports gray level images. Therefore you must convert the color field into a scalar field using *CastField* before you can invoke the image segmentation editor for the resampled labels.

2.10 Visualization of Vector Fields

This step-by-step-tutorial briefly explains some Mesh Option modules for vector field visualization. The use of these modules is explained by visualizing data representing the blood flow inside the carotid artery obtained with a simple numerical simulation. The image data used to segment the carotid are courtesy of Dr. Sell, University of Erlangen, Germany.

The following topics will be covered in this tutorial:

1. How to simply represent of a vector field
2. Use of Illuminated Stream Lines to visualize the flow
3. Extend the visualization with animated Particle Plot

2.10.1 Simple Vector Representation

As in the previous tutorials, we will use the file dialog to import data.

- Load the surface model of the carotids *bloodFlow.surf.am* from the tutorial directory `data/tutorials/vector`.
- Attach an *SurfaceView* module to this data object (see the *Surface reconstruction* tutorial).
- Load the data set called `bloodFlow.vector.am` from the tutorial directory `data/tutorials/vector`.
- Connect a *Vectors* module to the vector field *bloodFlow.vector.am*.

The vector field itself is stored in Amira's native *AmiraMesh* file format. The data represents the simulated flow inside the carotids computed on a regular grid. By selecting the green data icon *bloodFlow.vector.am*, you can find out in the Properties that the number of grid nodes in x,y,z-direction is 116 x 81 x 227.

Besides *SurfaceView*, two new objects are available now in the Pool, *Vectors* and *EmptyPlane*. The module *Vectors* specifies how the vectors are visualized and the module *EmptyPlane* identifies on which the plane.

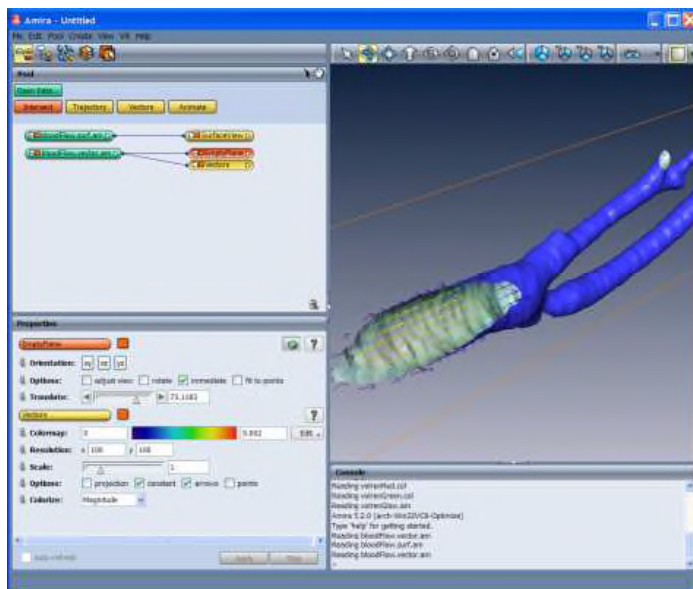


Figure 2.27: Surface of the carotis clipped and the inner vector field represented with simple vecotrs

- Set the *Resolution* of the *Vectors* module to 100 in X and Y
- Check in *Options* the toggles *constant* and keep *arrows* checked
- Right-click on the colorbar of the port *Colormap*, select the *physics* colormap and set the upper value to 0.002.
- In the module *EmptyPlane* click on the "YZ" button and on the "Clip" button on the upper-right corner of the Properties (beside to the question mark button).
- Rotate the object in the viewer and slide the slider of the port *Translate* of *EmptyPlane* to explore the vector field inside the carotid.

The vectors are represented as arrows equal in length, but varying in color. The color, taken from the colormap, indicates the vector's magnitude. To use the length of the arrows to indicate the magnitude, deselect the toggle *constant* in the *Options* port.

2.10.2 Illuminated Stream Lines

Illuminated Stream Lines is a technique for interactive 3D vector field visualization which makes use of large numbers of properly illuminated stream lines. A realistic shading model is employed which significantly increases realism of the resulting images and enhances spatial perception.

Now you will learn which tools are used for illuminated stream line visualization and how to use them to get a 3D impression of our blood flow vector field.

- Remove the *Vectors* module.
- Connect a *DisplayISL* module to the vector field *bloodFlow.vector.am* by right-clicking on it and selecting *Display-¿ Vectors*.
- Set *Num Lines* to 300.
- Press the *Apply* button.

A *TabBox* appears in the viewer. Only stream lines flowing through this box are visible. The green tabs at the corners and edges of the box allow you to change the dimensions of the box.

- Switch the viewer into interact mode.
- Try out what happens if you click with the left mouse button on one of the green tabs on the corners or edges of the *TabBox* and drag them around.
- To move the whole *TabBox*, click with the left mouse button in the box and move it.
- Try to get the *TabBox* into a shape and position as shown in the image.
- Press *Apply* button again in order to recalculate the stream lines.
- As we did before, slide the slider of the port *Translate* of *EmptyPlane* to explore the vector field inside the carotid.

You can colorize of the stream lines according to a scalar data set of the same size and even animate the lines.

- Load *bloodFlow.density.am* from the tutorial directory `data/tutorials/vector`. This data set represents the 3D scalar field of the simulated fluid density associated with the 3D vector field.
- Click on the white little square of the *DisplayISL* icon, select Color Field and attach *bloodFlow.density.am*.
- The port *Colormap* is now available in the Properties window of *DisplayISL*. Right-click on the color bar and select *temperature.icol*, set the colormap

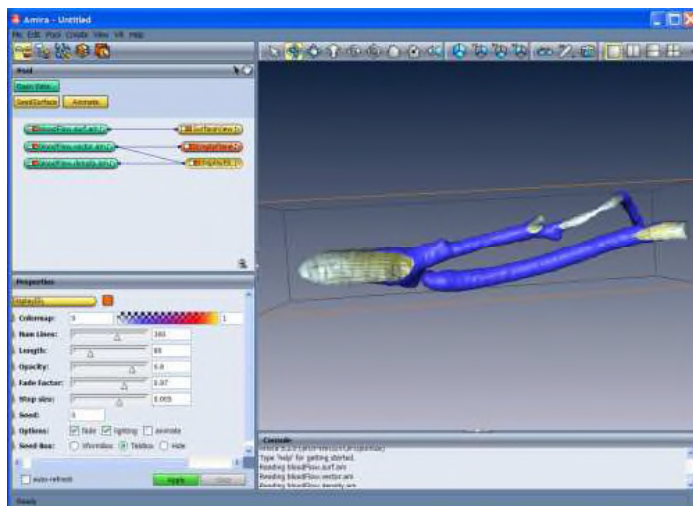


Figure 2.28: Illuminated blood flow stream lines with particle plot

interval to 0-1 and press *Apply*.

- To animate the stream lines just check the toggle *animate* in the port *Options*.

2.10.3 Animated Particle Plot

This module visualizes scalar vector fields by particles (cones) pointing in the direction of the local flow. The placement of particles can be done similar to the distribution modes of the *DisplayISL* module we have seen above. The particles are animated to generate a sequence of objects "walking" through the vector field.

- Remove the *DisplayISL* module.
- Connect a *ParticlePlot* module to the vector field *bloodFlow.vector.am*.
- In the port *Option* and *Option2* set *NumCones* to 10, *Height* to 0.8 and *Radius* to 0.5.
- Right-click on the slider of the port *Animate* and select *Loop*. Click now the play button to start the animation.

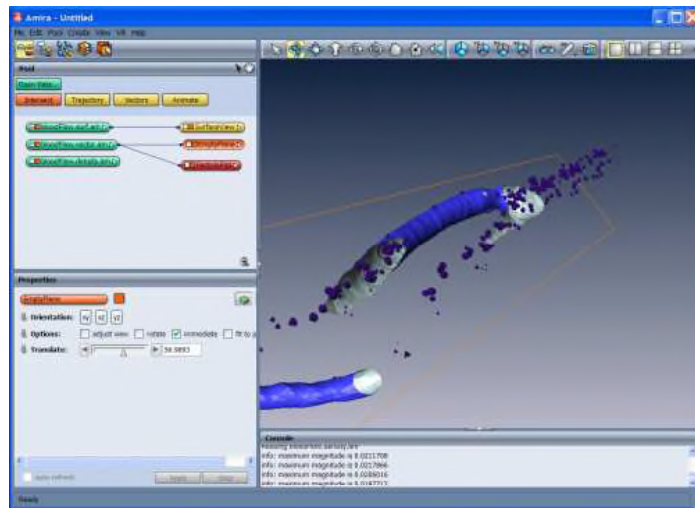


Figure 2.29: Blood flow visualized with particle plot and the reconstructed carotid surface

2.11 Creating animated demonstrations

In this tutorial you will learn how to use the *DemoMaker* module for creating an animated sequence of operations within Amira. In our example, we will visualize a polygon model using effects such as transparency, camera rotation, and clipping to make the visualization more meaningful and attractive.

The tutorial covers the following topics:

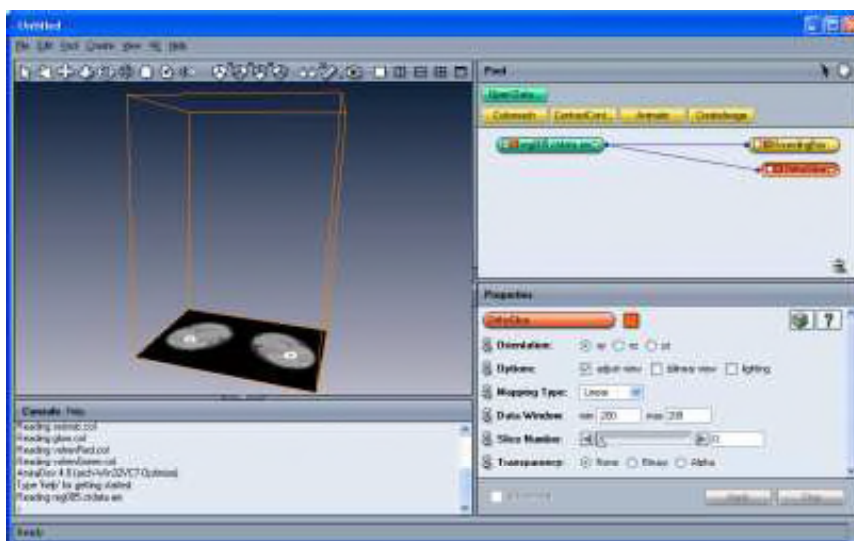
- creating an initial network for the demo
- animating an *OrthoSlice* module
- activating additional modules during the demo
- using a camera rotation or path
- editing or removing events that are already defined
- overlaying a bone model with a transparent skin model
- using clipping to make the skin appear gradually over the bone
- advanced clipping issues
- inserting breaks and defining demo segments
- using function keys for jumping between demo segments
- defining partial loops within the demo sequence

- storing and replaying a demo sequence

Once you have learned how to define an animated demo sequence, you can further learn how to record the demonstration into a movie file in Section 2.12.

2.11.1 Creating a Network

First, we need an Amira network that contains all the data and modules for the visualization and animation we want to do. In our example, we pick the medical CT scan data set `reg005`. Start by loading `data/medical/reg005.ctdata.am` from the `AMIRA_ROOT` directory. By right-clicking on the green data icon and selecting from the data set's popup menu, attach a *BoundingBox* module as well as a *OrthoSlice* module to the data. If you use the mouse to navigate around the model in the 3D viewer, you should manage to get a result similar to this:



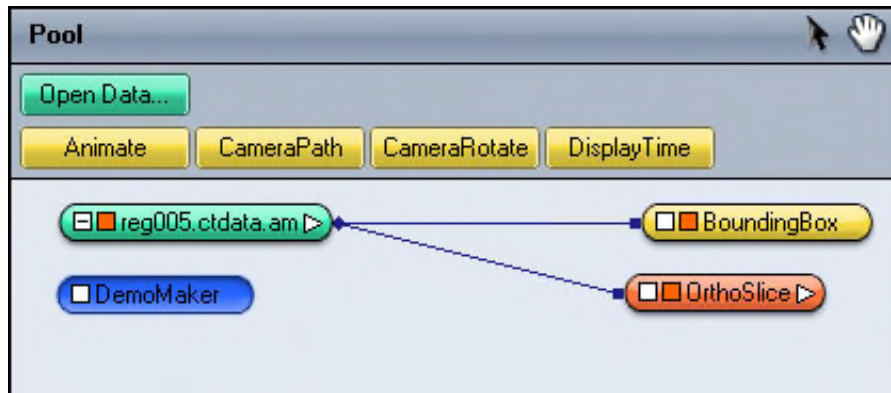
network)

(load

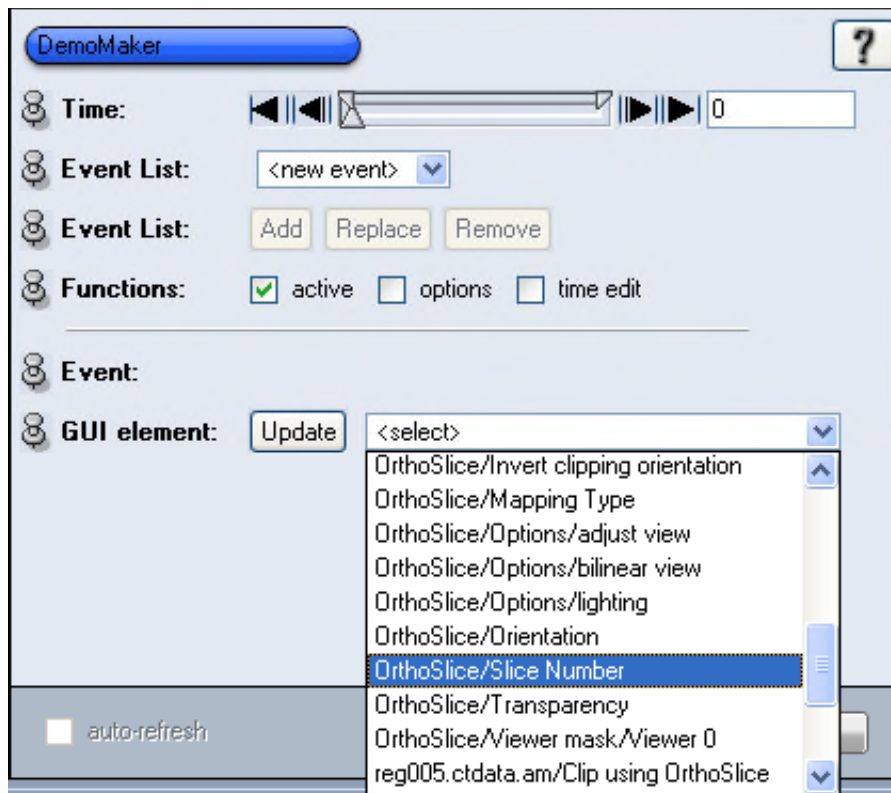
2.11.2 Animating an OrthoSlice Module

Let us move the *OrthoSlice* plane up and down to show what the data looks like. Note that the *OrthoSlice* module has a port called *Slice Number*. If you change the value of that slider, you see the plane move in the viewer.

Now let us animate this slider using the *DemoMaker* module as our first exercise. From the menu bar, select *Create/Animation/Demo/DemoMaker*. A blue script object appears in the Pool:



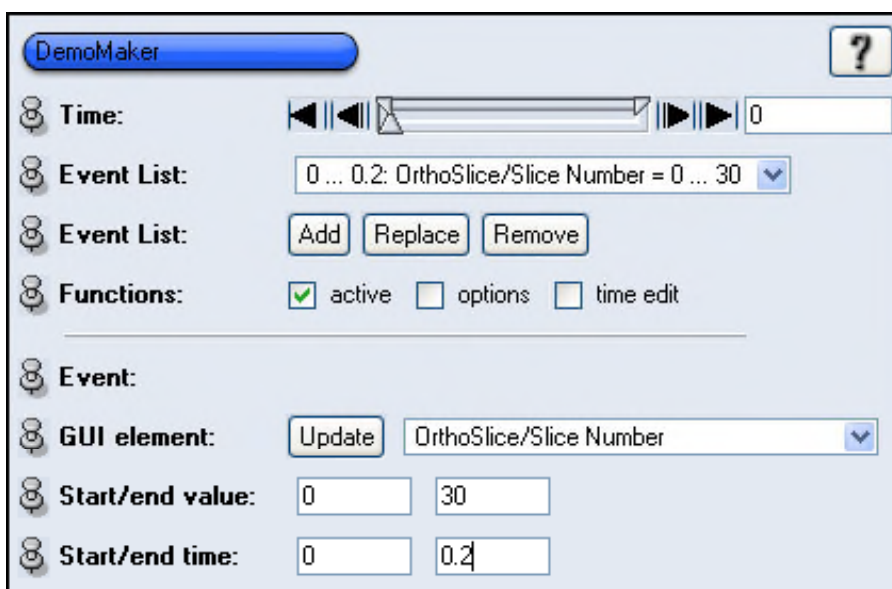
Click on the blue icon to see its user interface. Whenever you want to animate some port of the current network, you must select that port in the selection list called *GUI element*. Try to find the entry called *OrthoSlice/Slice Number*, which corresponds to the *Slice Number* port of the current *OrthoSlice* module. If you cannot find the entry, you may need to press the *Update* button to the left of the selection menu (see below for an explanation).



Once you have selected *OrthoSlice/Slice Number*, you see two more ports appear in the *DemoMaker* module: *Start/end value* as well as *Start/end time*. The start and

end value specify between which two values the OrthoSlice slider will be moved. Click the mouse into the start or end value fields, hold down the **Shift** key, and drag the mouse with **Shift** held down. With this feature called the *virtual slider* you can quickly set the desired value within the allowed range. Set the start value to 0 and the end value to 30. Then, set the start time to 0 and the end time to 0.2. These time values specify times on the *Time* slider of the *DemoMaker* module (first port in the module). You have now specified an *event* starting at time 0 and ending at time 0.2, varying the OrthoSlice slice number between 0 and 30 during that time.

Now press the *Add* button in the *Event List* port to add the newly defined event to the list of events. This list of events is represented in the selection menu above the *Add* button.



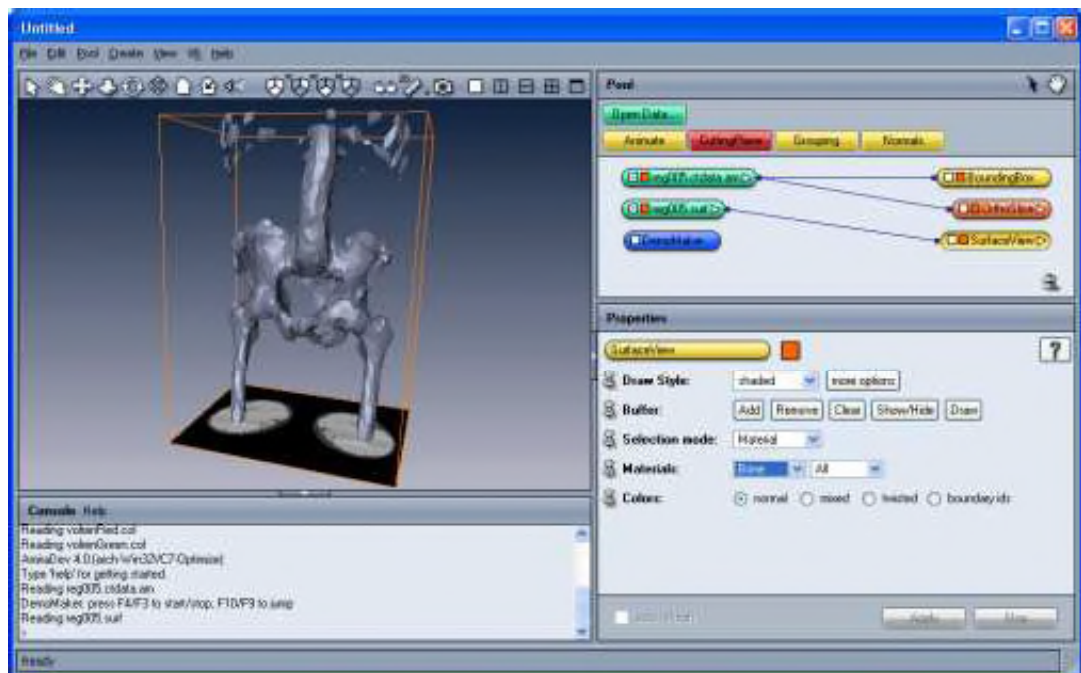
Test the result by pressing the play button of DemoMaker's time slider, represented by the triangle pointing to the right. When you press it, the time slider will start moving from the left to the right. When the time value is between 0 and 0.2, you should see that the OrthoSlice plane is moving between the specified start and end values in the viewer (load network). You can also play your demo backwards using the play button to the left of the time slider, or simply click somewhere on the time slider to jump to any point in time of the demonstration.

If the demo sequence runs too slow or too fast, you can adjust this by right-clicking anywhere on the time slider and selecting *Configure* from the popup menu. Change the *Increment* value in the dialog box that appears. A smaller increment will make the animation slower, whereas a larger increment makes it faster. If you choose an

increment value too large, the animation might become "jerky".

2.11.3 Activating a Module in the Viewer Window

Next, let us add a visualization of the bone structure in the data set after we have moved the OrthoSlice. Load the data set `data/medical/reg005.surf` in addition to the current network. Attach a *SurfaceView* module to it. Click on the yellow *SurfaceView* module to see its user interface. Press the *Clear* button in the *Buffer* port, then select *Bone* and *All* in the *Materials* port and press the *Add* button in the buffer port. This will visualize the bone surface of the model.



If you want to switch the bone visualization on and off manually, you would use the *viewer toggle* (orange rectangle) of the *SurfaceView* module. If you want to include this action in your demo sequence, you need to do the following:

- Click on the *DemoMaker* module.
- Click on the *Update* button in the *GUI element* port. This is necessary whenever you add new modules to the Pool after you have created the *DemoMaker* module.
- Select *SurfaceView/Viewer Mask/Viewer 0* from the *GUI element* list.
- Enter *on* in the *Toggle to value* port.
- Enter *0.2* in the *Trigger time* port.

- Press the *Add* button in the *Event List* port to add this newly defined event to the event list.

The screenshot shows a configuration window for a GUI element. It has three sections:

- GUI element:** A dropdown menu showing 'SurfaceView/Viewer mask/Viewer 0' and an 'Update' button.
- Toggle to value:** Two radio buttons, 'on' (selected) and 'off'.
- Trigger time:** A text input field containing '0.2'.

To test the newly added event, first toggle the *SurfaceView* module off using its orange viewer toggle icon. Now click to the very left of the *DemoMaker* time slider to jump to the start of the demo sequence. Click the play button. As before, the slice moves up. When it reaches the maximum value at time 0.2, the bone model is switched on (load network).

2.11.4 Using a Camera Rotation

To look at the 3D patient model from all sides, let us add a camera rotation to our demo sequence. Select *Create/CameraRotate* from the menu. Try the rotation by playing the time slider in the *CameraRotate* module. If you do not like the axis of rotation, reset the time slider to 0, navigate to a good starting view in the viewer window, and click on *recompute* in the *CameraRotate* module. Note that the values of the *CameraRotate* time slider range from 0 to 360.

Once you are satisfied with the camera rotation, add it to the event list:

- Click on the *DemoMaker* module.
- Click on the *Update* button.
- Select *CameraRotate / Time* from the *GUI element* list.
- Enter 0 and 360 as the start and end value.
- Enter 0.2 and 0.4 as the start and end time.
- Click on the *Add* button in the *Event List* port.

The screenshot shows a configuration window for the 'CameraRotate/Time' GUI element. It has three sections:

- GUI element:** A dropdown menu showing 'CameraRotate/Time' and an 'Update' button.
- Start/end value:** Two text input fields containing '0' and '360'.
- Start/end time:** Two text input fields containing '0.2' and '0.4'.

Now play the demo to see the result. After moving the slice and switching on the bone model, the view is rotated so that the bone can be seen from all sides (load network).

2.11.5 Editing or Removing an Already Defined Event

When you look at the demo sequence so far, you may think that it would be nice to wait for a short time before rotating the bone model. This can be done by starting the rotation at a later time step. We can easily correct this in the *DemoMaker* module:

- Select the *0.2 ... 0.4: CameraRotate* event in the *EventList* port.
- You see the start/end value and start/end time of this event appear in the lower part of the *DemoMaker* module.
- Change the start/end time to 0.3 and 0.5.
- Click on the *Replace* button in the *Event List* port. This replaces the currently selected event in the list by the event as defined in the lower part of the module.

Now you have moved the camera rotation event from 0.2-0.4 to 0.3-0.5 on the time line. Check the results by playing the time slider (load network).

Please note that you can delete an event from the list by simply selecting it from the *Event List* menu and clicking the *Remove* button.

2.11.6 Overlaying the Bone with Skin

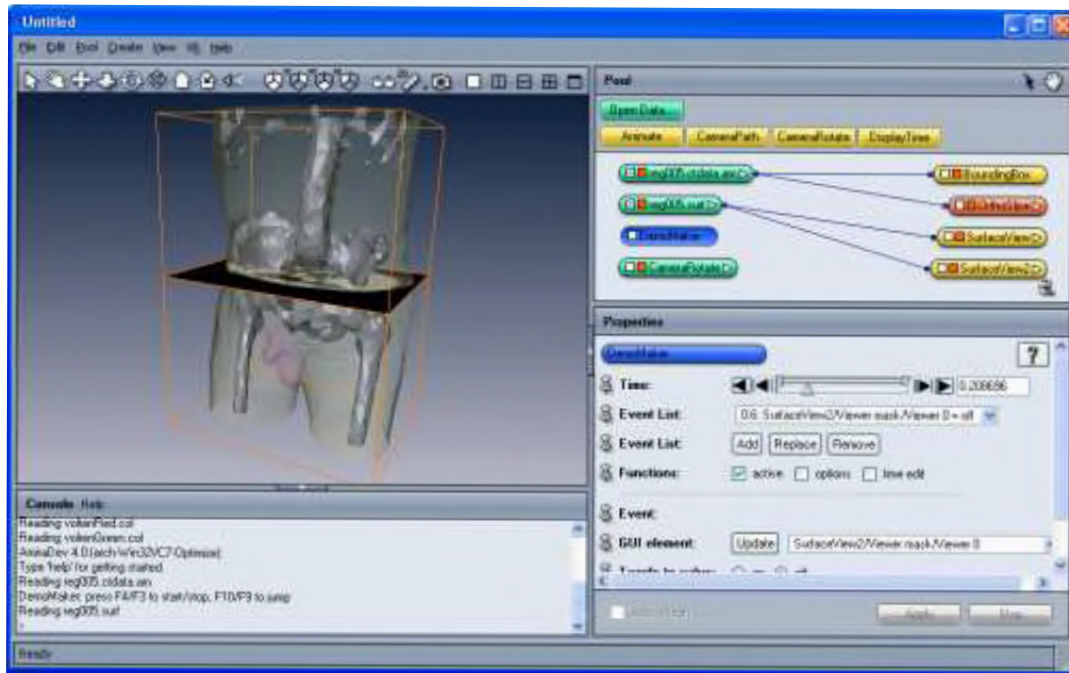
Now we want to show the patient's outer surface overlayed over the bone model.

- Attach a second *SurfaceView* module to the *reg005.surf* data set. Since *Exterior* and *All* are selected as the default materials, this brings up the patient's exterior surface.
- Click on the second *SurfaceView* module. It should be called *SurfaceView2*.
- Select *transparent* from the *Draw Style* port.
- It will be helpful to show the bone underneath the exterior surface, so jump to time step 0.2 or later in the *DemoMaker* module.
- Adjust the grade of transparency using the *BaseTrans* slider in *SurfaceView2*.
- Smooth out the outer surface by clicking on *more options* in the *Draw Style* port and selecting *Vertex normals*.

Like we did with the bone model, we can switch on the skin model at some point in the demo sequence:

- Click on the *DemoMaker* module.

- Click on the *Update* button in the *GUI element* port.
- Select *SurfaceView2/Viewer Mask/Viewer 0* from the *GUI element* list.
- Check *on* in the *Toggle to value* port.
- Enter *0.6* in the *Trigger time* port.
- Click on *Add* in the *Event List* port.



Again, check out the results by playing the demo sequence.

2.11.7 Using Clipping to Add the Skin Gradually

Instead of just switching the skin on at one point, we can make it appear gradually over the bone from bottom to top. In order to do so, we use the *OrthoSlice* plane to *clip* the skin model, and then move the *OrthoSlice* plane up.

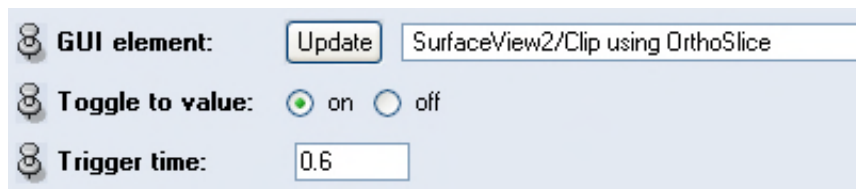
First, we need to move the *OrthoSlice* plane down again to where we want to start the clipping:

- Click on the *DemoMaker* module.
- Select *OrthoSlice/Slice Number* from the *GUI element* list.
- Enter 30 and 3 as the start/end values.
- Enter 0.3 and 0.5 as the start/end time.
- Click on *Add* in the *Event List* port.

Now, when you play the demo, the *OrthoSlice* plane will move down again during the camera rotation (load network).

Now, we will clip the skin model using the *OrthoSlice* plane:

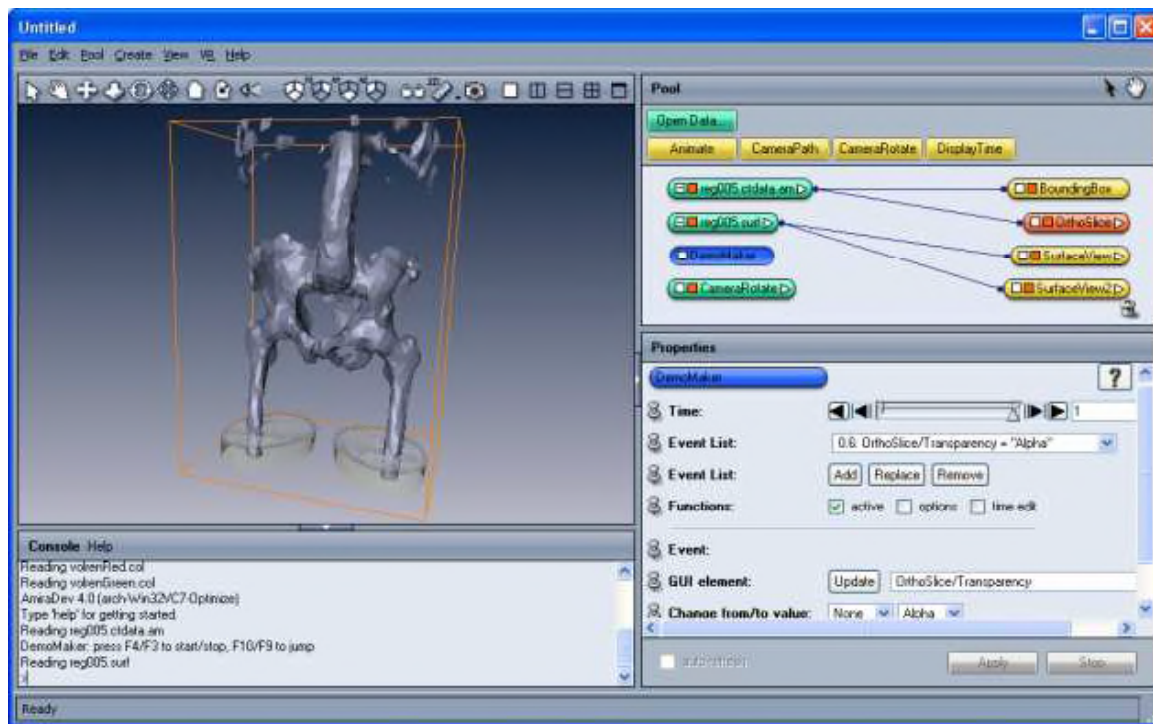
- Click on the *DemoMaker* module.
- Select *SurfaceView2 / Clip using OrthoSlice* from the *GUI element* list.
- Enter *on* as toggle value and *0.6* as trigger time.
- Click on *Add* in the *Event List* port.



When you run the animation now, you will not see the skin surface. This is because it is clipped above the *OrthoSlice* plane, and only visible below that plane. To see the partial surface below the plane, we must make the *Orthoslice* display transparent:

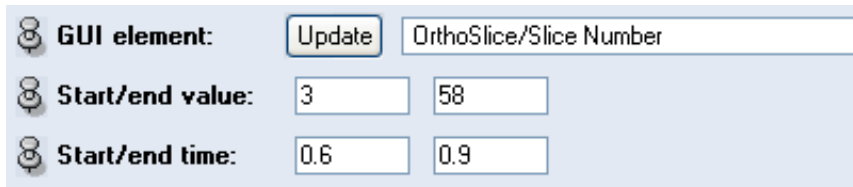
- Click on the *DemoMaker* module.
- Select *OrthoSlice / Transparency* from the *GUI element* list.
- Select *None* and *Alpha* as the from/to values.
- Enter *0.6* as the trigger time.
- Click on *Add* in the *Event List* port.

This way we have specified that at time 0.6, the *Transparency* port of the *OrthoSlice* module will be changed from value *None* to the new value *Alpha*. When running the demo sequence, the result should look like this:



As you see, part of the skin model is showing below the transparent *OrthoSlice* plane. To show all of the skin, we simply move the plane upwards pretty much the same way we did before:

- Click on the *DemoMaker* module.
- Select *OrthoSlice / Slice Number* from the *GUI element* list.
- Enter 3 and 58 as the start/end value.
- Enter 0.6 and 0.9 as the start/end time.
- Click on *Add* in the *Event List* port.



Now you see the skin slowly appearing over the bone as the clipping plane moves upwards.

As a last step, you might want to rotate the view again while the skin is appearing. You can simply reuse the old camera rotation during a second time range:

- Click on the *DemoMaker* module.

- Select *0.3 ... 0.5: CameraRotate* from the *Event List* menu.
- You will see start/end value and start/end time appear in the lower part.
- Change the start/end time to 0.6 and 0.9.
- Click on *Add* in the *Event List* port. This will leave the old event untouched and add a second camera rotation event to the list.

GUI element:	Update	CameraRotate/Time
Start/end value:	0	360
Start/end time:	0.6	0.9

You can check out the final animation by loading a saved network script.

2.11.8 More Comments on Clipping

Clipping can sometimes be a little bit more complicated than in our example, because clipping can be applied to a plane in two different orientations. This means that you can either clip away everything *above* the plane, or *below* the plane. Unfortunately it is not always obvious which of the two cases you are in.

However, you can simply invert the orientation of the clipping in *DemoMaker*. In our example, you would simply select *OrthoSlice / Invert clipping orientation* from the *GUI element* port and add that event at the very beginning of your demo sequence (e.g., at some time before the clipping takes effect).

GUI element:	Update	CameraRotate/Time
Start/end value:	0	360
Start/end time:	0.6	

- OrthoSlice/Colormap/max
- OrthoSlice/Colormap/min
- OrthoSlice/Contrast Limit/
- OrthoSlice/Data Window/max
- OrthoSlice/Data Window/min
- OrthoSlice/Frame
- OrthoSlice/Invert clipping orientation**
- OrthoSlice/Mapping Type
- OrthoSlice/Options/adjust view

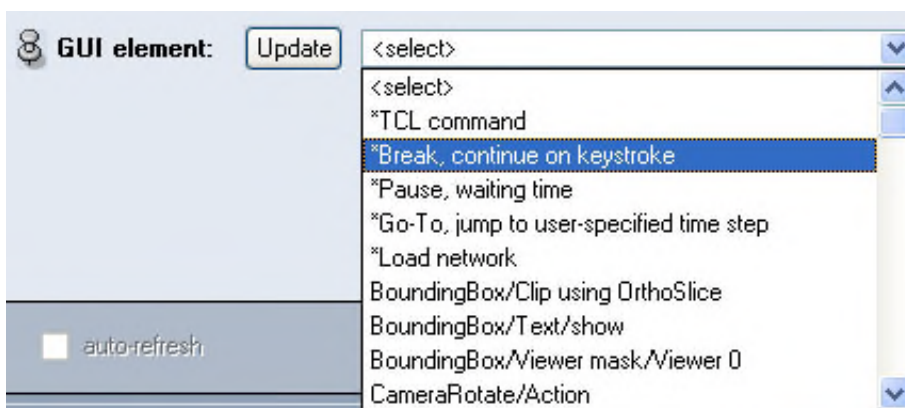
You do not need to use an *OrthoSlice* module to do clipping. As you have seen, the *OrthoSlice* might occlude parts of what you want to show. In that case, it is better to create an empty *ClippingPlane* module by selecting *Create/Clipping Plane* from the menu. Attach the module to the data set you want to clip (e.g., to *reg005.surf* in our example), and then use the *ClippingPlane* for clipping just as you used the *OrthoSlice* before.

2.11.9 Breaks and Function Keys

The demo sequence that we have created in this tutorial automatically runs through the complete time range that we defined. Sometimes it might be desirable to split the sequence into several segments, so that the demo will stop at some point and can be continued whenever the user desires to do so.

To take this into account, you can insert *breaks* in the *DemoMaker* event list. Let us insert one such break right after the bone model appears:

- Click on the *DemoMaker* module.
- Select **Break, continue on keystroke* from the *GUI element* list.
- Enter 0.21 as the trigger time.
- Click on *Add* in the *Event List* port.



This way the demo will stop at time 0.21, which is right after the time when the bone model is switched on (0.2). When you play the demo from the start, you will notice that after the bone is switched on, the demo will stop.

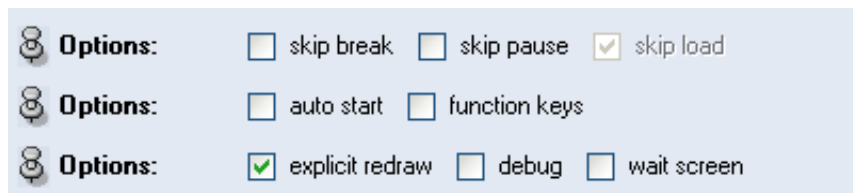
Let us insert a second break at time step 0.51, which is right before the skin is starting to show. Proceed as above, using a trigger time of 0.51 instead of 0.21 (load network).

If you run the demo from the very beginning, it will stop after the bone is displayed, and you can read a message in the console window telling you that *DemoMaker* just stopped and you may press F4 to continue. Try this by pressing the function key F4. The demo continues.

Likewise, the demo will stop just before showing the skin. Again, you can continue the demo by pressing F4. In general, at any point while the demo is running, you can press the F3 key to stop it manually. Pressing F4 will continue from the point where the demo stopped.

If you have defined breaks as we did above, there are two more interesting function keys that in some sense allow you to *navigate* through the demo segments: pressing F9 will jump back to the previous break or to the very beginning of the demo, and F10 will jump to the next break, or to the very end of the demo. If you use F9/F10 when the demo is stopped, it will just jump, and you need to press F4 to start playing it from the new time step. If you press F9/F10 while the demo is running, it will just jump to the new time step and continue running.

Please note that you can disable the breaks by checking the *skip break* toggle in the *Options* port of the *DemoMaker* module. You may even disable the definition of function keys by checking the *options* toggle in the *Functions* port, and then unchecking *function keys* in the second *Options* port. This is especially important if you want to use multiple *DemoMaker* modules, since only one of the modules can define the keys.

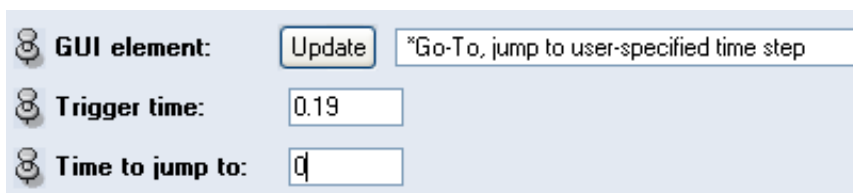


2.11.10 Loops and Go-to

One more feature that might be required for certain kinds of demos is the definition of loops. If you just want the whole demo to run in a loop, you can do this easily using the built-in features of the *time slider*: right-click on the slider and select *loop* or *swing*. Now if you play the time slider, it will start over from the beginning (loop mode), or play forwards, backwards, forwards... (swing mode).

However, you may want to define some part of the demo to run in a loop, and then stop the loop and continue with the demo upon key press. You can easily do this with the *go-to* feature of *DemoMaker*:

- Click on the *DemoMaker* module.
- Select **Go-to, jump to user-specified time step* from the *GUI element* list.
- Enter 0.19 as the *Trigger time*.
- Enter 0.0 as the *Time to jump to*.
- Click on *Add* in the *Event List* port.



GUI element: *Go-To, jump to user-specified time step

Trigger time:

Time to jump to:

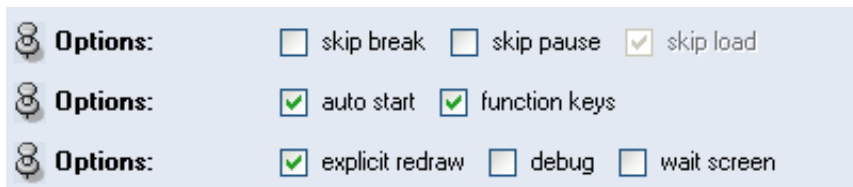
When you run the demo sequence now, it will loop in the first segment, only showing the *OrthoSlice* move up, jump down, move up again... You can stop this by clicking on the stop button of the *DemoMaker* time slider or by pressing F3. To continue after the loop, you need to jump to the next segment by pressing F10, and then start playing again by pressing F4.

2.11.11 Storing and Replaying the Animation Sequence

As you may have noticed by now, storing a demo sequence once you have defined it is quite easy: simply save the whole Amira network by selecting *File / Save Network...* from the menu. The *DemoMaker* module will be saved along with the network, and so will the demo sequence you have defined.

When you load the network back into Amira, the state of the network will be the same as it was when you saved it. This means that you should be careful to reset the *DemoMaker* time slider to 0 before saving the network, if you want the demo to start from the beginning.

After loading the network, you can start the demo by clicking on the play button of the *DemoMaker* module, or by pressing F4. If you want to run the demo automatically right after the network is loaded, you can use the *auto start* feature that you find when you check *options* in the *Functions* port:



Options: ☐ skip break ☐ skip pause ☒ skip load

Options: ☒ auto start ☒ function keys

Options: ☒ explicit redraw ☐ debug ☐ wait screen

Just check the *auto start* toggle and save the network. When you load it again, the demo will start running automatically (load network).

2.12 Creating movie files

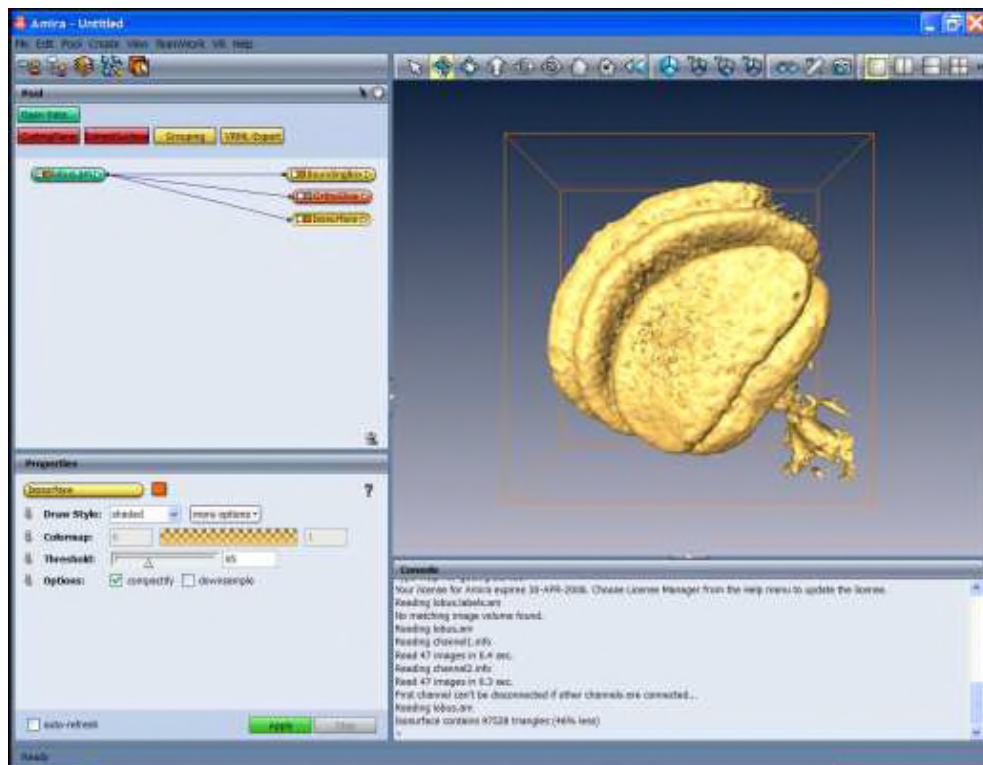
In this tutorial you will learn how to record a self-created animated sequence into a movie file using the *MovieMaker* module.

In our first example we will just use a camera path to animate the scene, whereas in our second example we will rely on the demo sequence created in Section 2.11.

2.12.1 Attaching MovieMaker to a Camera Path

If you have created a visualization of your data and want to create a movie showing this visualization from all sides or from certain interesting viewpoints, you can create an appropriate camera path and record a movie by following the camera along that path.

Let us create a simple example. Load the `lobus.am` data set from the tutorial subdirectory and attach an *Isosurface* module to it. Choose an isosurface threshold of 70 and press the *Apply* button. The result should look similar to this:

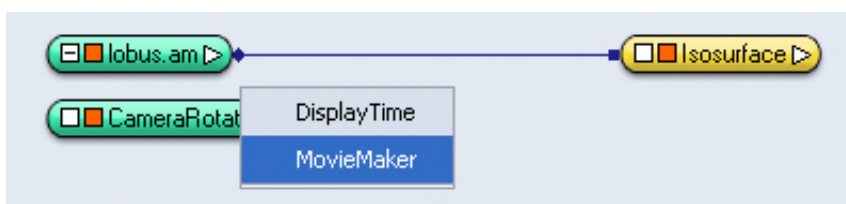


The easiest way to create a simple camera path is to use the *CameraRotate* module. Select *Create/CameraRotate* from the menu, and press the play button of the newly created module. You can watch the scene rotate in the viewer while the time slider is playing (load network).

To record an animated scene into a movie file, you need to attach a *MovieMaker* module to a module that possesses a *time slider port*. The movie is recorded by

going through the individual time steps and taking snapshots of the viewer along the way.

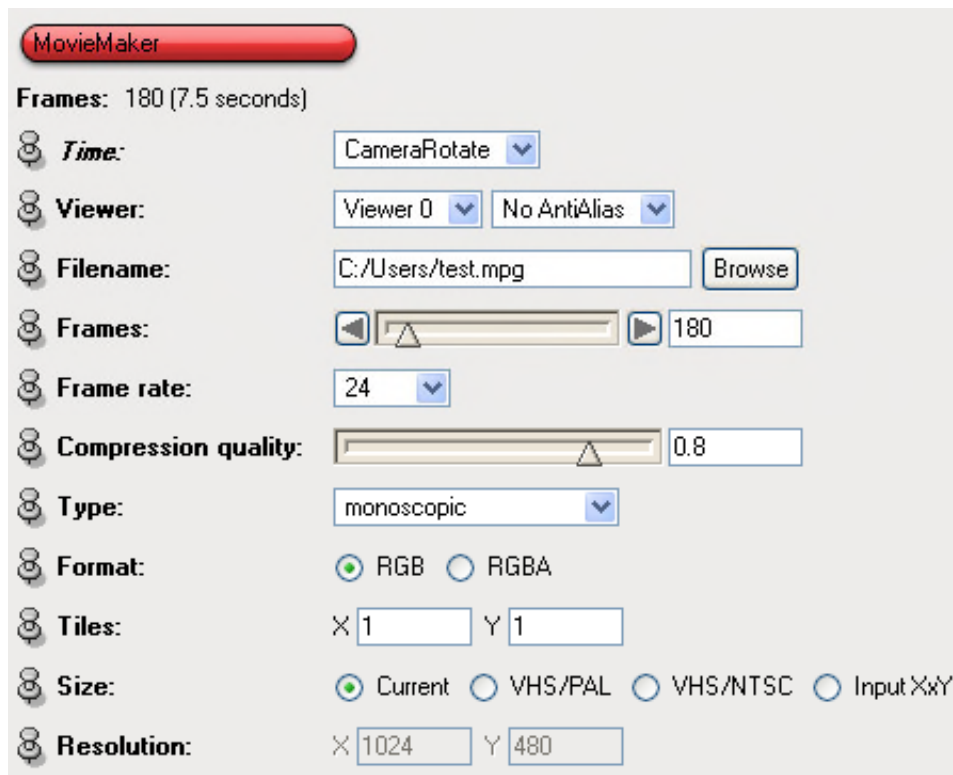
In our example, the *CameraRotate* module has a time slider, so we can attach a *MovieMaker* module to it by right-clicking on the *CameraRotate* icon in the Pool and selecting *MovieMaker* from the popup menu:



In the *MovieMaker* module, first click on the *Browse* button in the *Filename* port and enter a movie file name like `c : /tmp/test .mpg`. The `.mpg` suffix suggests that the movie file format will be MPEG, which is a widely accepted standard format for digital movies achieving a good compression ratio.

Next, adjust the parameters of the *MovieMaker* module to your liking, e.g., change the *number of frames*, the *image size*, or the *compression quality*. Please refer to the *MovieMaker* documentation for details.

In our example, let us choose 180 frames and leave all other parameters untouched. Since the *CameraRotate* module does a full rotation of 360 degrees, each of the 180 frames will represent a rotation of two degrees with respect to the previous frame. Press the *Create Movie* button to start recording.



MovieMaker

Frames: 180 (7.5 seconds)

Time: CameraRotate

Viewer: Viewer 0 No AntiAlias

Filename: C:/Users/test.mpg Browse

Frames: 180

Frame rate: 24

Compression quality: 0.8

Type: monoscopic

Format: ☒ RGB ☐ RGBA

Tiles: X 1 Y 1

Size: ☒ Current ☐ VHS/PAL ☐ VHS/NTSC ☐ Input XxY

Resolution: X 1024 Y 480

Wait for some time while the *MovieMaker* module drives the *CameraRotate* module and accumulates the snapshots. *Please note that the speed during the recording process is different than the playback speed of the movie.* Now view the resulting movie file `test.mpg` with a movie player of your choice (e.g., Windows Media Player or a similar tool). Experiment with the recording parameters until you get the desired result (e.g., control the file size and image quality by changing the *Compression quality* value, choose different image sizes to see up to which image size your computer is capable of smoothly displaying the movie, and change the number of frames to control the speed of the rotation).

2.12.2 Attaching MovieMaker to DemoMaker

Now we try to record a movie of a more complex animated scene. To this end, we load one of the networks that we have created in in Section 2.11: load network.

As you might remember, the basic idea of the *DemoMaker* module was that you define a set of events to be executed on a certain time line. Check this out by clicking the play button of the time slider in the *DemoMaker* module. You should see a nicely animated demonstration.

If you remember the previous section in this tutorial, you might already have an idea of how we can record this animated demonstration into a movie file. Like the *CameraRotate* module in the first example, the *DemoMaker* module is controlled via a *time slider* that we can attach to. So simply right-click on the *DemoMaker* icon in the Pool and attach a *MovieMaker* module. Like before, enter a movie file name and select the number of frames before you click on the *Create Movie* button to start recording.

2.13 Using MATLAB Scripts

In this tutorial you will learn how to integrate complex calculus into Amira using MATLAB (The MathWorks, Inc) by the means of the *CalculusMatlab* module. In order to use the *CalculusMatlab* module, MATLAB 7 must be correctly installed on your computer. The *CalculusMatlab* module establishes a connection to the MATLAB computational engine that was registered during installation. If you did not register during installation, on the Windows command line you can enter the command:

- `matlab /regserver`

The limitations of the *CalculusMatlab* module are listed in its *documentation*.

This tutorial covers the following topics:

1. Loading and executing a MATLAB script.
2. Lowpass filtering on images.
3. Controlling the script with a time slider.
4. Thresholding on a volume.

2.13.1 Lowpass Filtering on Images

In this section we will learn how to apply a lowpass filter on an Amira image using the MATLAB Fourier transformation. This example shows how to pass data and control variables from Amira to MATLAB, execute a MATLAB script, and import the data back into Amira.

- Load the *lena.png* image file located in subdirectory *data/tutorials/matlab*.
- Choose *Luminance* in the *Channel Conversion field* as shown in Figure 2.30.
- Right click on the green icon and choose *CalculusMatlab* from the Compute section.



Figure 2.30: Loading the image

A new red icon appears, the CalculusMatlab module that will try to connect to the MATLAB engine. This may take a while.

- Load the script *lowpass.m* located in subdirectory *data/tutorials/matlab* by clicking the *Read* button of the *File* port.
- Execute the script by clicking on the *buffer* button of the *Execute* port.
- Connect an *OrthoSlice* to the filtered image *result*.

The module uses the MATLAB computation engine which has its own user interface.

You can easily show or hide the MATLAB console using the checkbox in the options field. The MATLAB console is very useful for debugging purposes because it allows you to access variables of the MATLAB workspace. Any variable not cleared by the MATLAB "clear" command in the script is accessible in the MATLAB workspace, even after finishing the current CalculusMatlab computation (see the *CalculusMatlab* documentation).

In addition you can control scalar parameters of the script using time sliders:

- Create a time slider (*File/Create/Data/Time*).
- Connect the CalculusMatlab module to the time slider.
- Change the line `cutoff=0.05` to `cutoff=t` in the script (see the *CalculusMatlab*

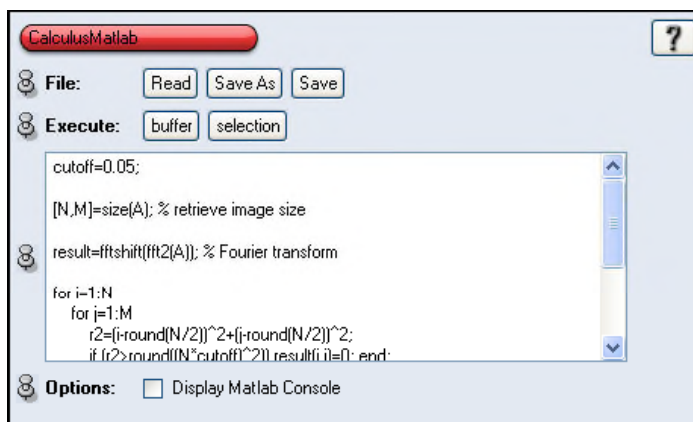


Figure 2.31: The CalculusMatlab module



Figure 2.32: Left: original Right: lowpass filtered

documentation for more information about the keyword t).

- Click on the time slider and adjust the value that will be assigned to w. Right mouse click in the text field of the time slider and select *Configure* to adjust the data range of the parameters.

Note: To handle RGBA image filtering, you must load the image with Color Field Channel Conversion and treat each channel separately in the script.

2.13.2 Thresholding on a Volume

In this section we will learn how to apply a threshold to a volume. This is done by setting a value for a threshold. If the value for the voxel is less than the threshold, the voxel value is assigned the value of zero. If it is above the threshold, it is assigned a value of 255.

- Load the file *lobus.am* located in subdirectory *data/tutorials*.
- Right click on the green icon and choose *CalculusMatlab* from the *Compute* section.
- Load the script *threshold.m* located in subdirectory *data/tutorials/matlab* by clicking the *Read* button of the *File* port.
- Execute the script by clicking on the *buffer* button of the *Execute* port.

A new green icon appears, the Lattice that will hold the threshold result. Connect an *OrthoSlice* or a *Voltex* to see the result.

Note: Any variable accessed by MATLAB and pushed back into the Amira workspace will lose its voxel size information. You will need to correct the voxel size manually using the *Crop Editor*.

2.14 Introduction to the Filament Editor

This section provides a step-by-step introduction to the Filament Editor.

The Filament Editor is a special purpose sub-application in Amira that is designed to extract complex three-dimensional networks of filamentous structures from image data and post-process such data by editing and annotating the network with label scalar data. In this tutorial we want to demonstrate the principles of the Editor by extracting the dendritic fiber network of an invertebrate neuron. The data set we will be working with depicts a neuron from the honeybee brain imaged with a confocal microscope and was kindly provided by Prof. R. Menzel, Free University of Berlin.

In this section the following topics will be discussed:

1. Exploring the volume data.
2. Automatic extraction of the network.
3. Interactive tracing.
4. Labeling and visualizing the network.

To access the Filament Editor, click the icon in the *sub-application bar*. We begin our work by loading the 3D image data set.

- Load the file `neuron.am` located in the `data/tutorials/neuron` subdirectory.

2.14.1 Exploration of the volume data

Once the image data has been loaded, the left-hand panel of the viewer shows a 2D slice of the volume, while the right-hand panel shows the 3D objects. In order to have a clear idea of how the neuron spreads out in space, we explore the volume by using slicing, windowing, and rendering the gray values.

- **Window Level** Select the *Window Level* icon, click the 2D viewer, hold the mouse button pressed, and drag the mouse cursor left/right to change the window width, or up/down to change the window center. Set the window level around 30-80. Click the 3D button to visualize the selected voxels using a shaded volume rendering.
- **Browse Slice** The browsing tool allows using the mouse to navigate through an image stack in the 2D viewer (or MPR viewer). Select the Browse Slice icon, click the 2D viewer, hold the mouse button pressed, and drag the mouse cursor up/down to scroll forward or backward through the image stack. If you are working with a wheel mouse you can use the wheel instead of this icon. Simply click the viewer and scroll the mouse wheel to scroll through the image stack.
- **Thick Slice** The thickness of the slice can be set by sliding the *Thickness* slider. When displaying a thick slice, data values are computed as maximum intensity of the values of the original slices. Set thickness to 25. Click the 3D slice button to visualize the thick slice in the 3D viewer.

2.14.2 Automatic extraction of the dendritic tree

In a first step you may want to automatically extract what will serve as a draft version of the neuron's skeleton. This can be achieved with the Filament Editor's *Auto Skeleton* tool.

- Set the *Window Level* to 37 137.
- Click the 3D button on beside the *Window Level* slider. The 3D viewer shows a shaded volume rendering of the neuron's main branches and gives

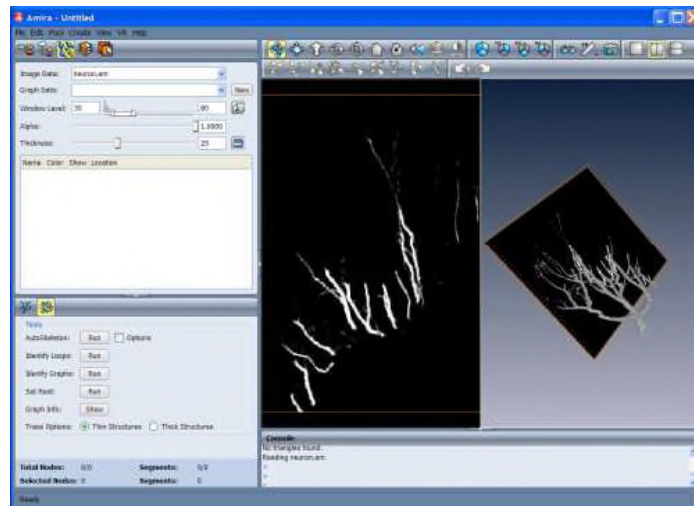


Figure 2.33: The Filament Editor immediately after the image data has been loaded. The left-hand panel of the viewer shows a 2D slice of the volume, while the right-hand panel shows the 3D objects. The 3D and slice rendering have been activated in the 3D viewer.

a rough estimate of what the automatic tracing procedure will consider as neuron.

- Select the *Tools* tab in the **Tool Box** and press the *Run* button of the *Auto-skeleton* port. After a few seconds you will see green lines and blue points in the 2D viewer. In the 3D viewer gray spheres are displayed together with the preview rendering. If you do not see them, click the icon *View All* (or the space bar) to center the 3D viewer on them.
- Adjust the *Alpha* slider, which is appeared when you activated the 3D rendering. You can see the generated graph through the rendering.

The *Auto Skeleton* tool traces connected regions according to a user-defined window level and converts the centerline of those regions into graphs composed of points, segments, and nodes, which are the elements of the data class *Spatial-Graph*. The result of the "skeletonization" is, therefore, a *SpatialGraph* object, which is being visualized in the 3D viewer as balls and lines, and as blue points connected by green lines in the 2D viewer.

Depending on the quality of the data and on the selected window level the result of skeletonization will typically show several disconnected elements and as well

as loops within the networks. Disconnected elements will break the single neuron into several sub-graphs. Loops, on the other hand, are parts of a graph where segments connect onto itself. In some biological objects like e.g. neurons loops must not occur. To identify graphs and loops the Filament Editor offers dedicated tools.

- Press the button *Identify Graphs* to automatically detect and label all graphs in the SpatialGraph object, that is all connected components. This creates a new label group in the Label Editor named *Identified_Graphs* under which all identified graphs are listed as *Graph1*, *Graph2*, ..., *GraphN*.
- To visually identify them just click on one of the items in the *Identified_Graphs* list. The selected item is highlighted in red in both viewers.
- Under *Identified_Graphs* select *Graph2* and shift-select *Graph25*, then remove them by clicking the *Delete Selection* icon.
- Press again the button *Identify Graphs*, now you should have only two graphs.
- Scale the nodes by moving the *Node Scaling Factor* slider in the *View Options* tab of the **Tool Box** (we will see the details in the next paragraph).

At this point we have only two graphs: Graph0 is the large tree, while Graph1 is just three segments and four nodes. Switch back to the *Tool* tab in the **Tool Box**.

- Select *Graph1* in the Label Editor under *Identified_Graphs*
- Activate the *Select Single* icon and Ctrl-click on the closest node of Graph0 in the 3D viewer.
- Select the *Connect Selected* icon to connect the graph.
- Press again *Identify Graphs*. At this point you should have only one graph.
- Press *Identify Loops* to get another label group *Identified_Loops*.
- Under *Identified_Loops* select *Loop2* and remove it by clicking the *Delete Selection* icon. Note that a segment is defined by two nodes. Therefore, when you remove a node the associated segment will be removed too. In this example this will not happen, but in general it is very likely that some connected segments are removed when a loop is deleted.

It should perhaps be noted that the *Auto Skeleton* tool is also available as a compute module in the Pool under the *Skeleton* category. Also, there is a rich set of tools available in the *Segmentation Editor* to perform the threshold segmentation necessary for the skeletonization process. Finally, in the case of a strict tree topology it

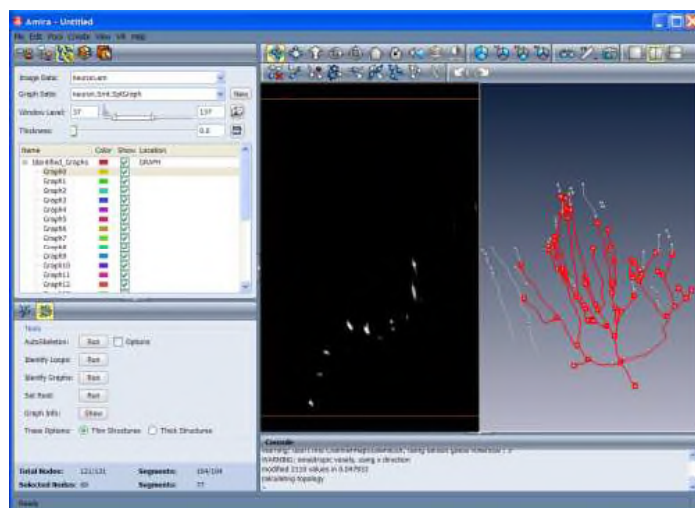


Figure 2.34: The SpatialGraph object obtained with the *Auto Skeleton* tool. Note the highlighted main graph.

may be advantageous to restrict the search to a tree. In this case you may want to use module *CenterlineTree* to extract a graph with guaranteed tree topology. Before closing this subsection, we should save the *SpatialGraph* data object we have created, it will be useful in further step of this tutorial.

- Switch back to the Pool, save the object called *neuron.Smt.SptGraph* in a directory of your choice and then remove it from the Pool.
- Switch again to the Filament Editor to prepare the next step.

2.14.3 Filament Tracing

The *Interactive Tracer* tool enables you to trace the filaments directly on the gray values using an innovative tracing algorithm developed by Visage Imaging. The user sets the starting and ending points of a segment and the *Interactive Tracer* finds the shortest line connecting the two points with respect to the user-defined data window. Now we can use the *Interactive Tracer* to re-trace the neuron segments. In order to clearly understand how to use the tracer, we will try to re-trace a small part the of the neuron tree we extracted in the previous paragraph.

- Press the *New* button at the *Graph Data* port to crate a new *SpatialGraph* data object.

- Click in the 2D viewer. Using the *Browse Slices* tool or the mouse wheel, slice through the volume until you reach the root of the neuron.
- Set the thickness of the *Thick Slice* at 10 for a better visualization.
- Activate the *Interactive Tracer* by clicking the *Trace Filament* icon in the toolbar and make sure that the *Thick structures* option is checked in the *Tools* tab.

Note that the *Interactive Tracer* is always active while the *Trace Filament* icon is highlighted and it can be triggered only if the 2D viewer is active.

If you now move the mouse over the 2D slice, the cursor has a cross shape whenever it is over black regions and turns into a crosshair (cross within circle) when it is over voxels that are within the window level. The cross-hair shape indicates that it is possible to set tracing key points. As you will see later, when the pointer is over a traced segment (green line) a small "P" appears in the crosshair. This indicates that a click would select the nearest point as a key point. Likewise, an "N" is shown in the cursor whenever the cursor is over a node and a click will select this node as a key point. Furthermore, the cursor turns red when the trace tool is in **append mode**, meaning that the next click will trigger a tracing action between the previously selected and the current key point.

- Start the tracing by clicking on a gray value of the tree root.
- Slice ahead and set further points, until you reach the first bifurcation and click on it.
- Deactivate the *Interactive Tracer* e.g. by clicking the *Select Single* tool or by clicking again its icon.

If you press *Identify_Graphs* graphs, you will see only one graph. You can also access some statistical information by clicking on *Graph Info* button in the *Tools* tab. This opens a spreadsheet window containing a list of all segments in the graph. The spreadsheet is interactive, i.e. if you click on a line, the corresponding segment will be highlighted in the viewer.

Repeat the actions to trace the first branching sections from the root.

2.14.4 Visualize the network

The *SpatialGraph* data object is able not only to store the spatial structure, i.e., the geometry of the network, but also to associate it with scalar and label data.

Labels can also be used to tag or annotate sub-graphs according to their topology.

In this simple example we want to tag the three main branches of the *Topology* as *Central*, *Left*, and *Right* of a neuron tree.

If you already extracted the graph as described in the second subsection you should now load the file you previously saved, otherwise you should go two steps back and read the subsection "*Automatic extraction of the dendritic tree*".

- Right-click into the Label Editor and select *Add graph label group*.
- Right-click on the created label and rename it *Topology*.
- Right-click on the *Topology* label and select *Add label*. Repeat this action to add three labels.
- Select and right-click on each of the created labels and rename them *Central*, *Right*, and *Left*.
- Select the *Select Lasso* icon and select-lasso the right-hand branch. You can use the *Select Lasso* tool in combination with *Shift* key to deselect-lasso or with the *Ctrl* key to add elements.
- Right-click on label *Right* and select *Assign selection*.
- Repeat the last two items to label *Central* and *Left* branches. Note that this is just an exercise, it does not really matter to exactly select the "Center", "Right" or "Left" segments.

Select now the *View Options* tab in the *Tool Box*

- Scale the nodes using the *Scaling Factor* slider.
- Colorize the nodes according to the label by selecting *Topology* in *Node Coloring*.
- Repeat the last item for the segments.
- Click on the color button of label *Right* and select a new color.

For advanced network visualization you can switch to the Pool or Tree View. If you do so, you may attach a *SpatialGraphView* module to the *SpatialGraph* object we created before.

2.14.5 Alternative way to extract a network using the Segmentation Editor

From the Filament Editor just switch to the *Segmentation Editor* by clicking the icon in the *Sub-application Tool Bar*. For an in-depth treatment of the Segmentation Editor we refer you to its documentation and tutorial. For this simple example you may try the following simple steps.

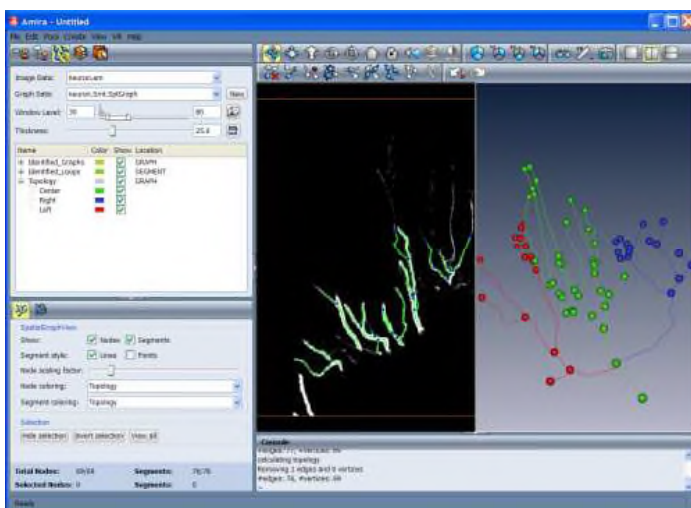


Figure 2.35: The visualization of the SpatialGraph obtained with the *AutoSkeleton* and *Tracing* tools.

- Create a new label field by pressing the *New* button in the *Label Data* port located in the upper part of the user-interface.
- Create a new material by pressing the *New* button in the *Materials* port.
- Right-click the new material in the *Material List*, select *Rename Material* and enter "neuron".
- In the area *Display and Masking* area set the threshold to 100-255
- In the tool box at the bottom part of the user-interface, select the *Threshold* tool and click the *Select* button.
- in the *Selection* box click the "+" button to assign the selected region to material "neuron".
- Switch back to the Filament Editor and select *neuron.Labels* from the *Image Data* pull-down menu. Now skeletonize the label field by pressing the *Run* button in the *Auto Skeleton* tool.

2.15 Introduction to the Multi-planar Viewer

The Multi-planar Viewer can be used to visualize one or two data set at the same time using a Multi Planar Reconstruction (MPR) and a 3D viewer. Addition-

ally, the image volumes can be registered automatically or manually using specific graphical interfaces.

In this tutorial we will try to illustrate how to:

1. Exploration of the volume data, including an associated label field
2. Explore two data sets in fusion mode
3. Register two data sets

Before we begin our work we need to define the terms *Primary* and *Overlay* in the context of the Multi-planar Viewer

1. **Primary** - Reference data set, this data set will be considered our principal data set to which a secondary data set could be eventually compared or registered. In the context of PET/CT fusion the CT data set would be used as reference data set as it provides the best anatomical reference.
2. **Overlay** - Secondary data set, which could be compared or registered to the Primary.

2.15.1 Exploration of the volume data

To activate the Multi-planar Viewer select its icon in the sub-application toolbar. On a first glance the Multi-planar Viewer looks similar to the Segmentation Editor: three slice views, a 3D viewer and a material list. In order to start exploring, load the Lobus data set located in the tutorial directory `data/tutorial/lobus.am`:

- **3D Volume Rendering** Select the 3D Rotate icon (it should have been selected by default as you start the Multi-planar Viewer) in the toolbar. Now rotate the object in the 3D viewer.
- **Volume Rendering Modes** Try to visualize the volume using VRT or MIP techniques and their optional rendering modes. To do so, switch between the radio buttons in the 3D Settings panel.
- **3D MPR** Click the icon in the 3D Settings panel to activate MPR in the 3D viewer. You should see now the volume rendered in 3D and the three MPR slices.
- **Window Level** Select the Window Level icon, click into one of the 2D viewers, hold the mouse button pressed, and drag the mouse cursor left/right to change the contrast (window width), or up/down to change the brightness (window center). Note how the bars in the graphical user interface (GUI)

change according to the pointer movements. Click in the 3D viewer and do the same.

- **Browse Slice** The browsing tool allows using the mouse to navigate through an image stack in the 2D viewer. Select the Browse Slice icon, click in a 2D viewer, hold the mouse button pressed, and drag the mouse cursor up/down to scroll forward or backward through the image stack. If you are working with a wheel mouse you can use the wheel to access the browse slice functionality. Simply click once the viewer to activate it and roll the mouse wheel to scroll through the image stack.
- **Thick Slice** The thickness of the slice can be set by sliding the Thickness slider. When displaying a thick slice, data values are computed as the average, the minimum or the MIP of the values of the included slices, according to the selected option.
- **Label field** Load the associated label field (`lobus.label.am`) from the tutorial directory and select the checkbox of the Label Field port. Visualize the materials in 3D with constant color, hide or fade them. Note that the port Label field refers to the Primary data set; to be activated the loaded LabelField has to have the same size as the Primary.
- **Extend viewers** Double-click one viewer to extend it, double-click to set the configuration back to 4-window layout again.

Note that clicking the right button in the viewers activates the context menu, which provides a quicker way to access tools and settings.

2.15.2 Explore two data sets using fusion mode

The process of *image fusion* consists of combining relevant information from two or more images into a single image using different color mappings. The Image Fusion became very relevant especially in medical science, where different imaging modalities can be obtained from the same patient. Typically, the multi-modal imaging includes magnetic resonance (MR), computed tomography (CT), and positron emission tomography (PET/SPECT). Image fusion can be very useful in microscopy imaging for alignment and registration of multi-channel, 3D, and 4D data sets.

When only one data set is available in the Pool, the Overlay Data port is not available. Load the Lobus2 data set (`data/tutorial/lobus2.am`) from the tutorial directory and the checkbox of the port will become active. Select the Overlay checkbox and Lobus2 will be shown in the available viewers. Change the min-

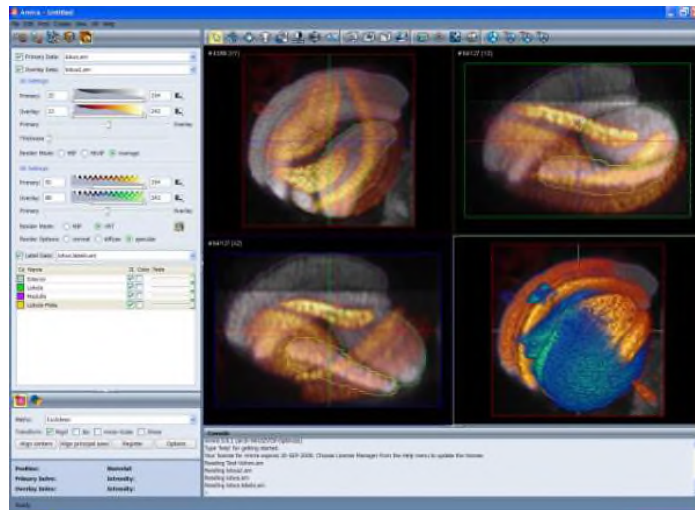


Figure 2.36: The Multi-planar Viewer immediately after the image data has been loaded.

imum and maximum values of the Overlay color map. To do so, select the 3D viewer to activate it, slide the colormap bar to the left and the right or change the upper and lower limits. You can do the same for the 2D viewers as well.

Select again the *Window Level* icon, keep the mouse button pressed and move the pointer: the window level changes only for the Primary data set. When Primary and Overlay images are loaded together, the Primary is the main object in the Multi-planar Viewer. The activated tools refer always to the Primary or to Primary and Overlay together, never to the Overlay individually.

You should be familiar now with the 2D and 3D visualization options. Therefore, experiment with changing the color maps or the blend of Primary and Overlay data sets.

To better explore internal structures of the two volumes you can use the 3D cropping and clipping tools offered by the Multi-planar Viewer.

- Set the 3D visualization mode to VRT with specular light effect.
- Click the Crop Corner icon. Browse the slides in 2D and rotate the volume in 3D to better see the cutting region.

The Crop Corner tool is very similar to the Corner Cut module available in the Pool. You can set the cutting boundaries directly by moving them in the 2D viewers. Experiment with the other cutting tools.

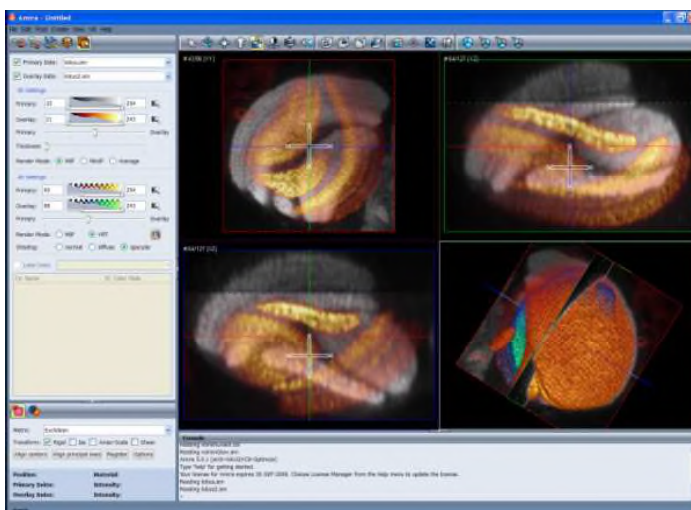


Figure 2.37: The Multi-planar Viewer with the Manual Registration tool activated.

2.15.3 Manually register two data sets

You might have noticed that as soon as Lobus2 was loaded, the icon Registration tool was activated. The registration tools work only when two data sets are loaded in the Multi-planar Viewer. Click now the Registration Tool icon and you will see an arrow-cross in the center of the 2D viewers of the Overlay (we assume that the Overlay is to be registered to the Primary). Move the pointer along the cross: the shape of the pointer changes: Arrow-cross - translate Turning arrow - rotate Arrowhead - scale the image volume

You can also digitally transform the volume by using the Transform Editor options tab in the Registration Toolbox at the bottom of the user interface. This editor corresponds to the user interface of the "Absolute" tab of the *Transform Editor* dialog. More information is available in its help file.

- Activate the Registration tool by clicking its icon.
- Try to manually align Lobus2 to Lobus.
- Use the Transform Editor options in the Registration toolbox to digitally refine the registration.

Once the two volumes are roughly aligned, use the Automatic Registration tool in the Registration toolbox to refine the results. You can get more information about

automatic registration by reading the Registration tutorial.