

amiraMol[®] 3.0 amiraDeconv[®] 3.0

Extensions for amira 3.0

Copyright Information

©1995-2002 Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Germany

©1999-2002 Indeed - Visual Concepts GmbH, Berlin, Germany

©1999-2002 TGS Template Graphics Software, Inc., USA

All rights reserved.

amira, amiraMol, and amiraDeconv are developed by Indeed - Visual Concepts GmbH (www.indeed3d.com) and are marketed and distributed exclusively by Template Graphics Software, Inc. (www.tgs.com).

This document contains proprietary information of Indeed - Visual Concepts GmbH, and is protected by German and international law. The contents of this document may not be translated, copied, or duplicated in any form, in whole or in part, without the express written permission of Indeed - Visual Concepts GmbH.

The information contained in this document is subject to change without notice. Neither Indeed - Visual Concepts GmbH nor their employees shall be responsible for incidental or consequential damages resulting from the use of this material or liable for technical or editorial omissions or errors made herein.

Contents

I	amiraMol	1
1	amiraMol Introduction	3
1.1	First Steps with Molecular Visualization in amira	3
1.1.1	Getting Started with Molecular Visualization	4
1.1.2	Selection, Labeling, and Masking	6
1.1.3	Alignment of Molecules	12
1.1.4	Molecular Surfaces	16
1.1.5	Sequential and Structural Alignment	19
1.1.6	Editing of molecules	20
1.1.7	Molecular Interfaces	23
1.1.8	Measurement	25
1.2	Molecular Data Structures	26
1.2.1	Internal Structure of Molecules	26
1.3	Displaying Molecules	27
1.3.1	Coloring Molecules	27
1.3.2	Selecting and Filtering atoms	28
1.4	Aligning Molecules	30
1.4.1	Alignment of Trajectories	30
1.4.2	Mean Distance Alignment	32
1.4.3	Sequence alignment	32
1.5	Visualizing Molecular Trajectories and Metastable Conformations	32

1.6	Atom Expressions	32
1.6.1	Overview	32
1.6.2	Grammar	33
1.6.3	Literals	33
1.6.4	Operators	34
1.6.5	Shortcuts	36
1.6.6	Further Examples	37
1.7	Molecule Selection Browser	37
1.7.1	The scroll window	37
1.7.2	Highlighting section	40
1.7.3	Menu bar	40
2	Alphabetic Index of Modules	43
2.1	AlignMolecules	43
2.2	AlignSequences	45
2.3	BondAngleView	47
2.4	BondCalculation	48
2.5	ClosestTimeStep	49
2.6	CompMolInterface	50
2.7	CompMolSurface	51
2.8	ConfigurationDensity	53
2.9	MeanMolecule	56
2.10	Measurement	57
2.11	MolSurfaceView	59
2.12	MoleculeLabel	61
2.13	MoleculeView	63
2.14	Observables	65
2.15	PrecomputeAlignment	66
2.16	SecStructureView	67
2.17	TubeView	71

3	Alphabetic Index of Data Types	73
3.1	MolSurface	73
3.2	MolTrajectory	74
3.3	Molecule	74
4	Alphabetic Index of File Formats	79
4.1	MDL	79
4.2	PDB	79
4.3	PSF/DCD (CHARMM)	81
4.4	Tripes	81
4.5	UniChem	81
4.6	ZIB Molecular File Format	81
5	Alphabetic Index of Editors	89
5.1	Molecule Attribute Editor	89
5.2	Molecule Editor	92
II	amiraDeconv	97
6	amiraDeconv Introduction	99
6.1	General remarks about image deconvolution	100
6.2	Data aquisition and sampling rates	101
6.3	Standard Deconvolution Tutorial	102
6.4	Blind Deconvolution Tutorial	109
6.5	Bead Extraction Tutorial	112
6.6	Performance issues and multi-processing	118
7	Module Reference	119
7.1	BeadExtract	119
7.2	Convolution	120
7.3	CorrectZDrop	122
7.4	DataPreprocess	122

7.5	Deconvolution	124
7.6	FourierTransform	126
7.7	PSFGen	127
8	Examples	129

Part I

amiraMol

Chapter 1

amiraMol Introduction

amiraMol is an amira extension providing support for the visualization as well as the analysis of molecules and dynamic molecular data.

The amiraMol documentation is organized into the following sections:

- *Getting started with molecular visualization*
- *Structure and interdependence of the molecular data structures*
- *Essentials for displaying a molecule*
- *Alignment facilities in amiraMol*
- *Visualizing dynamic data*
- *Atom expressions*

If you are not yet familiar with amira, please read the amira User's Guide first, or go back to it if you find any general amira terminology that you are not familiar with.

1.1 First Steps with Molecular Visualization in amira

This chapter gives you an overview of the visualization of molecular data sets. amira is not just able to display a 3D image of the molecule but also provides tools to investigate its distinct parts and properties. After reading the "getting started" introduction you may continue with any of the following tutorials.

- *Getting Started* - first steps
- *Selection, Labeling, and Masking* - exploring a molecule
- *Alignment of Molecules* - visualizing dynamical data

- *Molecular surfaces* - wrapping a molecule
- *Sequential and Structural Alignment* - comparing molecules
- *Molecule Editor* - interactive manipulation of the molecule
- *Molecular Interface* - computing intersection surfaces
- *Measurement* - measuring distances and angles within a molecule

Note: If you want to visualize your own data, please refer to the section about data import in the amira User's Guide. That section contains some general hints on how to import data sets into amira.

1.1.1 Getting Started with Molecular Visualization

In this section you will learn how to

1. load a molecular demo data set into amira,
2. view the molecule with the *MoleculeView* display module,
3. try out various color schemes,
4. select atoms in the viewer using the draw tool,
5. overwrite color scheme colors for selected atoms.

1.1.1.1 Loading Data into the System

In the following introduction we will consider a simple example to explain the basic functions of the *MoleculeView* module. Our example will be a small PDB (Protein Data Bank) structure consisting of 932 atoms in 213 residues.

- Load the file `2RNT.pdb` into the object pool from the directory `data/molecules/pdb`.

A green icon labeled `2RNT.pdb` will appear in the object pool. The green icon represents an object of type *Molecule*. Click on the green data icon with the left mouse button. In the working area below the object pool information about the molecule, such as the number of atoms etc., will be shown. In addition, other ports named *Transformation*, *Selection Browser*, and *Transform*, can be seen; they will be explained in later tutorials.

1.1.1.2 Displaying the Molecule with the MoleculeView Module

The *MoleculeView* module is the basic display module for visualizing molecules. It allows you to display atoms as plates or balls and bonds as lines or cylinders.

- Click again on the green data icon in the object pool, this time using the right mouse button.

A menu, containing several entries and submenus, will appear.

- Select the entry *MoleculeView*.

A new yellow icon labeled *MoleculeView* appears in the object pool. Yellow icons, in general, represent display modules, i.e., modules that visualize objects in the viewer. The blue line between the icons indicates a connection between the objects. In this case the *MoleculeView* module reads data from the object represented by the green icon and visualizes it.

The molecule is now displayed in the viewer as a wireframe. Using the left mouse button you can rotate the object; using the left and middle mouse buttons simultaneously you can zoom in and out. For these mouse operations to work, the viewer must be in *viewing mode*, in which case the mouse cursor is displayed as a hand.

Whenever the *MoleculeView* module is active, the little square on the yellow *MoleculeView* icon is orange. You can deactivate the module by clicking on the square with the left mouse button.

We will explore some basic ports of the *MoleculeView* module.

Mode port: Choose another mode to see both atoms and bonds of the molecule, or just atoms. If atoms are shown, use the *Atom Radius* port to adjust the size of the atoms as desired.

Quality port: If you choose the option *correct*, you can display a correct, i.e. three-dimensional, image of the balls and sticks. Consider the trade-off between correct representation and slower rendering performance. Use the *fast* mode if you want to display a large molecule. If your graphics hardware is fast enough, you can use the *correct* mode even for large molecules.

Complexity port: In order to allow interactive rendering, the default complexity of the scene is rather low. In most cases this will be sufficient. However, if you want to make screen shots or if you have small molecules, you might want to increase the complexity. The *Complexity* port is displayed only if *correct* quality is selected.

1.1.1.3 Changing the Color Scheme

The *MoleculeView* module allows the user to color the molecule according to levels, for example, atom level, residue level, secondary structure level, chain level, or user-defined levels. Each level has a number of attributes, the simplest of which is the *index* attribute. The default color scheme is to color the molecule according to the atom level's attribute *atomic_number*, i.e., the atom's type.

- Click the *Legend* button of the *Color* port to display a small window decoding the colors you see in the viewer window.
- Choose *residues* from the first pop-up menu and *type* from the second menu to color the molecule according to the residue type, here the amino acid type.

The default color map contains a constant color. Thus, currently all residues of the molecule have the same color.

- To change the color map, right-click on the color bar of the *Discrete CM* port which has appeared below the *Color* port and select any other color map.

- Try out different color maps to find a map that suits your purposes.

Some people prefer the *cpk* color scheme for atoms and the amino acid colors as they are used in the *Rasmol* program.

- You can set your preferences in the *amiraEdit* menu by selecting the entry *Preferences*.
- Press the *Molecules* tab and set your preferences. Pressing the *OK* button saves your preferences permanently.

Currently only amino acid colors are predefined. Thus, if the molecule contains residues other than the standard amino acids, those residues will be colored with the default color (black).

1.1.1.4 Using the Draw Tool

The draw tool appears in *amira* in several modules. It enables you to select objects or parts of an object by drawing a line in the viewer.

- Press the *Draw* button of the *Highlighting* port and draw a line in the viewer window around the group of atoms you want to select.

The atoms that were enclosed by the line will be highlighted. If the viewer is the active window, pressing the *d* key is equivalent to pressing the *Draw* button of the *Highlighting* port.

- To unhighlight all atoms, press the *Clear* button of the *Highlighting* port.
- Also try using the *Ctrl* and *Shift* keys while drawing a line.

1.1.1.5 Setting Colors for Selected Parts

The *Define Colors* port allows you to overwrite colors of the color scheme.

- Select some atoms using the draw tool.
- Press the *Set* button of the *Define Color* port.
- Change the current color of the *Color Editor* and press *OK*.

The previously highlighted atoms should now have the color selected in the color editor. This setting will be preserved even if the color scheme is changed. Unfortunately, the new setting will currently not appear in the color legend.

1.1.2 Selection, Labeling, and Masking

You already know how to load files and how to display molecules with the *MoleculeView* module. This tutorial will focus on exploring the structure of a loaded molecule. The tutorial consists of three subsections, in which you will:

1. Explore the possibilities for selecting within a molecule using the *MoleculeView* module.
2. Learn how to use the *MoleculeLabel* module.
3. Get to know the *Molecule Selection Browser*.

For this tutorial we will use the molecule `1IGM.pdb`.

- Load the file `1IGM.pdb` into the object pool from the directory `data/molecules/pdb`.
- Attach a *MoleculeView* module to the green object that has appeared,
- and change the *Mode* port to *balls and sticks*.

1.1.2.1 Interactive Selection with the MoleculeView Module

In the first tutorial you learned how to use the draw tool to select atoms within a molecule. The simplest way to select atoms, however, is to click on the atom of interest. In order to do so, the viewer must be in *interaction* mode. If the mouse cursor in the viewing window is depicted as a hand, you are in viewing mode. To change to interaction mode, you can either

- click on the arrow button in the upper right corner of the viewing window or press the `ESC` button while the viewing window is active.

The mouse cursor will change to an arrow.

- Now click one of the atoms.

The atom you selected should now be highlighted by a red frame around it. If you select another atom, the first atom will be unhighlighted. In order to select more than one atom

- press the `Ctrl` key and keep it pressed while clicking additional atoms.

Clicking a highlighted atom a second time unhighlights it.

- Now change the *Mode* port of the *MoleculeView* back to *sticks*.
- Select *residues* from the first menu of the *MoleculeView*'s *Color* port.
- Choose a suitable colormap from the list of pre-loaded colormaps as described in the first tutorial.

The residues should now be colored according to their type.

- Now select an atom that is currently unhighlighted.

As result the whole residue should now be highlighted. The reason for this is that picking is bound to the coloring. For example, if the color scheme is *atoms*, a click on an atom will only influence the

selection for this atom; if the scheme is *residues*, the atom's residue will be selected; if it is *chains*, the atom's chain will be selected, and so on. In some cases, you might wish to select a single atom even though the color scheme is not *atoms*, or you may want to select a residue when the scheme is not *residues*. The **Shift** key is used to deal with this situation. By default, **Shift**-clicking an atom will select the atom's residue. In the preference dialog you can change this behavior to always select atoms when **Shift**-clicking.

- Choose the *Preferences* entry from *amira's Edit* menu.
- Click on the tab labeled *Molecules*.
- Select *atoms* under *Selection using Shift Button* and press *OK*.

Shift-clicking should now always result in selecting a single atom independent of the color scheme.

1.1.2.2 Using the MoleculeLabel Module

So far you have seen that you can select atoms and groups of atoms by clicking on the molecule. The result of the picking is always printed in the console window. This might suffice in some cases. In other cases, however, it is necessary to label the molecule in the viewer so that you can easily track certain groups you are interested in. If you want to use this tool, here is how you do it.

- Click again on the icon *1IGM.pdb* with the right mouse button.
- Select *MoleculeLabel* from the *Display* submenu.

A second yellow icon labeled *MoleculeLabel* should have appeared in the object pool. By default, all clicks will now be handled by the *MoleculeLabel* module. This means that atoms or groups of atoms that were previously highlighted will now be labelled.

- Set the color scheme in the *MoleculeView* module to *residues*.
- Click on the molecule in the 3D viewer.

This action should cause the residue of the selected atom to now be labeled. **Shift**-clicking will result in the selected atom being labeled. Clicking a labeled atom or residue again will cause the label to disappear.

- Select a few residues by clicking and a few atoms by **Shift**-clicking.
- Click on the *MoleculeLabel* icon to view its ports in the working area.
- Change the color of the atom labels by pressing the color button of the *Color* port and selecting a different color in the color editor.
- In the *Levels* port select *residues*. The *Attributes*, *Level Option*, *Buttons*, *Font Size*, and *Color* ports now affect the residue labels.
- Increase the font size (which only increases the font size of the current level, i.e., residues).

- Select the *name* entry in the second menu of the *Attributes* port.

The selected residues will now be labeled by their types and names. In order to understand the *Options* port of the *MoleculeLabel* module

- deselect the last option, *replace attributes*,
- and set the entry of the second *Attributes* menu back to *disabled*.
- Now select a new residue.

The new residue will only be labeled by its type. In contrast, the old residues are still labeled with type and name.

If the first option of the last port, *handle clicks*, is deselected, mouse clicks will be handled just as if the module *MoleculeLabel* did not exist.

- Deselect the *handle clicks* option.
- Select any residue.

The residue is selected, not labeled.

1.1.2.3 Molecule Selection Browser

In this section you will learn the basics about the *molecule selection browser*, which is a very important feature for the investigation of a molecule. The selection browser is a flexible tool for selecting those parts of the molecule that you are interested in. Moreover, it allows you to easily combine different viewing modules into one viewer.

- Select the green icon *1IGM.pdb* by clicking on it.
- Press the *Show* button of the *Selection Browser* port to open the browser.

The scroll view of the browser currently contains three columns, one with the heading *MasterLevel*, the second with the heading *type*, and the third with the heading *MV* which stands for *MoleculeView*. In the first column, all residues of the molecule are displayed. The second column shows the residue type.

- Connect a *BondAngleView* to *1IGM.pdb* by right-clicking the icon and choosing *BondAngleView* from the *Display* submenu.

You will observe a new column in the selection browser, representing the *BondAngleView (BAV)* (see Figure 1.1).

Changing the Appearance of MoleculeView and BondAngleView

In order to get an image similar to Figure 1.3, we first need to set the ports in the *MoleculeView* and the *BondAngleView* modules correctly (Figure 1.2). We begin with the *MoleculeView* module.

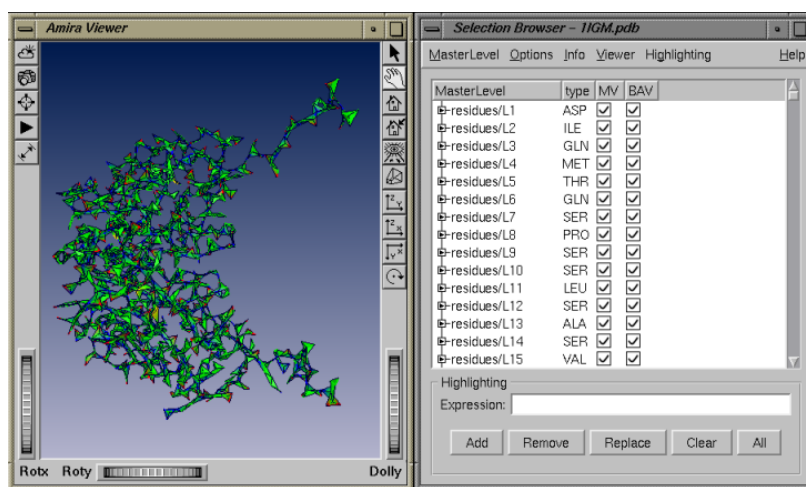


Figure 1.1: Left: MoleculeView and BondAngleView displaying the molecule simultaneously. Right: Selection browser after connecting two viewing modules to the molecule.

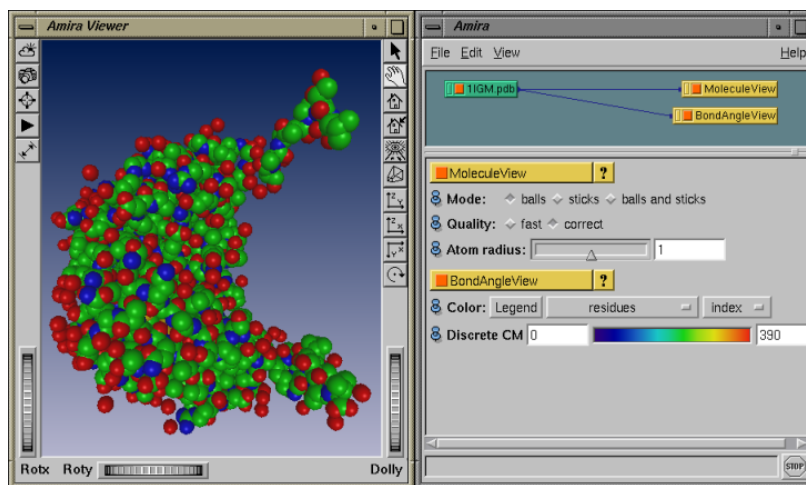


Figure 1.2: Setting the ports in the viewing modules.

- Set the *Mode* port to *balls*.
- Set the *Quality* port to *correct*.
- Set the *Atom Radius* port to 1.0.

For the *BondAngleView* continue as follows:

- Select the *residues* entry from the first menu of the *Color* port.
- From the second menu of *Color* port, select the *index* entry.
- Right-click on the color bar of the *Discrete CM* port and select the colormap *physics.icol*, if this colormap is pre-loaded. Otherwise load it from the directory *data/colormaps*.

We now only see the *MoleculeView*, since the triangles of the *BondAngleView* are hidden by the van der Waals spheres. In order to combine the two viewing modules into one image, continue with the next section.

Highlighting and Masking with the Selection Browser

We now want to use the selection browser to display parts of the molecule with the *MoleculeView* module and the rest with the *BondAngleView* module. In this section we will only use the selection browser, so all menu names etc. refer to this window.

- Select *chains* as new master level from the *MasterLevel* menu.

You will now see three entries in the *MasterLevel* column: *chains/L*, *chains/H*, and *chains/W*. This means that the level *chains* contains three groups. In order to view the group *chains/L* with the *MoleculeView* and the other two groups with the *BondAngleView*,

- remove the check marks for *chains/H* and *chains/W* from the column with heading *MV* by clicking on them.

To deselect the group *chains/L* for the *BondAngleView*,

- click on the respective box in the column *BAV*.

Now you should see an image that is pretty close to that in Figure 1.3. However, the *BondAngleView* displays more triangles than in the figure. The *BondAngleView* in Figure 1.3 only displays *backbone* atoms. In order to achieve this

- right-click on the heading labeled *BAV*.

A pop-up menu as in Figure 1.3 should appear.

- Select *Backbone* from the *Restrict to* submenu.

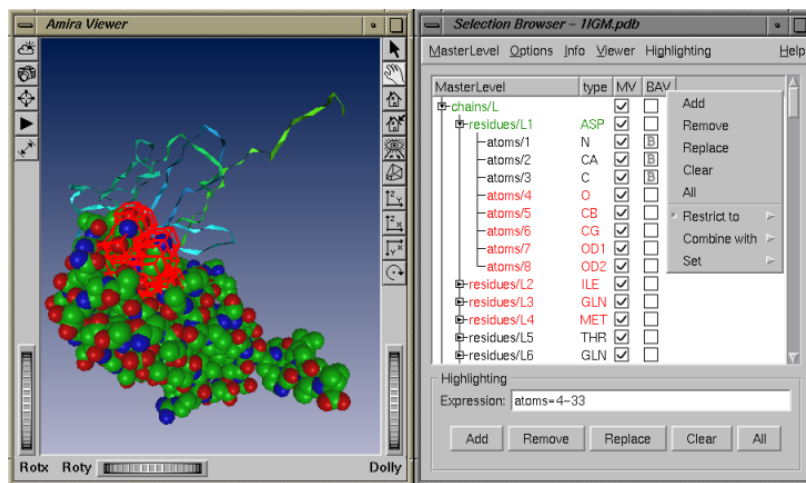


Figure 1.3: Using the Selection Dialog window.

The side-chain atoms should not be visible anymore. Next, you should explore the group *chains/L* a bit further.

- Click on the little '+'-icon left to it.

After this action the residues contained in chain *L* will have appeared.

- Click on the little '+'-icon left to the residue *L1*.
- Type `atoms/4-33` in the text field labeled *Expression*, below the scroll window, and press the *Replace* button.

Atoms 4 to 8 and residues *L2*, *L3*, and *L4* are highlighted in red. The residue *L1* and the chain *L* are highlighted in green. The color *red* means that the whole group is selected, i.e., all its elements. *Green* denotes that the group is partially selected.

To get familiar with the browser, play around with it. If you need further information, just press the **F1** button in the browser window or see the description of the *Selection Browser*.

1.1.3 Alignment of Molecules

In this section you will learn how to

1. align molecules by considering selected atoms,
2. compute a *mean molecule*,

3. compute a *configurational density*,
4. compare metastable molecular conformations.

1.1.3.1 Comparing two Molecules

In this section we will compare two different three-dimensional structures of the same molecule.

- Load the data file `alkane.zmf` from the directory `data/molecules/alkane`.
- Select *MolTrajectory* from the pop-up menu of the *alkane.zmf* icon.
- Select *Molecule* from the pop-up menu of the *MolTrajectory* icon.
- Attach a *MoleculeView* module to the molecule.
- Repeat the last two steps, creating two new objects, *Molecule2* and *MoleculeView2*

The *alkane.zmf* icon represents a bundle of molecular trajectories, in this case butane, pentane, and hexane. By attaching a *MolTrajectory* to it, we extract a single trajectory. By default, the first trajectory, which is butane, is selected. We can now extract single time steps from the trajectory by attaching a *Molecule* object to the trajectory. We have done this twice since we want to compare two time steps with each other. Currently, both molecules extract the same time step. This is the reason for only seeing a single molecule. In order to get more information about the data structures, go to the section on *molecular data structures*.

- Select the *MoleculeView* icon and change the *Mode* port to *balls and sticks* and the *Quality* port to *correct*.
- Repeat this action for the *MoleculeView2*.
- Select the *Molecule2* icon and change the value of the *Time* port to 2.

You should now see two butane molecules in the viewer, slightly displaced.

- Right-click on the small rectangle at the far left side of the *Molecule2* icon and select *AlignMaster* from the pop-up menu.
- Connect the blue line to the *Molecule* icon.
- Select the *Molecule2* icon.
- Press the *all* button of *Molecule2*'s *Select* port.

You have now aligned the molecules *Molecule* and *Molecule2* using a least squares fitting of all atoms. This only works for molecules with the same number of atoms. In the following we will align the two molecules by considering only three carbon atoms.

- Deactivate the *MoleculeView* by clicking on the orange square of the *MoleculeView* icon.
- **Ctr l**-click on three consecutive carbon atoms.

The frame of a red cube should appear around each of them.

- Activate the *MoleculeView* by clicking on the gray square on the *MoleculeView* icon.
- Select the *Molecule2* icon and press the *in slave* button of the *Select* port.

The three selected atoms should now fit better onto the corresponding atoms of the object labeled *Molecule*.

1.1.3.2 Computing a Mean Molecule

In this section we will compute a mean molecule of a metastable molecular conformation of butane.

- Remove the objects *Molecule* and *Molecule2*.
- Select the *MolTrajectory* icon.
- Load the data file *but_cluster_3_1.idx* from the directory *data/molecules/alkane*.

but_cluster_3_1.idx is a subtrajectory of *MolTrajectory*, i.e., a subset of all configurations in the trajectory.

- Attach a molecule to the icon *but_cluster_3_1.idx*.
- Select the *MeanMolecule* entry from the *Compute* submenu of *but_cluster_3_1.idx*'s pop-up menu.
- Right-click with the mouse on the little left-hand rectangle of the *MeanMolecule* icon and select *AlignMaster*.
- Connect the *AlignMaster* port to the *Molecule* icon by attaching the blue line to it.
- Select the *MeanMolecule* icon and press the *all* button of the *Select* port of the *MeanMolecule* module.
- Press the *DoIt* button to compute the mean molecule.
- Visualize the mean molecule *but_cluster_3_1.mean* by attaching a *MoleculeView* to it.
- Connect the *AlignMaster* port of the module *MeanMolecule* to the *but_cluster_3_1.mean* icon.
- Press the *DoIt* button of the *MeanMolecule* module two or three times.

You have now computed a mean molecule of the subtrajectory *but_cluster_3_1.idx*. The *AlignMaster* is used to align all time steps before accumulating the atom positions. Changing the *AlignMaster* to the previously computed mean molecule and repeating the computation of the mean molecule will improve it. This procedure should converge.

1.1.3.3 Computing and Visualizing a Configuration Density

For the subset of configurations contained in the subtrajectory *but_cluster_3_1.idx*, we will now compute a *configuration density* and visualize it with the *Isosurface* display module.

- Select the entry *ConfigurationDensity* from the submenu *Compute* of *but_cluster_3_1.idx*'s pop-up menu.
- Connect the *AlignMaster* port of the *ConfigurationDensity* icon to the *but_cluster_3_1.mean* icon.
- Select the *ConfigurationDensity* icon.
- Press the *all* button of the *Select* port.
- Press the *Field* button of the *Compute* port to compute the density.
- Visualize the created scalar field *but_cluster_3_1.scalar* with an isosurface by first selecting the *Isosurface* entry from the *Display* submenu of the icon's pop-up menu,
- second, setting the *Isosurface*'s *Threshold* value to 50,
- and third, pressing the *DoIt* button of the *Action* port.
- Try different *Threshold* values.

1.1.3.4 Comparing Metastable Molecular Conformations

The set of configurations in the subtrajectory *but_cluster_3_1.idx* belongs to a metastable conformation of butane. In this section we will compute the density of a second metastable conformation and compare the two with each other.

- Select the *MolTrajectory* icon.
- Load the data file *but_cluster_3_2.idx* from the directory *data/molecules/alkane*.
- Compute the mean molecule of the subtrajectory *but_cluster_3_2.idx* by repeating the steps described above for the subtrajectory *but_cluster_3_1.idx*.
- Visualize the second mean molecule with the *MoleculeView* module.
- Select three consecutive carbon atoms in the second mean molecule as was done in the first tutorial.
- Attach the *AlignMaster* of the object *but_cluster_3_2.mean* to the first mean molecule *but_cluster_3_1.mean*.
- Select the *but_cluster_3_2.mean* icon and press the *in slave* button of the mean molecule's *Select* port.

The two mean molecules should now be aligned to each other, i.e., the three selected carbon atoms should superimpose.

- Compute the density for the second subtrajectory using the *but_cluster_3_2.mean* molecule as *AlignMaster*.
- Visualize the computed density with an *Isosurface* module.
- Double-click on the colormap of the *Isosurface2* module and select a different color to better distinguish the two isosurfaces from each other.

You should now clearly see how the two conformations of butane differ. For larger molecules it might be interesting to color the isosurface according to the atom's colors. This can be done using the same *ConfigurationDensity* modules by selecting the *Color Field* option of the *Field* port. Attach the computed color field to the *ColorField* connection port of the *Isosurface* module.

Notice that you can also visualize the densities with the *Voltex* module of the *Display* submenu.

1.1.4 Molecular Surfaces

In section 1.1.1 of this tutorial you have seen how to visualize molecules with the *MoleculeView* module. Section 1.1.2, on selection, labeling and masking, showed you how to use *amira*'s facilities to select, mask, and label certain parts of the molecule. In this tutorial we will

1. compute a molecular surface with the *CompMolSurface* module,
2. compute the molecular surface for a restricted set of atoms,
3. compute partial surfaces,
4. explore the *MolSurfaceView* module,
5. get to know the picking facilities of the *MolSurfaceView*.

1.1.4.1 Compute Molecular Surfaces

In this section you will learn how to use the *CompMolSurface* module to compute molecular surfaces with different resolutions. You will also become familiar with some of the module's basic ports.

- Load the data file *2RNT.pdb* from the directory *data/molecules/pdb*.
- Attach the *CompMolSurface* module to the green icon by selecting the corresponding entry from the *Compute* submenu of *2RNT.pdb*'s pop-up menu.
- Select the *CompMolSurface* icon.
- Press the *DoIt* button of the *Action* port.
- Attach the *MolSurfaceView* module to the newly created green icon, *2RNT-surf*. You will find the entry in the *Display* submenu.

You should now see a gray solvent excluded surface which is still pretty coarse. If you want a surface with better resolution,

- increase the number of points per $\text{\AA}^2$.
- Press the *DoIt* button again.
- Try different resolutions.
- Now select the *no duplicate points* option in the *Options* port.
- Press the *DoIt* button.

Due to the last action, some sharp edges will have disappeared. If there are no duplicate points at sharp edges, the surface normals of the adjacent triangles will be interpolated, thus leading to a smoothing, which in this case might be undesired. However, this option is important if it is necessary to have a closed surface.

- Change the *Quality* port to *possibly incomplete*.
- Press the *DoIt* button.

You should observe that some atoms disappear. This is due to the fact that now only one surface component will be computed. If the molecular surface consists of only one component, the whole surface will be computed. Since the underlying algorithm does not need to touch every single atom, but approximately only every second atom, the algorithm is much faster. If performance is an issue, you might consider using this option.

1.1.4.2 Compute Partial Surfaces

In order to compute partial surfaces we need to select the atoms for which we want to compute their surface contribution.

- Open the selection browser. If you do not know how to do this, take a look at the *second tutorial*.
- Type `within(residues/HET105, 5)` in the browser's *Expression* command line.
- Press the selection browser's *Replace* button.

All atoms within a distance of 5 Å of the *HET105* residue will now be selected.

- Reset the *CompMolSurface*'s *Quality* port back to *correct*.
- Select *partial surface* from the *Options* port of *CompMolSurface*.
- Press the *DoIt* button.
- Select *adjacent patches* from the *Options* port of *CompMolSurface*.
- Press the *DoIt* button.

The last action causes the partial surface to be expanded by the toroidal patches adjacent to the partial surface.

1.1.4.3 Molecular Surface of a Restricted Set of Atoms

The molecule *2RNT.pdb* contains some water molecules, which are in most cases not desired when computing the molecular surface. In order to exclude the water molecule from the surface computation

- open the selection browser again.
- In the selection browser scroll to the end of the list of residues.

- Click on one of the strings *HOH* in the *type* column.

All residues of type *HOH* should now be highlighted.

- In the browser, right-click on the heading *CMS* which stands for *CompMolSurface* and select the entry *Remove*.
- Deselect *partial surface* in the *CompMolSurface* module.
- Press the *DoIt* button of *CompMolSurface*.

For the new surface only those residues that have a check mark in the selection browser were considered. You may combine this procedure with the partial surface computation described above.

1.1.4.4 Exploring the MolSurfaceView

The module *MolSurfaceView* is already attached to the molecular surface. A second connection exists to the molecule *2RNT.pdb*, from which data is read to enhance the *MolSurfaceView*.

- Compute the whole molecular surface with a resolution of 2 points per Å².
- Click on the *MolSurfaceView* icon to see its user interface in the working area.
- Select *molecule* for the *Color Mode*.
- Change the *Color* port's first menu entry to *residues*.
- Select an appropriate color map for the *Discrete CM* port by right-clicking on the color bar.
- Switch to interaction mode by clicking on the arrow button in the upper right corner of the viewing window.
- Click on the surface in the viewing window.

All triangles belonging to the picked triangle's residue will be highlighted. If the triangle belongs to two or even three (maximum) residues, all of those residues will be highlighted.

- Press the *Highlighting* port's *Clear* button to remove the selection.
- Change the *Pick Action* port to *by dist*.
- Pick any triangle of the surface.

All triangles further away from the picked triangle than the distance given by the *Selection Distance* port will be cut off. All triangles within this distance will remain, however, only if they are connected to the picked triangle without leaving the sphere around the picked point.

- Press the *All* button of the *Buffer* port to display the whole surface.
- Change the *Pick Action* port to *surface*.
- Shift-click on the surface in the viewing window.

All clicks on the surface will now be handled as you might be familiar with from the *SurfaceView* module.

1.1.5 Sequential and Structural Alignment

In this tutorial you will become familiar with the *AlignSequences* and *AlignMolecules* modules.

The *AlignSequences* tool facilitates the comparison of two sequences. It can be applied to both proteins and nucleic acids except for t-rna molecules containing modified bases. It is planned to extend this module in the future.

Having done the sequential alignment, you can use the correspondence produced by the sequence alignment to do a structural alignment of the molecules.

- Load the files *1IGM.pdb* and *2JEL.pdb* from the directory *data/molecules/pdb/* and connect a *MoleculeView* to each of them.

You should know how to do this from the *first tutorial*. For both *MoleculeViews*

- select *residues* from the first menu of the *Color* port.

A *Discrete ColorMap* with constant color will appear. To distinguish the molecules, choose a different color for one of them. In order to do so

- double-click on the colorbar and
- change the current color by dragging and dropping any of the custom colors.
- Press the *OK* button of the color editor.

1.1.5.1 Sequence Alignment

We will now use the *AlignSequences* module to align the sequences of the two molecules. In the next section we will use the associated amino acid pairs for a structural alignment.

- Right-click on the *1IGM.pdb* icon and select *AlignSequences* from the *Compute* submenu.
- Right-click the little rectangle inside the *AlignSequences* icon on the left side.

A pop-up menu displaying all connection ports opens.

- Select *MoleculeB* from it and connect the port to the *2JEL.pdb* icon by clicking on it.

There should now be two blue lines connecting the *AlignSequences* icon with the *1IGM.pdb* and *2JEL.pdb* icons, respectively.

- Select the module *AlignSequences* by clicking on its icon in the object pool.

- Choose *semiglobal* from the second menu of *Align Type* port.
- Press the *Align Sequences* button.

If the alignment has been successful, a window displaying several slightly different alignments will appear.

- Press the *AcceptAll* button.

This action results in the alignments being written to the molecules as new levels. In order to check this,

- type `1IGM.pdb list` in the amira console window.

In addition to levels such as *atoms*, *bonds*, *residues*, *etc.* you will also see levels named *semiGlobalSeqAlign1* to *semiGlobalSeqAlign10*. *2JEL.pdb* has the same levels with an equal number of groups.

1.1.5.2 Align Molecules by using the Mean Distance Criteria

We can now use the levels *semiGlobalSeqAlign** for a structural alignment. In order to do so,

- right-click the icon *1IGM.pdb* and select *AlignMolecules* from the *Compute* submenu.
- Right-click the little rectangle inside the *AlignMolecules* icon on the left side, select the *MoleculeB* entry, and connect the module to *2JEL.pdb*.
- Select the *AlignMolecules* icon to make it appear in the working area.
- Select any of the levels named *semiGlobalSeqAlign** in the *AlignLevel* port and press the *DoIt* button.

If you want to follow the alignment process,

- click the *show alignment* option before pressing the *DoIt* button.

Compare your result to the online demo.

1.1.6 Editing of molecules

An essential tool for manipulating the amiraMol molecular data structure is the *Molecule Editor*. It allows adding new bonds or changing the overall topology of the molecule as well as manipulating Cartesian or internal coordinates.

For each of these tasks, there is an example in the following section to give you an easy “learn by doing” introduction to the available features. Each action performed with the editor affects all currently selected atoms. Thus, it is necessary for you to be familiar with the functions of the *SelectionBrowser* beforehand (see section 1.1.2 on selection, labeling, and masking).

- Load the file `1HVRm.pdb` from the directory `data/molecules/pdb/` and connect a *MoleculeView* module to it.
- Select the *balls and sticks* representation for the *MoleculeView*.

The data structure loaded is an enzyme of the HIV virus in complex with the inhibitor XK2.

1.1.6.1 Invoking the Molecule Editor

To start the editor

- select the `1HVRm.pdb` object in the object pool and activate the green pencil button  in its user interface.

Now both the selection browser and the molecule editor interface will be visible.

1.1.6.2 Adding Bonds to a Part of a Molecule

The inhibitor XK2 is currently without bonds. To add bonds, carry out the following steps:

- Select the residue with the name *XK2* in the selection browser (it is at the end of the list).
- Switch to the *Tools* tab in the molecule editor and press the *bond length connect* button.

Bond length connect will create new bonds in the selected part of the molecule by comparing the distances between the atoms with a bond length table. The result should now look like Figure 1.4.

1.1.6.3 Splitting a Part of the Molecule

We now want to split the inhibitor from the protein.

- If it is not currently selected, reselect the *XK2* residue in the selection browser.
- Press the *split selected* button on the *Tools* tab of the molecule editor.

You will notice that the atoms of the inhibitor are no longer displayed by the *MoleculeView* and that a new object, *1HVRm2.pdb*, has appeared in the object pool. This object contains the previously selected atoms of the ligand.

1.1.6.4 Adding another molecule

We now want to manipulate the protein further by adding a water molecule to its active site.

- Load the file `h2o.pdb` from the directory `data/molecules/pdb/`.
- Go to the molecule editor and press the *add other* button.

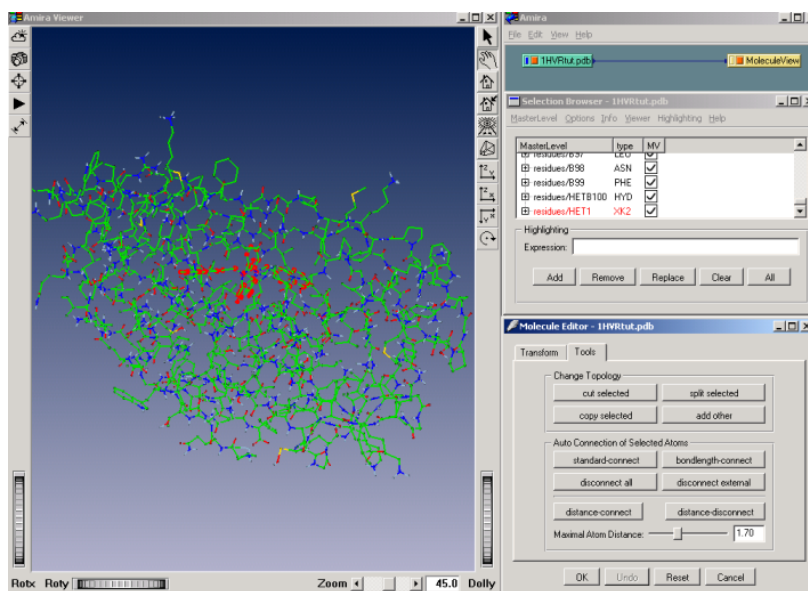


Figure 1.4: Adding bonds to the ligand

- In the window that has opened, all other molecules in the object pool will be shown. Select the *h2o.pdb* molecule and press OK.

The atoms of the water molecule have now been copied to the *1HVRm.pdb* object.

1.1.6.5 Moving Parts of the Molecule

To concentrate our view on the region of interest we will reduce the molecular view to amino acids A25 and B25 between which the water molecule should be placed.

- Type `r/name=?25` OR `r/type=H2O` into the selection browser and press the *add* button.
- Now move your mouse on the *MV* heading, press the right mouse button, and activate the *Replace* option.

With only the residues A25 and B25 and the water molecule displayed, all that is left to do is drag the water molecule to its correct location.

- Select the water molecule in the selection browser (the residue at the end of the list).
- Switch to the *Transform* tab of the molecule editor.

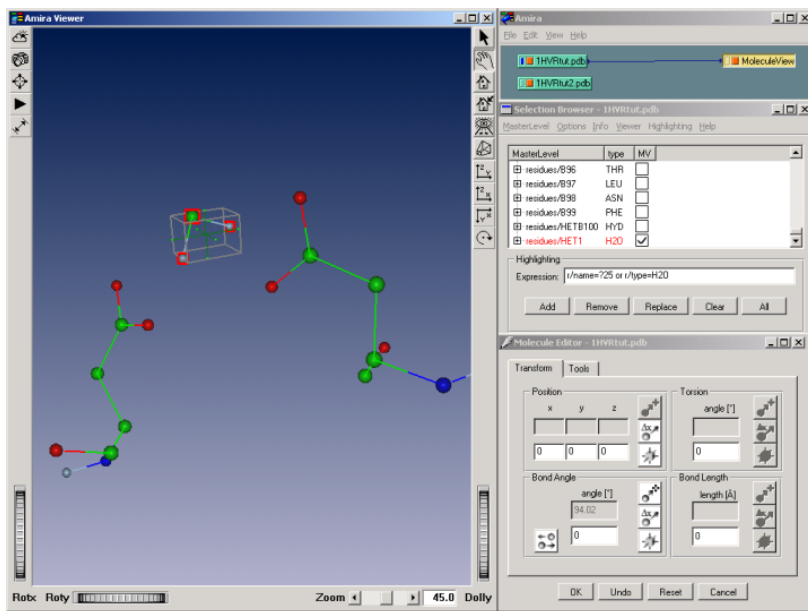


Figure 1.5: Moving the water molecule to the desired location

- Show the position of the transform dragger by pressing the  button in the *Position* section of the *Transform* tab.
- Left-click on the dragger and hold the mouse button down. You can now translate the dragger in different planes depending on the side you clicked on. Move the water molecule between the two amino acids as shown in Figure 1.5.
- To reorientate the water molecule, click on the green knobs of the dragger. They allow the dragger to rotate in different planes.

Repeat the steps described above until you are satisfied with the result.

1.1.7 Molecular Interfaces

This tool is particularly interesting to visualize contact areas between parts of a single molecule, or between different molecules, e.g. enzymes and their ligands. In this example we will consider the second case.

- Please load the file `t1gg.mol` from the directory `data/molecules/r_nuclease`.

1.1.7.1 Defining groups

In this section we will create a new level, called *interface*, for the molecule and define two groups of the level, *substrate* and *nuclease*, respectively.

- Select the `t1gg.mol` icon in the object pool by clicking on it.
- Now click in the console window to activate it, and press the TAB key.

The name of the selected icon, `t1gg.mol`, should appear in the console window. If it does not,

- please type `t1gg.mol`.
- On the same line, type `defregexp interface/substrate atoms/1461-1527`

The following text should now be written in the console window

```
t1gg.mol defregexp interface/substrate atoms/1461-1527
```

- Press the ENTER key.

You have now defined the group *substrate* in the level *interface*. The group consists of atoms 1461-1527. We will now define the group *nuclease*, containing atoms 1-1460.

- Press the TAB key again, then type `defregexp interface/nuclease atoms/1-1460`

If you now list all levels by typing

- `t1gg.mol list`

you will see an *interface* level containing two groups.

- Type `t1gg.mol list interface`

and all groups of the *interface* level will be printed in the console window.

1.1.7.2 Computing a Label Field

We will now create a label field containing two labels, *substrate* and *nuclease*, which will then be used for the computation of the interface surface. You will learn how to use the *CompMolInterface* module.

- Right-click the icon `t1gg.mol` and select *CompMolInterface* from the *Compute* submenu.
- Select the icon labeled `CompMolInterface`.
- Choose *interface* from the *Level* menu of the *MolecularInterface*.
- Press the *Generate Grid* button.

The result is an `Interface` object of type *UniformScalarField*.

1.1.7.3 Computing the Interface Surface with the SurfaceGen Module

With the *SurfaceGen* module we will now create an interface surface from the label field.

- Right-click the `Interface` icon and select the *SurfaceGen* entry from the *Compute* submenu.
- Press the *Triangulate* button of the *Action* port of the *SurfaceGen* module in the working area.

A green icon labeled `Interface . surf` has appeared in the object pool.

1.1.7.4 Visualizing the interface

Finally, we will view the computed interface in the viewer window via the *SurfaceView* module.

- Right-click on the `Interface . surf` icon and select *SurfaceView*.

The user interface of the *SurfaceView* module should now be visible in the working area.

- Choose *twisted* option from the *Colors* port.
- Select the *SubUnit2* entry from the first menu of the *Materials* port.
- Press the *Clear* button of the *Buffer* port followed by the *Add* button of the same port.

You should now see a gray surface in the viewer with some colored patches. The gray surface wraps the nuclease, the colored patches depict the interface between the nuclease and the substrate.

What you see should be similar to the Nuclease demo on the demo pages.

1.1.8 Measurement

In this tutorial you will learn how to measure distances and angles between atoms in a molecule. For this tutorial it is necessary for you to have already done the tutorial 1.1.1.

- Load any molecule from the directory `data/molecules/*` and connect a *MoleculeView* to it.
- Select the *MoleculeView* by clicking on its icon in the object pool.
- Select *balls and sticks Mode* and *fast Quality*.
- Next, right-click on the `MoleculeView` icon, and select *Measurement* from the pop-up menu.

The user interface of the *Measurement* tool should now be visible in the working area. An *Info* port tells you that you need to select 2, 3, or 4 atoms. In order to do so, switch to interaction mode by

- pressing the ESC key or by pressing the arrow button in the upper right corner of the viewer window.

You can now select single atoms in the viewer by clicking on them. If you click on one atom, the previously selected atom will be deselected. By using the `Ctrl` key, you can select more than one atom.

- Select 2, 3, or 4 atoms and observe what is being displaying in the user interface of the *Measurement* module.

The `Ctrl` key is also used to deselect atoms.

1.2 Molecular Data Structures

Several molecular file formats including, for example, PDB, Tripos, UniChem, and CHARMM, can be read and written by *amiraMol*. The information read from the data files will be stored in different types of objects: *Molecule*, *MolTrajectory*, and *MolTrajectoryBundle*.

Molecules. Single molecular configurations are represented as data objects of type *Molecule*. This kind of object can either be created by loading a file containing a single molecular structure or by attaching it to an object of type *MolTrajectory*, in which case it will act as a projector, extracting one of the trajectory's time steps.

Trajectories. The *MolTrajectory* data structure represents a series of molecular configurations of the same molecule. Again two cases are possible. Either the trajectory is loaded directly from a file. Or it can be attached to an object of type *MolTrajectoryBundle*, extracting one of the trajectories contained in that bundle.

Bundles of trajectories. A trajectory bundle reflects the case of a file containing more than one molecular trajectory. If such a file is loaded into *amira*, an object of type *MolTrajectoryBundle* will be created. The single trajectories can be accessed by attaching a *MolTrajectory* object.

1.2.1 Internal Structure of Molecules

An object of data type *Molecule* contains information about the structure of a molecule and the atomic coordinates of one of its configurations. The mandatory part of structural information concerns the number and types of all atoms contained in the molecule. All the topological information is organized in levels which are cliques of groups. Each group contains other groups or atoms. The simplest example is the level *bonds*, which consists of groups of two atoms.

Some levels can build hierarchies. For example, a residue consists of a number of atoms, and a couple of residues may form a secondary structure. These levels and groups can be defined by file readers or interactively via Tcl commands.

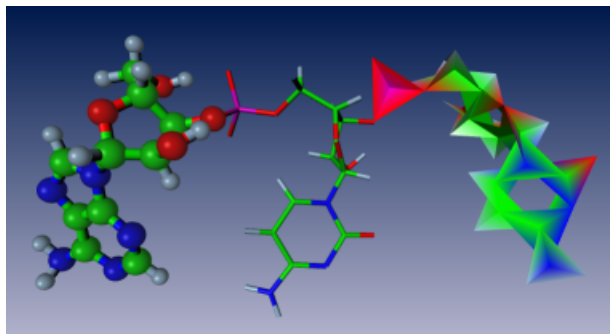


Figure 1.6: *MoleculeView* (left and middle) and *BondAngleView* (right) display the molecule simultaneously.

1.3 Displaying Molecules

Molecules can be visualized with the modules *MoleculeView*, *BondAngleView*, or *SecStructureView*. A sample output of the first two modules is shown in Figure 1.6.

In addition, there are two modules for generating molecular surfaces. The *CompMolInterface* module can be used to generate intramolecular interfaces, such as between single atoms or residues. Also, the interface between two molecules can be computed. The *CompMolSurface* module enables you to generate the *solvent accessible*, *solvent excluded*, and *van der Waals surface* of a molecule.

1.3.1 Coloring Molecules

The atom-oriented modules *MoleculeView*, *BondAngleView*, *ConfigurationDensity*, and *MolSurfaceView* allow a coloring on a per-atom basis, i.e., each atom is assigned a color. Balls in the *MoleculeView* will have the corresponding colors. Sticks will be split into halves colored according to their attached atoms' colors. In the *BondAngleView*, atom colors are assigned to the vertices and color is interpolated over the triangles.

To determine the coloring you can choose a level, e.g., atoms, residues, secondary structures, or chains. Furthermore, you can choose one of the level's attributes. The coloring will then be done according to the group's attributes such that all atoms of the same group will have the same color.

Color



A molecule may be colored according to various schemes. For example, each atom of a specific type may have the same color. Another possibility is to give all atoms belonging to a certain group the same color. The first menu of the *Color* port specifies the group level, i.e., atoms, residues, secondary structures, or chains. The second menu allows you to decide which attribute of the specified level should be considered to color its groups.

If you press the *Legend* button, a separate window will be opened, which displays a legend of the coloring, i.e. a table associating colors with attribute values according to the current coloring. Clicking on the text to the right of a color will select all atoms with this color.

Continuous CM



This colormap is used to map values of float attributes to colors. For optimal mapping, the colormap range must be set correctly. By default, the range is set to the minimum and maximum values that occur using the chosen color scheme, if the button in front of the first text field displays a capital “L”. The “L” means that the colormap uses a *local* range which allows you to modify the range without affecting the range of the same colormap used in other modules. By pressing the button you can toggle between *local* and *global* range.

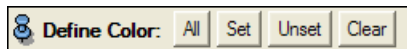
The default colormap has a constant color over the whole range. By leaving it constant you give all atoms the same color, which may be useful if you want to compare different molecules. For more information, see the section on *colormaps*.

Discrete CM



This colormap is used to map values of discrete attributes, like integers and strings, to colors.

Define Color



This port can be used to overwrite the standard colors of the selected color scheme. With the *All* button you can set a new color for all atoms. The *Clear* button unsets the userdefined colors for all atoms. The *Set* and *Unset* buttons operate on the set of highlighted atoms. These buttons can be used to set and unset the colors of those atoms, respectively.

1.3.2 Selecting and Filtering atoms

Various tasks require the selection of atoms in a molecule. The most prominent are the alignment of molecules and the selective display of molecule parts. *amira* offers two selection methods. You can select atoms visually via any of the viewing modules. Alternatively, selection can be done via the molecule’s *selection browser*, which can be opened by pressing the *Show* button of the molecule.

1.3.2.1 Selection of Atoms with a Viewing Module

The selection of atoms with a viewing module can be done in two ways. The first way is by clicking on certain displayed parts of the molecule. In general, this will highlight the selected parts. In most viewing modules the number of atoms that will be affected by the click depends on the selected group level of the *color port*. For example, if the atoms in the viewing module are colored according to the

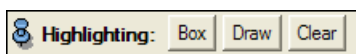
residue level, all atoms belonging to the same residue as the picked atom will get selected. If you pick another part, the previously highlighted atoms will be unhighlighted and the newly selected atoms are highlighted.

Ctrl-clicking a part of the molecule leaves the existing selection state of all atoms unchanged except for the newly selected atoms. If the selected atoms had been selected before, they will get deselected, otherwise they will get selected. On the one hand this allows you to add atoms to the selection, on the other hand it also allows you to deselect atoms.

If you Shift-click an atom, by default, the complete atom's residue will be selected. You may change the default behavior in the *Preferences* dialog of the *Edit* menu.

The Shift key can also be used in conjunction with the Ctrl key.

Highlighting



The second way to select parts of the molecule is by using the functionality of the port described below.

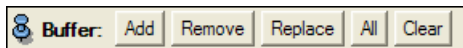
- If you press the *Box* button, the molecule's bounding box will be displayed. All atoms contained in the box will be highlighted. You can change the size of the box in *interaction mode* by clicking on the highlighted handles (usually in a light green) of the box. Keep the mouse button pressed and drag in the desired direction. Notice that the box will not exceed the molecule's bounding box. You can also move the box within the bounding box by clicking on one of the box's sides. Pressing the *box* button a second time hides the box.
- After pressing the *Draw* button, you can draw a line in the viewer window which selects all atoms that are inside the line. Pressing the Ctrl button while drawing inverts the selection. With the Shift button you can remove atoms from the selection.
- The *Clear* button deselects all atoms and hides the box if it is visible.

1.3.2.2 Filtering atoms

Filtering atoms gives us the ability to hide parts of the molecule for a certain module. All viewing modules and some computational modules, such as the *CompMolSurface* module, contain a *filter* that keeps track of all atoms of a molecule which are currently *in use*. The term *in use* means that the module will only see the *in use* atoms and regard all other atoms as non-existing. Thus, a viewing module will only display the *in use* atoms, and the computational module *CompMolSurface* will compute the molecular surface ignoring the *not in use* atoms. Each filter will register itself at the *molecule's selection browser*, so that you can easily determine the *in use* atoms of a module. For more information about how to do this, see the description of the *browser*.

In most modules, the filter's *Buffer* port will be visible, which adds to the convenience. The functionality of the *Buffer* port is described below.

Buffer



Pressing *Add* will append the selected atoms to the *in use* atoms. The *Remove* button will delete the selected atoms from the *in use* list. *Replace* will first clear the *in use* list and then add the selected atoms to it. Finally, the *All* and *Clear* buttons are shortcuts to delete or add all atoms from or to the *in use* list, respectively.

1.4 Aligning Molecules

To compare the structures of several molecules it is necessary to bring them into a suitable geometric arrangement relative to each other, because their relative positions in the common coordinate system are generally meaningless. By arranging the molecules in various different ways, you can investigate various aspects.

1.4.1 Alignment of Trajectories

amira offers a reusable component for the alignment of molecules. It appears in several modules, e.g. *Molecule*, *ConfigurationDensity* and *MeanMolecule*. Here, we will give an overview of the available functionality.

Two connection ports are supplied:

AlignMaster [optional]

To compute an alignment relative to a reference molecule, this port must be connected to the reference.

PrecomputedAlignment [optional]

This port can be connected to an alignment precomputed by using the module *PrecomputeAlignment*.

The control of the alignment is done via three ports, as described below:

Transformation



This port allows you to specify how the molecule should be aligned. The possible options are:

none: Atom coordinates are left as they are. Relative orientations and positions are displayed as specified in the data.

center of gravity: For every time step of a trajectory, the center of gravity is computed and positioned to the coordinate center by applying a translation.

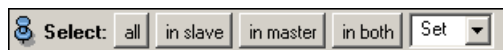
align to master: This mode requires a master molecule to be connected to *AlignMaster* port which is used as a reference to which other molecules, the *slaves*, are aligned. We consider correspondences between atoms from the reference molecule and atoms from the molecule to be aligned.

The alignment is done by minimizing the sum of squared distances between all corresponding atoms.

precomputed: This option is only enabled if the *PrecomputedAlignment* connection port is connected to an alignment that has been previously computed. If *precomputed* is chosen, the transformation will be taken directly from this alignment.

external: This option indicates that the molecule's global transform has been explicitly set, e.g. via the Tcl command *setTransform* or by using the *Transform Editor*.

Select



This port becomes important if *align to master* is selected. It gives control over the atoms in the slave and the master that are used for the alignment. In the general case master and slave molecules are different, i.e. their numbers of atoms differ, and an alignment requires an explicit specification of pairs of atoms. This can be done by highlighting atoms in both molecules via the *MoleculeView* module and then pressing the *in both* button. Depending on whether the option menu is set to *Set* or *Add*, the highlighted atoms will either replace an existing list of selected atoms or otherwise will be appended to it. The matching between atoms in the two molecules is defined by their order of selection.

The buttons *all*, *in slave*, and *in master* offer shortcuts for the frequent case of slave and master having the same number of atoms including the assumption that their order defines a natural matching. The easiest way is to use all atoms by pressing *all*. Selection of a subset can be done by highlighting in either the slave or the master and then pressing *in slave*, or *in master*, which will use the selected atoms in the respective molecule for the other molecule, too.

Selection



If *align to master* is selected, this port displays the existing state of selection that is used to compute the alignment. Possible forms are:

empty: No atoms are selected. At least three are necessary for an alignment.

all atoms: Master and slave have the same number of atoms and all are used for the alignment.

0, 5, 6: A list of indices implies that master and slave have the same number of atoms and identical subsets have been selected.

0 → 7, 5 → 6, 6 → 5: A list of index pairs (master index → slave index) indicates a matching resulting from individual selections in master and slave.

1.4.2 Mean Distance Alignment

Apart from the above mentioned alignment procedure, there exists a module that allows you to align two molecules by using the *mean distance* criterion instead of the *mean squared distance*. See *AlignMolecules* for more information.

1.4.3 Sequence alignment

amira allows you to *align the sequences* of two molecules. Three algorithms are available for use depending on the kind of data and your needs: local, semi-global, and global alignment. The algorithms work both for proteins and ribonucleic acids, whereas t-RNA molecules are currently not supported because they contain modified bases. This module can be very helpful when used in conjunction with the *AlignMolecules* module.

1.5 Visualizing Molecular Trajectories and Metastable Conformations

A *MolTrajectory* can be visualized via animation of single time steps by attaching a *Molecule* module to the *MolTrajectory* and then attaching a *MoleculeView* or *BondAngleView* to the *Molecule*. The animation is controlled via the *Molecule*'s *Time* port.

For *MolTrajectories* that represent metastable conformations, the modules *MeanMolecule* and *ConfigurationDensity* can be used. The *MeanMolecule* module aligns all steps of a trajectory and computes a mean molecule by averaging every atomic coordinate over all time steps. Using the *ClosestTimeStep* module, the member of the trajectory that is most similar to the mean molecule can be found, which can be used as a representative of the metastable conformation.

The *ConfigurationDensity* module gives an impression of the fuzziness of the conformation by computing a probability density for the positions of atoms and bonds within a molecular trajectory. This density can then be visualized with the *Isosurface* or *Voltex* module.

1.6 Atom Expressions

1.6.1 Overview

Atom expressions are a query language to find and select atoms of certain properties in the molecule for further action. The most important application of atom expressions in *amira* is the highlighting section of the *Selection Browser*.

In *amira*, a molecule is separated into groups of different levels. Each group contains a set of attributes. (More details of this concept can be found in the description of the *Attribute Editor*). Atom expressions are a simple form of a relational query language which accesses these attributes.

The most simple form of an atom expression is an *atom specifier*. This is a literal defining a level, one of its attributes, and a condition for this attribute. For example, `atoms/atomic_number=8` defines all atoms whose atomic number attribute equals 8 (i.e., all oxygens).

Such atom specifiers can be combined with logical operators like **AND**, **OR** and **NOT**. For example, `atoms/atomic_number=8 AND NOT atoms/charge=0` will select all charged oxygen atoms.

There are additional operators like **WITHIN**, **BONDED** and **GROUP** which apply certain conditions to coordinates or bond structure. These are explained in section on *operators*.

1.6.2 Grammar

All possible syntaxes of atom expressions are shown in the following grammar. The different literals and operators are further explained in the following sections.

```

atomExpr →      ( atomExpr )
                | NOT atomExpr
                | atomExpr AND atomExpr
                | atomExpr OR atomExpr
                | WITHIN (atomExpr,float)
                | BONDED (atomExpr[,int])
                | GROUP (groupParts)
                | CS
                | atomSpecifier
atomSpecifier →  hierName/[attrName=]ID
groupParts →     groupPart
                | groupPart,groupParts
groupPart →      groupSpecifier
                | groupSpecifier [bondOrderSymbolChar] groupSpecifier
groupSpecifier → [!] elementSymbol index

```

1.6.3 Literals

As mentioned in the overview, the simplest form of an atom expression is an atom specifier. An atom specifier consists of three literals: *hierName*, the optional *attrName*, and *ID*.

hierName stands for a name of a hierarchy level (e.g., `residues`). The following abbreviations can be used for the most common levels:

a=atoms, r=residues, b=bonds, s=secondary_structure

attrName is optional and specifies the name of an attribute (e.g., `temperature`, `occupancy`, `type`, ...) of the given level. If it is omitted, the ID is assumed to specify the attribute name or index as shown by the list command. If an attribute name is given, the ID is assumed to stand for values of the attribute.

To see which hierarchy levels and respective attributes are defined for a given molecule, take a look at the *Color port* which is used in several modules. The right pull-down menu will show all available attributes for the level chosen in the left pull-down menu.

ID specifies an identifier of a member of the given hierarchy and attribute. It can be the name/index of the group (e.g., L112 for residue 112 on chain L) or the type (e.g., ASP for all aspartate residues). Wildcards such as * and ? may be used, or ranges (e.g., L112-L115) may be given. For the atom level there is a special case to be considered. Molecules in *amira* usually contain an atomic number attribute instead of an element symbol attribute. To select atoms via their element symbol you can simply type *a/element*. Thus the atom specifier for all oxygen atoms *a/O* is equivalent to the atom specifier *a/atomic_number=8*.

Another atom expression form involving several literals is the *groupParts* expression which is used with the **GROUP** operator. Operators will be explained in the next section.

1.6.4 Operators

Logical Operators

Several *atomSpecifier* combinations can be used in one expression by linking them logically via the operators **AND**, **OR**, and **NOT** (&, |, and !). Priorities can be specified using usual parentheses (and).

WITHIN(*atomExpr*, *float*)

This operator selects all atoms which are nearer than *float* Å to any of the atoms specified by *atomExpr*.

BONDED(*atomExpr* [, *int*])

With this operator, all atoms that are recursively connected to any atoms specified by *atomExpr* will be chosen. You can optionally specify an integer value defining the maximal bond steps. If this is omitted, there will be no limit.

CS

CS specifies all currently selected (highlighted) atoms.

GROUP(*groupParts*)

The **GROUP** operator is a powerful tool to find functional groups by searching for a certain atom and bond pattern in the molecular graph. To define a graph as a search pattern (*groupParts*), you must divide it into linear pieces of sequential atoms (a *groupPart*). Each atom must be defined by its element symbol and an index which distinguishes it from other atoms with the same symbol but other indices (*groupSpecifier*). Thus, C1C2C3 would be a *groupPart* consisting of three different *groupSpecifiers* representing a chain of three carbons. This group part can now be combined with another group part by using one of its *groupSpecifiers* as branch point (e.g., C2O1H1 for a hydroxyl group branching from the second carbon of the chain). At the beginning of each group specifier, there can be an optional '!'. If it is given, the groupSpecifier is only used for matching, but the corresponding atoms will not be selected. (Thus !C1O1H1 would find the hydroxyl groups without selecting the flanking carbon, which must be given, however, to avoid selecting structures, such as OH groups in H2O). If the atomSpecifier in question appears several times (to define branch points), it is sufficient to mark it with the exclamation mark once.

Consecutive *atomSpecifiers* in a *groupPart* are usually not divided from each other. This means that there must be a bond of any type between the consecutive atoms. If you want to define the bond order further, you can give an optional *bondOrderSymbol*. ('-' for a single, '=' for a double, '#' for a triple and '~' for an aromatic bond).

Take a carboxyl group as an example. To define the group pattern, the central carbon atom (C2) needs to be connected to three atoms: two oxygens (O1, O2), and one other carbon (C1). This can be done in the following way: `GROUP ( ! C1C2O1, C2O2H1 )`. Thus, the hydroxyl group (O2H1) is given as a branch of the CCO chain. There are, of course, several other ways to split this branch into linear pieces which you can easily find yourself. If your molecule contains the bond type attribute, you can also make use of the double bond. Thus the expression becomes `GROUP ( ! C1-C2=O1, C2-O2-H1 )`.

Notice: The keywords do not need to be in capital letters. Lower case letters, even a combination of lower and upper case letters, works as well.

1.6.5 Shortcuts

amira also provides pre-defined shortcuts that have been assembled using the previously mentioned syntactical elements. The shortcuts can be found in your local *amira* directory in `share/molecules/atomExpr.cfg`, and can be edited and supplemented.

The standard aliases included in the current *amira* release are:

acidic	acidic amino acids
acyclic	acyclic amino acids
aliphatic	aliphatic amino acids
alkali	atoms which are alkali metals
alkaliearth	atoms which are alkali earth metals
amino	amino acids
aromatic	aromatic amino acids
at	adenine or thymine
backbone	atoms of protein or DNA/RNA
basic	basic amino acids
buried	amino acids which are usually found inside the protein
cg	cytosine or guanine
charged	charged amino acids
cyclic	cyclic amino acids
h2o	water molecules
helix	helices
halfmetallic	atoms which have halfmetallic properties
halogens	halogenic atoms
hetero	heterogenic atoms
hydrophobic	hydrophobic amino acids
ions	charged heterogenic atoms
metallic	atoms which have metallic properties
neutral	neutral amino acids
noblegas	atoms which have noblegas properties
nonmetallic	atoms which have nonmetallic properties
nucleic	nucleic acids
nucleicbackbone	backbone atoms of RNA/DNA
polar	polar amino acids
proteinbackbone	backbone of a polypeptide
purine	adenine or guanine
pyrimidine	cytosine or thymine
sheet	sheets
sidechain	atoms of the proteine/RNA/DNA which do not belong to the backbone
site	
surface	amino acids which tend to be found on the surface of molecules
turn	

1.6.6 Further Examples

- `atoms/5-8`
all atoms whose index is in the range 5 to 8, inclusive.
- `s/type=helix AND NOT (a/C OR a/N)`
all atoms which belong to helices, except C and N atoms.
- `r/type=A*`
all atoms which belong to residues whose type name begins with the letter A.
- `BONDED(a/4 OR a/100, 6)`
all atoms which are connected via at most 6 steps to the two specified atoms
- `WITHIN(r/A11, 3.1) AND C`
all carbon atoms which are not away further than 3.1 angstroms from atoms of residue 11 on chain A
- `GROUP(C1C2C3C4C5C6C1)`
all cycles consisting of 6 carbons (e.g., cyclohexane).
- `acidic AND helix`
all atoms of acidic amino acids which belong to helices

1.7 Molecule Selection Browser

The selection browser is a handy tool for investigating the topology of a molecule. It consists of three main parts: the menu bar, a scroll window, and a highlighting section. In the scroll window all items of a chosen level will be displayed. Most of the pop-up menus in the menu bar allow you to control what is being shown in the scroll window. Finally, the highlighting section provides further functionality for the selection of items in the scroll window.

The selection browser enables you to quickly select parts of the molecule, which will then be highlighted in the browser as well as in the viewer by the display modules connected to the molecule. The highlighting allows a correlation between the topology and the molecule's three-dimensional structure to be established. Highlighted atoms can be used for masking. Thus, you can concentrate on specific regions of interest. Furthermore, masking allows different representations to be combined quickly and easily.

1.7.1 The scroll window

1.7.1.1 Displayed information

In this window all items of a chosen level are displayed hierarchically in the first column. An item might consist of a single atom or a group of items.

If there is additional information for an item, it will be displayed in the following columns. If the information is not available, the entry will be empty. Additional information typically includes the item's type and the membership of the item in some level. Note that only the direct membership will be considered, i.e., if an item is not a member of some level, but a member of some group, that is an item of the level, the membership information will not be displayed. An exception is being made if *atoms* is chosen as the master level. In this case, *amira* displays all groups of the information level the atom belongs to.

If there are any display modules or other modules containing a filter connected to the molecule (see *introductory section on the display of molecules*), there will be more columns to the right of the info columns - one per module. These columns, which we will call *module columns*, are used to display whether an item is *in use* of a module. For each item a rectangle will be drawn in each *module column*. If the item is completely *in use* of a module, the rectangle contains a check mark. If it is partially *in use*, i.e. some but not all of its group items are *in use*, the check mark will be gray. There is one other symbol that might be displayed in the rectangle, a 'B'. The 'B' stands for backbone and indicates that the atom belongs to those atoms currently defined as the backbone. This will be described in the following subsections. The 'B' might be black or gray. Again, gray means the atom is not *in use* of the module. A black letter indicates that the atom is *in use*, but simply because it belongs to the backbone.

1.7.1.2 Item and information columns

If an item is a group, i.e., it contains items, a little icon will be placed to the left of the item name. You can click on the icon to open/close the item, i.e., to show or hide the contained items, respectively.

In order to select an item in the scroll window, you can click on the respective item, which will be highlighted immediately. To toggle the selection of an item, you must **Ctrl**-click the item. When **Shift**-clicking an item a whole range will be selected. You can also drag the mouse with the left mouse button pressed over a range of items which selects all of them. **Ctrl**-click and drag again toggles the selection. If an item is fully selected, it will be highlighted in red. If at least one but not all of its subitems are selected, *amira* considers the item to be partially selected and highlights it in green.

It is also possible to select items by type or other information in the information columns. If you click on the displayed information for an item, all items with the same information will also be selected.

1.7.1.3 Module columns

If there are any modules containing a filter connected to the molecule, such as *MoleculeView*, *Sec-StructureView*, *CompMolSurface*, or *MolSurfaceView* there will be module columns in addition to the master and information columns - one per display module. Each of these columns contains a check box for each item displayed. The check box contains information about whether an item is *in use* of a module or not. Before the different states of a check box are explained, we will make a short side trip. For convenience, all atoms are classified as belonging to the backbone or not. You can restrict your selection to only backbone atoms, which means that an atom will only be *used* by the module if it belongs to the backbone. Apart from that you may also want to *use* at least all backbone atoms plus

additional atoms. Therefore, internally there are two arrays, one containing all backbone atoms and one containing so-called *selected* atoms. Depending on the backbone mode, there are three choices:

- all atoms contained in either array will be *used*, or
- all atoms contained in both arrays will be *used*, or
- only those atoms contained in the *selected* array will be *used*.

Using those two arrays allows you to quickly change between backbone atoms and all others without losing the previous atom masking.

Now let's go back to the check box states. There are five different states of a check box, including one state with two different meanings.

- The box is empty: the complete item is not shown by the viewer.
- The box contains a black check mark: the complete item is shown.
- The box contains a gray check mark and the item is an atom: the atom is a member of the *selected* array, but is still not shown due to the backbone mode, i.e., it does not belong to the backbone.
- The box contains a gray check mark and the item is not an atom: the item is only partially shown in the viewer.
- The box contains a black 'B' (only for atoms): the atom is shown simply because it belongs to the backbone.
- The box contains a gray 'B' (only for atoms): the atom is not shown, but belongs to the backbone.

Clicking on a box toggles its state. You can also click on one item and then drag the mouse pointer over multiple items. All of these items will be set to the same state as the first item.

In addition to directly clicking on a box, there is another way to change the state of the check boxes. If you click on the header of a module column with the right mouse button, a pop-up menu will appear containing the following entries:

- Add: adds the currently highlighted atoms to the 'selected' array.
- Remove: removes the highlighted atoms from the selection.
- Replace: replaces selection by highlighted atoms.
- Clear: clears the 'selected' array.
- All: selects all atoms.
- Restrict to: this pop-up menu allows you to restrict the set of displayed atoms to some predefined set.
- Combine with: if one of the entries in this pop-up menu is selected, the chosen set will be the minimum set that is being displayed by the respective display module.
- Set: the 'selected' array is set to the chosen predefined set of atoms. Starting from this selection

you can add or remove parts of the molecule.

1.7.2 Highlighting section

This browser section consists of a text field and a button row. In the text field you can type *atom expressions*. The expressions you type will be saved in a history. Using the Up and Down keys you can go back to previous expressions.

1.7.2.1 Buttons

The buttons work in conjunction with the expression in the text field. When pressing any of the first three buttons, the atom expression will be evaluated, and the atoms corresponding to the expression will be *added to*, *removed from*, or *replace* the highlighting, respectively. The last two buttons clear the highlighting or highlight all items, respectively.

1.7.3 Menu bar

The menu bar contains a couple of pop-up menus which will be explained below.

1.7.3.1 MasterLevel

The *MasterLevel* menu contains all levels currently defined in the molecule's *topology*. All groups of the current master level will be displayed in the scroll window.

1.7.3.2 Options

The *Options* menu allows you to quickly open and collapse items, i.e., to show and hide the items contained in other items.

1.7.3.3 Info

With the help of the *Info* menu, you can display additional information columns, such as type, or the item's membership in a group of some level.

1.7.3.4 Modules

If you have many display modules connected to the molecule, the large number of columns might be confusing. If you are only working with one or a few of the display modules, you might want to hide the other columns. This can be done with the help of this menu.

1.7.3.5 Highlighting

By default, all display modules that allow highlighting will highlight all selected atoms. The *Highlighting* menu enables you to turn highlighting for each display module on and off.

Chapter 2

Alphabetic Index of Modules

2.1 AlignMolecules

This module enables you to align two arbitrary molecules to each other. However, the molecules need to fulfill one precondition: they both must have (at least) one level, the *align level*, with the same name and an equal number of groups. The algorithm works as follows: for each group of the first molecule all its atoms attract all atoms of its corresponding group of the other molecule, whereby the correspondence is established by the index of the group in the *align level*. One possibility to create such levels is by using the module *AlignSequences*.

The algorithm works iteratively. Starting from an arbitrary position, a force and a rotation are calculated at each step resulting from the attraction forces of the atoms. The algorithm stops if the transformation from one step to another is smaller than some epsilon. The alignment found is not necessarily the global minimum of the sum of distances between the corresponding atoms. However, it will always be a local minimum.

If the global minimum is prominent, it can be assumed that this minimum will be found from each starting position. In order to find out whether there exist other equally good local alignments, it is possible to compute alignments starting from 20 different positions. The different alignments will be stored in a list. You can then compare the alignments visually as well as by the distance between the two molecules.

Connections

MoleculeA [required]

First molecule to which the second molecule will be aligned.

MoleculeB [required]

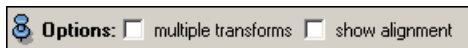
Second molecule to be aligned to the first one.

Slave [optional]

You can connect any data object to this port. The same transformation that is applied to the second molecule will be applied to this data object as well.

Ports

Options



- *multiple transforms*: If this toggle is not selected, a local minimum will be found starting from the current positions of the molecules. If the toggle is selected, molecule B will be translated such that the centers of the molecules according to the groups found in the *Align Level* are at the same position. Starting from this position, 20 rotations will be applied to molecule B, and these new positions will be used as starting points for the alignment.
- *show alignment*: If this option is set, you can observe the alignment process visually. Otherwise, only the final alignment will be shown.

Transforms



If you compute the alignment with the *multiple transforms* toggle switched on, this slider appears. Once the computation is finished, i.e., for all 20 starting positions at which the alignment was computed, you can step through all alignments that differ by an epsilon. The epsilon can be set by the command *setEpsilon*. Notice that if there is only one local alignment, it can be assumed that it is also the global one. However, there is no guarantee.

Align Level



Select the level which is used for alignment.

Action



Compute alignment and transform the second molecule.

Commands

getEpsilon

Prints the current epsilon that is used for comparison of the transformations.

setEpsilon <value>

Set epsilon to value, which should be smaller than 1.0. Check the current epsilon before setting a new one with the command *getEpsilon*.

2.2 AlignSequences

This module aligns the sequences of two molecules to each other. Proteins as well as nucleic acids can be aligned. Currently, it is not possible to align t-RNA sequences because they contain modified bases deviating from the standard data format. The module will be adjusted and extended in the near future. The output of the alignment is displayed in a separate window. Currently you may choose between *local*, *semiglobal*, and *global* alignment. The algorithms are based on the Waterman algorithms for pairwise sequence alignments using a *Blosum* weighting matrix for protein sequences and a *Transition/Transversion* weighting matrix for nucleic acids. Consider the following three cases to decide which algorithm to use:

- *local alignment*

The local alignment algorithm is the best algorithm to use if you want to match a short sequence, possibly a subsequence or motif of some molecule, against a long one. You can specify a motif (e.g., a promoter sequence) and search for it in the molecule of interest. Beware, the local alignment algorithm is not useful for aligning two long sequences to each other, e.g., two related proteins. You might end up with a very large number of equally evaluated sequences, fragmenting the molecules in many pieces. For alignment of two long sequences, you should rather use one of the next two algorithms instead.

- *global alignment with gap function*

In this case, you want to match two related long sequences to each other, e.g., two related enzymes. The algorithm matches the whole sequences penalizing long gaps proportionally less than many short gaps.

- *semiglobal alignment*

If you have no idea about the relationship of your two test molecules, it is best to use the semiglobal algorithm first because it will scan for the possibly best matching parts of your molecules without having to align the whole sequences as with the global algorithm.

Connections

MoleculeA [required]

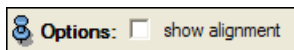
First molecule to be aligned.

MoleculeB [required]

Second molecule to be aligned.

Ports

Options



Toggle the button to show/hide the alignment. This toggle is sensitive if an alignment exists, i.e., if it has previously been computed.

Input

 **Input:** ☐ molecules A+B ☒ molA/motif ☐ molB/motif

Which sequences should be taken as input for the alignment. Three options exist:

- molecule A and B.
- molecule A and the specified motif (see below).
- molecule B and the motif.

Motif

 **Motif:**

You can specify a motif which will be searched for in a longer sequence. This port appears if one of the last two options is chosen in the *Input* port.

Align Type

 **Align Type:**

You can align proteins as well as ribonucleic acids. Specify the type of molecule in the first pop-up menu. In the second menu you can select one of three algorithms as explained above: *local*, *semiglobal*, or *global* alignment.

Align Sequences

 **Do it:** ☒ Align Sequences

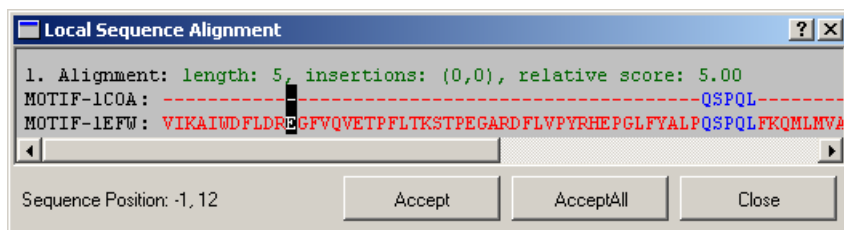
Compute alignment and display it.

Commands

`setLimit <value>`

With this command you restrict the number of alignments computed from one start position of the alignment matrix. The default is 10.

AlignView



For each alignment, its number and additional information are displayed in the first line. Currently the information includes the length of the alignment, the number of insertions for both sequences, and the relative score, i.e., the score divided by the length of the alignment. The latter might be interesting when comparing alignments between homologous sequences.

The *AlignView* displays corresponding residues in blue. To provide an idea about where in the sequences similarities have been found, the rest of the sequence is displayed as well (in red). To find out where in the sequence you are, just click at the position within the sequence you are interested in. Immediately, the positions for both sequences will be shown in the lower-left corner of the window. The clicked position will be marked by a black rectangle.

Apart from just showing alignments, the *AlignSequences* module allows you to add alignments as levels to the aligned molecules. However, since there might be a large number of alignments and you might only be interested in a few, the window allows you to select alignments. In order to select a single alignment, left-click on the row with the label *i. Alignment*. The label will turn black. If you want to select more than one alignment, use the `Ctrl` key. An alignment can be unselected by just clicking on it again. All currently selected alignments are highlighted by a black frame around the label. Those alignments can be written to the molecules by pressing the *Accept* button. If all alignments are desired, press the *AcceptAll* button.

When an alignment is written to a molecule as a new level, all letters of the sequence that are not matched, i.e., that correspond to a blank, are left out. Both molecules will have a new level with exactly the same name and length. Thus, those levels can be used by the *AlignMolecules* module.

2.3 BondAngleView

The *BondAngleView* offers an alternative to the *MoleculeView*. This viewing module, however, does not display atoms and bonds but, as the name suggests, bond angles, i.e. for every three atoms connected by two bonds a triangle will be shown. The vertex colors are linearly interpolated across the triangle. Apart from coloring the molecule as is done in the *MoleculeView*, the bond angles can be colored according to a specified color field, e.g., the electrostatic potential, and a colormap. Here, in addition to interpolating the colors, texture mapping can be applied which interpolates the texture coordinates.

Connections

Data [required]

Molecule to be visualized.

ColorField [optional]

Scalar field that can be displayed on the bond angle by pseudo-coloring.

Colormap [optional]

Color map needed in conjunction with the color field.

Continuous CM [optional]

The colormap that is used to map values of float attributes to colors.

Discrete CM [optional]

The colormap that is used to map values of discrete attributes, like integers and strings, to colors.

Ports

Draw Style



This port is inherited from the *ViewBase* module.

Colormap



This port becomes visible only if a scalar field has been connected to the color field port.

Transparency



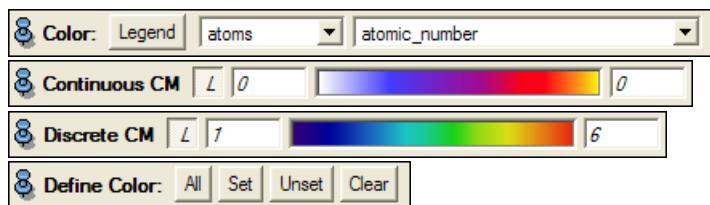
If the draw style is set to *transparent*, this port lets you configure the degree of transparency.

Color Mode



Allows you to switch between *normal* coloring mode and coloring according to a scalar field.

Coloring



For an explanation of the four ports above, see the description in the general section on *display of molecules*.

2.4 BondCalculation

The module calculates bonds in a given molecular structure by analyzing the distances between the individual atoms of the structure. A maximal distance value will be used to determine whether two atoms are connected or not.

This module is especially useful when being used with a molecule that is derived from a trajectory. Each newly displayed time step will cause the bonds to be recalculated. A similar module for bond calculation is available via the *molecule editor*. This editor module offers advanced functionality but cannot be used dynamically in the work flow of the object pool.

Connections

Data [required]

Molecule for which a new molecule with a complete bond structure should be computed.

Ports

Maximal atom distance



The value determines the maximal distance between two atoms (in Å) below which they are considered to be connected by a bond.

Action



Starts the computation of the new molecule.

2.5 ClosestTimeStep

For a given molecule, this module searches for the molecule within a molecular trajectory with the smallest mean squared distance. The reference molecule and the molecules in the trajectory must have the same number of atoms.

The module is especially helpful in conjunction with the module for the computation of the *mean molecule*. It allows you to find the time step in the trajectory that best approximates the mean molecule.

Connections

Data [required]

Molecular trajectory for which the mean molecule should be computed.

ReferenceMolecule [required]

The molecule for which the closest time step is searched.

Ports

Action



Start search for the closest time step in the trajectory. The index of the closest time step together with the distance will be printed in the *console window*.

2.6 CompMolInterface

This computational module provides a way to generate intramolecular and intermolecular interfaces. This is done by creating an object of type *LabelField*, which is a three-dimensional grid holding at each node a label that represents an area of the molecule, plus additional information. The usual way to visualize a label field in order to see molecular interfaces is to use the *SurfaceGen* module to compute a surface representing the interfaces. This surface can then be viewed with the *SurfaceView*.

Connections

PrimaryMolecule [required]

Molecule for which intramolecular interfaces should be computed. In case of the computation of intermolecular interfaces, this is one of the molecules for which the interfaces should be generated.

SecondaryMolecule [optional]

Second molecule for the computation of intermolecular interfaces.

Ports

Resolution



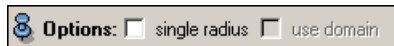
Resolution of the label field.

Level



This option menu allows you to choose the level of molecular structure for which intramolecular interfaces should be extracted. For example, you might want to compute atomic interfaces on a very low level, or you might just want to generate interfaces between functional groups on a higher level or even secondary structures. This port disappears if the *SecondaryMolecule* connection is used.

Options



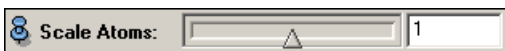
The *Options* toggle list enables you to select two options. By selecting *single radius* all atoms will be considered as having the same radius, which will be equal to the *Scale Atoms* value (see below). The second option, *use domain*, allows you to specify a range of atoms, or residues, depending on the specified level. All atoms belonging to this range will be assigned to one subunit, the rest to a second subunit. Currently only two regions can be defined this way.

Domain

A control interface for the 'Domain' option. It features a blue icon with a lock symbol on the left, followed by the text 'Domain:'. To the right of the text are two adjacent input fields, both containing the number '0'.

Visible only if the toggle *use domain* is selected in the *Options* port. In conjunction with the chosen level, this range specifies which atom will be assigned to *Subunit1*. All other atoms are assigned to *Subunit2*.

Scale Atoms

A control interface for the 'Scale Atoms' option. It features a blue icon with a lock symbol on the left, followed by the text 'Scale Atoms:'. To the right is a slider control with a triangular handle, and a numeric input field on the far right containing the value '1'.

Multiply all atomic radii by this value.

Enlarge Atoms

A control interface for the 'Enlarge Atoms' option. It features a blue icon with a lock symbol on the left, followed by the text 'Enlarge Atoms:'. To the right is a slider control with a triangular handle, and a numeric input field on the far right containing the value '0'.

Enlarge all atomic radii by this value.

Action

A control interface for the 'Action' option. It features a blue icon with a lock symbol on the left, followed by the text 'Action:'. To the right is a button with a red square icon and the text 'Generate Grid'.

Start computation of label field.

2.7 CompMolSurface

This module computes molecular surfaces. Three types of molecular surfaces can be generated: the *van der Waals (vdW) surface*, the *solvent accessible surface (sas)*, and the *solvent excluded surface (ses)*. The *vdW* surface encloses all van der Waals spheres of the molecule's atoms. The *sas* encloses all van der Waals spheres extended by some probe radius. Finally, the *ses* encloses the subspace which is not accessible to a probe sphere in the presence of the molecule represented by its van der Waals spheres.

The implemented algorithm allows the computation of the full molecular surface as well as of partial surfaces. If the partial surface option is selected, the surface will only be computed for all highlighted atoms (see, e.g., the description of the *Molecule Selection Browser*). This module will register itself at the Selection Browser, which allows you to restrict the computation of the molecular surface to a subset of its atoms. For example, if the molecule consists of several chains, you can compute the molecular surface of a single chain by deselecting the other chains.

Apart from the option to compute either the full or a partial surface, you can also choose between two algorithms that differ in the *quality* of the surface generated, and also in speed. If time does not matter, you should always use the default option, *correct*. However, if you are interested in the dynamic behavior of the molecular surface, time does matter and you might want to use the second algorithm. This algorithm works as follows. We start with an arbitrary atom contributing to the surface. For this we compute the atom's surface contribution. All atoms adjacent to this atom's surface are stored in a list. Next, we take the first atom from the list and deal with it in the same fashion as for the very first atom. Thus, in the case of a vdW surface or an *sas* we end up with the same surface as with the *correct* algorithm if the surface consists of one component. In case of the *ses* we run into problems if two components are further away from each other than the probe's diameter. Furthermore, the *faster* algorithm might not compute all cavities, whereas the *correct* algorithm will.

A maximum of two molecules may be connected to the module, thus enabling the surface computation of a complex consisting of two molecules. This can be interesting, for example, if you want to compute the surface before and after a docking.

As a result of the computation a new object, i.e., the molecular surface, will appear in the object pool. To visualize the surface, attach the *MolSurfaceView* module to it.

Connections

Molecule [optional]

The molecule for which the molecular surface should be computed.

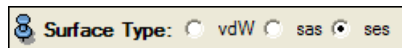
Molecule2 [optional]

If you want to compute the surface of a complex of two molecules, you must connect this port to a second molecule.

Either of those two ports needs to be connected to some molecule in order to compute a molecular surface.

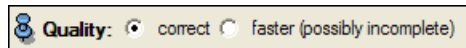
Ports

Surface Type



Choose between *van der Waals*, *solvent accessible*, and *solvent excluded* surface.

Quality



The default option computes the molecular surface correctly, i.e., with all, possibly disconnected, components and all enclosed cavities. The second algorithm is faster because it does not compute the molecular surface contribution for each atom, but only for surface atoms. If the surface is not connected, it might miss parts of the surface.

Probe Radius

A control panel for the Probe Radius. It features a blue icon of a person on the left, followed by the text "Probe Radius:". To the right is a horizontal slider with a triangle marker, and a numerical input field on the far right containing the value "1.4".

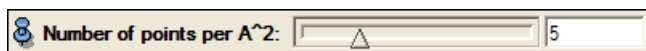
The radius of the solvent probe sphere. This port is only visible for the surface types *sas* and *ses*.

Edge length

A control panel for the Edge length. It features a blue icon of a person on the left, followed by the text "Edge length:". To the right is a horizontal slider with a triangle marker, and a numerical input field on the far right containing the value "0.480562".

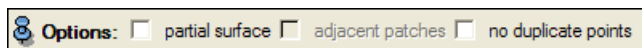
Choose the approximate edge length of the surface's triangles. This port corresponds to the number of points per $\text{\AA}^2$, and vice versa. Thus, if you modify the value of this port, the value of the other port will be changed too.

Number of points

A control panel for the Number of points. It features a blue icon of a person on the left, followed by the text "Number of points per A^2:". To the right is a horizontal slider with a triangle marker, and a numerical input field on the far right containing the value "5".

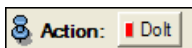
With this port you define the approximate number of points per $\text{\AA}^2$ and thereby the granularity of the surface.

Options

A control panel for the Options. It features a blue icon of a person on the left, followed by the text "Options:". To the right are three checkboxes: "partial surface" (checked), "adjacent patches" (unchecked), and "no duplicate points" (unchecked).

If the option *partial surface* is selected, only the surface of the currently highlighted atoms will be computed. The second option, which is only active if the first option is checked and the surface type *ses* is selected, initiates the computation of the highlighted atom's adjacent patches, i.e., toroidal and spherical concave patches.

Action

A control panel for the Action. It features a blue icon of a person on the left, followed by the text "Action:". To the right is a button with a red square icon and the text "DoIt".

Start computation.

2.8 ConfigurationDensity

This module enables you to compute a probability density for the positions of atoms and bonds within a molecular trajectory. The input is an object of type *MolTrajectory*. As output either a scalar field or a color field is generated. In order to visualize the computed density you can, for example, apply the volume rendering module, *Voltex*.

Since the time steps of the molecular trajectory can be arbitrarily rotated and translated, we must perform an alignment for each time step to fit it best to some chosen reference. This is done internally, but you must specify how the molecules should be aligned to each other. There are four options from which to choose. The first uses all atoms for alignment, which is the recommended option. In the

second you can select a few atoms (see below for more information). The third and fourth use *none* and *center of gravity* alignment, respectively.

For the representation of the molecule two kinds of geometric objects can be used: spheres and sticks. Here, spheres represent the positions of the atomic nuclei and the sticks the existing bonds within the molecule. You can compute the density for sticks, atoms, or sticks and atoms.

Connections

Data [required]

The molecular trajectory for which the density should be computed.

AlignMaster [optional]

The molecule to which each time step of the trajectory will be aligned.

PrecomputedAlignment [optional]

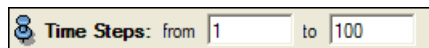
Instead of aligning all time steps of a trajectory to the *AlignMaster* molecule, you can use a pre-computed alignment to align the time steps.

Continuous and DiscreteColormap [optional]

These two colormaps are used for the color management of the computed volume, in case a color field should be generated.

Ports

Time Steps

A port widget for 'Time Steps' with a blue icon. It contains the text 'Time Steps: from' followed by a text box with '1', 'to' followed by a text box with '100'.

Specifies range of time steps for which the density should be computed.

Voxel Size

A port widget for 'Voxel Size' with a blue icon. It contains the text 'Voxel Size:' followed by a slider control and a text box with '0.1'.

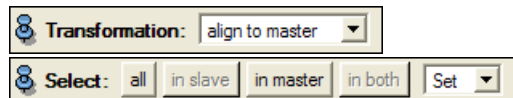
This value determines the size of a voxel, i.e., its height, width, and depth, of the field storing the probability density.

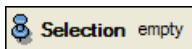
Grid Dimension

A port widget for 'Grid Dimension' with a blue icon. It contains the text 'Grid Dimension:'.

The dimension of the field grid resulting from the specified voxel size.

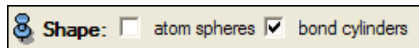
Alignment

A port widget for 'Alignment' with a blue icon. It contains two rows of controls. The first row has 'Transformation:' followed by a dropdown menu showing 'align to master'. The second row has 'Select:' followed by four buttons: 'all', 'in slave', 'in master', and 'in both', and a 'Set' button with a dropdown arrow.



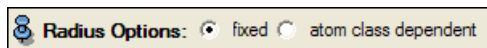
The three ports above specify how the molecules are aligned to each other. See the description in the general section on *alignment of molecules* for a detailed explanation.

Shape



Choose the geometry used to represent the molecule. If you have a highly varying trajectory, the recommended shape is *cylinders*.

Radius Options



Use a unique radius for all atom spheres (*fixed*) or take for each sphere the van der Waals radius scaled by the value of port *Atom Radius* (*atom class dependent*).

Atom Radius



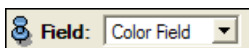
Radius of the atom spheres.

Bond Radius



Radius of cylinders representing bonds.

Field



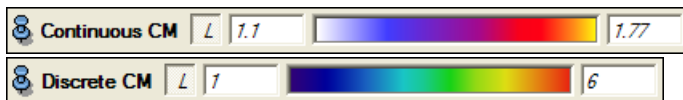
This menu contains two options, *Scalar Field* and *Color Field*. Choose the first if you only want the density. If you choose the second option, in addition to the density, the color information will be stored according to the color scheme (see *Atom Colors* port).

Atom Colors



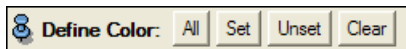
Determines how the field regions will be colored. For more information see the description of *Color* port in the section on displaying molecules.

Colormaps



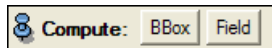
These two colormaps are used to color the atoms and sticks according to the selected color scheme in *Atom Colors*. See the description in the section on *display of molecules* for a detailed explanation.

Define Color



The port is described in the section on *display of molecules*.

Compute



Invoke actions.

- Since for the computation of the grid dimension all molecules need to be aligned, for a long trajectory this might take some time. So it may not be desirable to recompute the grid dimension all the time. If you are interested in the size of the grid before invoking the computation of the field, the *BBox* button must be pressed. The bounding box of the field, including all alignments, will be computed and from this, the grid dimension.
- Press the *Field* button to start the computation of the field. If the grid dimension is not up to date, it will be computed first.

2.9 MeanMolecule

This module computes the *mean* molecule of a molecular trajectory by averaging the atom positions. In order to do this, the molecules need to be aligned to some reference molecule which can either be a molecule from the trajectory or some other molecule which can be connected to the module via the *AlignMolecule* connection port.

By computing the mean molecule several times, each time taking the previous mean molecule as align molecule, you get a mean molecule of increasing quality.

Connections

Data [required]

Molecular trajectory for which the mean molecule should be computed.

AlignMaster [optional]

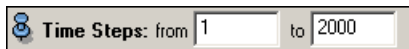
In order to compute the mean molecule, all molecules of the trajectory need to be aligned with respect to a certain molecule. If you don't want to align the molecules to a time step of the trajectory, you need to connect this port to some other molecule. In order to do so, right-click on the little rectangle of the module icon in the *object pool*. As result a pop-up menu appears. Select *AlignMolecule* and connect it to a molecule in the object pool. This connection is also needed if you want to use *selected atoms* for alignment instead of *all atoms*.

PrecomputedAlignment [optional]

Instead of aligning all time steps of a trajectory to the *AlignMaster*, you can use a precomputed alignment to align the time steps.

Ports

Time Steps

A port control box with a blue icon on the left. The text reads "Time Steps: from 1 to 2000". The numbers 1 and 2000 are in input fields.

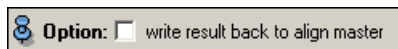
Specify the range of molecules used for the computation.

Alignment

Three stacked port control boxes. The top box has a blue icon and the text "Transformation: align to master" with a dropdown arrow. The middle box has a blue icon and the text "Select: all in slave in master in both" with a "Set" button and a dropdown arrow. The bottom box has a blue icon and the text "Selection empty".

The three ports above specify how the molecules are aligned to each other. See the description in the general section on *alignment of molecules* for a detailed explanation.

Option

A port control box with a blue icon on the left. The text reads "Option: ☐ write result back to align master".

Select this option if you want to iteratively improve the mean molecule.

Action

A port control box with a blue icon on the left. The text reads "Action: ☒ Dolt".

Start computation of mean molecule.

2.10 Measurement

This module lets you determine distances, angles, and dihedral angles between atoms of the molecule. To obtain a measurement, you need to select the atoms that are to be examined. In order to select objects in the viewer, you need to switch into the interaction mode of the viewer (for example by pressing the ESC key as a toggle) and **Ctr** L-clicking on the desired atoms. You can read about other ways to select structures in the documentation of the *Selection Browser*.

Connections

Module [required]

A module of type *MoleculeView* which can be used for the selection of atoms.

Ports

The following ports will only be visible if you have selected two atoms for distance measurement, three atoms for angle measurement, or four atoms for measurement of dihedral angles.

Selected atoms

 **Selected atoms:** 7 6 5

This port shows the indices and sequence of the atoms that are used for the measurement.

Options

 **Options:** ☒ show mean + standard deviation

This port is visible if your base molecule is derived from a trajectory. If you activate the toggle, not only the measurement of the currently active time step, but also the mean value of the complete trajectory and its standard deviation, will be evaluated. For large trajectories this may take some time as all atomic coordinates have to be loaded for each time step. So if you do not need this option, leave it unchecked.

Angle

 **Angle:** 107.86°

This port shows the value of the measurement. For two selected atoms, this is the distance; for three, the angle; and for four atoms, their dihedral angle. (The dihedral angle is defined as the angle between the two planes determined by the first three selected atoms and the last three selected atoms). The distance is given in angstroms, the angles in degrees.

Time-Mean

 **Time-Mean** 112.891 ± 6.07893°

This port is visible if your base molecule is derived from a trajectory and the *Time-mean* toggle is activated. It will show the mean value of the measurement by determining the arithmetic mean over all molecules that are part of the trajectory. In addition, the standard deviation of the time-mean will be shown.

Time Plot

 **Time Plot**

Like the time-mean port, this port is only visible for trajectories. By pressing the *Show* button you can see the plot of the measurement against the time steps of the trajectory. The vertical line of the plot indicates the position of the currently shown time step. The horizontal lines show the mean and the standard deviation.

Selection

 **Selection**

With the *Clear* button you can deselect all atoms.

2.11 MolSurfaceView

This module visualizes data objects of type *MolSurface*. Apart from a connection to the molecular surface, this module also should be connected to the molecules the surface was generated from, in order to exploit the full functionality of the *MolSurfaceView*. If the molecular surface was generated in the current *amira* session and the molecules were not deleted in between, the *MolSurfaceView* will be automatically connected to these molecules.

Like the other molecular visualization modules, this module allows you to color the surface according to various different color schemes, using the *coloring component*. You can also highlight parts of the surface by clicking on it. How the click is interpreted depends on the current color scheme. If default colors are used, clicking is interpreted as in the *SurfaceView* module. Otherwise, clicking is handled differently depending on the color scheme.

Connections

Data [required]

Object of class *MolSurface*.

ColorField [optional]

Scalar field to color the surface according to physical or chemical properties.

Colormap [optional]

Colormap for pseudocoloring the surface according to the values in a scalar field.

Molecule [optional]

Object of type molecule. Needed if information about the surface is required.

Continuous CM [optional]

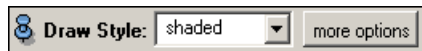
Needed for coloring the molecular surface according to atomic information.

Discrete CM [optional]

Needed for coloring the molecular surface according to atomic information.

Ports

Draw Style



Colormap

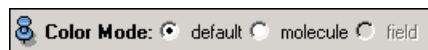


BaseTrans



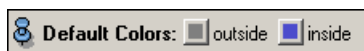
See the description of MolSurfaceView's *base class* for an explanation of the first three ports.

Color Mode



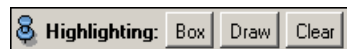
There are three color modes. The first mode uses default colors to color the outside triangles differently than the inside triangles. If one or two molecules are connected to the *MolSurfaceView*, you can choose the second mode to color the molecular surface according to the *coloring* component(s). The last mode, finally, allows you to color the surface by use of a scalar field (representing chemical or physical properties). This mode can only be chosen if the *ColorField* port is connected to a field.

Default Colors



You can change the default colors by clicking on them. A color editor will appear, enabling you to change the color.

Highlighting



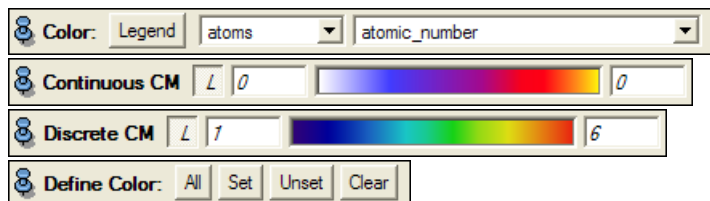
This port provides functionality for selecting (or highlighting) atom surface patches in the viewer window. For more information, see the general section on *highlighting*.

Buffer



This port is the front end to a filter that determines the atom surface patches being displayed by the *MolSurfaceView*. The functionality of the port's buttons are described in the general section on *highlighting*.

Coloring



The four ports above belong to the *coloring* component. There will be one component for each molecule connected to the *MolSurfaceView*.

2.12 MoleculeLabel

This module allows you to label certain parts of the molecule. If a *MoleculeLabel* module is connected to an object of type *Molecule*, all clicks in the viewing window normally handled by a viewing module connected to the same molecule will instead be handled by the *MoleculeLabel* module, by default. Handling of a click by the *MoleculeLabel* module means that it will display a new label. The kind of label to be displayed depends on the viewing module and its current color scheme. For example, if the current coloring level is *atoms*, a label for the selected atom will be displayed; if it is *residues*, a label for the residue the atom belongs to will be displayed. Up to two data items per label can be displayed. Which data is displayed can be set in the *Attributes* port (see below). Clicking a selected label will hide it.

The user interface is built as follows. The *levels* port allows you to choose a label level. The other ports, except the last one, *Options*, correspond to this level. They allow you to specify how the labels of the selected level will be displayed, i.e., what data is displayed for each label, and in which color and size labels are displayed. Do not assume that a label of the selected level will appear if you click parts of the molecule. This solely depends on the color scheme of the viewing module as mentioned above. You may also hide all labels of the current level.

The last port, *Options*, does not refer to any level, but determines the overall behavior of the module.

There are a few Tcl commands that allow you to set labels. The `setLabelString` command enables you to set arbitrary labels to an atom or a group of atoms. See the description of commands at the end of this page for more information.

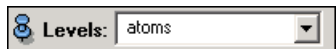
Connections

Data [required]

The molecule to be labeled.

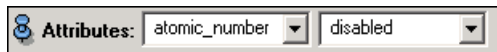
Ports

Levels



Choose a level to change the display properties of all labels of this level. The *Attributes*, *Level Option*, *Buttons*, *Font size*, and *Color* ports have individual values for the chosen level.

Attributes



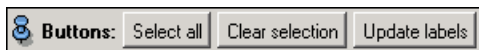
Port to select two attributes that will make up the label string for the current level. By default, only one attribute is displayed.

Level Option



If the level option *is visible* is set, all currently selected labels of this level are displayed. In order to hide all labels of a certain level without removing them, deselect this option.

Buttons



The three buttons of this port affect only the labels of the current level. The first button displays all level labels, the second one removes them all. The third button updates the label strings according to the currently selected attributes, then updates the positions of the labels to display them as well as possible.

Font size



Change the font size of the level's labels.

Color



Color of the level's labels. To change the color, click on the *Color* button. As result, a color editor will be displayed in which you can choose the color you wish.

Options



Overall options. If the first option is selected, clicks that are normally handled by the viewing modules will instead be handled by the *MoleculeLabel* module, resulting in new labels being displayed and already visible labels being removed. The second option allows you to compose labels of one level differently. If the option is not set, the currently displayed labels will not be changed if the *Attribute* port changes. Otherwise, they will be.

Commands

There are a few commands to set labels via the command line interface.

```
setLabel <levelName/{groupName|groupRange}> [<attr1>] [<attr2>]
```

This command sets (a) new label(s) for a group or a range of groups specified by the first argument. If one or two attributes are given (as strings), those attributes will be assigned as labels to the group(s). If no attribute is given, the currently selected attributes of the level are used. For more information about specifying a group or a range of groups, see the `list` command in the description of *Molecule*.

Example:

```
setLabel residues/NIL14-NIL98 name index
```

`unsetLabel <levelName/{groupName|groupRange}>`

Hides the label(s) of the specified group or range of groups.

`setLabelString <levelName/groupName> <string>`

Assign the label 'string' to the specified group.

2.13 MoleculeView

The *MoleculeView* allows you to display molecules in three different representations: atom spheres, wire frame, and ball-and-stick. For each of these representations there are two *Quality* modes, *fast* and *correct*.

Atoms can be colored according to the *groupings* defined on the molecule, e.g., atoms, residues, and chains. Moreover, the displayed part of the molecule can be restricted by using the *selection browser* as well as by interaction in the viewer.

In addition to just visualizing molecules, this module provides tools that facilitate interaction with the molecule. For example, you can select atoms to find out which index they have and which group they belong to. The ability to select atoms is also a precondition for the alignment procedure described in the section on *aligning molecules*.

In order to measure distances and angles in the molecule, right-click the *MoleculeView* in the *object pool*, which will open a pop-up menu with the entry *Measurement*.

Connections

Data [required]

The molecule to be visualized.

ReferenceMolecule [optional]

A reference molecule must be connected if you want to study the alteration of the torsion angles from one time step of a molecular trajectory to another (see the *Options* port below for more information).

Continuous CM [optional]

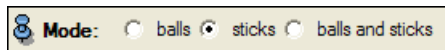
The colormap that is used to map values of float attributes to colors.

Discrete CM [optional]

The colormap that is used to map values of discrete attributes, e.g., integers and strings, to colors.

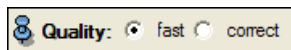
Ports

Mode



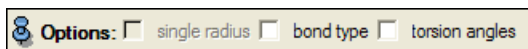
Select one of three main display modes: balls, sticks, or balls and sticks. The default is sticks.

Quality



The appearance of the balls and sticks can be altered by selection of one of two different qualities, a *fast* mode and a *correct* mode. The fast mode shows sticks as lines and spheres by rendering a square with the image of the sphere. This permits the display of large numbers of spheres at interactive speed. However, showing the image of a sphere means the 3D expansion of the sphere in only two directions is taken into account. As long as none of the spheres intersect, there will be no visible difference between spheres drawn in *fast* mode and spheres drawn in *correct* mode. In the correct mode, spheres are represented by textured triangular meshes and sticks by cylinders.

Options

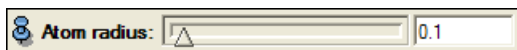


Single radius: display all atoms with the same radius.

Bond type (only in fast quality mode): display correct bond type, e.g., single, double, triple, or aromatic.

Torsion angles: display torsion angles by the use of a coil around the stick.

Atom radius



If the option *single radius* is selected, the value given here will be interpreted as the sphere radius for all atom spheres. Otherwise, the value scales the van der Waals radius of the atom.

Bond radius



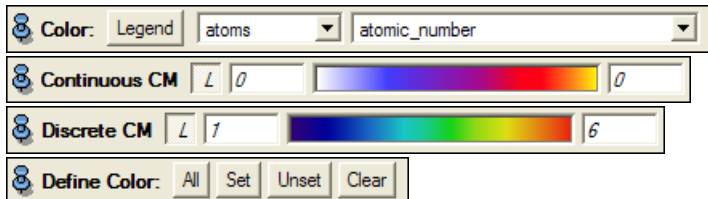
Radius of cylinders representing bonds (only visible in *correct* display mode).

Twist factor



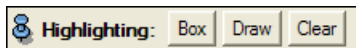
Allows adjustment of the twist of the coil.

Coloring



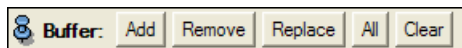
The four ports above are described in the general section on *visualization of molecules*.

Highlighting



This port provides functionality for selecting (or highlighting) atoms in the viewer window. For more information, see the description of the highlighting port in the general section on *visualization of molecules*.

Buffer



This port is the front end to a filter that determines the set of atoms being displayed by the *Molecule-View*. The buttons are described in the section on *visualization of molecules*.

Commands

`setTextureAntiAliasing <value>`

This command enables (1) or disables (0) antialiasing for the textures used in fast sphere mode. Antialiasing has no impact on the rendering speed. However, if antialiasing is enabled and you have many different colors, the generation of the textures might be considerably slower.

2.14 Observables

This module is used to display information about a molecular trajectory. All properties stored in the trajectory can be displayed, e.g., *potential energy*, *kinetic energy*, *total energy* etc.

Connections

Data [required]

The molecular trajectory for which properties should be displayed.

Time [optional]

A time object that may be connected to the module in order to synchronize, e.g., a viewing module with the line in the plot window, indicating the position of the time step within the trajectory.

Ports

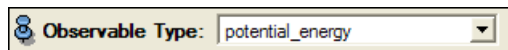
Time



This port is most useful in conjunction with a viewing module in order to see the property of the displayed time step. In order to create a time object, click the *Time* slider with the right mouse

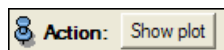
button and select *Create time* from the menu. As a result a time object will appear in the object pool, which should be connected with an object of type *Molecule* which can be visualized with, for example, the *MoleculeView* module.

Observable Type



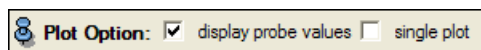
Choose the type of property that should be plotted.

Action



Plot the *Observable Type* within a new window.

Plot Option



Select the first toggle if the values of the plotted properties at the current time are shown in the plot window. If the second option is selected, only a single plot will be shown, and changing the *Observable Type* will update the current plot.

2.15 PrecomputeAlignment

This module can be used to generate a data object that contains an alignment for every step of a whole *trajectory*.

Connections

Data [required]

The molecule must be connected to the *molecular dynamics trajectory* that the alignment should be computed for.

AlignMaster [optional]

Reference molecule.

PrecomputedAlignment [optional]

You will find the description of the above two connections in the section on *alignment of molecules*.

Ports

Mode



Two modes are available. In the first mode (*simple alignment*), the module acts as a recorder for the time-step-based alignments described in the section *Aligning Molecules*. In the second mode (*multiple alignment*), the mean squared distance of every time step to all other time steps is minimized. A reference molecule can be connected to specify atoms to be taken into account. If a reference is connected, the mean of the aligned time steps will be aligned to the reference.

Alignment



You find the description of the three ports above in the section on *alignment of molecules*.

Action



Pressing this button starts the computation of the alignment.

2.16 SecStructureView

This module provides the functionality for viewing the secondary structure of a molecule. It relies on the information given in the data entry of the connected molecule. Helices, sheets, and turns can only be displayed if information about these structures has been read in (for example from a *pdb* file).

The backbone will be interpolated by using the coordinates of those atoms whose comment is set to “N”, “C” and “CA” (if the structure is a polypeptide) or respectively “O5*”, “O3*” and “C3*” (if the structure is a DNA or RNA). If the atom-comment attribute does not exist (for example, molecules that have been imported from UniChem files) no backbone can be viewed.

The user interface of the module includes general ports to configure the overall appearance of the molecule, as well as specific ports for each secondary structure type. The latter let you adjust type-specific parameters, such as colors or widths. To switch between the specific ports, just click on the structure type you want to configure in the *View Options* port.

Connections

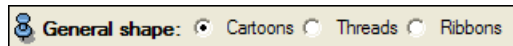
Data [required]

The molecule that is to be viewed.

Ports

General Ports:

General Shape



This port determines the general shape for viewing the secondary structure. The following view modes are available:

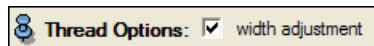
- **Cartoons:** In the *cartoon* mode, you can select a view mode for each secondary structure type separately. The backbone and the turns will be shown as tubes that run through the central atoms of each residue. For helices and sheets, additional (more abstract) view modes like cylinders or arrows are available. You can configure these submodes in the specific port sections for the secondary structure type. This is the mode most frequently used for representing the secondary structure of a molecule. However, due to its complex triangulation, the rendering performance is distinctly worse than that of the other two modes.
- **Threads:** The backbone will be drawn as a set of lines running parallel to the peptide planes of the amino acids. This mode will show the surface orientation of the backbone, thus making parts containing symmetries (like helices or sheets) easily recognizable. It is also the view mode which can be rendered fastest.
- **Ribbons:** Same as the thread mode, but the areas between the lines will be a filled surface, thus giving a better surface impression with the trade-off of slightly lower rendering performance.

General Threads Width



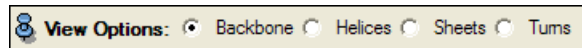
With this port you can adjust the width of the threads. A similar port exists for the ribbon view mode.

General Threads Options



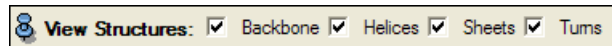
With the *width adjustment* option enabled, the widths of backbone parts which belong to helices or sheets will be increased by the factor 1.5. All other parts will be decreased by the factor 0.5. Thus helices and sheets will be easier to recognize. A similar port exists for the ribbon view mode.

View Options



To switch between the options of the different structure types, you can use these radio buttons. The option ports for the different structure will then be visible.

View Structures



This port can be used to filter out the display of all sheets, helices, or turns. The backbone parts will be displayed as random coil tubes instead.

Filter

The secondary structure view also employs a filter which can be used to hide certain parts of the molecule. You can hide sections by using the *selection browser*. This port is explained further in the section *visualization of molecules*.

Complexity



The *Complexity* port allows the adjustment of the overall exactness of the structures displayed. It will influence the number of points with which the backbone coordinates will be interpolated as well as the detail of triangulation or the number of lines in the thread view mode. If you are viewing large molecules on a machine with slow graphics hardware, turn this value down to a more appropriate value.

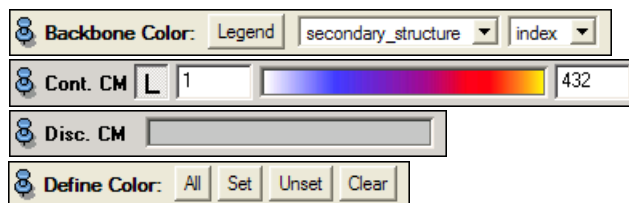
Backbone Ports:

Backbone Tube Radius



This port lets you change the backbone radius.

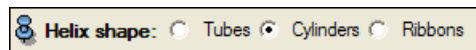
Coloring



The four ports above are described in the section on *visualization of molecules*. Similar ports exist for helixes, sheets, and turns.

Helix Ports:

Helix shape



This port lets you choose between different methods for viewing helices in *cartoon* mode.

- **Tube:** The tube view will show the helix by drawing a tube connecting the interpolated backbone coordinates.

- **Cylinder:** The cylinder view will approximate the axis using a least squares error minimization for the sum of the distances between the backbone coordinates of the helix and the axis. These axes will be used to center the cylinders for the helices.
- **Ribbon:** The ribbon view is similar to the tube representation except that the backbone is shown using a surface whose normals are set perpendicular to the peptide planes.

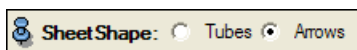
Helix Radius



With this port you can adjust the radius of the shown tubes, or cylinders.

Sheet Ports:

SheetShape



This port lets you choose between different methods for viewing sheets in *cartoon* mode.

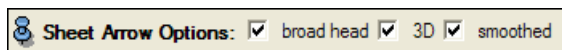
- **Tube:** The tube view will display the sheet using a tube connecting the interpolated backbone coordinates.
- **Arrow:** The arrow view will display arrows that indicate the orientation of the peptide planes, thus showing the surface determined by the hydrogen bonds between the strands.

Sheet Arrow Width



This port is visible if you view the sheets as arrows. It determines the width of the arrows.

Sheet Arrow Options



The *broad head* and *3D* options configure the style of the arrows further. The *smoothed* option will Bezier interpolate the backbone coordinates of the sheet to produce a smoother visualization of the surface.

Sheet Radius



If you view the sheets as tubes, this port lets you configure the radius of the tubes.

Turn Ports:

Turn radius

A control box for the 'Turn radius' parameter. It features a blue icon on the left, followed by the text 'Turn radius:'. To the right is a horizontal slider bar with a triangle marker, and a numeric input field on the far right containing the value '0.3'.

This port lets you specify the radius of the *turn* tubes.

2.17 TubeView

With the *TubeView* module you can connect an arbitrary set of atoms with a tube. The atom coordinates will be interpolated by a cubic spline. To define the atom set, you can use *atom expressions*. The order with which the atoms are connected is determined simply by their sequence.

Connections

Data [required]

The molecule you want to be shown.

Ports

Part

A control box for the 'Part' parameter. It features a blue icon on the left, followed by the text 'Part:'. To the right is a text input field containing the value 'backbone'.

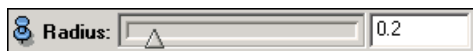
This port lets you enter an *atom expression*. The atom expression will determine the set of atoms to be connected. The default value is *backbone*, which means that all peptide and DNA/RNA backbone atoms will be connected.

Distance Cutoff

A control box for the 'Distance Cutoff' parameter. It features a blue icon on the left, followed by the text 'Distance Cutoff'. To the right is a horizontal slider bar with a triangle marker, and a numeric input field on the far right containing the value '20'.

All atoms that are further away than the specified value will not be connected.

Radius

A control box for the 'Radius' parameter. It features a blue icon on the left, followed by the text 'Radius:'. To the right is a horizontal slider bar with a triangle marker, and a numeric input field on the far right containing the value '0.2'.

This port specifies the radius of the tube.

Complexity

A control box for the 'Complexity' parameter. It features a blue icon on the left, followed by the text 'Complexity:'. To the right is a horizontal slider bar with a triangle marker, and a numeric input field on the far right containing the value '0.3'.

This port specifies the complexity of the tube triangulation as well as the number of interpolation points to be added between each atom coordinate.

Coloring

A control box for the 'Coloring' parameter. It features a blue icon on the left, followed by the text 'Color:'. To the right are two dropdown menus. The first dropdown menu is set to 'Legend' and the second is set to 'atoms'.

Disc. CM

Cont. CM L 1 1844

Define Color: All Set Unset Clear

You will find the description of the four ports above in the section on *visualization of molecules*.

Chapter 3

Alphabetic Index of Data Types

3.1 MolSurface

The data class *MolSurface* represents molecular surfaces in Amira, generated by the computational module *CompMolSurface*. It is derived from the class *Surface* and hence, everything that can be done with an object of type *Surface* can also be done with an object of type *MolSurface*. Additional information stored in class *MolSurface* includes:

- the molecular surface type, i.e. van der Waals, solvent accessible, or solvent excluded surface,
- the solvent probe radius,
- for each molecule that contributed to the surface, its number of atoms,
- and for each point or triangle, the index of the atom it belongs to.

The entire functionality of the editors provided for objects of type *Surface* is available, including simplification. However, some care needs to be taken. If you modify the number of points or triangles, the module *MolSurfaceView* will no longer be able to color the surface according to atom attributes. In contrast, other operations, such as edge flipping to improve the surface quality, will not affect the representation.

Connections

Master [optional]

Connection to the module that generated the molecular surface.

Commands

For a description of Tcl commands, see the documentation of the *Surface* data class in the *amira User's Guide*.

3.2 MolTrajectory

An object of type *MolTrajectory* represents a series of molecular configurations. A trajectory can be created by loading a file containing a trajectory, or it can be attached to an object of type *MolTrajectoryBundle*, extracting one of the trajectories contained in that bundle.

Connections

Bundle [optional]

The trajectory may be connected to a *trajectory bundle*. If so, the *Trajectory* port will appear in the user interface of the data object which enables you to control which member of the bundle this trajectory represents.

Ports

Trajectory



The *Trajectory* port is visible only if the trajectory object is connected to a trajectory bundle. The menu offers lets you choose from the trajectories contained in the bundle.

3.3 Molecule

An object of data type *Molecule* contains information about the structure of the molecule. Typical information that is stored are the types and positions of the molecule's atoms and the bonds between the atoms, plus the type of each bond. Furthermore, the data object stores information about groups of atoms in a hierarchical way. For example a functional group consists of a number of atoms, several functional groups may form a residue, and a couple of residues may form a secondary structure. This allows quick traversal the structure of a molecule's structure.

Molecules can be visualized with the following viewing modules: *MoleculeView*, *BondAngleView*, *SecStructureView*, or *TubeView*. In addition, there are two modules that generate molecular surfaces. The *CompMolInterface* module can be used to generate intramolecular interfaces, such as between single atoms or residues. Also, the interface between two molecules can be computed. The *CompMolSurface* module provides the functionality of computing the *solvent excluded*, *solvent accessible*, or the *van der Waals* surface.

When comparing the structures of several molecules with each other, one is faced with the problem of aligning molecules to each other. This can be easily done by connecting the *AlignMaster* connection port to a second molecule which will serve as reference. There are several alignment modes. If the molecules have the same number of atoms you can align the two molecules to each other by using all atoms, whereby the *i*'th atom of the first molecule corresponds to the *i*'th atom of the second molecule. You can also select atoms in either the *Slave* or the *Master* molecule. The *Slave* is the molecule to be

aligned. If the molecules have different numbers of atoms, the only way to align molecules is to select atoms in both molecules.

Connections

Data [optional]

The molecule may be connected to a *molecular dynamics trajectory*. If so, a *Time* port will appear in the *working area* which enables you to load new time steps into the molecule.

Time [optional]

Sometimes it is necessary to synchronize two or more time-dependent data objects, such as a molecule from a trajectory and the corresponding electrostatic field. In this case, the *Time* port can be connected to a *Time* object.

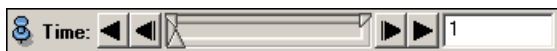
AlignMaster [optional]

PrecomputedAlignment [optional]

You find the description of the two ports above are described in the section on *alignment of molecules*.

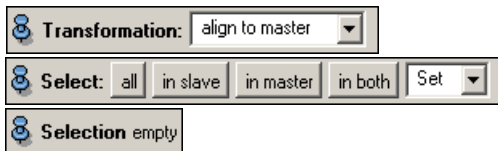
Ports

Time



The *Time* port is visible only if the molecule object is time dependent, i.e., it is connected to a molecular dynamics trajectory. In the slider you can select a specific time step. If you wish to step through the trajectory, use the buttons next to the slider. For continuous animation, use the outer buttons.

Alignment



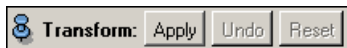
You will find the description of the above three ports in the section on *alignment of molecules*.

Selection Browser



Press this button if you wish to open the selection browser for this molecule.

Transform



If you want to transform all atom coordinates of the molecule by the current transformation in order to save the molecule with the transformed coordinates to a file, press the *Apply* button. As long as the *Transform Editor* is active you can also undo your actions.

Commands

You can use the *console window* for entering additional commands for molecules. The easiest way to use the console for a molecule is to click on the molecule object in the object pool and then press the TAB key in the console window to display the name of the selected object. After the name you can use the following selection commands:

`list`

Lists all levels defined for the molecule.

`list <levelName>`

Lists all groups of level *levelName*.

`list <levelName/[groupName|groupRange]>`

Prints information about a single group or multiple of groups specified by a group range.

`define <levelName/groupName> <groups>`

This command defines a new group *groupName* in the level *levelName*. If *levelName* does not yet exist, it will be added as a new level. *groups* is a list of groups of possibly different levels. Groups can be specified using ranges.

Example: The following command defines group *one* in level *test* consisting of atoms 1 and 3 and residues 4 to 6.

- `define test/one atoms/1 atoms/3 residues/4-6`

`defregex <levelName/groupName> <atomExpr>`

Define a new group by using *atom expressions*. The new group will only contain atoms, i.e., no higher level groups.

`sel <atomExpr>`

Lets you add atoms specified by *atomExpr* to the current selection.

Example: The following command will select all carbon atoms which are located in helices.

- `sel a=C AND s/type=helix`

`desel <atomExpr>`

Lets you remove atoms from the selection.

`applyTransform`

The current transform is applied to the molecule, i.e., the original atomic positions are replaced by the transformed positions.

Chapter 4

Alphabetic Index of File Formats

4.1 MDL

The *sdf* file format is part of the set of MDL file formats used for saving chemical structures. *amira* uses and writes the information of the atom and bond blocks.

4.2 PDB

PDB is a file format for storing structures of biomolecules. The format is used for archiving molecules in the online protein data base. [<http://www.rcsb.org/pdb/>]

Supported records

amira uses most, but not all, information supplied in a PDB file. The following record types will be parsed:

HELIX SHEET TURN SITE MODEL ENDMDL ATOM HETATM TER CONECT END

The rest will be skipped.

When saving a structure, the following record types will be written:

HEADER TITLE AUTHOR

SEQRES HELIX SHEET TURN SITE SSBOND

ATOM HETATM TER CONECT MASTER END

amira supports PDB files that follow the conventions of PDB format version 2.0 or higher.

How to identify PDB groups in *amira*

In *amira* all levels (atoms, residues, bonds , ...) contain a set of information fields called attributes. Every level contains the field 'index' which is an internal identifier that does not necessarily correspond to the succession of the groups in the file.

If the molecule has been read from a PDB file then the **atom** index usually corresponds to the sequence number (columns 7 - 11) minus 1 of the respective ATOM records.

The ‘comment’ field contains the information of the field named ‘atom name’ (columns 13 - 16) in the PDB file.

The ‘name’ field of **chains** contains the chainID (column 22) of the respective ATOM record. HETATM records whose chainID field is empty will not belong to any chain in amira. The chain name of atoms belonging to ATOM records whose chainID field is empty will be set to ‘NIL’.

The ‘name’ field of **residues** is composed of the chain name and the residue sequence number (columns 23 - 26). If the residue is a het-group, the name will start with ‘HET’.

The field ‘type’ contains the residue names (columns 18 - 20). If none is given, the field will be set to ‘UNK’ (for unknown).

Example:

```
ATOM    441  CB  VAL L   58    21.025  37.362  -2.515  1.00  8.15  1IGM 562
```

The atom will be known with index ‘441’ (assumed that there are no irregularities in the PDB file) and comment ‘CB’. It will belong to the residue with name ‘L58’ and of type ‘VAL’ which is sited on the chain with name ‘L’.

Possible problems

Some molecular modelling packages use slightly altered conventions for saving PDB files. If you encounter any problems in reading such files please check the following section and adjust your files as necessary.

amira can read **several models of one molecule as a trajectory** if the different versions are separately declared between MODEL and ENDMDL records. If there are several possible conformations in one model (by using alternate location identifiers), amira will read only the first version (that one with the altLocId ‘A’).

Molecules that are declared as models need to have the same number of atoms as amira will only read the topology once. Only the coordinates are read in for each timestep. If you have **several different molecules** in one PDB file, you can separate them by END records. amira will load every molecule ended by an END records separately, creating its own data object.

The **automatic creation of bonds** of peptide and RNA/DNA residues is done by comparing the atoms of the residues with a residue database. The method requires the atom names and the residue names to follow the PDB standard. Sometimes third party programs write atoms like O3* as O3’. This will prevent amira from recognizing the atoms in the database. Connections between consecutive residues is done by checking the sequence of residues of each chain. **Bonds of non-standard residues** must be declared in the connect records.

Columns 77-78 should contain the element name of the atom. In some PDB files, this field is empty or contains other information. In this case, amira uses the first two letters of the atom name to determine

the atom type. However, in special cases this is not unequivocal. The field is not allowed to contain non-numerical information.

All information in columns 79-80 is interpreted as the charge of the atom. Thus, line numbers in this field will lead to strange charge values.

You can get complete documentation of the file format at <http://www.rcsb.org/pdb/>.

4.3 PSF/DCD (CHARMM)

This is a format used by CHARMM. It consists of two files, one containing the structural information (suffix .psf), and one containing the trajectory, i.e., the atom coordinates (suffix .dcd).

amira can read this format and translates the sections of the psf file into grouping levels.

4.4 Tripos

The *mol2* file format is used to save Tripos Sybyl molecule information. See more at <http://www.tripos.com/services.old/mol2/index.html>.

amira uses and writes the information given in the molecule atom and bond records.

4.5 UniChem

The UniChem file format (extension *uni*) is used by the UniChem molecular modelling software to save the structure of molecules and computational results. See more at <http://www.oxmol.com/software/unichem/>.

amira uses all structural information supplied in the COORDINATES and BONDS blocks. Files that are exported by amira follow file format version 4.0 and can be used with the UniChem software.

4.6 ZIB Molecular File Format

The *zmf* file format is a structured file format developed for exchanging molecular data.

This description is divided into two parts. First we describe the general and context-independent file structure. Second, we explain our structuring conventions for molecular data.

Structured files

A structured file consists of a header and two sections: a plain text section describing the file's structure, and a binary section containing the actual data:

```

STRUCTURED FILE V0.1 BINARY_BE MOLECULE_TRAJECTORY
TYPE
struct {
    atoms : array of struct {
        id      : integer*4;
        name     : string;
        mass     : real*4;
        properties : integer*4[8];
    };
};
DATA;
..... binary data .....

```

The file structure is similar to a structured data type in C or PASCAL. There are two composite data types (**array** and **struct**), and three elementary data types (**integer**, **real**, and **string**). Finally there exists a special type (**extern**) for including other structured files.

struct contains named fields of arbitrary heterogeneous type.

array contains numbered fields of the same type. Arrays are one-dimensional only. Their size can either be specified in the type declaration (Example: `array[3] of integer`) or in the data section.

real and **integer** contain numerical, possibly multi-dimensional, data. The number of bytes in an element may be specified using “*n”. At the moment only 4-byte types are allowed. The rank (number of dimensions) must be specified in the type declaration. The number of elements in each dimension may be specified in the type declaration or in the data section (indicated by “.”). Example: `integer*4[4,.,3]`.

string contains ASCII data of arbitrary length. Again the length can be specified in the type declaration (Example: `string*22`) or in the data section.

extern is a special kind of a **string**. The length must be specified in the data section. The string is interpreted as a structured file whose contents should appear as part of the current file structure.

The data section is written as follows:

struct: The individual members are written in the given order.

array: The elements are simply written one after another. If the array size is unspecified, it is written in front of the array as a 4-byte integer.

string and **extern**: The data is written byte by byte. If the size is unspecified in the data type, it is prepended as a 4-byte integer.

real and **integer**: Individual elements are written in the format specified in the file header (big endian or little endian). Floating point numbers are in IEEE format. Matrices are written in FORTRAN order. If any dimension is unspecified in the data type, it is written as a 4-byte integer before the data. If more than one dimension is unspecified, they (and only they) are written before the data in their

natural order. If a dimension is specified in the data type, it is never written in the data. Example:

```
integer[2,.,4] = [ [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ],  
                  [ [13,14,15,16], [17,18,19,20], [21,22,23,24] ] ]
```

is written as a sequence of 4-byte integers.

```
3 1 13 5 17 9 21 2 14 6 18 10 22 3 15 7 19 11 23 4 16 8 20 12 24
```

The leading 3 specifies the dimension unspecified in the data type. The 24 numbers that follow it are the entries of the array.

Molecular data - a minimal file

When reading a structured file containing molecular data, `amira` expects the file structure to have certain elements. Moreover the file may have arbitrary additional elements, which will be ignored. The minimal structure a molecular data file must supply is:

```
struct {  
    molecules : array of struct{  
        name : string;  
        groupings : struct {  
            atoms : array of struct{  
                id : integer*4;  
                type : integer*4;  
                name : string;  
            };  
        };  
        trajectory : array of struct{  
            coordinates : real*4[3,.];  
        };  
    };  
    typbase : struct {  
        atoms : array of struct{  
            id : integer*4;  
            name : string;  
            number : integer*4;  
            radius : real*4;  
        };  
    };  
};
```

In the following we will use a notation inspired by C to denote elements of the structure. A `struct`'s field will be specified as the `struct` followed by a "." and the name of the field. For array elements we will write the array followed by brackets ("[]"). For example, "`molecules[].groupings.atoms[].id`" means the field "id" in an element of the array "atoms" which is a field of the struct "groupings" which is a field of an element of the array "molecules".

The overall type of the file is a `struct` containing the fields **molecules** and **typbase**. **molecules** is an array of elements, each of which describes a molecule. **molecules[].name** contains the name of the molecule.

molecules[].groupings specifies the topology of the molecule, i.e., which atoms it contains. Every element of **molecules[].groupings.atoms** describes one atom of the molecule. **atoms[].id** is a unique numerical identifier. **atoms[].type** is a numerical reference to a field in the **typbase** described below. **atoms[].name** is a textual identifier unique with respect to all atoms. It is used to specify this atom on the command line.

molecules[].trajectory is an array with every element describing one time step of a molecule trajectory. **trajectory[].coordinates** has the same number of elements as **molecules[].groupings.atoms** and every element specifies a position for the corresponding atom.

typbase is structured analogously to **molecules[].groupings**. Corresponding to **molecules[].groupings.atoms**, the array **typbase.atoms** holds information common to atoms of the same type. Specifically, the type information of an atom is stored in the **molecules[].groupings.atoms[].type**'th element of **typbase.atoms**.

typbase.atoms[].id holds a numerical identifier that should be identical to the element index. **typbase.atoms[].name** is a textual label for making the types user transparent. **typbase.atoms[].number** and **typbase.atoms[].radius** are the atomic number and radius of an atom having this type.

Bonds

Bonds can be specified by adding an array named **bonds** to **molecules[].groupings**:

```
bonds : array of struct{
    id : integer*4;
    type : integer*4;
    components : integer*4[2];
    index : integer*4;
};
```

bonds[].id is a unique numerical identifier which should be greater than any value specified in **atoms[].id**. Analogous to **atoms[].type** for atoms, **bonds[].type** specifies a numerical type for every bond, although this information is not yet interpreted by amira. The array **bonds[].components** holds the numerical identifiers of the two atoms connected by the bond. **bonds[].index** finally indicates if it is a single (1), double (2), triple (3), or an aromatic bond (4). The value 8 indicates a bond of unknown type.

Molecular data - a big example

The information stored in the described file format can be divided into different degrees of generality. Information stored in **typbase** is specific to the molecular force field used in the calculations leading to a molecular trajectory. It would be redundant to store this information in every file for which the same molecular force field was used. Thus it can be stored in an extra file, which is then referenced by an **extern** declaration.

A similar mechanism applies if more than one trajectory is calculated for the same molecule. The topological information about atoms making up the molecule and bonds connecting them will be the same for all the trajectories. On the other hand, we do not want a file for every single molecule topology. Thus our trajectory can reference another file containing a multitude of molecular topologies and just supply the time steps itself.

The following is an example realizing these concepts:

trajectory file:

```
struct {
    typbase    : extern;
    molecules : array of struct{
        id          : integer*4;
        observations : struct {
            global_obs : string[.];
        };
        text          : string[.];
        name          : string;
        molbase       : extern;
        trajectory    : array of struct{
            global_obs : real*4[.];
            coordinates : real*4[3,.];
        };
    };
};
```

molbase file:

```
struct {
    typbase : extern;
    molecules : array of struct{
        id          : integer*4;
        observations : struct {
            global_obs : string[.];
        };
        text : string[.];
    };
};
```

```

name      : string;
groupings : struct {
    atoms      : array of struct{
        id      : integer*4;
        type     : integer*4;
        name     : string;
    };
    bonds       : array of struct{
        id      : integer*4;
        components : integer*4[2];
        type     : integer*4;
        index    : integer*4;
    };
    dihedrals   : array of struct{
        id      : integer*4;
        components : integer*4[4];
        type     : integer*4;
    };
    residues    : array of struct{
        id      : integer*4;
        name     : string;
        remark   : string;
        type     : integer*4;
        from_id  : integer*4;
        to_id    : integer*4;
    };
};
trajectory : array of struct{
    coordinates : real*4[3,.];
};
};

```

typbase file:

```

struct {
    atoms : array of struct{
        id      : integer*4;
        type     : integer*4;
        name     : string;
        number   : integer*4;
        periodic_row : integer*4;
        mass     : real*4;
    };
};

```

```

        charge      : real*4;
        partial_bond_charge_increment : real*4[2];
        radius      : real*4;
        N_i         : real*4;
        A_i         : real*4;
        g_i         : real*4;
        alpha       : real*4;
        datyp       : string;
        properties  : integer*4[8];
        equivalence : integer*4[4];
    };
};

```

Some yet undescribed features appear in this example. The first is the possibility to supply scalar observables corresponding to the single time steps. Possible values are different energies like kinetic and potential energy. The names of the observables are read from **molecules[].observations.global_obs[]** in the trajectory file. Their values are stored in **molecules[].trajectory[].global_obs[]**.

Besides **bonds**, the file can contain arbitrary groups of atoms or groups of groups. These groups are organized in levels which appear here as the elements of **molecules[].groupings**.

Every level is an array whose elements describe the particular groups. Every group is a struct with the fields **id** and **type**, and also fields specifying the contents of the group. **id** is numerical identifier unique over all groups specified for the molecule. Specifically, the identifiers in one level should be consecutively numbered and bigger than all the identifiers used in the preceding levels.

There are two ways of specifying a group. The general way is to specify all elements explicitly. In that case, the group contains a field **components[]** containing the ids of all elements of the group. If the elements of a group cover a consecutive sequence of ids, it is sufficient to specify only the first and the last id. In that case the group contains the fields **from_id** and **to_id**.

dihedrals are groups of four atoms connected by three bonds like a chain. Every such group defines two intersecting planes. The angle of intersection is called a *dihedral angle* or *torsion angle*. For performance reasons *amira* will read *dihedrals* only if the molecule has less than 500 atoms. Torsion angles can be displayed with the *MoleculeView*.

residues are groups representing the building blocks of a complex molecule, for example the amino acids of a protein. All groupings can be used to color the molecule in the *MoleculeView* or in the *BondAngleView* using the general *coloring facilities*.

Chapter 5

Alphabetic Index of Editors

5.1 Molecule Attribute Editor



This tool allows you to edit attributes of groups contained in a *Molecule* data object.

In amira, each molecule consists of several grouping levels, which are chemical subdivisions of the molecule of different degrees of complexity. The levels are ordered in a hierarchy in the way they depend on each other. The most basic subdivision and therefore the root of the hierarchical tree is the atom level. All bonds or residues are subdivisions which contain sets of atoms, while secondary structures consist of sets of groups of the *residues* level, and so on. In the language of database systems, levels are entities and groups are instances of these entities. Figure 5.1 shows the entity relationship model for the most common grouping levels.

Each level has a set of attributes. Attributes are properties of groups of type string, integer, or float. The number of levels and attributes depends on the file format from which the molecule is read.

The attribute editor (Figure 5.2) offers two different tools:

- addition, deletion, or renaming of attributes from a level
- changing of attribute values of certain groups of a level

The pull-down widget at the top of the window lets you choose the level to which the action will be applied. The table below will show all attribute names of the level in the left column. The second column will contain the corresponding attribute values of all completely selected groups of the given level.

The coloration of these table cells can change depending on the attribute and the selected groups:

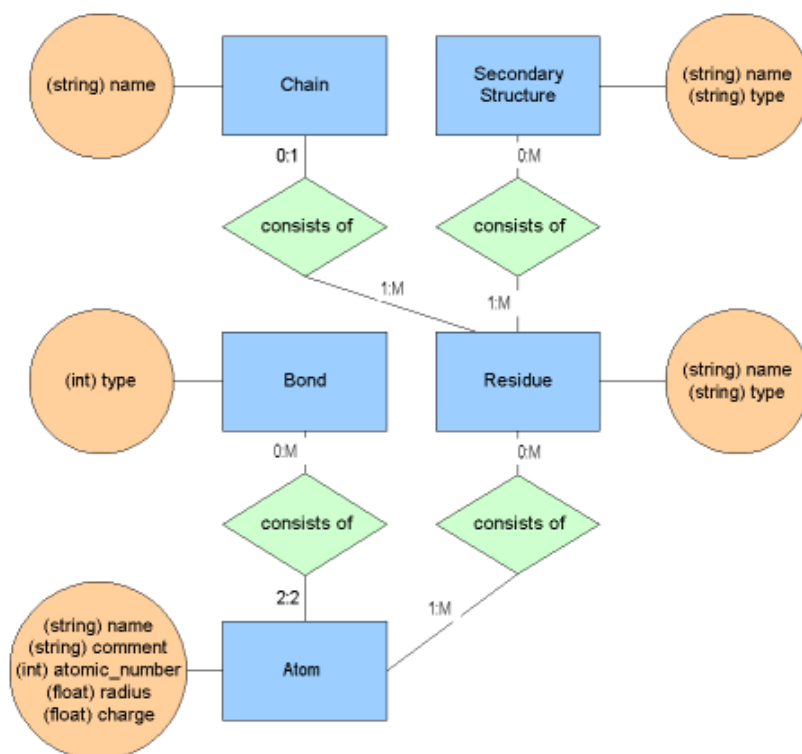


Figure 5.1: Entity relationship model of the most common levels

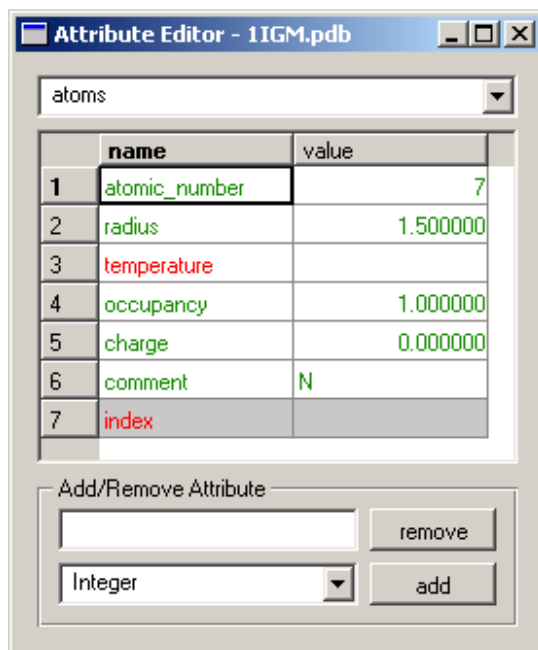


Figure 5.2: Attribute Editor

- gray background: attribute can not be changed or deleted (attribute is used as an index by amira)
- white background: attribute can be changed and deleted
- black text: only one group of the level is selected
- green text: several groups of the level are selected but all of their attribute, values are equal
- red text: several groups of the level are selected and at least two attribute values of these groups differ (in this case no attribute value will be displayed in the right column)

To change the name of an attribute, simply left-click in the respective cell of the left column and enter the desired name. To change attribute values of the currently selected groups, click in the cell in the right column. Except for index attributes (gray background), all values can be changed. If several groups are selected and no value is displayed (because some values differ), the adjustment of the value in the empty table cell will reset the value of all groups to the given choice. Therefore the color of this attribute will change from red to green.

To select groups you can use the *selection browser*.

The lower part of the window lets you add or delete attributes by typing the name into the text box and using the appropriate button. When adding attributes, you also must choose the internal format type (string, integer, or float) with the pull-down menu.

5.2 Molecule Editor



This tool can be used to change the geometry and topology of a *Molecule* data object. To change group attributes, you can use the *Molecule Attribute Editor*.

Most of the editor's functions are applied to sets of atoms, which can be selected by using the *selection browser* or by directly clicking on atoms in the viewer.

The molecule editor, Figure 5.3, has two tabs:

- The *Transform* tab contains all tools to change the geometry of selected atoms by adjusting positions, bond lengths, torsional angles, and bond angles.
- The *Tools* tab offers methods to split or copy parts of the molecule, as well as an interface for adding or removing bonds between atoms.

Transform Tab

The transform tab is divided into four different sections. Each section can be used to adjust certain coordinates of the currently selected atoms. The different coordinate types are:

- *Position* is the Cartesian coordinate of an atom. If several atoms are selected, only relative changes are allowed.

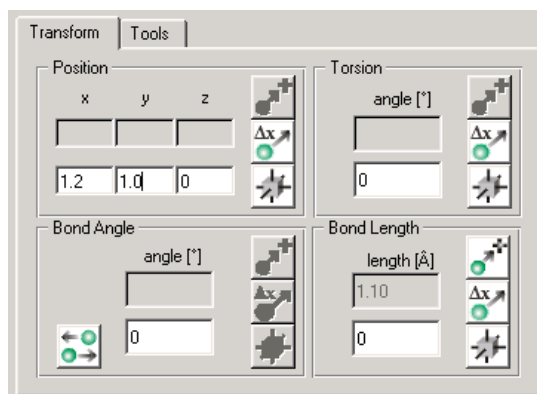


Figure 5.3: Transform Tab

- *Bond Length* is the distance between two selected atoms.
- *Bond Angle* is the planar angle between three selected atoms.
- *Bond Torsion* is the dihedral angle between four atoms.

Coordinates can be set absolute or relative to their current value. In each section, the upper row (with gray background) displays the current absolute value. In the lower row (with white background) you can enter a new value. In the right part of each section are three buttons. The first button transforms the coordinate while assuming that the entered value is an absolute value. The second button simply applies the transformation with the entered value text as a relative difference. The third button will activate a dragger in the viewer for adjusting the coordinate interactively. Figure 5.4 shows the bond torsion dragger as an example. The *Measurement* module can be used to display currently edited bonds or angles.

Which buttons can be used and which are disabled depends on the number of currently selected atoms.

An additional toggle button can be found in the *Bond Angle* section. If this toggle is activated, the interactive adjustment with the dragger will cause both bonds to be bent symmetrically, otherwise only one bond will be bent.

The position dragger can be moved on the plane determined by the face of the cubic dragger you click on. You can additionally rotate the selected groups by dragging the green knobs.

Tools Tab

The tools tab is subdivided into two sections:

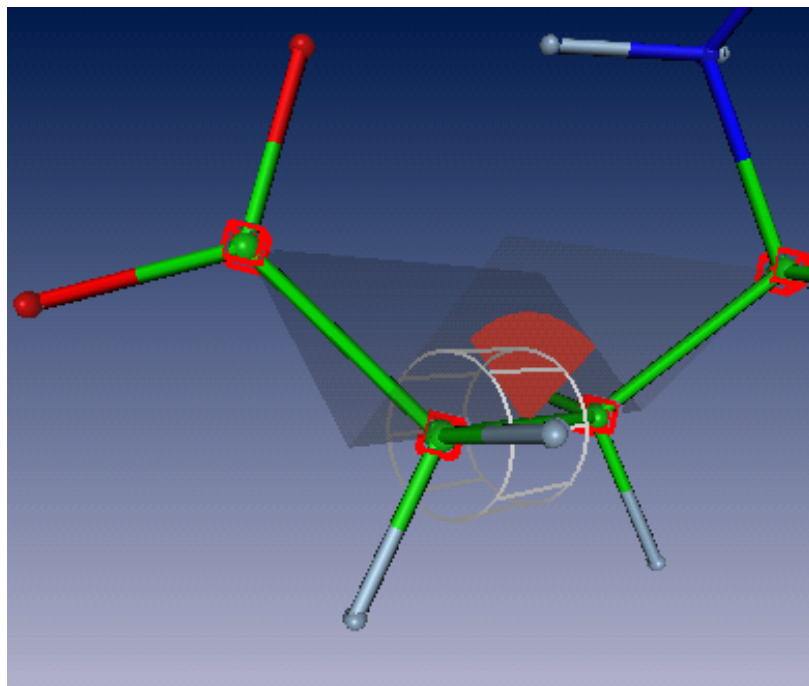


Figure 5.4: Editing a dihedral while viewing the angle with the measurement module

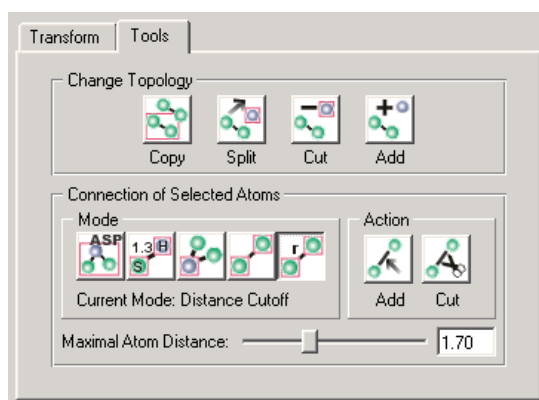


Figure 5.5: Tools Tab

Change Topology

The *cut* button will cut the currently selected groups out of the molecule. The *split* button will do the same but will copy the groups into a new molecule which will be added in the object pool. The *copy* button will leave the current molecule unchanged while copying the selected groups into a new molecule which will be added to the object pool. When pressing the *add* button, a window will open which will let you choose another molecule in the object pool whose groups you want to add to the current molecule.

Connection

The *Connection* section offers different options for influencing the bonding of the currently selected atoms. On the right side of the interface are buttons for adding or removing bonds between the currently selected atoms. The set of bonds which will be added or removed will depend on the connection mode that you can choose on the left side of the interface.

- *Standard*: This is the most reliable method for adding bonds to a protein or DNA/RNA. It will look up all residues in a residue database and add bonds accordingly. This method will only work for molecules that contain the residue type attribute. For connections between different residues it will check all residues on a chain sequentially. When used together with the *cut* action all bonds will be removed which can not be found in the database.
- *Bond length table*: This option lets you add bonds between selected atoms by looking up their bond lengths in the file *bondLengths.cfg* which can be found and edited in your local *amira/share/molecules* directory. If the distance of two atoms does not deviate further than a certain threshold from the bond length between the respective elements in the table, the bond will be added. This method is able to distinguish between single, double, triple, and aromatic bonds. It should be the first choice for non-standard residues or molecules not containing residue information. Just like the *Standard* option this mode can also be used together with the *cut* action. In this case all bonds which deviate too strongly from the bond length table will be removed.
- *External*: This mode can only be used together with the *cut* action. It will remove all bonds between selected and unselected atoms thus enabling you to disconnect certain parts of the molecule from the rest.
- *All*: This mode will add or remove all possible bonds between the selected atoms.
- *Distance Cutoff*: This last mode uses a distance cutoff for deciding which bonds to add or remove. The *Maximal Atom Distance* slider determines the maximal distance of two bonded atoms in Å. When using this mode together with the *add* action all bonds between atoms which are nearer than the cutoff distance will be added. Equally, the *cut* action will remove all bonds whose length is greater than the threshold.

Button Group

The *OK* button will close the editor and accept all changes made to the molecule. The *Cancel* button lets you return from the editor restoring the original state. Resetting the molecule without canceling the editor can be done with the *Reset* button. The *Apply* button will apply the changes you have made to the molecule object. This means that any further reset will return the molecule to the current state. With the *Undo* button any transformation can be undone up to 10 levels back from the current state.

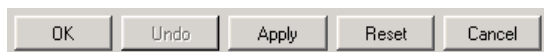


Figure 5.6: Button Group of the Molecule Editor

Part II

amiraDeconv

Chapter 6

amiraDeconv Introduction

The amira Deconvolution Package provides powerful algorithms for improving the quality of microscopic images recorded by 3D widefield and confocal microscopes. Two different methods are supported, namely a so-called non-blind and a blind deconvolution method, both based on iterative maximum-likelihood image restoration. In the first case a measured or computed point spread function (PSF) is required. In the second case the PSF is estimated along with the data itself.

The documentation of the amira Deconvolution Package is separated into the following parts:

- *General remarks about image deconvolution*
- *Data acquisition and sampling rates*
- *Standard deconvolution tutorial*
- *Blind deconvolution tutorial*
- *Bead extraction tutorial*
- *Performance issues and multi-processing*

The following modules are part of the amira Deconvolution Package:

- *BeadExtract* - obtain a PSF from a bead measurement
- *Convolution* - convolve two 3D images
- *CorrectZDrop* - corrects attenuation in z-direction
- *DataPreprocess* - background and flatfield correction
- *Deconvolution* - the actual deconvolution front-end
- *FourierTransform* - computes FFT and power spectrum
- *PSFGenerate* - calculates a theoretical PSF

6.1 General remarks about image deconvolution

Deconvolution is a technique for removing out-of-focus light in a series of images recorded via optical sectioning microscopy. Intended to investigate 3D biological objects, optical sectioning microscopy works by creating multiple images (optical sections) of a fluorescing object, each with a different focus plane. However, besides the in-focus structures the images usually also contain out-of-focus light from other parts of the object, causing haze and severe axial blur. This is even the case for a confocal laser scanning microscope, where most of the out-of-focus light is removed from the image by a pinhole system. Mathematically, the image produced by any microscopic system can be described as the convolution of the ideal unblurred image of the specimen and the microscope's so-called point spread function (PSF), i.e., the image of an ideal point light source. With the inverse of this process, called deconvolution, a deblurred image of the specimen can be obtained, provided the point spread function is known or at least, can be estimated.

The *amira* deconvolution package mainly provides two variants of a powerful iterative maximum-likelihood image restoration algorithm, namely a non-blind one and a blind one. The difference between them is that in the first case a measured or computed point spread function is used, while in the second case the PSF is estimated along with the data itself. Maximum-likelihood image restoration can be considered as the de-facto standard for deconvolution of 3D optical sections. Although computationally quite expensive, the method is able to significantly enhance image quality. At the same time it is very robust and insensitive with respect to noise artifacts. However, it should be noted that, although rejecting most of the out-of-focus light, by no means all of it is rejected. Therefore, some noticeable haze remains in the images. Also, the images retain a substantial axial smearing in z-direction, which cannot be removed by any deconvolution algorithm.

At the first sight one may wonder why both a non-blind and a blind deconvolution algorithm are provided in *amira*, although blind deconvolution seems to be more general since the PSF is calculated automatically. One answer is that blind deconvolution is computationally even more expensive than non-blind iterative maximum-likelihood image restoration. The other answer is that in a blind deconvolution algorithm a meaningful estimate of the PSF can only be computed if severe constraints are imposed. For example, a trivial solution of the blind deconvolution problem would be a deblurred image which is identical to the input image and a PSF with the shape of an ideal delta peak. Obviously, this solution isn't useful at all. Therefore, if for example confocal data is to be deconvolved the algorithm fits the actual PSF in such a way that it looks like a possible measured PSF of a confocal microscope. More precisely, the fit is constraint to be in agreement with the experimental parameters (the refractive index of the medium, the numerical aperture of the objective and the voxel-sizes). Sometimes, this can lead to wrong results, for example when the confocal pinhole aperture of the microscope wasn't stopped down sufficiently during confocal image acquisition, in which case the microscope actually didn't behave like a true confocal microscope. As a matter of fact you should try which approach provides the best results for your own image data, blind deconvolution or non-blind deconvolution with either a measured or an automatically computed PSF.

6.2 Data aquisition and sampling rates

In order to obtain best quality when deconvolving microscopic images some fundamental guidelines should be obeyed during image aquisition. Good results may be obtained even if some of these guidelines are not followed exactly, but in general the chances to get satisfactory results improve if they are. Below we discuss the most important recommendations.

Adjusting the scanned image volume

The region of interest should be centered in the middle of the image volume, as the optics of the microscope has usually the least aberrations in this region and it helps to avoid possible boundary artifacts, which can arise during the deconvolution procedure. Especially for widefield data it is important to record a sufficiently large (preferably empty) region below and above the actual sample. Ideally, this region should be as large as the sample itself. For example, if the sample covers 100 micrometer in z-direction, the scanned image volume should range from 50 micrometer below the sample to 50 micrometer above it.

Choosing the right sampling rate

The sampling rate is determined by the pixel sizes in x and y direction as well as the distance between two subsequent optical sections, both measured in micrometers. Generally speaking, image deconvolution works best if the data is apparently oversampled, i.e., if the pixel or optical section spacing is smaller than required. The maximal required sampling distance (Nyquist sampling) to avoid ambiguities in the data can be obtained from considerations in Fourier-space yielding

$$d_{xy} = \frac{\lambda}{4 NA},$$

where λ denotes the wavelength and NA is the numerical aperture of the microscope. Similar considerations yield for the maximal distance between adjacent imaged planes:

$$d_z = \frac{\lambda}{2 n (1 - \cos(\alpha))},$$

where n denotes the refractive index of the object medium and α the aperture half angle as determined by $NA = n \sin(\alpha)$.

For a confocal microscope both the in-plane sampling distance and the axial sampling distance need to be in theory approximately 2 times smaller. However, this requirement is far to strict for most practical cases and even in the widefield case, approximately fulfilling the above requirements is often sufficient.

The total number of optical sections is obtained by dividing the height of the image volume by the sampling distance d_z . It should be mentioned that deconvolution also works if the sampling distances

are not matched rigorously, but matching them improves the chances to get good results. In general oversampling the object is less harmful than undersampling it, with one exception: In case of confocal data the sampling distance d_{xy} should not be much smaller than indicated, if the blind deconvolution algorithm or the non-blind deconvolution algorithm together with a theoretically computed confocal PSF are used. Otherwise the unconstrained Maximum Likelihood algorithm and the predominant noise in the data might lead to unsatisfactory results.

Black level and saturation

Before grabbing images from the microscope's camera the light level should be adjusted in such a way that saturated pixels, either black or white ones, are avoided. Saturated pixels are pixels which are clamped to either black or white, because their actual intensity values are outside the range of representable intensities. In any case, saturation means a loss of information and thus prevents proper post-processing or deconvolution. At the same time an extensively high blackground level should be avoided, since it decreases the dynamic range of the imaging system and the deconvolutions perform worse. This means that empty regions not showing any fluorescence should appear almost black. A background level close to zero is especially important when bead measurements are performed in order to extract an experimental point spread function. Details are discussed in a separate tutorial about *bead extraction*.

6.3 Standard Deconvolution Tutorial

This tutorial explains how 3D image data sets can be deconvolved in *amira*. It is assumed that the reader is already familiar with the basic concepts of *amira* itself. If this is not the case, it is strongly recommended to work through the *standard amira tutorials* first. In this section the following topics are covered:

1. Prerequisites for deconvolution
2. Resampling a measured PSF
3. Deconvolving an image data set
4. Calculating a theoretical PSF

As an example we are going to use a confocal test data set (*polytrichum.am*) provided with the *amira* deconvolution package. The data file is located in the directory *Amira-3.0/data/deconv*.

- Load the data set *polytrichum.am*.
- Visualize it for example using a *ProjectionView* module.

The data set shows four chloroplasts in a spore of the moss *polytrichum commune*.

Prerequisites for deconvolution

Besides the image data itself for the standard non-blind deconvolution algorithm also a so-called *point spread function* (PSF) is required. The PSF is the image of a single point source, or approximately, the image of a single fluorescing sub-resolution sphere. PSF images can either be computed from theory (see below) or they can be obtained from measurements. In the latter case tiny so-called *beads* are recorded under the same conditions as the actual object. This means that the same objective lens, the same dye and wavelength, and the same immersion medium are used. Typically, the images of multiple beads are averaged to obtain an estimate of a single PSF. *amira* provides a module called *BeadExtract* facilitating this process. The use of this module is discussed in a *separate tutorial* about bead extraction. At this point let us simply load a measured PSF from a file.

- Load the data set *polytrichum-psf.am*.
- Use the *ProjectionView* module to visualize it.

The PSF appears as a bright spot located in the middle of the image volume (Figure 6.1). It is important that the PSF is exactly centered. Otherwise, the deconvolved data set will be shifted with respect to the original image. Also, it is important that the PSF fades out to black at the boundaries. If this is not the case, the black level of the PSF image needs to be adjusted using the *Arithmetic* module. Finally, neither the PSF nor the image to be deconvolved should exhibit *intensity attenuation artifacts*, i.e., image slices with decreased average intensity due to excessive light absorption in other slices. If such artifacts are present, they can be removed using the *CorrectZDrop* module.

Resampling a measured PSF

Next, select both, the PSF and the image data. You'll notice that the voxel sizes of both objects are not the same. It is recommended to adjust different voxel sizes of PSF and image data prior to deconvolution using the *Resample* module. The deconvolution module itself also accounts for different voxel sizes, but it does so by using point sampling with trilinear interpolation. This is ok as long as the voxel size of the PSF is larger than that of the image data. However, in our case the voxel size of the PSF is smaller than that of the image data, i.e., the resolution of the PSF higher. Using the *Resample* module provides slightly more accurate results here, since all samples will be filtered correctly using a Lanczos kernel.

- Connect a *Resample* module to *polytrichum-psf.am*.
- Connect the *Reference* port of the *Resample* module to the image data set *polytrichum.am*
- In the *Mode* port of the *Resample* module, choose *voxel size* (see Figure 6.2).
- Resample the PSF by pressing the *DoIt* button.

The *voxel size* option means that the PSF will be resampled on a grid with exactly the same voxel size as the image data set, which is connected to the *Reference* port. While the original PSF had a resolution of 12 x 12 x 30 voxels, the resampled one only has 12 x 12 x 16 voxels. However, the extent

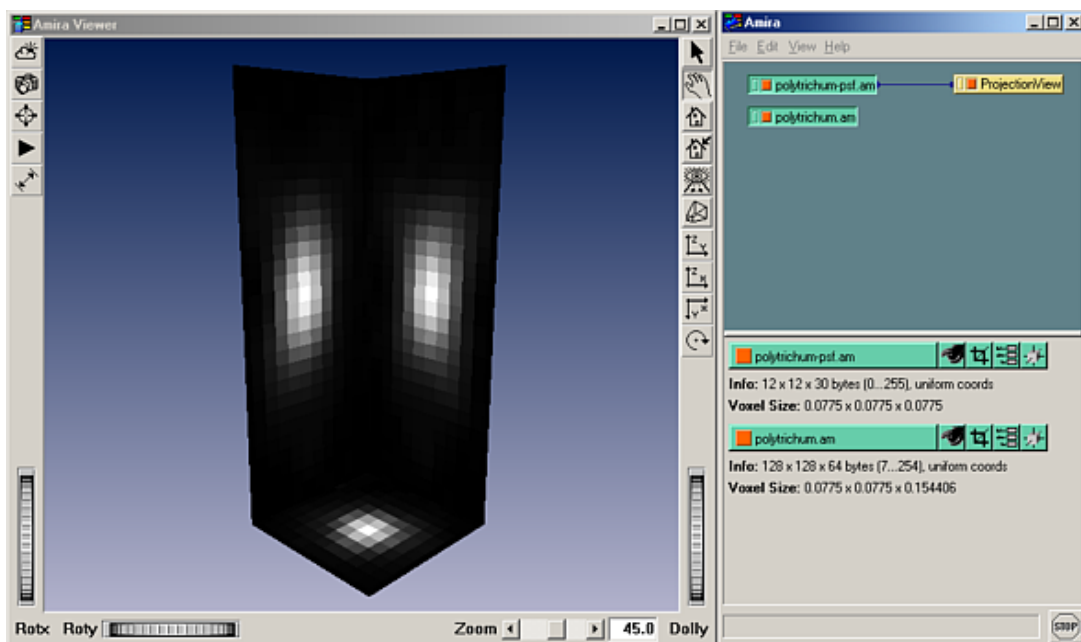


Figure 6.1: Maximum intensity projection of *polytrichum-psf.am*

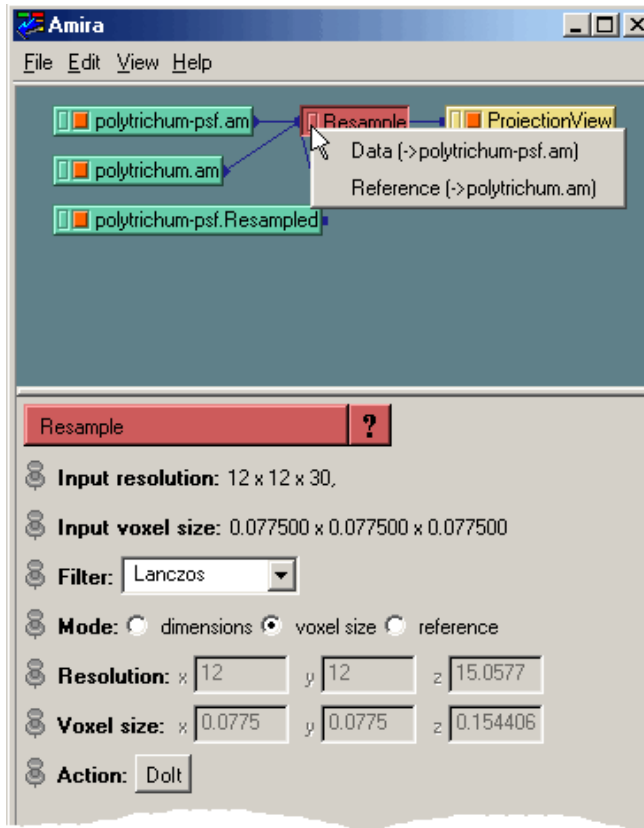


Figure 6.2: Resampling a PSF using the *Resample* module.

of a single voxel in z-direction is bigger now.

Deconvolving an image data set

After a suitable PSF has been obtained we are ready for deconvolving the image data set. This can be done by attaching a *Deconvolution* module to the image data.

- Connect a *Deconvolution* module to *polytrichum.am*.
- Connect the *Kernel* port of the *Deconvolution* module to the resampled PSF *polytrichum-psf.Resampled*.

Once the deconvolution module is connected to its two input objects, some additional parameters need to be adjusted (for a detailed discussion of these parameters see also the *reference documentation* of the *Deconvolution* module itself). Figure 6.3 shows these settings:

Border width: For deconvolution the image data has to be enlarged by a guardband region. Otherwise boundary artifacts can occur, i.e., information from one side of the data can be passed to the other. There is no need to make the border bigger than the size of the PSF. However, if the data set is dark at the boundaries a smaller border width is sufficient. In our case, let us choose the border values 0, 0, and 8 in the x, y, and z direction.

Iterations: The number of iterations of the deconvolution algorithm. Let us choose a value of 20 here.

Initial estimate: Specifies the initial estimate of the deconvolution algorithm. If *const* is chosen a constant image is used initially. This is the most robust choice, yielding good results even if the input data is very noisy. We keep this option here.

Overrelaxation: Overrelaxation is a technique to speed up the convergence of the iterative deconvolution process. In most cases the best compromise between speed and quality is *fixed* overrelaxation. Therefore we keep this choice also.

Method: Selects between standard (non/blind) and blind deconvolution. Let's specify *standard* here.

The actual deconvolution process is started by pressing the *DoIt* button. Please press this button now. The deconvolution should take about 1-2 minutes on a modern computer. During the deconvolution the progress bar informs you about the status of the operation. Also, after every iteration a message is printed in the *amira* console window indicating the amount of change of the data. If the change seems to be small enough you can terminate the deconvolution procedure by pressing the stop button. However, note that the stop button is evaluated only once between two consecutive iterations.

When deconvolution is finished a new data set called *polytrichum.deconv* appears in the object pool. You might take a look at the deconvolved data by moving the *ProjectionView* connection line from *polytrichum.am* to *polytrichum.deconv*.

Calculating a theoretical PSF

Sometimes bead measurements are difficult to perform, so that an experimental PSF cannot easily be obtained. In such cases a theoretical PSF can be used instead. *amira* provides the module *PSFGenerate*, allowing you to calculate theoretical PSFs. The module can be created by selecting *PSFGenerate* from the *Edit Create* menu of the *amira* main window.

Once the module is created again some parameters have to be entered. The *resolution* and the *voxel size* can be most easily specified by connecting the *Data* port of the *PSGGenerate* module to the image data set to be convolved. In our case please connect this port to *polytrichum.am*.

In order to generate a PSF you also need to know the numerical aperture of the microscope objective, the wavelength of the emitted light (to be entered in micrometers!), and the refractive index of the immersion medium. In our test example these values are NA=1.4, lambda=0.58, and n=1.516 (oil medium). Also, change the microscopic mode from widefield to confocal.

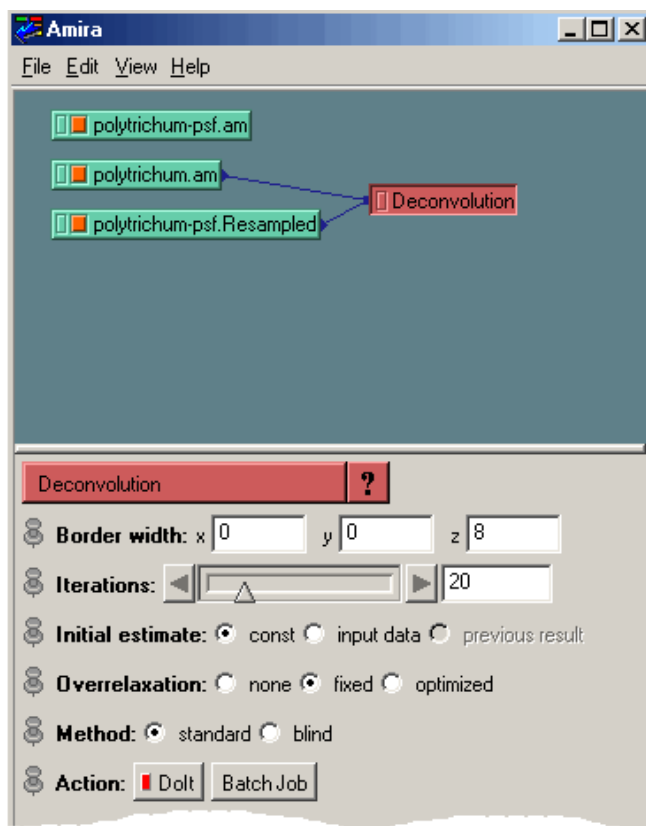


Figure 6.3: Deconvolution module attached to *polytrichum.am*.

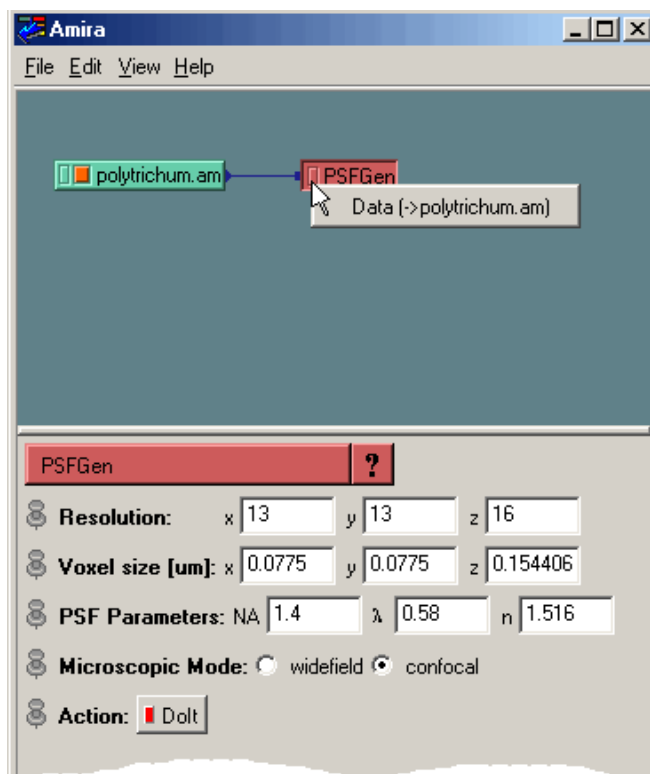


Figure 6.4: The PSFGen module calculates theoretical PSFs.

After pressing the *DoIt* button the computed PSF appears as an icon labelled *PSF* in the object pool. You can compare the theoretical PSF with the measured one using the *OrthoSlice* module. You'll notice that the measured PSF appears to be slightly wider. This is a common observation in many experiments.

Once you have computed a theoretical PSF you can perform non-blind deconvolution as described above. However, for convenience the *Deconvolution* module is also able to compute a theoretical PSF by itself. You can check this by disconnecting the *Kernel* port of the *Deconvolution* module. If no input is present at this port, additional input fields are shown, allowing you to enter the same parameters numerical aperture, wavelength, refractive index, and microscopic mode as in *PSFGenerate*. After these parameters have been entered the deconvolution process again can be started by pressing the *DoIt* button.

Note, that any previous result connected to the *Deconvolution* module will be overwritten when starting the deconvolution process again. Therefore, be sure to disconnect a previous result if you want to

compare deconvolution with different input PSFs.

6.4 Blind Deconvolution Tutorial

This tutorial explains how blind deconvolution can be performed in *amira*. At the same time it describes how deconvolution jobs can be processed using the *amira* job queue. Like in the previous tutorial it is assumed that the reader is already familiar with the basic concepts of *amira* itself. If not, we recommend to work through the *standard amira tutorials* first.

A blind deconvolution example

Let us start by loading a raw image data set first.

- Load the file *alphalobe.am* from the directory *Amira-3.0/data/deconv*.
- Visualize the data set by attaching a *ProjectionView* module to it.

The data set has been recorded using a widefield microscope. It shows a stained neuron, the alpha-neuron, from a part of the brain of a bee called the optical lobe. Compared to the confocal data set used in the standard deconvolution tutorial *alphalobe.am* is much bigger. It has a resolution of 248 x 248 x 256 voxels with a uniform voxel size of 1 micrometer. In the xy-plane of the projection view the structure of the neuron can be clearly identified. However, the contrast of the image is quite poor because there is a significant amount of out-of-focus light or haze present. With *amira*'s blind deconvolution algorithm we can enhance the image data without needing to know an explicit PSF in advance.

- Attach a deconvolution module to *alphalobe.am*.
- Adjust the parameters like shown in Figure 6.5.

The individual parameters have the following meaning:

Border width: Like for standard non-blind deconvolution the image data has to be enlarged by a guardband region. Otherwise boundary artifacts can occur, i.e., information from one side of the data can be passed to the other. In our case we only provide a small guardband region of 8 voxels in x- and y-direction. In z-direction we don't provide any border because there are sufficiently many empty slices below and above the actual neuron. The resulting size of the data arrays on which the computations are performed then is 256 x 256 x 256, which is nice because it is a power of two and thus the fast Fourier-transforms, the computationally most expensive part of the deconvolution algorithm, can be executed somewhat faster.

Iterations: We choose a value of 25 here. Depending on the data, usually at least 10 iterations are required. With overrelaxation being enabled (see below), results usually don't improve much after 40 iterations.

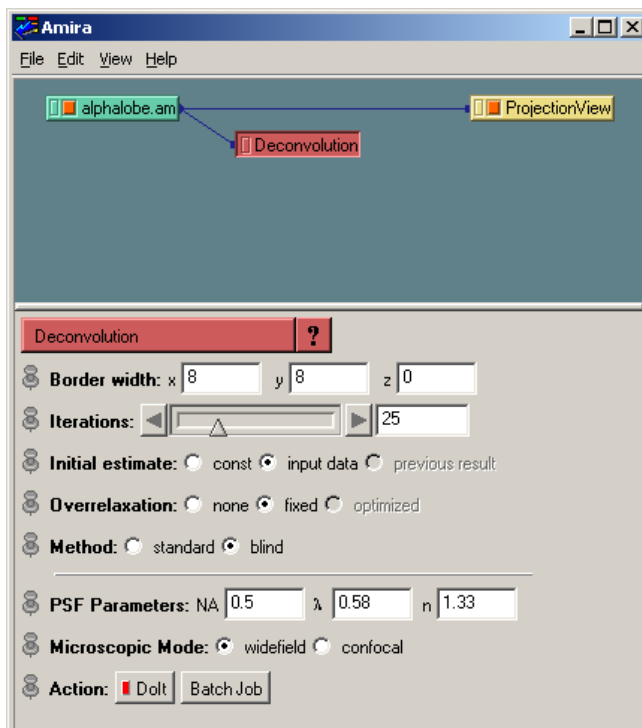


Figure 6.5: Parameters for blind deconvolution.

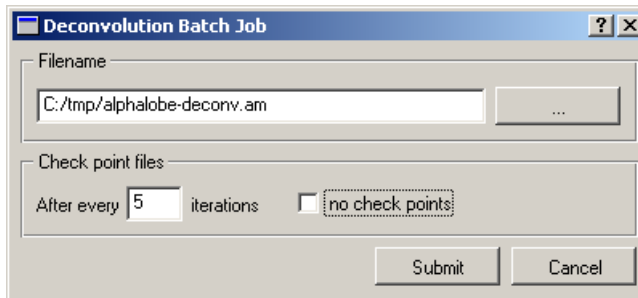


Figure 6.6: Dialog for submitting a deconvolution job.

Initial estimate: Specifies the initial estimate of the deconvolution algorithm. Since there is not much noise present in the original alphalobe images it is safe to chose *input data* here. This causes the algorithm to converge even faster.

Overrelaxation: Overrelaxation is a technique to speed up the convergence of the iterative deconvolution process. We enable overrelaxation by chosing the *fixed* toggle.

Method: We chose *blind* here in order to select the blind deconvolution algorithm.

PSF Parameters: For *alphalobe.am* the numerical aperture is 0.5, the wavelength is 0.58 micrometers, and the refractive index is 1.33 (water). These parameters are required in order to apply certain constraints to the estimated point spread function. They are also used in order to compute an initial PSF. If a data set would be connected to the *Kernel* port of the deconvolution module, this data set would be used as the inital PSF with the given PSF parameters still acting as constraints. For example, you could provide a measured PSF and let it be fitted to the actual data by the deconvolution algorithm.

Microscopic mode: *alphalobe.am* is a widefield data set, so select this option here.

Submitting a deconvolution job

After all parameters have been entered, the deconvolution process can be started. On a modern computer blind deconvolution of our test data set roughly takes about 20 minutes. Especially, if you want to deconvolve multiple data sets at once it is inconvenient to do this in an interactive session. Therefore multiple deconvolution jobs can be submitted to the *amira* job queue and then, for example, processed over night. This works as follows:

- Press the *Batch Job* button of the *Action* port. A dialog as shown in Figure 6.6 pops up.
- In the dialog choose a file name under which you want to save the deconvolved data set, e.g. `C:/Temp/alphalobe-deconv.am`.
- Modify the text field, so that check point files are written after every 5 iterations.

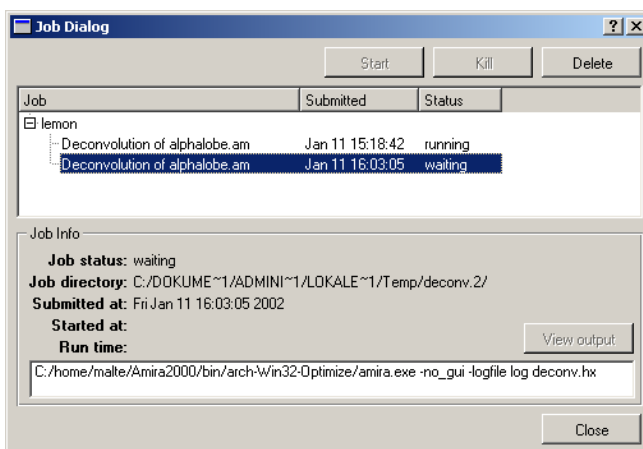


Figure 6.7: The amira job dialog showing a pending deconvolution job.

Check point files are used to store intermediate results. With the above settings the deconvolved data is written into a file after every 5 iterations. Check point files are called like the final result, but a consecutive number is inserted just before the file name suffix. For example, if the result file name is `C:/Temp/alpha-lobes-deconv.am` the check point files are called `C:/Temp/alpha-lobes-deconv-0005.am`, `C:/Temp/alpha-lobes-deconv-0010.am` and so on. Now we are ready to actually submit the batch job.

- Press the *Submit* button of the deconvolution dialog. After a few seconds the amira batch job dialog appears, compare Figure 6.7.
- Select the deconvolution job and press the *Start* button.

You now have to wait about 20 minutes until the deconvolution job is finished. Once the job queue has been started you can quit *amira*. The batch jobs will be continued automatically. If *amira* is still running when the deconvolution job exits then the result will be loaded automatically in *amira*. Otherwise you have to restart *amira* and load the deconvolved data set manually.

6.5 Bead Extraction Tutorial

Non-blind deconvolution is a powerful and robust method for enhancing the quality of 3D microscopic images. However, the method requires that the image of the point spread function (PSF) responsible for image blurring is provided. As stated in the *standard deconvolution tutorial*, the PSF can either be calculated theoretically or it can be obtained from a bead measurement. *amira* provides a special-purpose module called *BeadExtract* which facilitates the extraction of PSF images from one or multiple bead measurements. In this tutorial the use of the module shall be explained. The following topics are

covered:

1. Bead measurements
2. Projection View and Projection View Cursor
3. Resampling and averaging the beads

Bead measurements

The PSF is the image of a single point source recorded under the same conditions as the actual specimen. It can be approximated by the image of a fluorescing sub-resolution microsphere, a so-called bead. Performing good bead measurements requires some practice and expertise. In order to obtain good results the following hints should be obeyed:

1. Use the right beads. It is important that the bead size is smaller than approximately $1/2$ full width at half maximum (FWHM) of the PSF. Good sources for obtaining beads suitable for PSF measurements are Molecular Probes (<http://www.probes.com/>) or Polysciences (<http://www.polysciences.com/>).
2. The beads must be solid. Besides solid beads there are also beads with the shape of a spherical shell, allowing to check the focus plane of a microscope. Such beads cannot be used as a source for PSF generation in the current version of amira.
3. Don't record clusters of multiple beads. Sometimes multiple beads may glue together, appearing as a single big bright spot. Computing a PSF from such a spot obviously leads to wrong results.
4. Note that beads are not resistant to a variety of embedding media. In particular beads will be destroyed in xylene based embedding media such as Permount (Fisher Scientific) and methyl salicylate (frequently used to clear up the tissue). As a substitute you might use immersion oil instead which has a similar refractive index as e.g. methyl salicylate.
5. The sample and the beads should always be imaged as close to the coverslip as possible. When it is not possible to attach the sample to the coverslip, the beads should also be imaged in a comparable depth, embedded in the same mounting medium. Imaging the beads with better quality than the sample will yield a slightly blurred deconvolution result. However, when the PSF used for deconvolution is too wide, artefacts can arise during deconvolution.

The objective lense should always be selected according to the mounting medium, i.e. if the sample is attached to the slide and embedded in a buffer of refractive index close to water, a severe loss of image quality can be expected when using an oil-immersion objective without a correction collar. Deconvolution of properly imaged data will always be superior to deconvolution of data suffering from aberrations.

6. Problems occur if the mounting medium remains liquid. In this case the sample distribution may not be permanent. If your specimen is to be embedded in water you can try to immerse the beads in an agarose gel of moderate concentration instead. Attaching the small beads to the coverslip (for example by letting them dry) is often also sufficient for immobilisation.

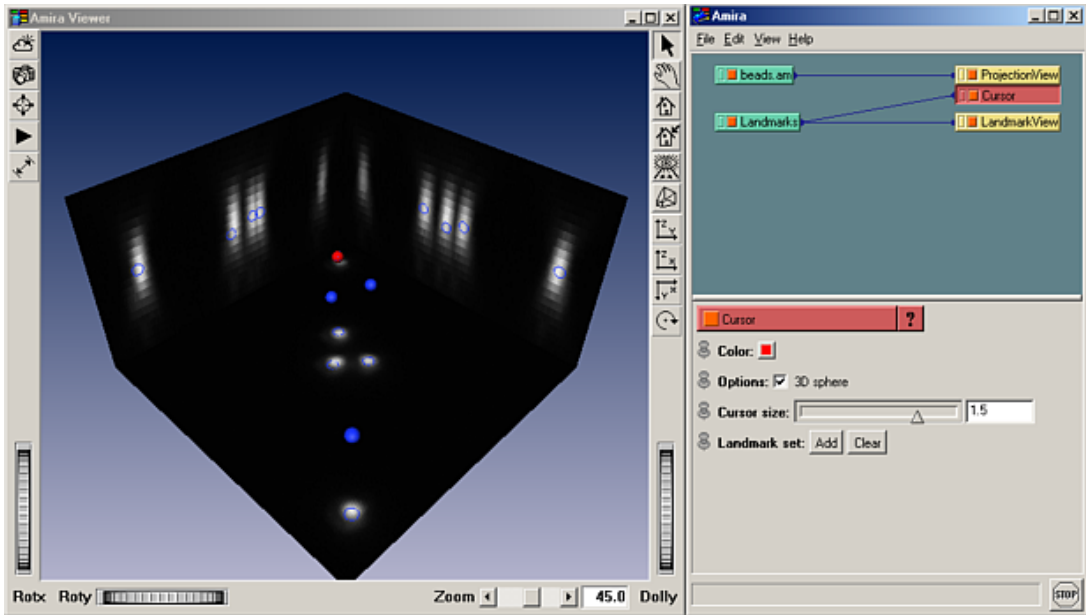


Figure 6.8: Individual beads can be interactively identified using a *Cursor* module.

An example of an image data set containing multiple beads is provided in the file *beads.am* in the directory *Amira-3.0/data/deconv*.

- Load the data set *beads.am*.
- Visualize the data using a *ProjectionView* module.

Projection View and Projection View Cursor

The bead data set contains five different beads which can be clearly seen in the three orthogonal planes of the *ProjectionView* module. In order to obtain a single PSF we first want to select several “good” beads. These beads are then resampled and averaged, thus yielding the final PSF. A bead can be considered as “good” if it is clearly visible and if it is not superimposed by other beads (even when defocused),

Selecting “good” beads is an interactive process. It is most easily accomplished using the *ProjectionView*’s *Cursor* module. This module allows you to select a point in 3D space by clicking on one of the three planes of the *ProjectionView* module. The third coordinate is automatically set by looking for the voxel with the highest intensity. Points selected with the *Cursor* module can be stored in a *LandmarkSet* data object.

- Attach a *Cursor* module to the *ProjectionView* module.
- Click on any bead on one of the three planes.
- Store the current cursor position in a *LandmarkSet* object by pressing the *Add* button.
- Select and add some other beads too.

The landmarks need not to be located exactly at the center of a bead. The exact center positions can be fitted automatically later on.

You can remove incorrect bead positions from the landmark set by invoking the landmark set editor. In order to activate the editor select the landmark set object and press the green button with the disk-shaped symbol. If you want to add additional bead positions to an existing landmark set object, make sure that the *master port* of the landmark set object is connected to the *Cursor* module. Otherwise, a new landmark set object will be created.

Resampling and averaging the beads

Now we are ready to extract and average the individual beads. This is done by means of the *BeadExtract* module.

- Connect a *BeadExtract* module to the *Landmarks* object.
- Make sure, that the *Data* port of *BeadExtract* is connected to the bead data set *beads.am*. If the landmarks are still connected to the beads via the *Cursor* and *ProjectionView* modules, the connection is established automatically.

The *BeadExtract* module provides two buttons called *Adjust centers* and *Estimate size*, which should be invoked in a preprocessing step before the beads are actually extracted.

The first button (*Adjust centers*) modifies the landmark positions so that they are precisely located in the center of gravity of the individual beads.

The second button (*Estimate size*) computes an estimate for the number of voxels of the PSF image to be generated. This button is only active if no PSF image is connected as a result to *BeadExtract*. If there is a result object, the resolution and voxel sizes of the result are used and the port becomes insensitive.

Any of the actions of the two preprocessing buttons can be undone using the *Undo* button. This can be necessary for example if two beads are too close so that no correct center position could be computed. In general, overlaps between neighbouring beads should be avoided. Small overlaps might be tolerated because during resampling intensities are weighted according to the influence of surrounding beads.

- Perform the preprocessing steps *Adjust centers* and *Estimate size*.
- Compute a resampled and averaged PSF by pressing the *Exzract beads* button.

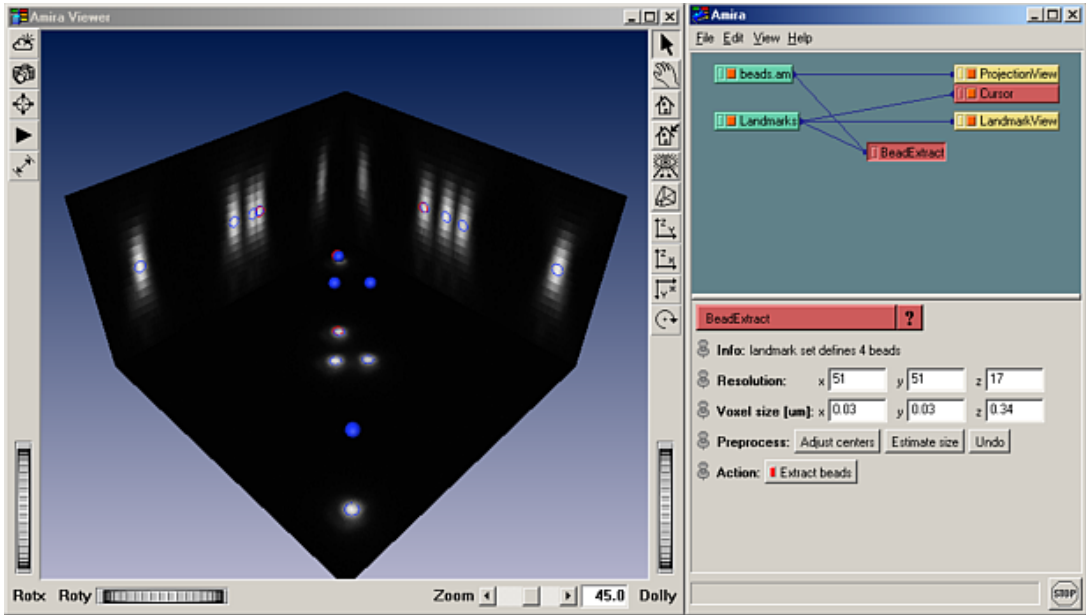


Figure 6.9: The *BeadExtract* module resamples and averages multiple beads.

The data type of the resulting PSF will be *float*, irrespectively of the data type of the input image. The individual beads will be weighted on a per-voxel basis and added to the result. No normalization will be performed afterwards. You may investigate the resulting PSF image using any of the standard visualization modules. In Figure 6.10 a volume rendering of the resulting PSF is shown.

In some cases you may want to average multiple beads recorded in different input data sets. This can be easily achieved by creating a *Landmarks* object for each input data set. For the first input data set extract the beads as described above. For the other input data set also use the *BeadExtract* module. However, make sure that the PSF obtained from the first input data set is connected as a result object to *BeadExtract* before pressing the *DoIt* button. In order to use an existing PSF as a result object connect the *Master* port of the PSF to *BeadExtract* (once this is done the *Resolution* and *Voxel size* ports of *BeadExtract* become insensitive, see above). If an existing result is used new beads simply will be added into the existing data set. Therefore datasets should be scaled in intensity according to their quality prior to bead extraction and summation to obtain a suitable weighting of the individual extracted beads in the final result.

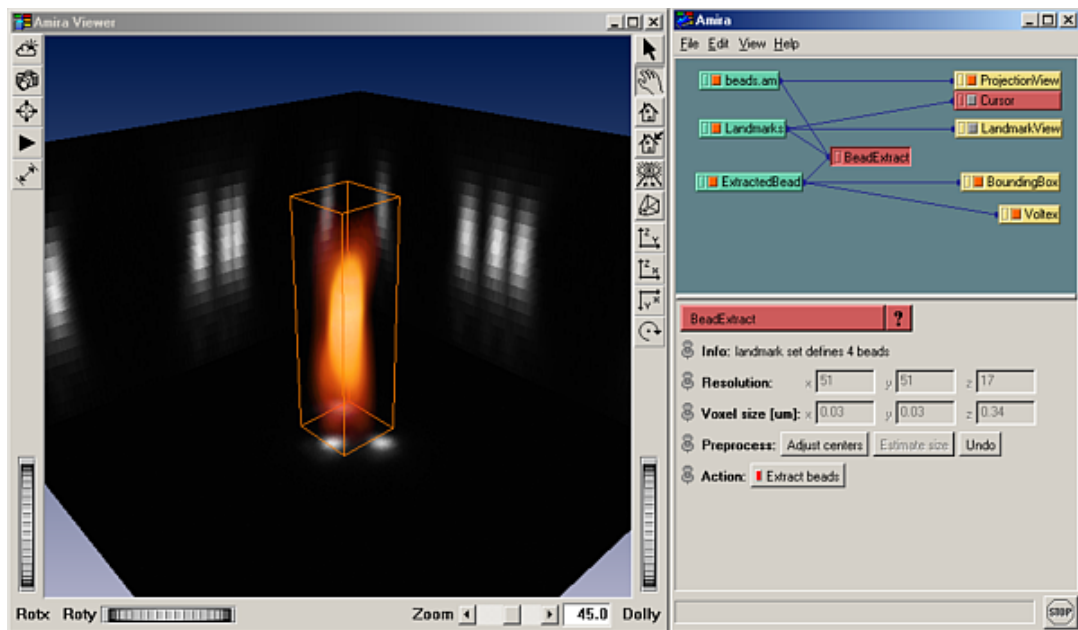


Figure 6.10: The final PSF visualized using a *Vortex* module.

6.6 Performance issues and multi-processing

Iterative maximum-likelihood deconvolution essentially is the most powerful and most robust technique for the restoration of 3D optical sections. However, it is also computationally very demanding. It can take several minutes (sometime even hours) to process large 3D data sets. This is not due an improper implementation but due to the algorithm itself. Both, the blind and the non-blind variant of the method, rely heavily on fast Fourier-transforms in order to efficiently compute convolutions. If you want to improve performance, try to adjust the size of your data volumes so that the number of voxels plus the border width is a power of two. Sometimes, it is worth to enlarge the border width a little bit in order to get a power of two. Although the algorithm works with data of any size, powers of two can be transformed somewhat faster.

Another issue is memory consumption. Internally, several copies of the data set need to be allocated by the deconvolution algorithm. These copies should all fit into memory at the same time (a specific variant of the algorithm suitable for working under low memory conditions will be provided in a later version). Besides the input data itself the following number of working arrays are required by the different methods:

- 3 working arrays for the non-blind algorithm with no or with fixed overrelaxation
- 5 working arrays for the non-blind algorithm with optimized overrelaxation
- 5 working arrays for the blind algorithm

The number of voxels of a working array is the product of the number of voxels of the input data set plus the border width along each spatial dimension. The primitive data type of a working array is a 4-byte floating point number. For example, if the number of voxels of the input data set plus the border width is $256 \times 256 \times 256$ (as for the *alphalobe.am* data set in the blind deconvolution tutorial), each working array will be about 64 MB, irrespectively of the primitive data type of the input data set. Therefore at least 192 MB ($3 \times 4 \times 256 \times 256 \times 256$ bytes) are required for non-blind deconvolution with fixed overrelaxation, and 320 MB ($5 \times 4 \times 256 \times 256 \times 256$ bytes) for blind deconvolution. Keep this in mind when configuring the computer on which to perform deconvolution! However, also note that for most platforms it usually doesn't make sense to have more than 1.5 GB of main memory. For more memory a 64-bit operating system is required. On a Windows PC, it is strongly recommended not to use Windows 98 or Windows ME if you have more than 512 MB of main memory.

Finally, it should be mentioned that the deconvolution algorithm can make use of a multi-processor CPU board. Although you don't get twice the performance on a dual-processor PC, a speed-up of almost 1.5 can be achieved. On default, *amira* uses as many processors as there are on the computer. If for some reason you want to use less processors you can set the environment variable `AMIRA_DECONV_NUM_THREADS` to the number of processors you actually want to use simultaneously.

Chapter 7

Module Reference

7.1 BeadExtract

This module is part of the Amira deconvolution package. It can be used to resample and average the image of one or multiple beads, i.e., fluorescing sub-resolution microspheres, thereby obtaining an approximation of a point spread function (PSF) required for non-blind deconvolution.

The module must be connected to the image data set containing the measured beads as well as to a landmark set indicating the center positions of all beads to be resampled and averaged. The whole process of obtaining a point spread function from a bead measurement is described in a separate *tutorial on bead extraction*. Please refer to this tutorial for more information on how to use the module.

Connections

Data [required]

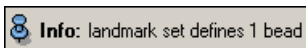
Connection to a uniform image data set containing measured beads.

Landmarks [required]

Connection to a landmark set indicating the center positions of the beads to be resampled and averaged.

Ports

Info



Displays the number of beads to be resampled and averaged.

Resolution

A control panel for the Resolution parameter. It features a blue circular icon with a white '8' on the left. To its right is the label 'Resolution:'. Further right are three input fields: 'x' with the value '33', 'y' with the value '33', and 'z' with the value '65'.

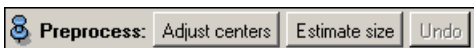
Specifies the number of voxels of the final PSF image to be generated. If a PSF image is connected as a result object to this module, the port becomes insensitive and the number of voxels of the result object are used.

Voxel size [um]

A control panel for the Voxel size parameter. It features a blue circular icon with a white '8' on the left. To its right is the label 'Voxel size [um]:'. Further right are three input fields: 'x' with the value '0.04', 'y' with the value '0.04', and 'z' with the value '0.04'.

Specifies the voxel size of the final PSF image to be generated in micrometers. If a PSF image is connected as a result to this module, the port becomes insensitive and the voxel size of the result object is used.

Preprocess

A control panel for the Preprocess section. It features a blue circular icon with a white '8' on the left. To its right is the label 'Preprocess:'. Further right are three buttons: 'Adjust centers', 'Estimate size', and 'Undo'.

Adjust centers: If this button is pressed the landmark positions of the data set connected to port *Landmarks* are shifted to the center of gravity of the corresponding bead. It is required that the initial landmark positions are inside the bead images and that neighbouring beads do not overlap significantly. Otherwise incorrect center positions may be computed.

Estimate size: If this button is pressed the desired number of voxels of the final PSF image specified in port *Resolution* are computed automatically. Before this is done the beads' center positions already should have been adjusted. The estimate is computed by determining the extent of the biggest spot around any landmark. Again, it is required that neighbouring beads do not overlap significantly. The *Estimate size* button becomes insensitive if a result object is connected to the module. In this case always the size of the existing PSF image will be used.

Undo: Undos the effect of any of the previous two buttons. For example, if wrong center positions have been computed after pressing *Adjust centers*, the original landmark positions can be restored using the *Undo* button.

Action

A control panel for the Action section. It features a blue circular icon with a white '8' on the left. To its right is the label 'Action:'. Further right is a button with a red square icon and the text 'Extract beads'.

Actually extracts the beads with the center of the resulting dataset corresponding to the current positions of the landmarks. The image volume around every landmark is extracted and resampled using a Lanczos filter (compare the *Resample* module). The resampled bead images are then added to the result object. The final PSF is not normalized.

7.2 Convolution

This module convolves two uniform 3D data objects with each other by Fourier transforming the two inputs, multiplying them, and then transforming them back. The module is part of the Amira

deconvolution package. It can be used for example to verify the results of image deconvolution.

The bounding box and voxel sizes of the input data set and the convolution kernel are ignored by this module. Use the *Resample* module to make sure that the resolution of both inputs is identical.

Connections

Data [required]

The data set to be convolved.

Kernel [required]

The convolution kernel.

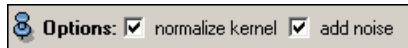
Ports

Border width

A port control for 'Border width' with a blue icon on the left. It contains three input fields: 'x' with value '12', 'y' with value '12', and 'z' with value '30'.

Defines the size of the border region. A border region is necessary if the input data set doesn't fade out to black at the boundaries.

Options

A port control for 'Options' with a blue icon on the left. It contains two checked checkboxes: 'normalize kernel' and 'add noise'.

If *normalize kernel* is selected, the integral of the convolution kernel (connected to port *Kernel*) will be normalized to one. If this is not the case the intensities of the convolved data set will be scaled by the actual integral.

If *apply noise* is selected, the values of the convolved data set will be multiplied by random numbers uniformly distributed around 1 (white noise).

Noise level

A port control for 'Noise level' with a blue icon on the left. It contains a slider and a numeric input field with the value '0'.

This port will only be shown if the *apply noise* option of the *Options* port has been selected. It specifies the amount of noise applied to the output, i.e., the range of the random numbers around 1, by which the result is multiplied.

Action

A port control for 'Action' with a blue icon on the left. It contains a red square icon and the text 'Dolt'.

Actually convolves the input data sets.

7.3 CorrectZDrop

This module lets you fix artifacts in 3D microscopic images caused by light absorption in other slices. If such artifacts are present, the average intensity in lower slices seems to be decreased. This so-called *z-drop* or *intensity attenuation* can be corrected automatically by fitting an exponential curve to the average intensities in each slices, or manually by providing a user-defined formula.

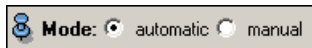
Connections

Data [required]

The image data exhibiting a z-drop artifact. Scalar fields with uniform or stacked coordinates as well as *multi-channel fields* are supported.

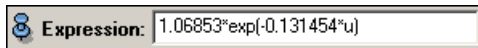
Ports

Mode



Lets you select between automatic mode and manual mode.

Expression



This port is only available if manual mode has been selected. It provides a text field where you can enter a formula specifying a factor used to multiply the intensity values in each slice. Within the formula the variable u specifies the slices. u will take the value 0 for the first slice and 1 for the last slice. For the other slices it takes intermediate values depending on the actual slice location (this makes support of stacked coordinates easy). In automatic mode the following formula $a * \exp(b * u)$ will be used, where a and b are fitted automatically. If you first perform an automatic z-drop correction and then switch to manual mode, the fitted exponential will be displayed in the port's text field.

Action



Starts z-drop correction.

7.4 DataPreprocess

This module is part of the Amira *deconvolution package*. It can be used to apply both a background and a flatfield correction to a raw 3D image stack.

For the background correction a single background image has to be provided at the *background* port of the module. The background image should be a nearly black image recorded with the camera's shutter

closed. This image is subtracted from all slices of the 3D input data set, thus compensating for any dark current of the camera's CCD detector.

For the flatfield correction a single flatfield image has to be provided at the *flatfield* port of the module. The flatfield image should be an unfocused almost white image taken from a drop of homogeneously fluorescing dye. The intensities of the 3D input image are then scaled according to the normalized intensities of the flatfield images. The input image gets brighter at pixels where the flatfield is dark and vice versa. In this way non-uniform sensitivity of the camera's CCD detector is compensated. If both a flatfield and a background image are present, the background is subtracted from the flatfield too.

Connections

Data [required]

The raw 3D image stack to be corrected. Any regular scalar field with uniform coordinates is supported.

Background [optional]

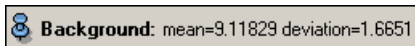
An optional 2D background image with the same number of voxels in x and y direction as the 3D input image. If an input is present at this port a background correction is performed (see above).

Flatfield [optional]

An optional 2D flatfield image with the same number of voxels in x and y direction as the input image. In an input is present at this port a flatfield correction is performed (see above).

Ports

Background



Displays the mean value and the standard deviation of the background image, if such an image is present. Only the first slice is considered.

Flatfield



Displays the mean value and the standard deviation of the flatfield image, if such an image is present. Only the first slice is considered.

Action



Starts the computations and produces a corrected output data set.

7.5 Deconvolution

This module is the front-end for deconvolving 3D microscopic images. Two different iterative maximum-likelihood image restoration algorithms are provided, a non-blind one and a blind one. For a general description of the deconvolution process please refer to the *tutorials* provided with the Amira deconvolution package.

The resulting deconvolved data set will be stored in the object pool. If no input PSF is specified or if the blind deconvolution algorithm has been selected, also the estimated PSF will be stored in the object pool.

Connections

Data [required]

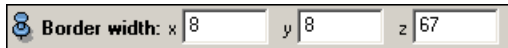
The data set to be deconvolved.

Kernel [optional]

The point spread function (PSF) to used for deconvolution. If no PSF is specified, an estimated PSF is calculated automatically based on the numerical aperture of the microscope, the wavelength of the emitted light, and the refractive index. If a blind deconvolution is to be performed, an input PSF (if connected) will be used as an initial estimate.

Ports

Border width

A control panel for border width. It features a blue icon on the left, followed by the text "Border width:". To the right are three input fields labeled "x", "y", and "z". The "x" field contains the value "8", the "y" field contains "8", and the "z" field contains "67".

Defines the size of the border region. A border region is necessary if the input data set doesn't fade out to black at the boundaries. For performance reasons it might advisable to choose values such that the sum of the size of the input data set and the border width results in a power of two. For example, if the data set consists of 118 slices, a border width of 10 slices in z direction would be a good choice.

In case of widefield data it is sometimes advisable to have the border width in z direction exactly as large as the data set itself. If this is the case, the border region will be initialized by mirroring the values from the actual data volume. Otherwise, the values of the first slice and of the last slice will be interpolated linearly.

Iterations

A control panel for iterations. It features a blue icon on the left, followed by the text "Iterations:". To the right is a slider control with a triangular knob in the center. Further right is a numeric input field containing the value "60".

Specifies the number of iterations of the deconvolution procedure.

Initial estimate

A control panel for the initial estimate. It features a blue icon on the left, followed by the text "Initial estimate:". To the right are three radio buttons. The first radio button is selected and is labeled "const". The second radio button is labeled "input data". The third radio button is labeled "previous result".

Specifies the initial estimate of the deconvolution algorithm. If *const* is chosen a constant image is used initially. Often, this yields smoother results than the second option, namely *input data*. The third option (*previous result*) is only available if the input data set has already been deconvolved previously. Use this option if you want to apply some additional deconvolution iterations.

Overrelaxation



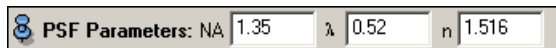
Overrelaxation is a technique to speed up the convergence of the iterative deconvolution process. In most cases *fixed* overrelaxation is a good choice. For non-blind deconvolution also an *optimized* overrelaxation technique is available. This method further accelerates convergence but is more memory and time consuming.

Method



This port specifies whether standard (non-blind) or blind deconvolution should be used.

PSF Parameters



Parameters for calculating the point spread function. This port will only be shown if standard (non-blind) deconvolution has been selected and no input PSF was specified, or if blind deconvolution has been selected, in which case these parameters act as constraints.

NA denotes the numerical aperture of the microscope. *lambda* denotes the wavelength of the emitted light in micrometers with the voxel sizes also being interpreted in micrometers. Finally, *n* denotes the refractive index of the specimen.

Microscopic Mode



Selects whether the input image has been recorded using a widefield microscope or using a confocal microscope. This is important for the PSF generation as well as for the selection of appropriate constraints during blind deconvolution.

Action



The *DoIt* button actually starts the deconvolution process. Since deconvolution is a time consuming operation, it optionally can be performed as a batch job. A batch job can be submitted using the *Batch job* button. If this button is pressed, first a dialog is popped up allowing you to specify the filename of the final deconvolved data set as well as the number of optional check point files (compare Figure 6.6). A check point stores an intermediate result obtained after a certain number of iterations have been performed. The actual deconvolution job is started via the Amira job dialog.

The job dialog is popped up automatically after the job has been finally submitted, but this may take a few seconds if there are currently no jobs running.

7.6 FourierTransform

This module compute a discrete forward or backward Fourier transform a scalar input data set with uniform coordinates. Alternatively, the power spectrum, i.e., the squared magnitude of the Fourier transform can be computed.

The origin of the input data set will be ignored by this module. Also, instead of being expressed in wave numbers the bounding box of a Fourier transformed data set will be the same as the bounding box of the input.

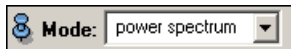
Connections

Data [required]

The input data to be Fourier transformed.

Ports

Mode



Option menu specifying the action to be performed.

If a real scalar field is connected to the module, three options are available, namely *forward*, *forward complex*, and *power spectrum*. If *forward* is selected, the output is stored in a special so-called half-complex format. Such an output can be back-transformed, but not many other operations can be performed on it. If *forward complex* is selected, the output will be a complex-valued scalar field with the same number of voxels as the input object. Finally, if *power-spectrum* is selected, the output is a real-valued scalar field with the same number of voxels as the input object containing the squared magnitude of the Fourier transform.

If a complex scalar field in half-complex format is connected, the only option is *backward*, indicating a backward Fourier-transform.

If an ordinary complex scalar field is connected to the module, the three options *forward*, *backward*, and *power spectrum* are available. The first two options specify a forward and backward Fourier transform, respectively. The output is a complex scalar field with the same resolution as the input. If *power-spectrum* is selected, the output is a real-valued scalar field with the same number of voxels as the input object containing the squared magnitude of the Fourier transform.

Action



Starts the computation.

7.7 PSFGen

This module can be used to compute a point spread function (PSF) for deconvolution of widefield and confocal microscopic image data. PSF computation is based on electromagnetic vector theory. The module is part of the *Amira deconvolution package*. It can be created by selecting it from the *Edit Create* menu of the Amira main window.

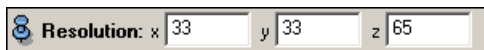
Connections

Data [optional]

A uniform scalar field (usually the image data to be deconvolved) can be connected to this port. The values of the *Resolution* and *Voxel size* ports will be set automatically to the values of the input field then.

Ports

Resolution

A control panel for the Resolution port. It features a blue circular icon with a white 'B' on the left. To its right is the text 'Resolution:'. Further right are three input fields: 'x' with the value '33', 'y' with the value '33', and 'z' with the value '65'.

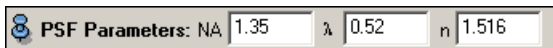
The number of voxels of the PSF image to be generated.

Voxel size [um]

A control panel for the Voxel size port. It features a blue circular icon with a white 'B' on the left. To its right is the text 'Voxel size [um]:'. Further right are three input fields: 'x' with the value '0.04', 'y' with the value '0.04', and 'z' with the value '0.04'.

The voxel size in microns of the PSF image to be generated.

PSF Parameters

A control panel for the PSF Parameters port. It features a blue circular icon with a white 'B' on the left. To its right is the text 'PSF Parameters:'. Further right are four input fields: 'NA' with the value '1.35', ' λ ' with the value '0.52', and 'n' with the value '1.516'.

Parameters for calculating the point spread function. *NA* denotes the numerical aperture of the microscope. *lambda* denotes the wavelength (as measured in vacuum) of the emitted light in micrometers. For confocal data the excitation and emission wavelength are assumed to be identical. In this case it might prove useful to compensate by supplying a value between excitation and emission wavelength as parameter. Finally, *n* denotes the refractive index of the specimen.

Microscopic Mode

A control panel for the Microscopic Mode port. It features a blue circular icon with a white 'B' on the left. To its right is the text 'Microscopic Mode:'. Further right are two radio buttons: 'widefield' (which is selected) and 'confocal'.

Selects whether the PSF of a widefield microscope or of a confocal microscope should be computed.

Action

A control panel for the Action port. It features a blue circular icon with a white 'B' on the left. To its right is the text 'Action:'. Further right is a button with a red square icon and the text 'DoIt'.

Actually computes the PSF.

Chapter 8

Examples

Example 1: Confocal Data

The original data set is provided under *Amira-3.0/data/deconv/polytrichum.am*. The images below were created using the *ProjectionView* module.

The properties of the data set are as follows:

- Numerical aperture 1.4
- Wavelength of the emitted light 0.58 micrometers
- Refractive index 1.516 (oil)

Example 2: Widefield Data

The original data set is provided under *Amira-3.0/data/deconv/alphalobe.am*. The images below were created using the *ProjectionView* module.

The properties of the data set are as follows:

- Numerical aperture 0.5
- Wavelength of the emitted light 0.58 micrometers
- Refractive index 1.33 (water)

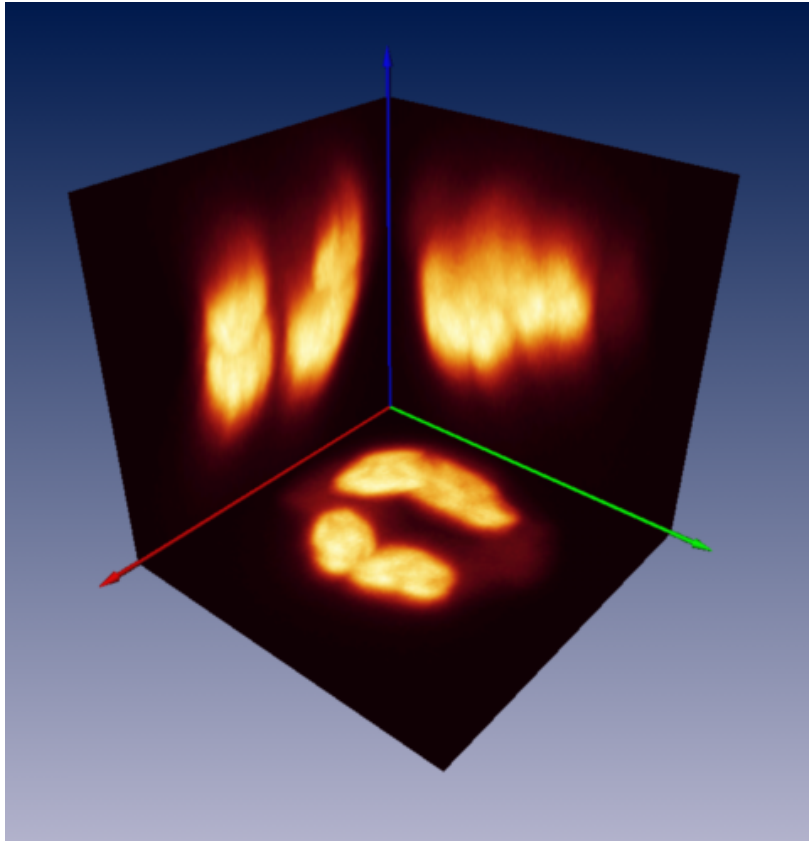


Figure 8.1: *polytrichum.am* before deconvolution.

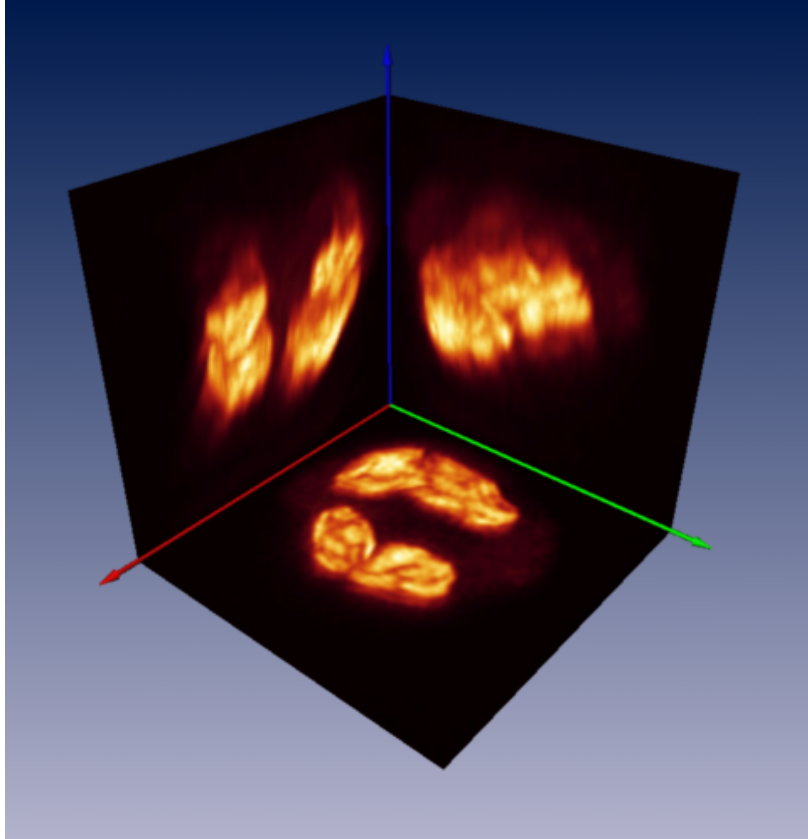


Figure 8.2: *polytrichum.am* after deconvolution.

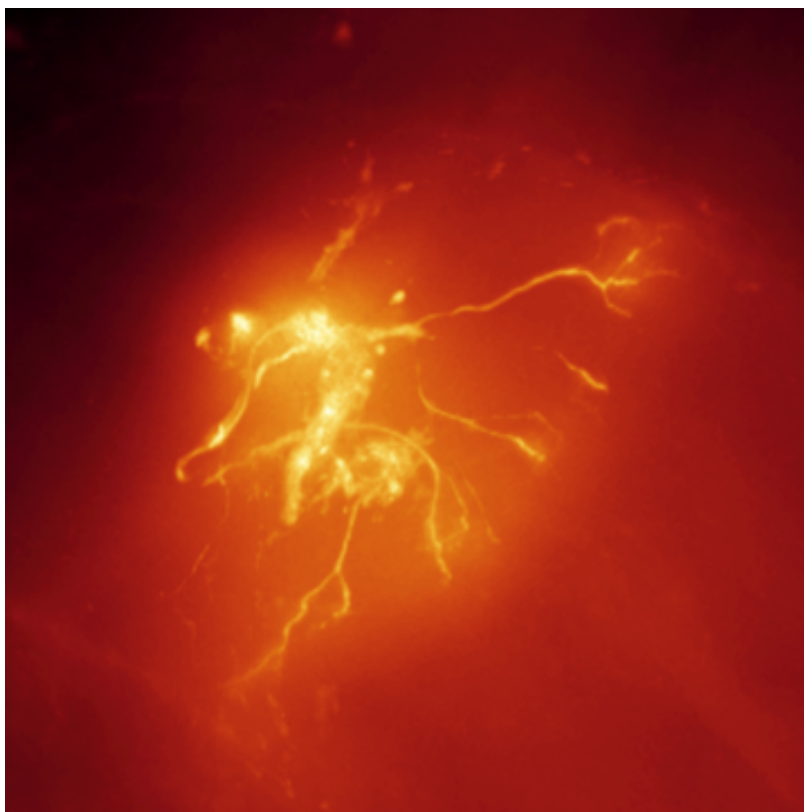


Figure 8.3: XY-maximum intensity projection of *alphalobe.am* before deconvolution.

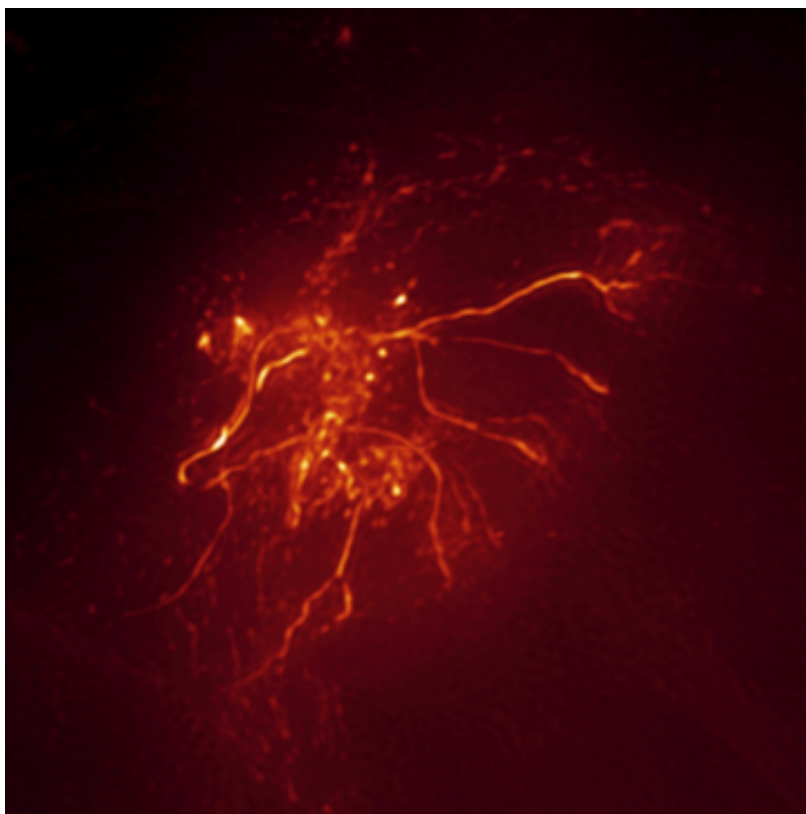


Figure 8.4: XY-maximum intensity projection of *algalobe.am* after deconvolution.

Index

- abberation, 113
- agarose gel, 113
- AlignMolecules, 43
- AlignSequences, 45
- axial blur, 100
- background correction, 123
- batch job, 112
- bead extraction, 113
- BeadExtract, 119
- beads, 113
- black level, 102
- BondAngleView, 47
- BondCalculation, 48
- border width, 106, 109, 124
- boundary artifacts, 101
- CCD detector, 123
- check point files, 112
- ClosestTimeStep, 49
- CompMolInterface, 50
- CompMolSurface, 51
- ConfigurationDensity, 53
- confocal microscope, 100, 101
- Convolution, 120
- CorrectZDrop, 122
- coverslip, 113
- DataPreprocess, 122
- Deconvolution, 124
- deconvolution
 - blind, 100, 109
 - non-blind, 100
 - standard, 102
- embedding medium, 113
- flatfield correction, 123
- Fourier transform, 118
- FourierTransform, 126
- immersion medium, 113
- in-plane sampling, 101
- initial estimate, 106, 111, 125
- intensity attenuation, 103, 122
- job dialog, 112, 126
- Lanczos filter, 103
- landmark set, 119
- maximum-likelihood method, 100, 124
- MDL, 79
- MeanMolecule, 56
- Measurement, 57
- memory consumption, 118
- microsphere, 113, 119
- Molecule, 74
- Molecule Attribute Editor, 89
- Molecule Editor, 92
- MoleculeLabel, 61
- MoleculeView, 63
- MolSurface, 73
- MolSurfaceView, 59
- MolTrajectory, 74
- multi-processing, 118
- noise, 100, 102, 121
- numerical aperture, 101, 124
- Nyquist sampling, 101

- Observables, 65
- oil immersion, 113
- optical sectioning microscopy, 100
- out-of-focus light, 100, 109
- overrelaxation, 106, 111, 125
- oversampling, 102

- PDB, 79
- performance, 118
- power spectrum, 126
- PrecomputeAlignment, 66
- PSF, 100, 103, 119
 - theoretical, 106
- PSF/DCD (CHARMM), 81
- PSFGen, 127

- reampling, 103
- refractive index, 113
- refractive index, 101, 124

- sampling rate, 101
- saturation, 102
- scanned volume, 101
- SecStructureView, 67

- Tripos, 81
- TubeView, 71

- undersampling, 102
- UniChem, 81

- vector theory, 127

- wavelength, 101, 124
 - emission, 127
 - excitation, 127
- widefield data, 101

- z-drop, 122
- ZIB Molecular File Format, 81