Part VI

Microscopy Option User's Guide

10 Deconvolution

The deconvolution modules of the Microscopy Option provide 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 deconvolution documentation is organized as follows:

- *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 provided:

- *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

Examples:

- *Confocal data set*
- *Widefield data set*

10.1 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 modules mainly provide 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 first sight, one may wonder why both a non-blind and a blind deconvolution algorithm are provided; blind deconvolution seems to be more general because 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 an 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 constrained 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.

10.2 Data acquisition 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 micrometers in the z-direction, the scanned image volume should range from 50 micrometers below the sample to 50 micrometers above it.

Choosing the Right Sampling Rate

The sampling rate is determined by the pixel sizes in the x and y directions 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

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

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

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

where n denotes the refractive index of the object medium and $alpha$ 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 too strict for most practical cases and even in the widefield case, approximately fullfilling 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 the 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, a high background level should be avoided because it decreases the dynamic range of the imaging system and the deconvolution works 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 is a separate tutorial about *bead extraction*.

10.3 Standard Deconvolution Tutorial

This tutorial explains how 3D image data sets can be deconvolved in Amira. It is asumed 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

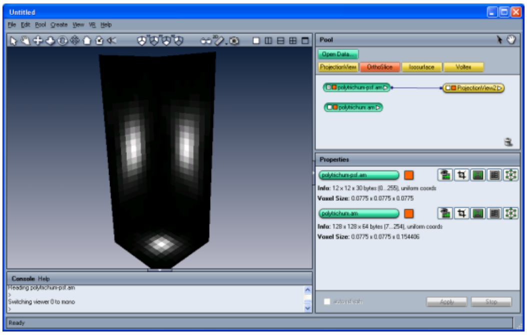
As an example we are going to use a confocal test data set (*polytrichum.am*) provided with the Amira deconvolution modules. The data file is located in the directory *Amira-5/data/deconv*.

- 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 a so-called *point spread function* (PSF) is also required. The PSF is the image of a single point source, or as a close approximation, 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.



polytrichum-psf.am
Info: 12 x 12 x 30 bytes [0...255], uniform coords
Voxel Size: 0.0775 x 0.0775 x 0.0775

polytrichum.am
Info: 128 x 128 x 64 bytes [7...254], uniform coords
Voxel Size: 0.0775 x 0.0775 x 0.154406

```
Reading polytrichum-psf.am
>
Switching viewer 0 to mono
>
```

Ready

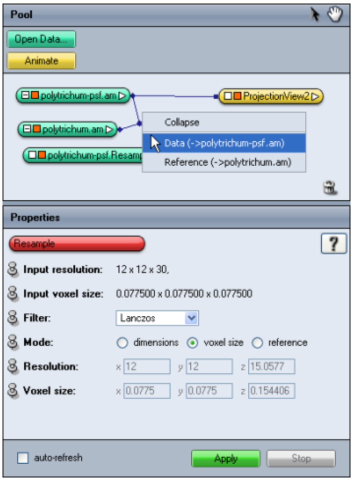
Figure 10.1: Maximum intensity projection of *polytrichum-psf.am*

- 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 10.1). 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 is 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



Input resolution: 12 x 12 x 30,
Input voxel size: 0.077500 x 0.077500 x 0.077500
Filter: Lanczos
Mode: dimensions [✓ voxel size] reference
Resolution: x 12 y 12 z 15.0577
Voxel size: x 0.0775 y 0.0775 z 0.154406

Figure 10.2: Resampling a PSF using the *Resample* module.

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 10.2).
- Resample the PSF by pressing the *Apply* 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 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 10.3 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.

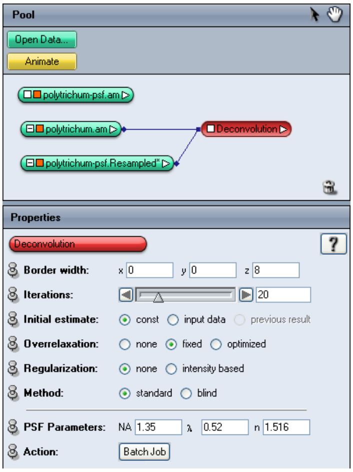
Initial estimate: Specifies the initial estimate of the deconvolution algorithm. If *const* is chosen a constant image is used initially. This is the most robust choice, yielding good results even if the input data is very noisy. We keep this option here.

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 us specify the *standard* option here.

The actual deconvolution process is started by pressing the *Apply* button. Please press this button now. The deconvolution should take about 10 seconds 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 Pool. You might take a look at the deconvolved data by moving the *ProjectionView* connection line from *polytrichum.am* to *polytrichum.deconv*.

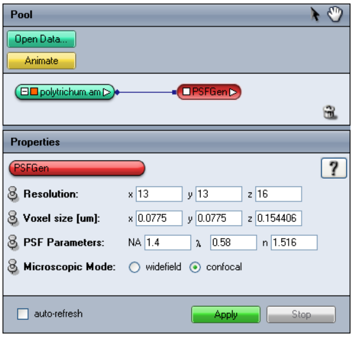


Border width: x 0 y 0 z 8
Iterations: 20
Initial estimate: [✓ const] input data previous result
Overrelaxation: none [✓ fixed] optimized
Regularization: [✓ none] intensity based
Method: [✓ standard] blind
PSF Parameters: NA 1.35 λ 0.52 n 1.516
Action: Batch Job

Figure 10.3: Deconvolution module attached to *polytrichum.am*.

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 *PSFGen*, allowing you to calculate theoretical PSFs. The module can be created by selecting *PSFGen* from the *Create Others* 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 *PSGGen* 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.
After you press the *Apply* button, the computed PSF appears as an icon labelled *PSF* in the 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



Resolution: x 13 y 13 z 16
Voxel size [um]: x 0.0775 y 0.0775 z 0.154406
PSF Parameters: NA 1.4 λ 0.58 n 1.516
Microscopic Mode: widefield [✓ confocal]

Figure 10.4: The PSFGen module calculates theoretical PSFs.

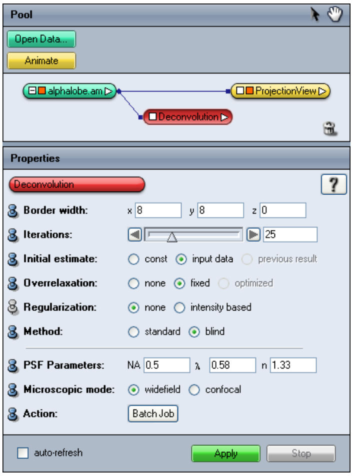
slightly wider. This is a common observation in many experiments.

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 *PSFGen*. After these parameters have been entered, the deconvolution process again can be started by pressing the *Apply* 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.

10.4 Blind Deconvolution Tutorial

This tutorial explains how blind deconvolution can be perfomed 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.



Properties

Deconvolution

Border width: x 8 y 8 z 0
Iterations: 25
Initial estimate: const [✓ input data] previous result
Overrelaxation: none [✓ fixed] optimized
Regularization: [✓ none] intensity based
Method: standard [✓ blind]
PSF Parameters: NA 0.5 λ 0.58 n 1.33
Microscopic mode: [✓ widefield] confocal
Action: Batch Job

auto-refresh Apply Stop

Figure 10.5: Parameters for blind deconvolution.

A Blind Deconvolution Example

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

- Load the file *alphalobe.am* from the directory *Amira-5/data/deconv*.
- Visualize the data set by attaching a *ProjectionView* module to it.

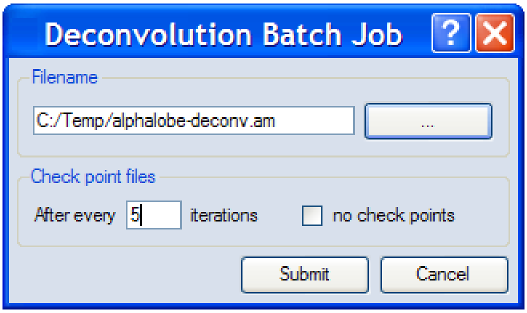
The data set has been recorded using a standard fluorescence microscope under so-called *widefield* conditions. It shows a neuron from the alpha-lobe of the honeybee brain. Compared to the confocal data set used in the standard deconvolution tutorial, *alphalobe.am* is much bigger. It has a resolution 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 10.5.

The individual parameters have the following meaning:
Border width: As 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 do not 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. Because 256 is a power of two ($2\hat{8}$), 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 usally don't improve much after 40 iterations.
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.
Regularization: We chose *none* here in order to do no regularization.
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



C:/Temp/alphalobe-deconv.am

After every 5 iterations

☐ no check points

Figure 10.6: Dialog for submitting a deconvolution job.

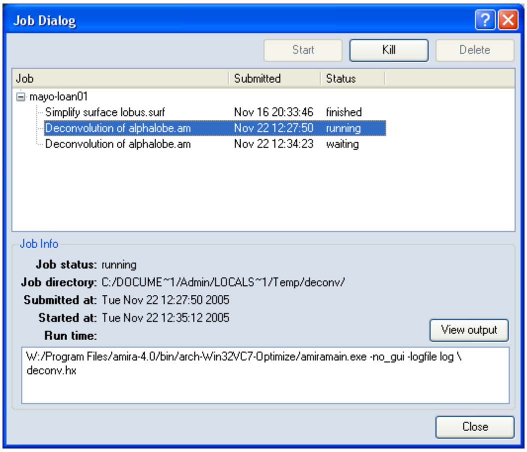
jobs can be submitted to the Amira job queue and then, for example, processed overnight. This works as follows:

- Press the *Batch Job* button of the *Action* port. A dialog as shown in Figure 10.6 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.

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 named 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/alphalobe-deconv.am`, the check point files are named `C:/Temp/alphalobe-deconv-0005.am`, `C:/Temp/alphalobe-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 10.7.
- 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



Job	Submitted	Status
mayo-loan01		
Simplify surface lobus.surf	Nov 16 20:33:46	finished
Deconvolution of alphalobe.am	Nov 22 12:27:50	running
Deconvolution of alphalobe.am	Nov 22 12:34:23	waiting

Job Info

Job status: running
Job directory: C:/DOCUME~1/Admin/LOCALS~1/Temp/deconv/
Submitted at: Tue Nov 22 12:27:50 2005
Started at: Tue Nov 22 12:35:12 2005
Run time:

```
W:/Program Files/amira-4.0/bin/arch-Win32VC7-Optimize/amiramain.exe -no_gui -logfile log \
deconv.hx
```

Figure 10.7: The Amira job dialog showing a pending deconvolution job.

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.

10.5 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 mesurements. 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 appropriate beads. It is important that the bead size be 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 mircoscope. 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 refractive index similar to methyl salicylate, for example.
5. Sample and 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, artifacts 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 allways be supperior to deconvolution of data suffering from aberrations.

6. Problems occur if the mounting medium remains liquid. In that 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 immobilization.

An example of an image data set containing multiple beads is provided in the file *beads.am* in the directory *Amira-5/data/deconv*.

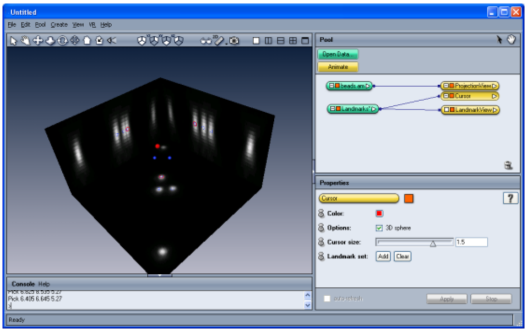
- Load the data set *beads.am*.
- Visualize the data using a *ProjectionView* module.

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 Landmark Editor button. If you want to add additional bead positions to



```
Pick 6.405 6.645 5.27
```

Figure 10.8: Individual beads can be interactively identified using a *Cursor* module.

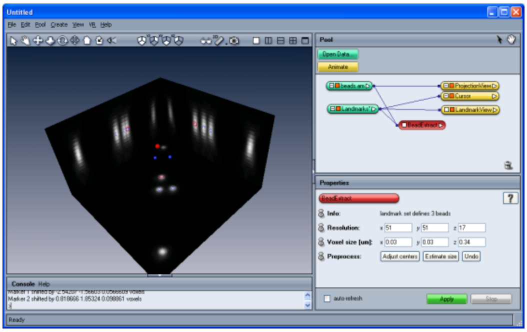
an existing landmark set object, make sure that the *master port* of the landmark set object is connected to the *Cursor* module. Otherwise, a new landmark set object will be created.

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.



Info: landmark set defines 3 beads
Resolution: x 51 y 51 z 17
Voxel size [um]: x 0.03 y 0.03 z 0.34
Preprocess: Adjust centers Estimate size Undo

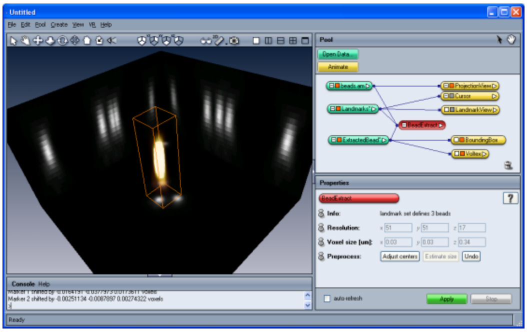
```
Marker 2 shifted by 0.818666 1.85324 0.098861 voxels
```

Figure 10.9: The *BeadExtract* module resamples and averages multiple beads.

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 neighboring 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 *Apply* button.

The data type of the resulting PSF will be *float*, irrespective 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 10.10 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 *Apply* 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 data sets should be scaled in intensity according to



BeadExtract

Info: landmark set defines 3 beads

Resolution: x 51 y 51 z 17

Voxel size [um]: x 0.03 y 0.03 z 0.34

Preprocess: Adjust centers | Estimate size | Undo

```
Marker 2 shifted by -0.00251134 -0.0087897 0.00274322 voxels
```

Figure 10.10: The final PSF visualized using a *Voltex* module.

their quality prior to bead extraction and summation to obtain a suitable weighting of the individual extracted beads in the final result.

10.6 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 it 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 with 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 x 256 x 256 (as for the *alphalobe.am* data set in the blind deconvolution tutorial), each working array will be about 64 MB, irrespective of the primitive data type of the input data set. Therefore at least 192 MB (3x4x256x256x256 bytes) are required for non-blind deconvolution with fixed overrelaxation, and 320 MB (5x4x256x256x256 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.

Finally, it should be mentioned that the deconvolution algorithm can make use of a multi-processor CPU board. Although you do not get twice the performance on a dual-processor PC, a speed-up of almost 1.5 can be achieved. By 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.

Example 1: Confocal Data

The original data set is provided under *Amira-5/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)

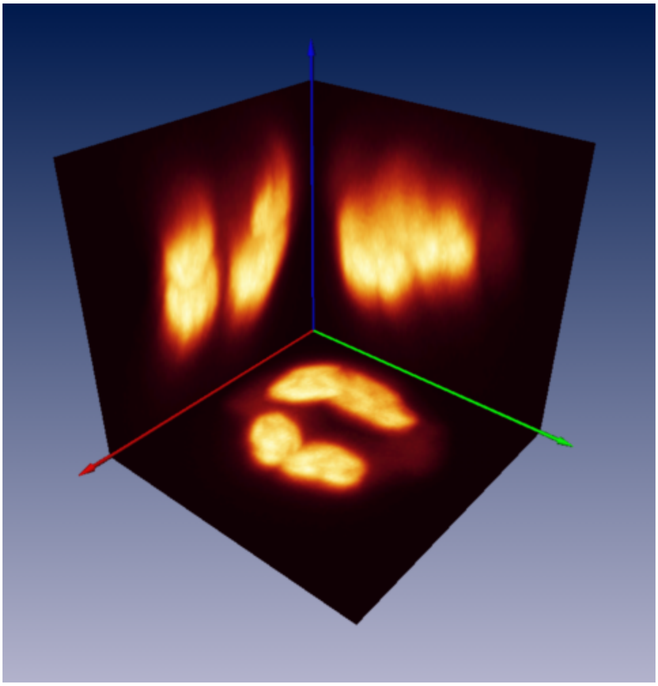


Figure 10.11: *polytrichum.am* before deconvolution.

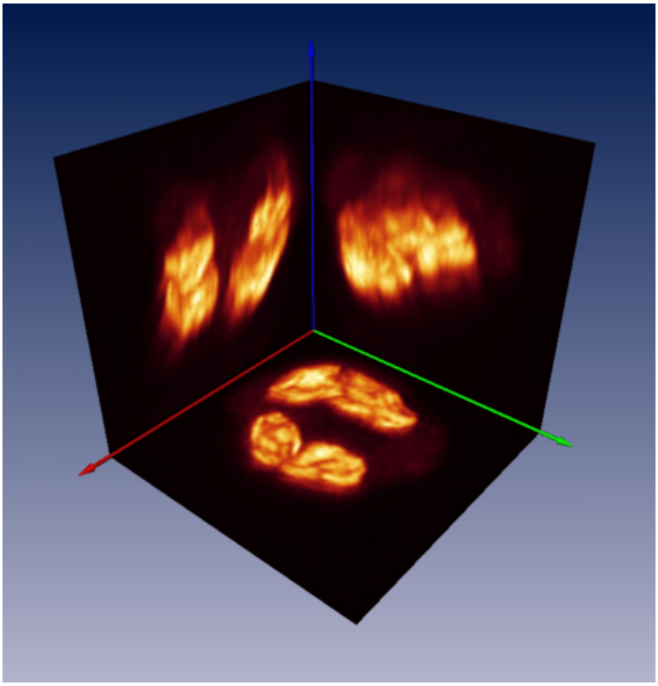


Figure 10.12: *polytrichum.am* after deconvolution.

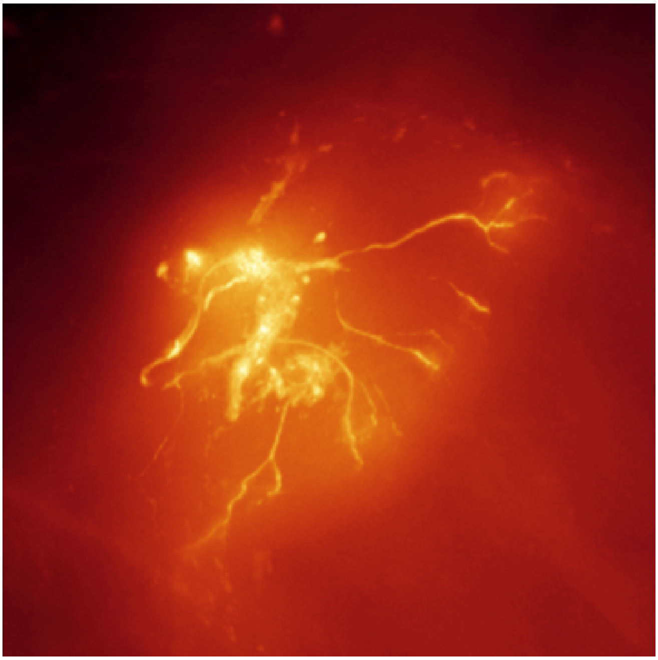


Figure 10.13: XY-maximum intensity projection of *alphalobe.am* before deconvolution.

Example 2: Widefield Data

The original data set is provided under *Amira-5/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)

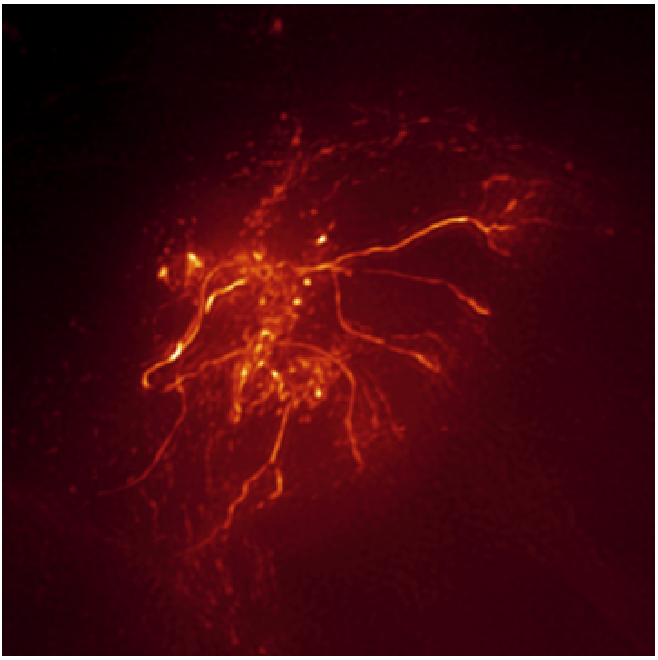


Figure 10.14: XY-maximum intensity projection of *alphalobe.am* after deconvolution.

11 Working with Multi-Channel Images

This is a step-by-step tutorial on how to visualize multi-channel image data. To follow this tutorial you should be familiar with the basic concepts of Amira. In particular you should be able to load files, to interact with the 3D viewer, and to connect display modules to data modules. All these issues are discussed in the *getting started* section.

We are going to load a set of multi-channel images into the workspace, attach a *MultiChannelField* group object to the data, and visualize it with several display modules. The steps are:

1. Load data into Amira.
2. Create a *MultiChannelField* and attach channels to it.
3. Using *OrthoSlice* with a *MultiChannelField*.
4. Using *ProjectionView* with a *MultiChannelField*.
5. Using *Voltex* with a *MultiChannelField*.
6. Saving multi-channel images in a single AmiraMesh file.

11.1 Loading Multi-Channel Images into Amira

The data we will be working with in this tutorial are confocal stacks of the prothoracic ganglion of the locust *Locusta migratoria*. They were kindly provided by Dr. Paul Stevenson, University of Leipzig, Germany. Two different channels were recorded and stored as separate files.

Amira supports a number of proprietary multi-channel formats of several microscope manufacturers (e.g., Leica and Zeiss). In such formats all channels are stored in a single file. Therefore the first steps described in this tutorial, namely grouping the channels manually, can often be omitted.

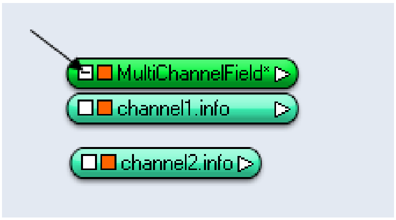


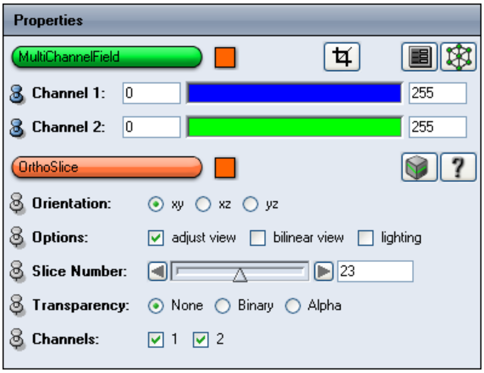
Figure 11.1: Data objects are connected to the *MultiChannelField* object with a right mouse click on the white square indicated by the arrow.

- Load channel 1 data by selecting the file `/data/multichannel/channel1.info`
- Load channel 2 data by selecting the file `/data/multichannel/channel2.info`
- Create a *MultiChannelField* object by selecting *MultiChannelField* from the *Create* menu of the Amira main window.

A dark green icon is displayed in the Pool. After you select the object, an info port is displayed saying ”no channels connected”.

- Connect channel1.info to the *MultiChannelField* by selecting ’Channel 1’ from the *MultiChannelField’s* connection menu (right mouse click in the small field on the left side of the icon) and releasing it on the channel1.info icon.
- Repeat the above procedure with channel2.info

When the *MultiChannelField* is selected you will note that two entries, *channel 1* and *channel 2*, appear in the module’s control panel. Each entry has two range text fields and a colormap area. The range text fields work very much like those in *OrthoSlice*. Press the right mouse button over the colormap area to bring up a context menu that will allow you to connect a colormap, edit the colormap, and so forth. If constant color is selected, double clicking in the colormap area pops up a color dialog that lets you freely define the color of each channel.
Now perhaps it is a good idea to activate the pins corresponding to Channel 1 and Channel 2 in the Properties Area. This will keep the control elements of the *MultiChannelField* module permanent in the Properties Area.


Properties

MultiChannelField

Channel 1: 0 255

Channel 2: 0 255

OrthoSlice

Orientation: [✓ xy] xz yz

Options: [✓ adjust view] bilinear view lighting

Slice Number: 23

Transparency: [✓ None] Binary Alpha

Channels: [✓ 1] [✓ 2]

Figure 11.2: When connected to a *MultiChannelField* object the *OrthoSlice* module has additional check boxes corresponding to the number of connected channels.

11.2 Using OrthoSlice with a MultiChannelField

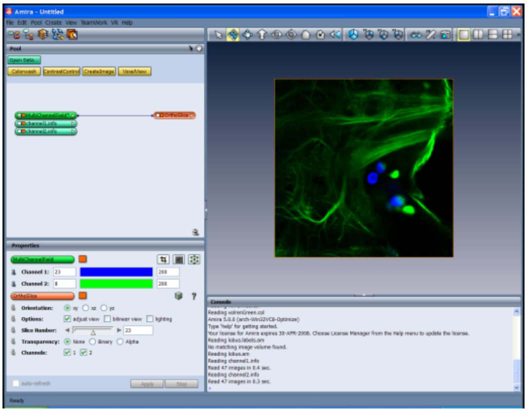
- Connect an *OrthoSlice* module to the MultiChannelField by right clicking on the icon and selecting OrthoSlice from the context menu.

When selecting the *OrthoSlice* module, you will see that there are two additional check boxes in its control panel corresponding to the two channels. Clicking these check boxes lets you selectively switch on/off each channel. First, we adjust the intensity mapping of each channel separately.

- Switch off channel 2 by deselecting its check box.
- Enter 23 and 200 in the min and max range fields of channel 1.

As a result, weak stainings – potentially unspecific staining – disappear and those structures that exhibit good staining become even more intense.

- Click off channel 1 and click on channel two.
- Enter the values 8 and 200 in the min and max text fields of channel 2. Move through the slices to see the results.



Channel 1: 23 200
Channel 2: 8 200
Orientation: [✓ xy] xz yz
Options: [✓ adjust view] bilinear view lighting
Slice Number: 23
Transparency: [✓ None] Binary Alpha
Channels: [✓ 1] [✓ 2]

```
Reading volrenGreen.col
Amira 5.0.0 (arch-Win32VC8-Optimize)
Type 'help' for getting started.
Your license for Amira expires 30-APR-2008. Choose License Manager from the Help menu to update the license.
Reading lobus.labels.am
No matching image volume found.
Reading lobus.am
Reading channel1.info
Read 47 images in 0.4 sec.
Reading channel2.info
Read 47 images in 0.3 sec.
>
```

Figure 11.3: Multi channel data visualized using the *OrthoSlice* module.

11.3 Using ProjectionView with a MultiChannelField

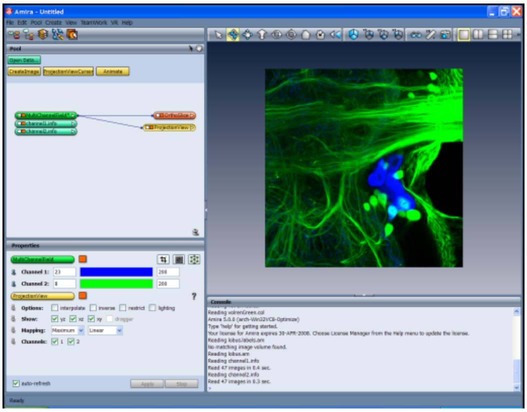
- Switch off the *OrthoSlice* by clicking on the viewer toggle of its icon (orange rectangle).
- Connect a *ProjectionView* module to the MultiChannelField by right clicking on the icon and selecting ProjectionView from the Display submenu.

As with the *OrthoSlice*, two new check boxes are shown in the module's control panel which can be used to display channels separately or simultaneously. In this way you may efficiently adjust the color and intensity of each channel before displaying them simultaneously.

11.4 Using Voltex with a MultiChannelField

- Switch off the ProjectionView by clicking on the viewer toggle of its icon (orange rectangle).
- Connect a *Voltex* module to the MultiChannelField by right clicking on the icon and selecting Voltex from the Display submenu.

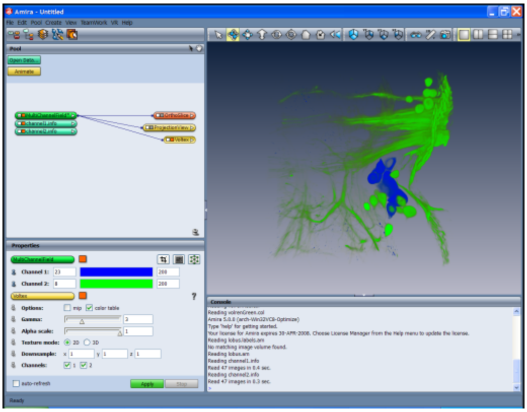
Here also, two channel check boxes are available. Furthermore, the familiar colormap field is missing. Instead there is a slider labelled *Gamma*. Now the color



Channel 1: 23 200
Channel 2: 8 200
Options: ☐ interpolate ☐ inverse ☐ restrict ☐ lighting
Show: ✓ yz ✓ xz ✓ xy ☐ dragger
Mapping: Maximum Linear
Channels: ✓ 1 ✓ 2
✓ auto-refresh

```
Reading volrenGreen.col
Amira 5.0.0 (arch-Win32VC8-Optimize)
Type 'help' for getting started.
Your license for Amira expires 30-APR-2008. Choose License Manager from the Help menu to update the license.
Reading lobus.labels.am
No matching image volume found.
Reading lobus.am
Reading channel1.info
Read 47 images in 0.4 sec.
Reading channel2.info
Read 47 images in 0.3 sec.
>
```

Figure 11.4: Multi channel data visualized using the *ProjectionView* module.



Channel 1: 23 200
Channel 2: 8 200
Options: ☐ mip ✓ color table
Gamma: 3
Alpha scale: 1
Texture mode: ● 2D ○ 3D
Downsample: x 1 y 1 z 1
Channels: ✓ 1 ✓ 2
☐ auto-refresh

```
Reading volrenGreen.col
Amira 5.0.0 (arch-Win32VC8-Optimize)
Type 'help' for getting started.
Your license for Amira expires 30-APR-2008. Choose License Manager from the Help menu to update the license.
Reading lobus.labels.am
No matching image volume found.
Reading lobus.am
Reading channel1.info
Read 47 images in 0.4 sec.
Reading channel2.info
Read 47 images in 0.3 sec.
>
```

Figure 11.5: Multi channel data visualized using the *Voltex* module.

of each channel is determined by that defined in the *MultiChannelField* and the *Gamma* slider controls the steepness of the alpha value (opacity) mapping used for volume rendering. Because volume rendering makes intensive use of hardware texture mapping and most consumer graphics adapters are limited in texture memory size, it is recommended to enter at least factors of 2 2 1 in the *Downsample* text fields of the Voltex module.

- Press the *Apply* button.

Each time you want to display another channel, you must press the *Apply* button again.

11.5 Saving a MultiChannelField in a Single AmiraMesh File

When the *MultiChannelField* icon is selected in the Pool, choose *Save Data As* from the File menu, enter a filename, and click OK. The data will be stored in AmiraMesh format so that each time you load the data the two channel stacks and the *MultiChannelField* group object will be restored.

Part VII

Skeleton Option User's Guide

12 Skeleton Option User's Guide

This is a step-by-step tutorial on how to use *Large Disk Data* to analyze microvascular networks in human brain tissue. Please note that another *tutorial* is available to learn how to extract filament networks from vessels or neuron images. To follow this tutorial you should be familiar with the basic concepts of Amira. In particular you should be able to load files, to interact with the 3D viewer, and to connect display modules to data modules. All these issues are discussed in the *getting started* section.

We are going to load 4 overlapping bricks of a large data set. The goal is to merge these bricks into one large volume (Large Disk Data). The volume will be stored on disk only – subvolumes can be loaded into memory. Some basic operations can directly be applied to the large volume.

For the tutorial you should have access to a directory to which you're allowed to write files.

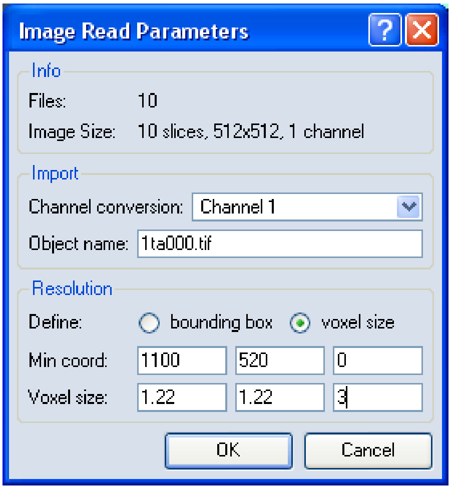
We don't provide any *TIFF* data with this tutorial. Hopefully you have a couple of blocks (*AmiraMesh* format) to test the algorithm on. Generally it is a good idea to import them into Amira and specify an approximate bounding box near the correct position.

12.1 Importing your Image Data

You should have your image data as stacks of numbered 2D images. The topmost slice should have the lowest number. File formats recognized by Amira can be found in the *File Formats section* of the user's guide. A good choice is *TIFF* because it provides lossless compression and is readable on many different systems.

You should also know the position in 3D of the lower left front corner of the brick you're going to import and the voxel size.

Choose *File/Open Data...*. A *File Dialog* pops up. You can now select all of the 2D images comprising the brick. After you press *Load*, another dialog pops up:



Info
Files: 10
Image Size: 10 slices, 512x512, 1 channel

Import
Channel conversion: Channel 1
Object name: 1ta000.tif

Resolution
Define: bounding box [✓ voxel size]
Min coord: 1100 520 0
Voxel size: 1.22 1.22 3

Enter the position and the voxel size of your block. After you press *OK*, the files are loaded into one block. A new green icon will appear in the Pool. Select it and select *File/Save Data As...* to store it on disk. Name it *1ta.am*. In this way you should proceed with all of your data.

12.2 Arranging the Bricks

After importing your data, you should copy the files to another directory where all the processing will be done. In this way the original data is not touched and you can revert back to it if something goes wrong.
Amira has a special data object to store links to files on disk and arrange them in 3D. It is called *Mosaic*.

- Create a Mosaic by selecting *Mosaic* from the *Create/Skeleton* menu of the Amira main window.

A green icon appears. When you select it you see that it contains no bricks. The buttons below the info line are used to add data objects.

- Press the *add files* button.
- Select the files, e.g `1ta.am`, `1tb.am`, `2ta.am`, `2tb.am` in the directory `AMIRA_ROOT/data/tutorials/skeleton`. You can select

multiple files at once by clicking on the first one and shift clicking on the last one.

- Press *Load*.

The selected files are added to the Mosaic. The Info port shows the overall number of the bricks added up to now. You can visualize the bricks with the *DisplayMosaic* module.

- Create a *DisplayMosaic* module by right clicking on the Mosaic and selecting *Skeleton/DisplayMosaic* from the context menu.
- Select the yellow DisplayMosaic icon and switch the highlighted brick by dragging the slider in the interaction area.
- Save the Mosaic.

12.3 Aligning Bricks

A special module allows to exactly align the bricks based on their gray values.

- Attach an *AlignBlocks* module to the Mosaic by selecting *Skeleton/AlignBlocks* in the data context menu.
- If you saved the mosaic already, its filename appears in *Mosaic name*.
- Press the *Apply* button to start the processing.

A maximum intensity projection (mip) of each block is computed. These mips are aligned and the resulting transformations are applied to the bricks on disk.

12.4 Filtering, Correcting Z-Drop, Resampling

It is possible to apply the same operation to all the bricks at once. To do this you must create a template for this operation. This could either be one brick with an editor (e.g. *Digital Image Filters*) attached or a compute module (e.g. *Resample*). We're going to demonstrate these two examples now. But first we should correct for the Z-Drop:

- Load one brick of the mosaic by using the *File/Open Data...* menu.
- Attach a *Deconv/CorrectZDrop* module to the brick.
- Press *Apply* to start the correction procedure and check the result.

- Attach an *ApplyTemplateToMosaic* script object by right clicking on the Mosaic and selecting *ApplyTemplate* from the *Skeleton* submenu in the context menu.
- Attach the *Template* connection of the ApplyTemplateToMosaic module to the CorrectZDrop module.
- Select *ApplyTemplateToMosaic* and press *Apply*.

Next we're going to apply a digital filter to all blocks:

- Load one brick of the mosaic by using the *File/Open Data...* menu.
- Select the data icon of the brick.
- Attach a *Digital Image Filter* by pressing the *Digital Filters* button in the Properties Area.
- Select *Gauss* from the *Filter* port, and apply it. To check the results you can attach an *OrthoSlice*.
- Attach the *Template* connection of the ApplyTemplateToMosaic module to the filter object (with the data attached).
- Select *ApplyTemplateToMosaic* and press *Apply*.

Another useful filter might be *Median2D* or the *Median3D*. For *Median3D*, select *Median* from the first pulldown menu of the *Filter* port, and *3D* from the second pulldown menu. The application of the latter filter takes some time but leads to good results.

The script object starts to load each brick of the mosaic, applies the filter to it, and writes it back to the same location on disk. *There are no warnings about this overwrite.*

In the next step we're going to resample every brick to an isotropic voxel size. This is only an optional step and might lead to smoother central lines in the following processing. But it increases size of the data on disk by a factor of about three. You should carefully consider whether you want to perform this step or not.

- Attach a *Resample* module to the brick loaded before and select *Mode: voxel size* and adjust *Voxel size: z* to get an isotropic result.
- Press *Apply* to start the resampling procedure and check the result.
- Attach the Resample module to the *Template* connection of the ApplyTemplateToMosaic module.
- Press *Apply* of the ApplyTemplateToMosaic module.

All bricks are resampled and saved to the same position. It is a good idea to sample all bricks to an isotropic voxel size. It improves the result of the distance map and the skeletonization we're going to apply.
As a standard prefiltering procedure you should:

- Apply the ZDropCorrection.
- Apply a 2D median filter.
- Apply a 3D Gaussian filter with a small sigma (1 or smaller).

If the results are not satisfactory, you should try to extend the prefiltering step.

12.5 Creating the Large Disk Data

The next step is to create a new Large Disk Data object and sample the bricks onto it. The overlapping regions can be blended with each other and a border can be added.
Note: The thinning algorithm expects a black border around the data. The border should be at least of size *lenOfEnds* used during thinning (see below). By default a border of 15 voxels on each side in each dimension. Be sure to check this if you manually set *lenOfEnds*.

- Attach a *MosaicToLargeDiskData* to the Mosaic by right clicking on the Mosaic and selecting *Skeleton/MosaicToLargeDiskData* from the submenu in the context menu.
- Select the red MosaicToLargeDiskData icon.

You can see some options in the Properties Area. The default options are fine for the tutorial.

- A default filename derived from the mosaic is displayed in the *Filename* port. You might want to override it.
- Press the *Apply* button.

A new green icon which represents the new data object will appear in the Pool. After this the bricks will be loaded one after the other and will be sampled. This may take some time.

- Select the new green icon (titled Image).

In the Properties Area some information about the data stored on disk is displayed. Next,

- Delete or switch off the DisplayMosaic module.
- Connect a BoundingBox to the Mosaic icon.
- Connect a BoundingBox to the Image icon.

The second box is slightly bigger than the bounding box of the Mosaic. This is due to the border added by the MosaicToLargeDiskData module.

12.6 Accessing the Large Disk Data

You can't directly visualize the Large Disk Data by e.g. attaching an OrthoSlice. Before you can do this, you must select a subvolume and load this subvolume into the Pool. The Subvolume will be an ordinary Amira field and you can use all the modules that you normally use. It may be a good idea to clean up the Pool now, but it's not required. The Mosaic is no longer needed.

- Connect an *LatticeAccess* to the Image object by right clicking on it and selecting *LatticeAccess* from the popup menu.
- Select the red *LatticeAccess* icon.

In the viewer you can see a dragger box in one corner of the bounding box of the Large Disk Data. You can click and drag the corners or the faces of the box to specify the subvolume you want to load. In the Properties Area the corresponding dimensions are displayed.

- Drag the box somewhere inside the volume (For this you need to switch the viewer into interaction mode).
- Press the *Apply* button.
- Attach an OrthoSlice to the new green icon (Image.view).
- Select the *LatticeAccess* object, then toggle on the *auto-refresh* check box.
- Drag the box in the viewer.

By setting *auto-refresh* on, every time you drag the box an automatic reload is started and all modules downstream of the view are recomputed. This is an easy way to scan through the large volume. Try different display modules on the Image.view, e.g., an isosurface.

12.7 Computing directly on the Large Disk Data

Some computation modules are able to handle the Large Disk Data directly. These include thresholding, computation of a distance map, thinning, extracting a line set from a voxel skeleton, and computation of the thickness of the lines (evaluating the distance map at the points of the lineset). All these steps are presented in this subsection.
The first step is to apply a simple thresholding.

- Attach a *Threshold* to the *Image* icon by right clicking on it and selecting *Threshold* from the *Skeleton* submenu in the popup menu.
- Select an appropriate threshold in the Properties Area.
- Select a filename you want to store the result to. In the tutorial we will use the default name *Image.labels*.
- Press the *Apply* button.

A new green icon that contains the labels will appear. Connect an *LatticeAccess* module to it as described above and have a look at the results.
You might want to correct the result of the segmentation procedure manually. This might be useful to fill big vessels or remove uninteresting parts. Amira has a *segmentation editor* to perform this task. Due to the size of the data, you will have to work on subblocks of the whole data set.
In the next step we'll calculate a distance map of the object.

- Attach a *ChamferMap* to the *Image.labels* icon by right clicking on it and selecting *ChamferMap* from the *Skeleton* submenu in the popup menu.
- Specify an filename (the default is OK for the tutorial).
- Press *Apply*.

A new green icon named Image.dm will appear. Connect an *LatticeAccess* module and have a look at the distance map.
The thinning procedure needs the labels and the distance map as input.
Note: The thinning algorithm automatically detects endpoints of vessels. A parameter is used to distinguish them from ”noise” on the surface of the vessels to avoid spurious branches. You might want to change this parameter manually in the console. Use `Thinner setVar lenOfEnds 10` to set the length of the ends to 10 voxels before they are detected as unconnected ends. This is a rather large value leading to only a few branches. The drawback is that you also might miss real endpoints. It will be really hard to detect such errors during the network

check. But in general we think it is a good idea to avoid spurious branches directly during thinning.

- Attach a *Thinner* to the *Image.labels* icon by right clicking on it and selecting *Thinner* from the *Skeleton* submenu in the popup menu.
- Connect the port for the distance map to the Image.dm icon. You can achieve this by right clicking on the white square on the left side of the ExtThinner icon and selecting Distmap. A blue line is attached to the mouse pointer; after you click on the Image.dm icon, the two modules are connected.
- Specify a filename (the default is fine for the tutorial).
- Press *Apply*.

A new green icon named Image.thinned will appear and the thinning process is started. It may take some time before it finishes. We will directly go on and convert the result into a lineset before visualizing it.

- Attach a *TraceLines* to the *Image.thinned* icon by right clicking on it and selecting *TraceLines* from the *Skeleton* submenu in the popup menu.
- Unselect the *cluster* toggle in the Properties Area.
- Press *Apply*.

The new icon that is visible now in the Pool is a lineset, which you are probably already familiar with. You can visualize it by connecting a *LineSetView*.

- Create an *LineSetView* module by right clicking on the Image.lineset and selecting LineSetView from the context menu.

The lines are rather jaggy because they connect centers of voxels. To get smoother lines you can use a Tcl command in the console.

- Type `Image.lineset smooth` into the Amira console window. You can repeat this if you would prefer even smoother lines.

You can use the *CheckNetwork* module from the *Skeleton* submenu in the context menu to remove short ends.
In the last step of this subsection we will compute a thickness for every point on the lines. For the thickness we use the values of the distance map.

- Attach an *EvalOnLines* to the *Image.lineset* icon by right clicking on it and selecting *EvalOnLines* from the *Compute* submenu in the popup menu.

- Connect the port for the distance map to the Image.dm icon. You can achieve this by right clicking on the white square on the left side of the EvalOnLines icon and selecting *Field*. A blue line is attached to the mouse pointer; after you click on the Image.dm icon, the two modules are connected.
- Press *Apply*.

The module doesn't create a new data icon. It is more like an editor and changes the connected lineset. It adds a data value for every vertex in the lineset and calculates the value of the field at the point of the vertex. You can visualize the data with the LineSetView.

- Select the LineSetView by clicking on it.
- In the Properties Area there is a drop down menu called *ColorMode*. Click on it and select *Data 0*.
- Right click on the rectangular area in the row *Colormap*. A popup menu appears. Select *physics.icol*.
- Change the range of the colormap by clicking into text field right of the colormap and type in 15.

You see that the lines are now colored. The color is an indicator for the local radius of the original object.

12.8 Region of Interest

During visualization of large data sets there is often the need to restrict the displayed geometry to a subvolume of the total data set. It would be nice if different modules shared the same volume and the volume could be changed simultaneously for all of them. In Amira there is a special module that provides this possibility; it is called *SelectRoi*. You can attach it to every spatial data object. Some display modules have a connection called *ROI* that can be attached to the SelectRoi module to restrict the view.

- Remove all objects except for the Image.lineset and the LineSetView.
- Create a *SelectRoi* module by right clicking on the Image.lineset and selecting SelectRoi from the *Display* submenu of the context menu.
- Connect the Connection named ROI of the LineSetView to the SelectRoi module. To do this, right click on the white square on the left side of the LineSetView icon and select *ROI*. A blue line is attached to the mouse

pointer; after you click on the SelectRoi icon, the two modules are connected.

- Switch the viewer to interaction mode and click and drag one of the green squares. This will adjust the Region of Interest and the LineSetView will adopt the new restriction immediately.
- By clicking and dragging on the (invisible) faces of the cuboid you can move it to another position.

When working with a small subset of the lineset, it is possible to do more involved visualizations that require more graphics power. For example, the lines can be displayed as tubes that reflect the local thickness.

- Choose a rather small part of the lineset.
- Select the *LineSetView*.
- Click on the *Shape* drop down menu and select *Circle*.
- Click on the *ScaleMode* drop down menu and select *Data 0*.
- Move the *ScaleFactor* slider to 2.

In the viewer the lines are now displayed as tubes. The thickness is scaled with the data associated with the lines.

Note: The data value associated with the lines is the local radius. The LineSetView scales by the local diameter. To scale to the physical size you therefore must use a ScaleFactor of 2.

In the next step we're going to load a part of the image data that is also determined by the SelectRoi module. You can then easily load the same subvolume from different lda files if you connect all *LatticeAccess* modules to one common SelectRoi module.

- Load the file `Image` that you saved before.
- Attach an *LatticeAccess* module to it (see above if you don't know how).
- Connect the Connection named ROI of the *LatticeAccess* to the SelectRoi module. This is done the same way as with the LineSetView.
- Select the *LatticeAccess* module, then press the *Apply* button.
- Attach a ProjectionView display module to the newly displayed green *Image.view* icon.
- You can do the same for the `Image.labels` file.

All *LatticeAccess* modules you created are now restricted to the same volume and can easily be moved by one click-and-drag operation.

12.9 Check Network

In this subsection we'd like to present a module that can be used to jump to all endpoints of a lineset and create some nice views for checking if the endpoints are fine or if they should be edited.

- Create a *CheckNetwork* module by right clicking on the *Image.lineset* and selecting *CheckNetwork* from the *Skeleton* submenu of the popup menu.
- Connect the *SelectRoi* connection of the *CheckNetwork* to the *SelectRoi* module (right click on the white square at the left of CheckNetwork, select SelectRoi, click on the yellow SelectRoi icon).
- Select the *LatticeAccess* module, and toggle on the *auto-refresh* check box.
- Adjust the size of the lines by selecting the *LineSetView* and changing the *ScaleFactor* slider to approximately 0.15.
- Select the *CheckNetwork* module.
- Press the *Next Endpoint* button.
- By repeating the last step you can jump through all endpoints.

12.10 Coloring a Lineset According to its Depth Value

It can be useful to color the lines in different ways. In the next example we're going to color the lineset by the local z value. This is done in two steps:

- Create a Scalarfield which provides the depth (z value).
- Evaluate this value on the lines and use a LineSetView.

To create the Scalarfield:

- Select *Create/Data/Scalarfield*.
- Select the newly created icon.
- Type *z* into the *Expr* field.

The next step it to evaluate this scalarfield on the lineset. You can do this by selecting the lineset and typing into the console.
Hint: Press the `<TAB>` key to get the name of the selected module.

- Type `lines computeData scalarfield 2` to evaluate the data (Fill in your specific names for `lines and scalarfield`). The 2 indicates to store the data values as data 2 in the lineset.
- Attach a LineSetView and use *data2* for color coding.

Index

VISAGE IMAGING®
Visioneering Science for Life

For contact details, please see our Internet page at:

http://www.amira.com/contact.html

Ref: 750-70280-0522

3D VISUALIZATION OF DICOM IMAGES FOR RADIOLOGICAL APPLICATIONS

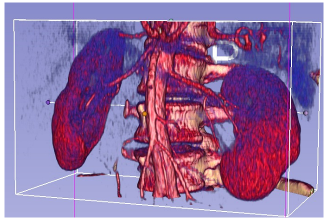
Sonia Pujol, PhD, Harvard Medical School
Surgical Planning Laboratory, Brigham and Women's Hospital

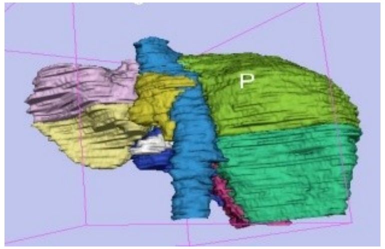
Kitt Shaffer, MD, PhD, Boston University
Vice-Chairman for Education, Boston University School of Medicine

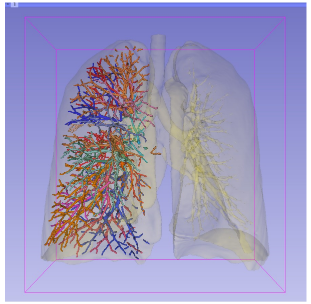
Ron Kikinis, MD, Harvard Medical School
Surgical Planning Laboratory, Brigham and Women's Hospital

3D Visualization of DICOM images for Radiological applications

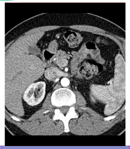
Following this tutorial, you will be able to load and visualize DICOM volumes with 3D Slicer, and to interact in 3D with structural images and models of the anatomy.



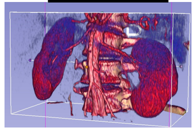




Overview

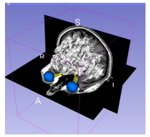


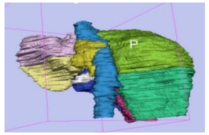
Part I: Introduction to the 3DSlicer software



Part II: 3D Data Loading and visualization of DICOM images

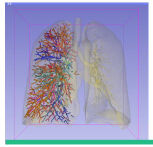
- Volume Rendering of thoraco-abdominal CT data
- Surface Rendering of MR head data





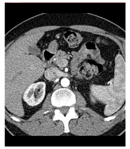
Part III: 3D interactive exploration of the anatomy

- Exploration of the Segments of the liver
- Exploration of the Segments of the lung



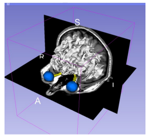
Tutorial Datasets

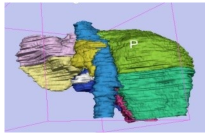
The tutorial data include 4 datasets:



3D Visualization DICOM images part 1:

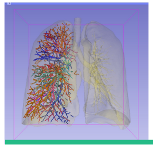
- dataset1_Thorax_Abdomen
- dataset2_Head



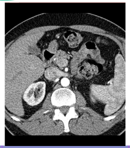


3D Visualization DICOM images part 2

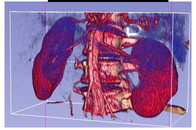
- dataset3_Liver
- dataset4_Chest

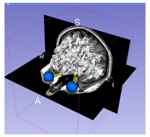


Overview



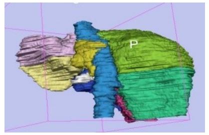
Part I: Introduction to the 3DSlicer software

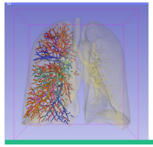




Part II: 3D Data Loading and visualization of DICOM images

- Volume Rendering of thoraco-abdominal CT data
- Surface Rendering of MR head data

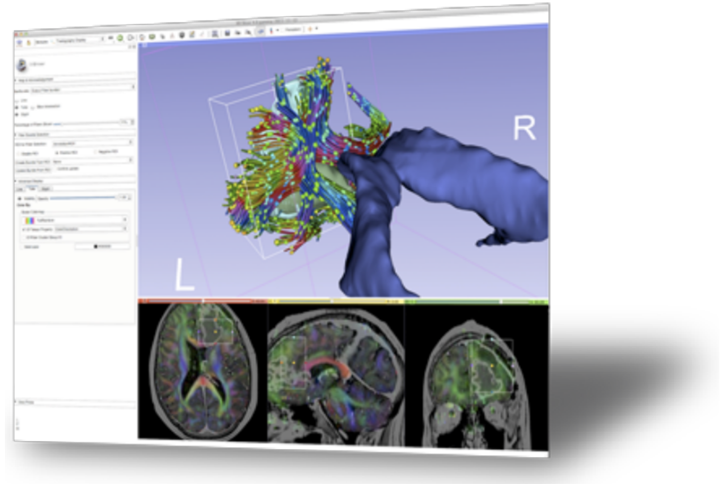




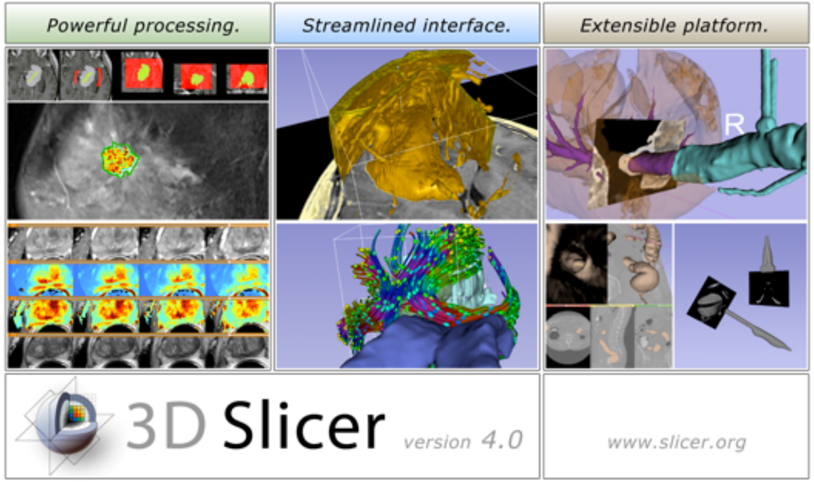
Part III: 3D interactive exploration of the anatomy

- Exploration of the Segments of the liver
- Exploration of the Segments of the lung

Introduction to the 3DSlicer software



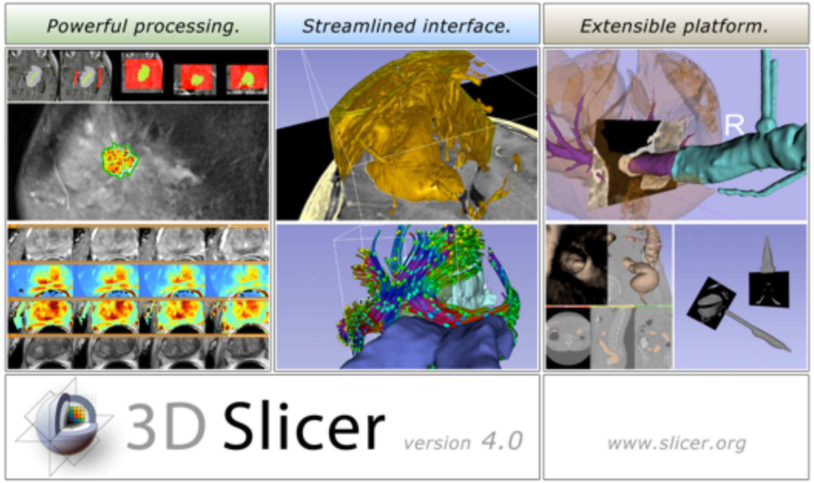
3DSlicer



3D Slicer is a freely available open-source platform for segmentation, registration and 3D visualization of medical imaging data.

3D Slicer is a multi-institutional effort supported by the National Institute of Health.

3DSlicer

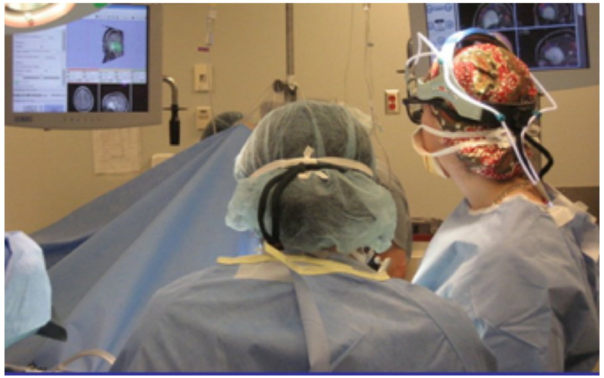


- 3DSlicer version 4.3 is a multi-platform software running on Windows, Linux, and Mac OSX
- Slicer is distributed under a BSD license with no restriction on use
- Slicer is a tool for research, and is not FDA approved

Disclaimer
It is the responsibility of the user of 3DSlicer to comply with both the terms of the license and with the applicable laws, regulations and rules.

An interdisciplinary platform

An open-source environment for software developers



An end-user application for clinical investigators and scientists

A software platform that is both easy to use for clinical researchers and easy to extend for programmers

3DSlicer History

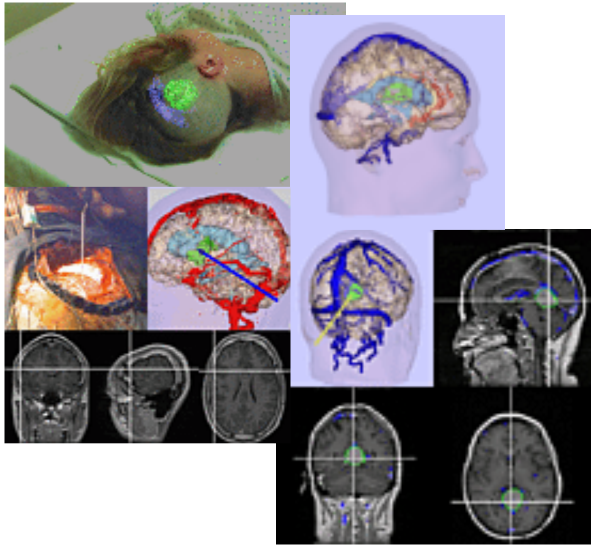
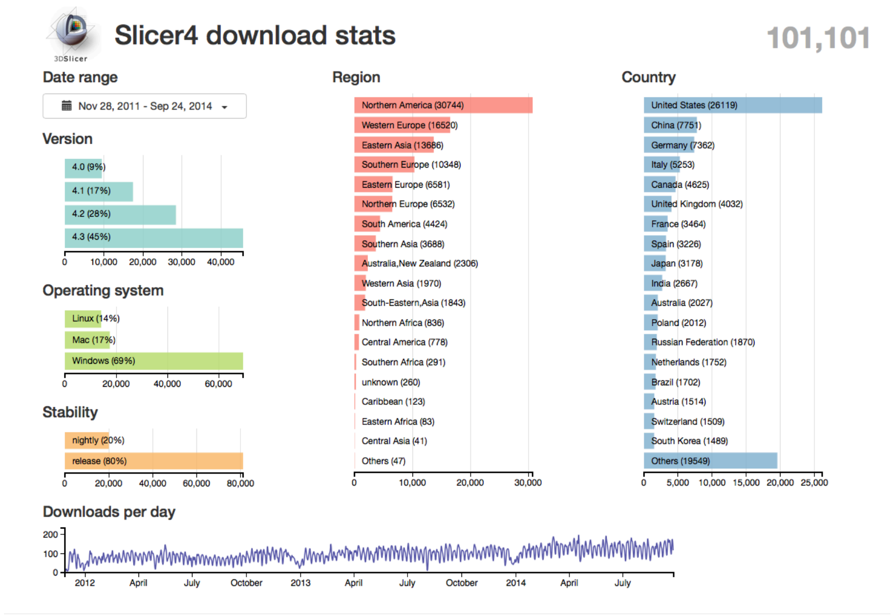


Image Courtesy of the CSAIL, MIT

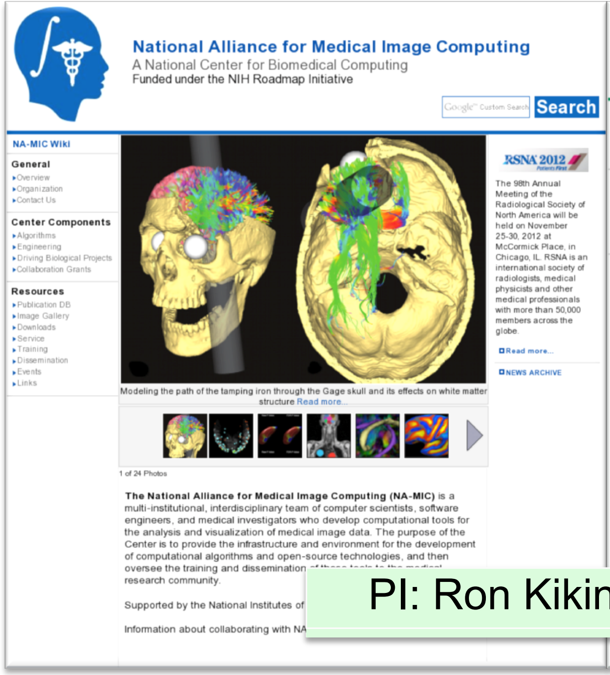
- 1997: Slicer started as a research project between the Surgical Planning Lab (Harvard) and the CSAIL (MIT)

3DSlicer History



- 1997: Slicer started as a research project between the Surgical Planning Lab (Harvard) and the CSAIL (MIT)
- 2014: Multi-institution effort to share the latest advances in image analysis with the clinical and scientific community

A multi-institution: NA-MIC, NAC, NCIGT



National Alliance for Medical Image Computing
A National Center for Biomedical Computing
Funded under the NIH Roadmap Initiative

NA-MIC Wiki

General
- Overview
- Organization
- Contact Us

Center Components
- Algorithms
- Engineering
- Driving Biological Projects
- Collaboration Grants

Resources
- Publication DB
- Image Gallery
- Downloads
- Service
- Training
- Dissemination
- Events
- Links

Modeling the path of the tamping iron through the Gage skull and its effects on white matter structure Read more...

1 of 24 Photos

RSNA 2012

The 98th Annual Meeting of the Radiological Society of North America will be held on November 25-30, 2012 at McCormick Place, in Chicago, IL. RSNA is an international society of radiologists, medical physicists and other medical professionals with more than 50,000 members across the globe.

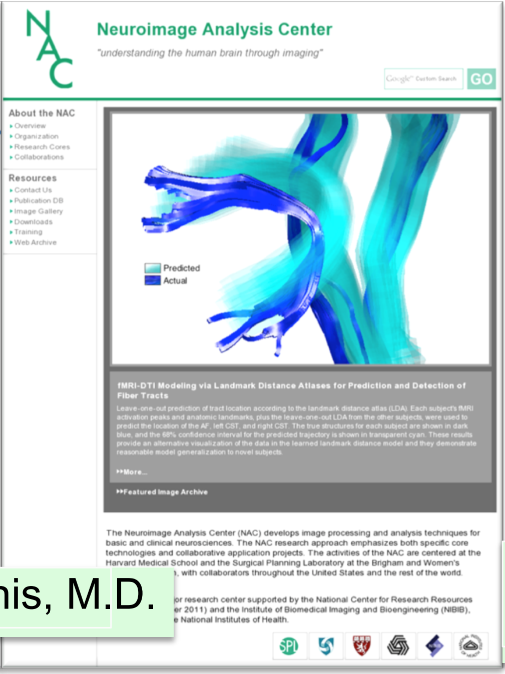
Read more...

NEWS ARCHIVE

The National Alliance for Medical Image Computing (NA-MIC) is a multi-institutional, interdisciplinary team of computer scientists, software engineers, and medical investigators who develop computational tools for the analysis and visualization of medical image data. The purpose of the Center is to provide the infrastructure and environment for the development of computational algorithms and open-source technologies, and then oversee the training and dissemination of these tools to the medical research community.

Supported by the National Institutes of

Information about collaborating with NA



Neuroimage Analysis Center
"understanding the human brain through imaging"

About the NAC
- Overview
- Organization
- Research Cores
- Collaborations

Resources
- Contact Us
- Publication DB
- Image Gallery
- Downloads
- Training
- Web Archive

fMRI-DTI Modeling via Landmark Distance Atlases for Prediction and Detection of Fiber Tracts

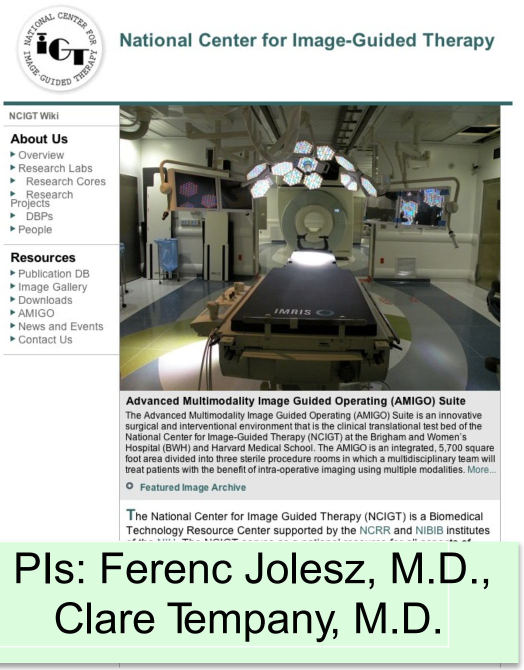
Leave-one-out prediction of tract location according to the landmark distance atlas (LDA). Each subject's fMRI activation peaks and anatomic landmarks, plus the leave-one-out LDA from the other subjects, were used to predict the location of the AF, left CST, and right CST. The true structures for each subject are shown in dark blue, and the 68% confidence interval for the predicted trajectory is shown in transparent cyan. These results provide an alternative visualization of the data in the learned landmark distance model and they demonstrate reasonable model generalization to novel subjects.

▸▸More...

▸▸Featured Image Archive

The Neuroimage Analysis Center (NAC) develops image processing and analysis techniques for basic and clinical neurosciences. The NAC research approach emphasizes both specific core technologies and collaborative application projects. The activities of the NAC are centered at the Harvard Medical School and the Surgical Planning Laboratory at the Brigham and Women's
, with collaborators throughout the United States and the rest of the world.

or research center supported by the National Center for Research Resources
er 2011) and the Institute of Biomedical Imaging and Bioengineering (NIBIB),
e National Institutes of Health.



National Center for Image-Guided Therapy

NCIGT Wiki

About Us
- Overview
- Research Labs
- Research Cores
- Research Projects
- DBPs
- People

Resources
- Publication DB
- Image Gallery
- Downloads
- AMIGO
- News and Events
- Contact Us

Advanced Multimodality Image Guided Operating (AMIGO) Suite

The Advanced Multimodality Image Guided Operating (AMIGO) Suite is an innovative surgical and interventional environment that is the clinical translational test bed of the National Center for Image-Guided Therapy (NCIGT) at the Brigham and Women's Hospital (BWH) and Harvard Medical School. The AMIGO is an integrated, 5,700 square foot area divided into three sterile procedure rooms in which a multidisciplinary team will treat patients with the benefit of intra-operative imaging using multiple modalities. More...

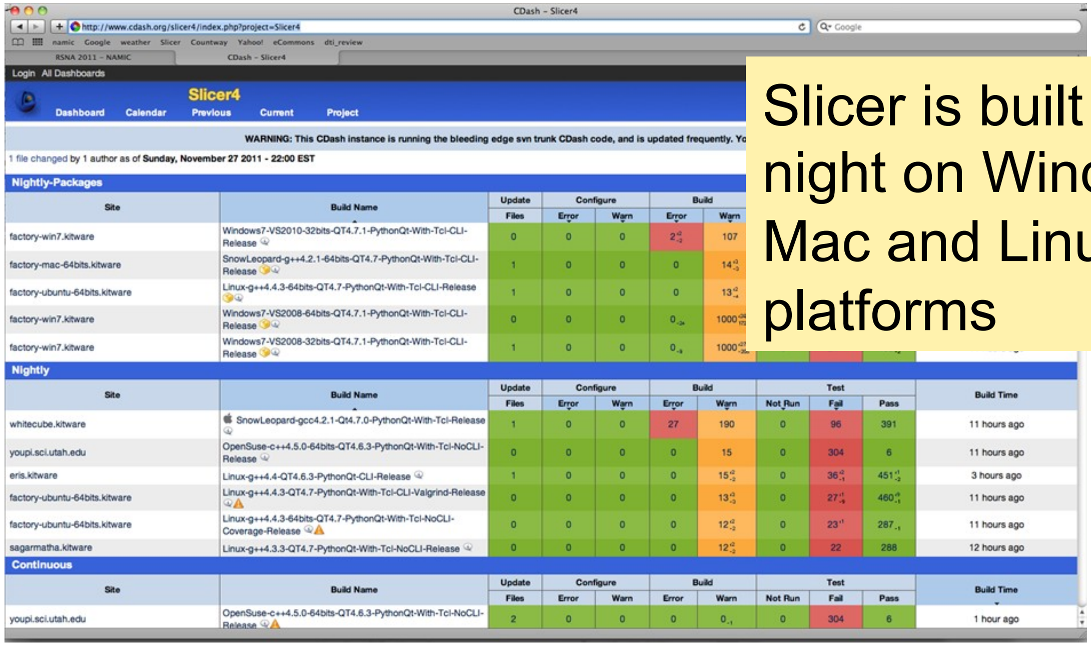
Featured Image Archive

The National Center for Image Guided Therapy (NCIGT) is a Biomedical Technology Resource Center supported by the NCRR and NIBIB institutes

PI: Ron Kikinis, M.D.

PIs: Ferenc Jolesz, M.D., Clare Tempany, M.D.

Slicer: Behind the scenes



CDash - Slicer4

http://www.cdash.org/slicer4/index.php?project=Slicer4

Login All Dashboards

Slicer4

Dashboard Calendar Previous Current Project

WARNING: This CDash instance is running the bleeding edge svn trunk CDash code, and is updated frequently. Yo

1 file changed by 1 author as of **Sunday, November 27 2011 - 22:00 EST**

Nightly-Packages

Site	Build Name	Update	Configure		Build	
		Files	Error	Warn	Error	Warn
factory-win7.kitware	Windows7-VS2010-32bits-QT4.7.1-PythonQt-With-Tcl-CLI-Release	0	0	0	2	107
factory-mac-64bits.kitware	SnowLeopard-g++4.2.1-64bits-QT4.7-PythonQt-With-Tcl-CLI-Release	1	0	0	0	14
factory-ubuntu-64bits.kitware	Linux-g++4.4.3-64bits-QT4.7-PythonQt-With-Tcl-CLI-Release	1	0	0	0	13
factory-win7.kitware	Windows7-VS2008-64bits-QT4.7.1-PythonQt-With-Tcl-CLI-Release	0	0	0	0	1000
factory-win7.kitware	Windows7-VS2008-32bits-QT4.7.1-PythonQt-With-Tcl-CLI-Release	1	0	0	0	1000

Nightly

Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
whitecube.kitware	SnowLeopard-gcc4.2.1-Qt4.7.0-PythonQt-With-Tcl-Release	1	0	0	27	190	0	96	391	11 hours ago
youpi.sci.utah.edu	OpenSuse-c++4.5.0-64bits-QT4.6.3-PythonQt-With-Tcl-NoCLI-Release	0	0	0	0	15	0	304	6	11 hours ago
eris.kitware	Linux-g++4.4-QT4.6.3-PythonQt-CLI-Release	1	0	0	0	15	0	36	451	3 hours ago
factory-ubuntu-64bits.kitware	Linux-g++4.4.3-QT4.7-PythonQt-With-Tcl-CLI-Valgrind-Release	0	0	0	0	13	0	27	460	11 hours ago
factory-ubuntu-64bits.kitware	Linux-g++4.4.3-64bits-QT4.7-PythonQt-With-Tcl-NoCLI-Coverage-Release	0	0	0	0	12	0	23	287	11 hours ago
sagarmatha.kitware	Linux-g++4.3.3-QT4.7-PythonQt-With-Tcl-NoCLI-Release	0	0	0	0	12	0	22	288	12 hours ago

Continuous

Site	Build Name	Update	Configure		Build		Test			Build Time
		Files	Error	Warn	Error	Warn	Not Run	Fail	Pass	
youpi.sci.utah.edu	OpenSuse-c++4.5.0-64bits-QT4.6.3-PythonQt-With-Tcl-NoCLI-Release	2	0	0	0	0	0	304	6	1 hour ago

Slicer is built every night on Windows, Mac and Linux platforms

Slicer Training events

- Hands-on training workshops at national and international venues
- More than 2,700 clinicians, clinical researchers and scientists trained since 2005

Slicer Training events

RSNA 2011

Major international conferences

- **RSNA** 2008, 2009, 2010, 2011, 2012, 2013, 2014
- **MICCAI** 2008, 2009, 2011, 2012, 2013, 2014
- **SfN** 2009, 2011
- **SPIE** 2012, 2013, 2014
- **CAOS** 2010
- **CARS** 2010, 2012, 2013

RSNA Activities

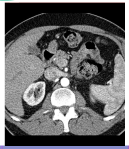
Hands-on refresher courses

- 3D Visualization of DICOM images for Radiology Applications
- Quantitative Imaging for Clinical Research and Practice

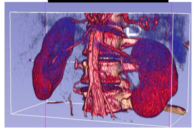
Quantitative Imaging Reading Room Exhibit

- 3DSlicer: An Open Source Platform for Segmentation, Registration, Quantitative Imaging, and 3D Visualization of Multi-Modal Image Data.

Overview

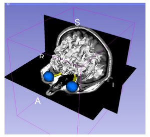


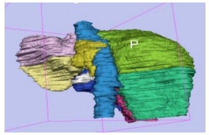
Part I: Introduction to the 3DSlicer software



Part II: 3D Data Loading and visualization of DICOM images

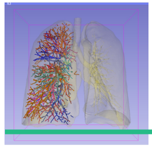
- Volume Rendering of thoraco-abdominal CT data
- Surface Rendering of MR head data



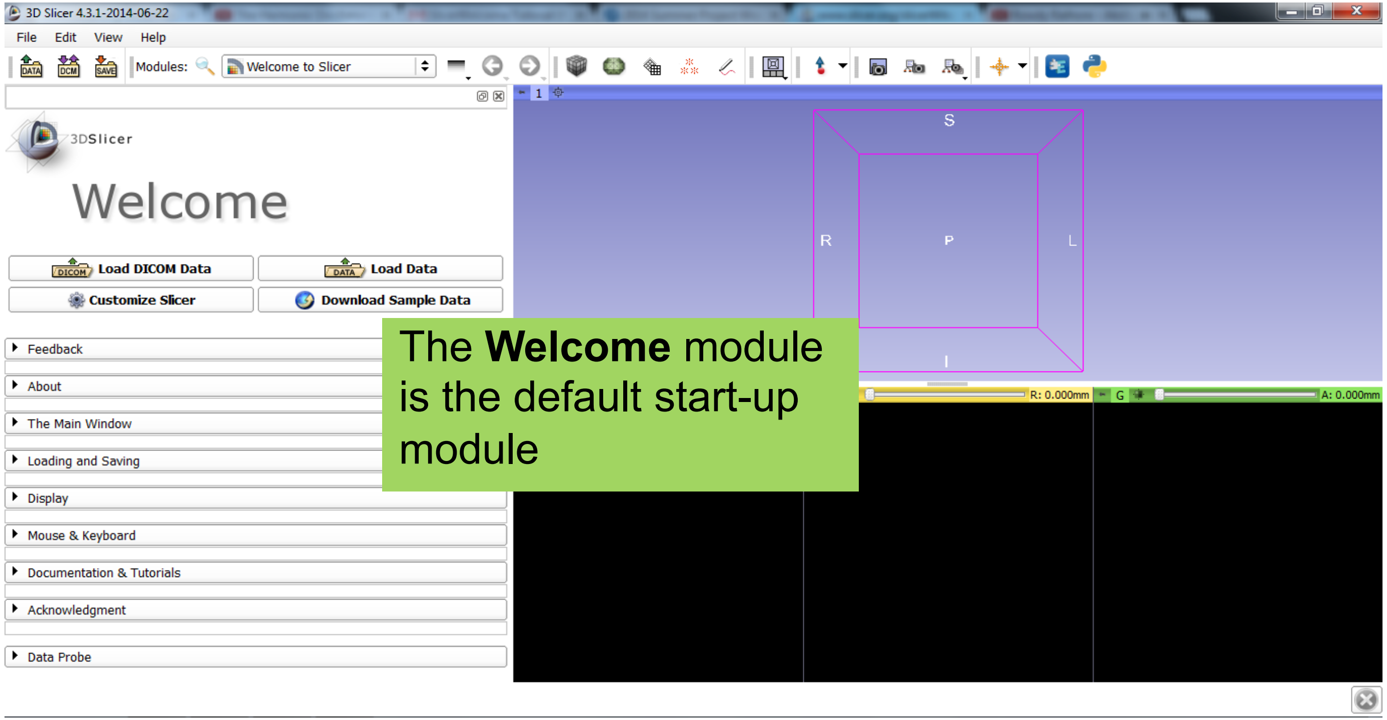


Part III: 3D interactive exploration of the anatomy

- Exploration of the Segments of the liver
- Exploration of the Segments of the lung

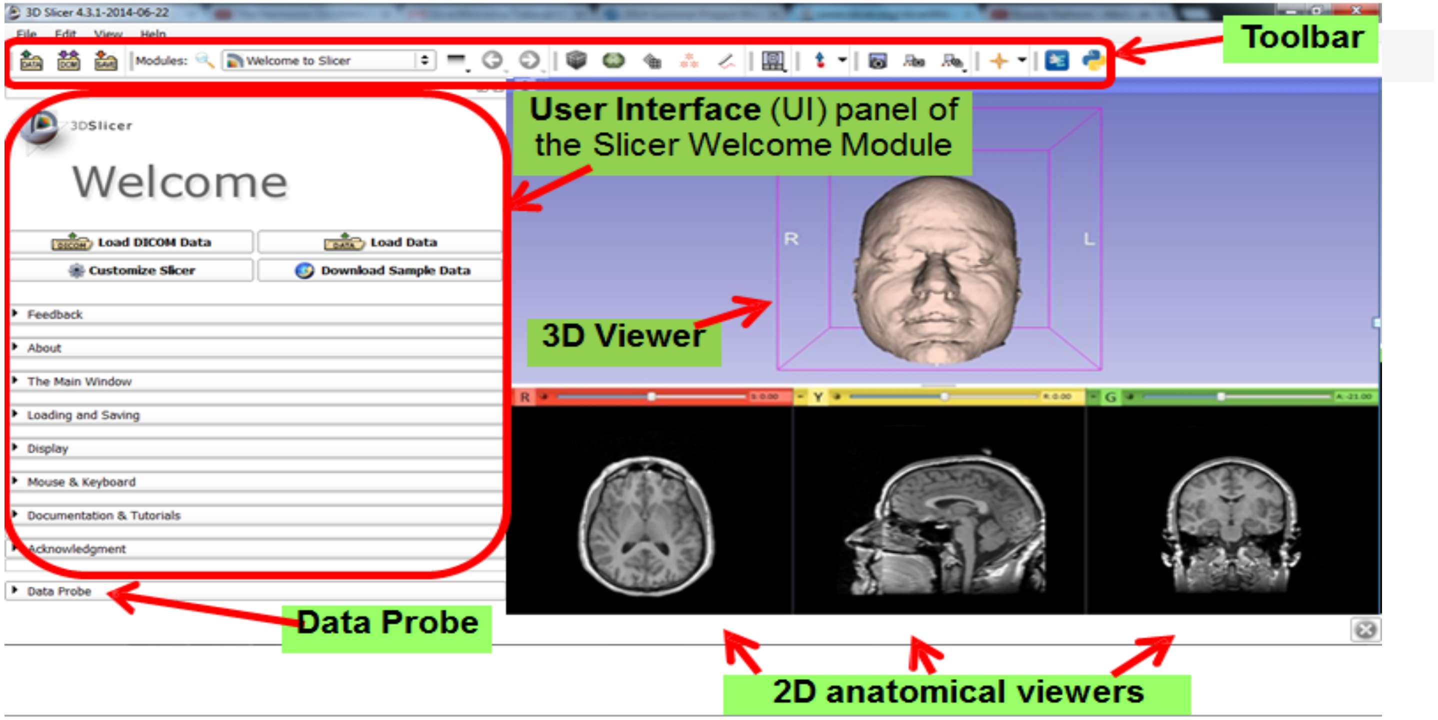


Welcome to Slicer4

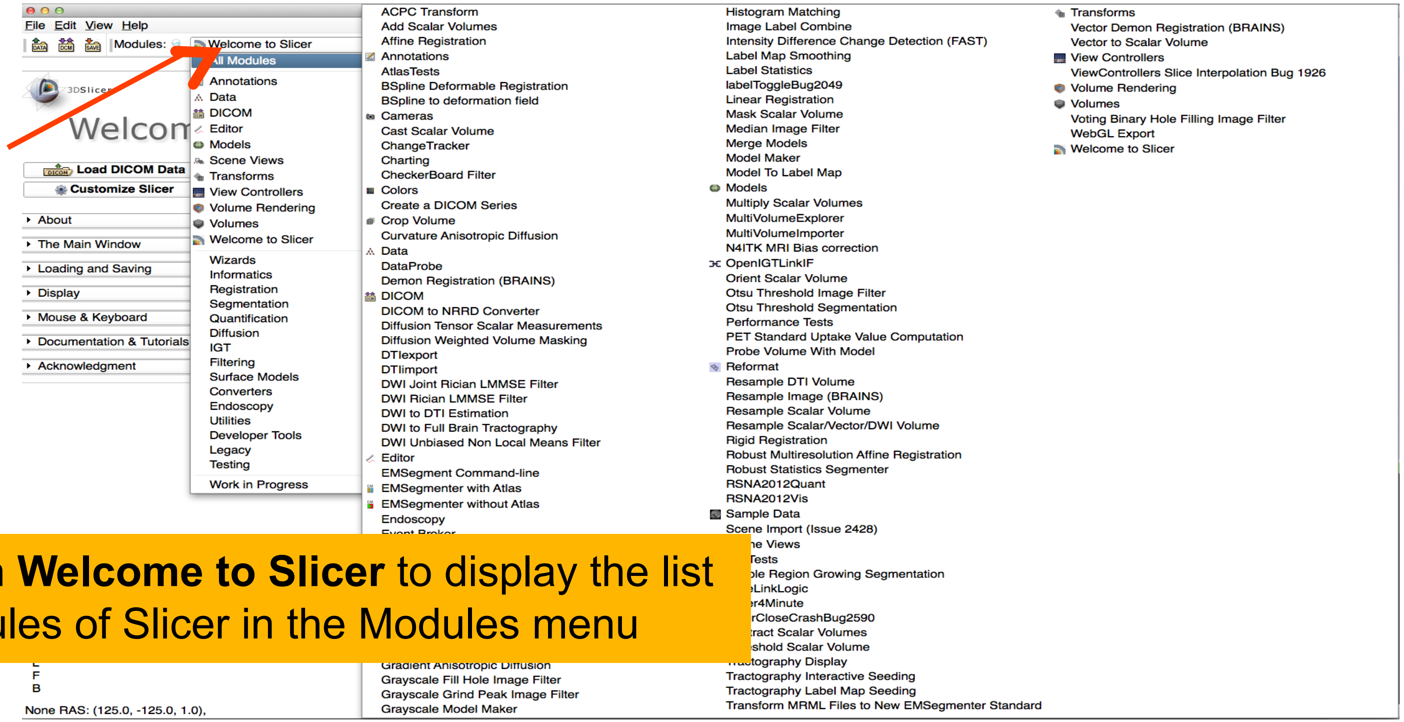


The **Welcome** module is the default start-up module

Navigating the Application GUI

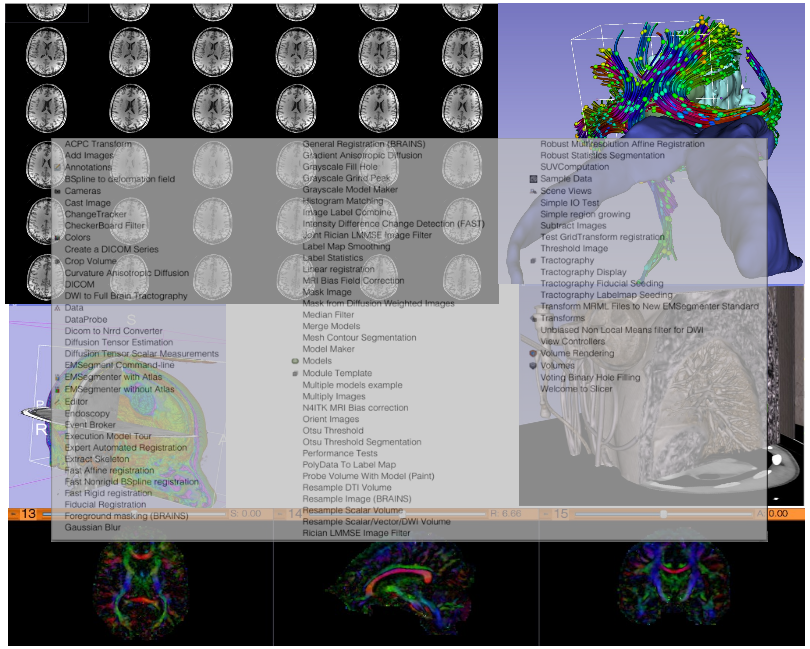


Welcome to Slicer4.3.1.1

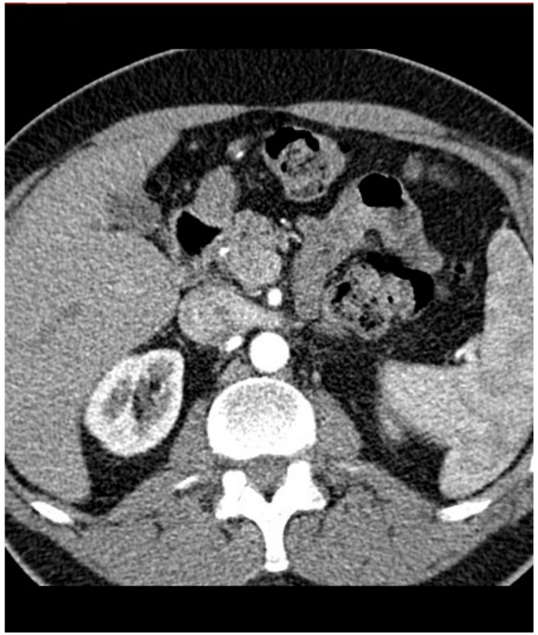


Click on **Welcome to Slicer** to display the list of modules of Slicer in the Modules menu

Welcome to Slicer4



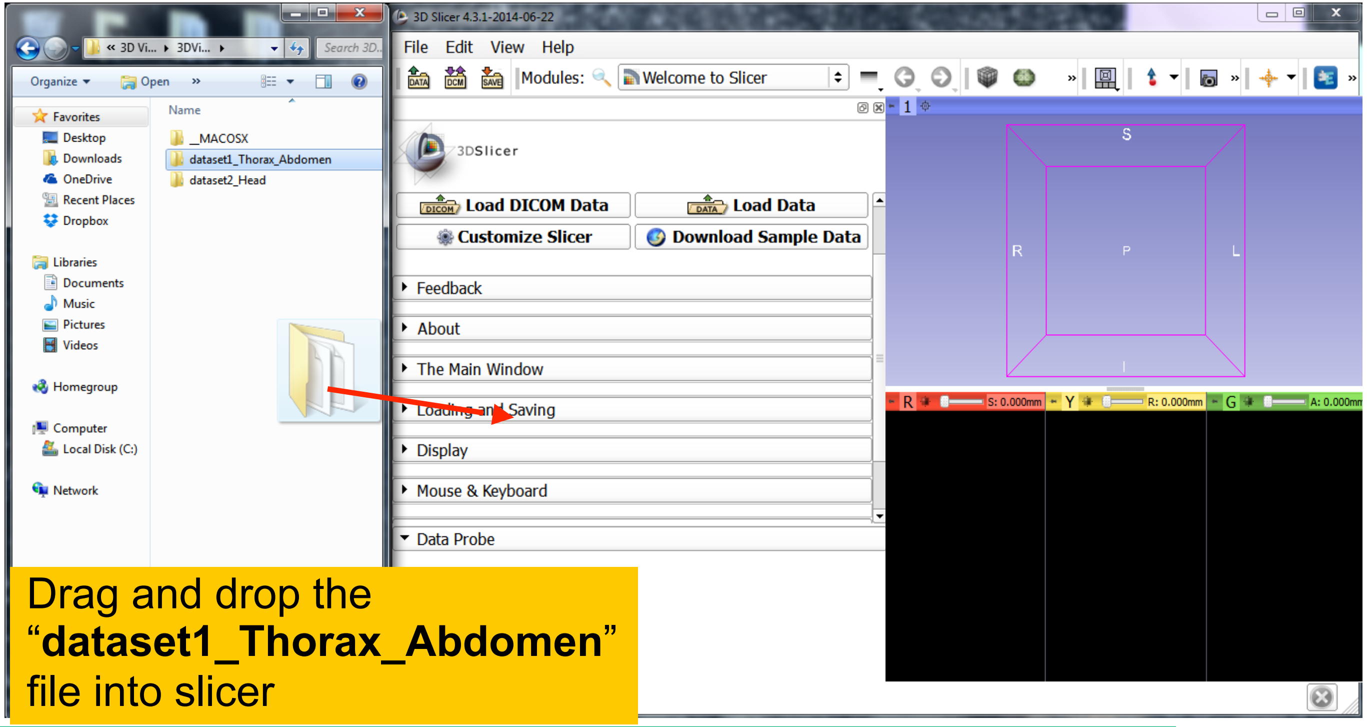
Slicer4.3.1 contains more than 100 modules for image segmentation, registration and 3D visualization of medical imaging data



Part 1:

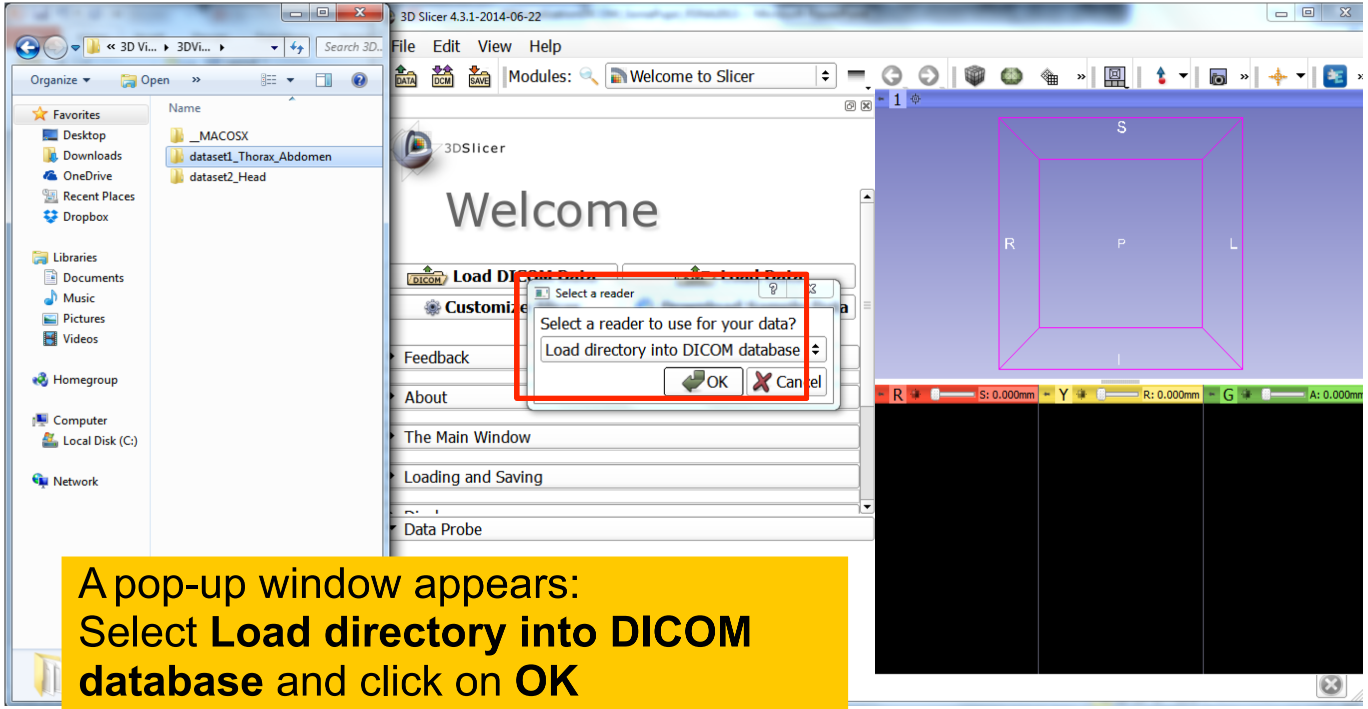
Loading a DICOM Volume

Loading a DICOM volume



Drag and drop the "**dataset1_Thorax_Abdomen**" file into slicer

Loading a DICOM volume



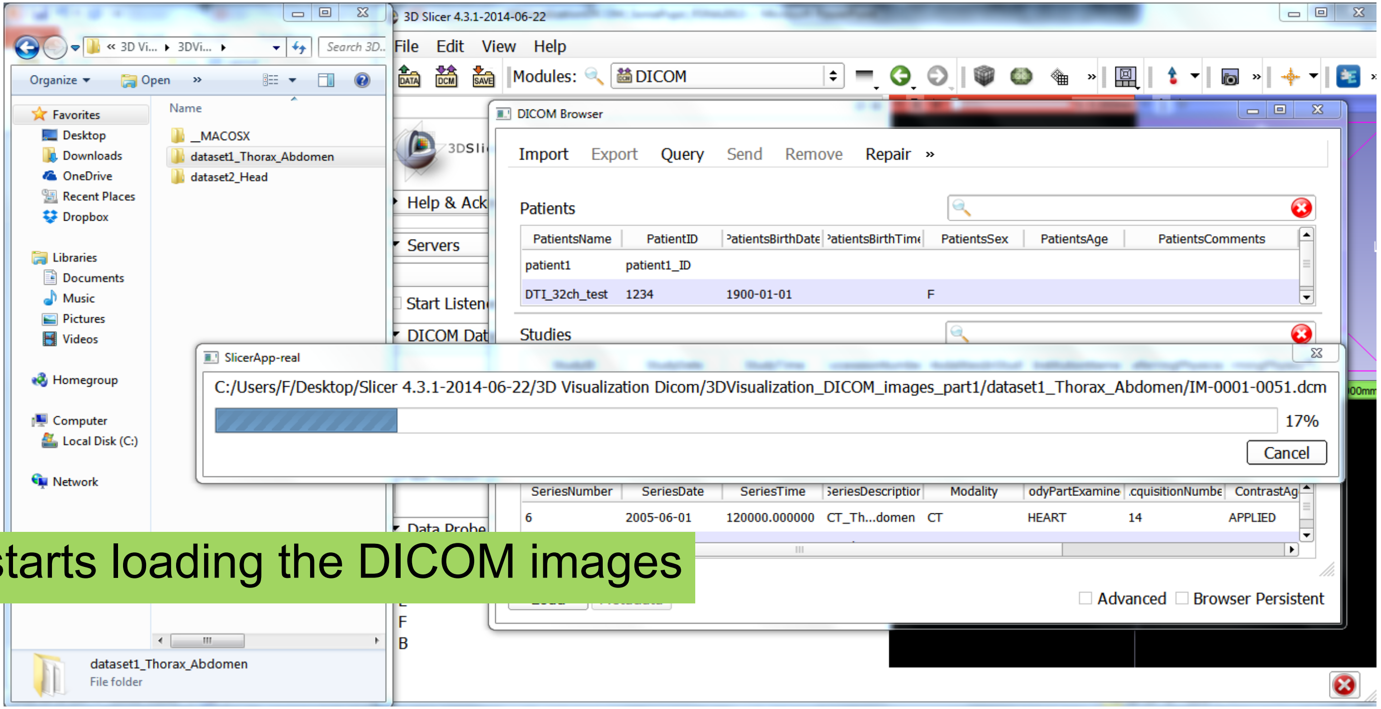
Select a reader

Select a reader to use for your data?

Load directory into DICOM database

A pop-up window appears:
Select **Load directory into DICOM database** and click on **OK**

Loading a DICOM volume



Patients

PatientsName	PatientID	PatientsBirthDate	PatientsBirthTime	PatientsSex	PatientsAge	PatientsComments
patient1	patient1_ID					
DTI_32ch_test	1234	1900-01-01		F		

Studies

SlicerApp-real

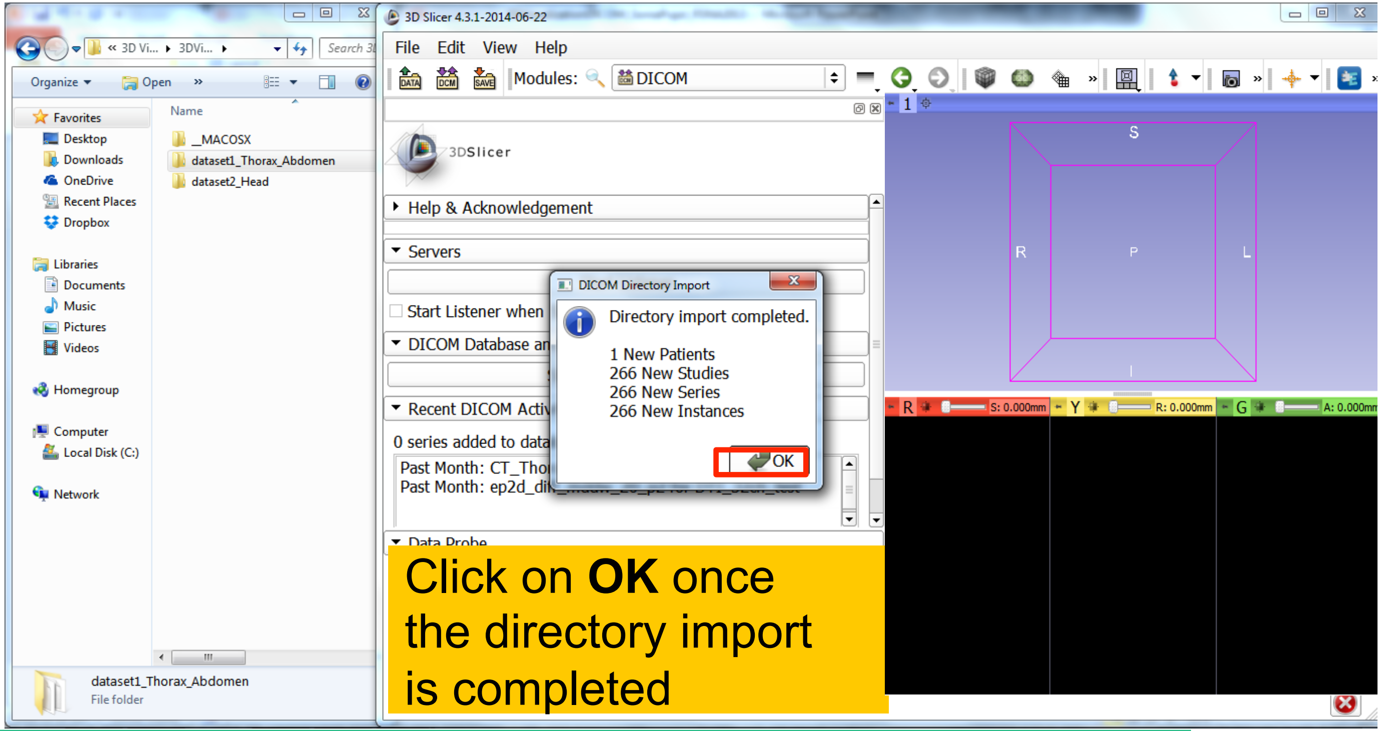
C:/Users/F/Desktop/Slicer 4.3.1-2014-06-22/3D Visualization Dicom/3DVisualization_DICOM_images_part1/dataset1_Thorax_Abdomen/IM-0001-0051.dcm

17%

SeriesNumber	SeriesDate	SeriesTime	SeriesDescription	Modality	BodyPartExamined	AcquisitionNumber	ContrastAg
6	2005-06-01	120000.000000	CT_Th...domen	CT	HEART	14	APPLIED

Slicer starts loading the DICOM images

Loading a DICOM volume



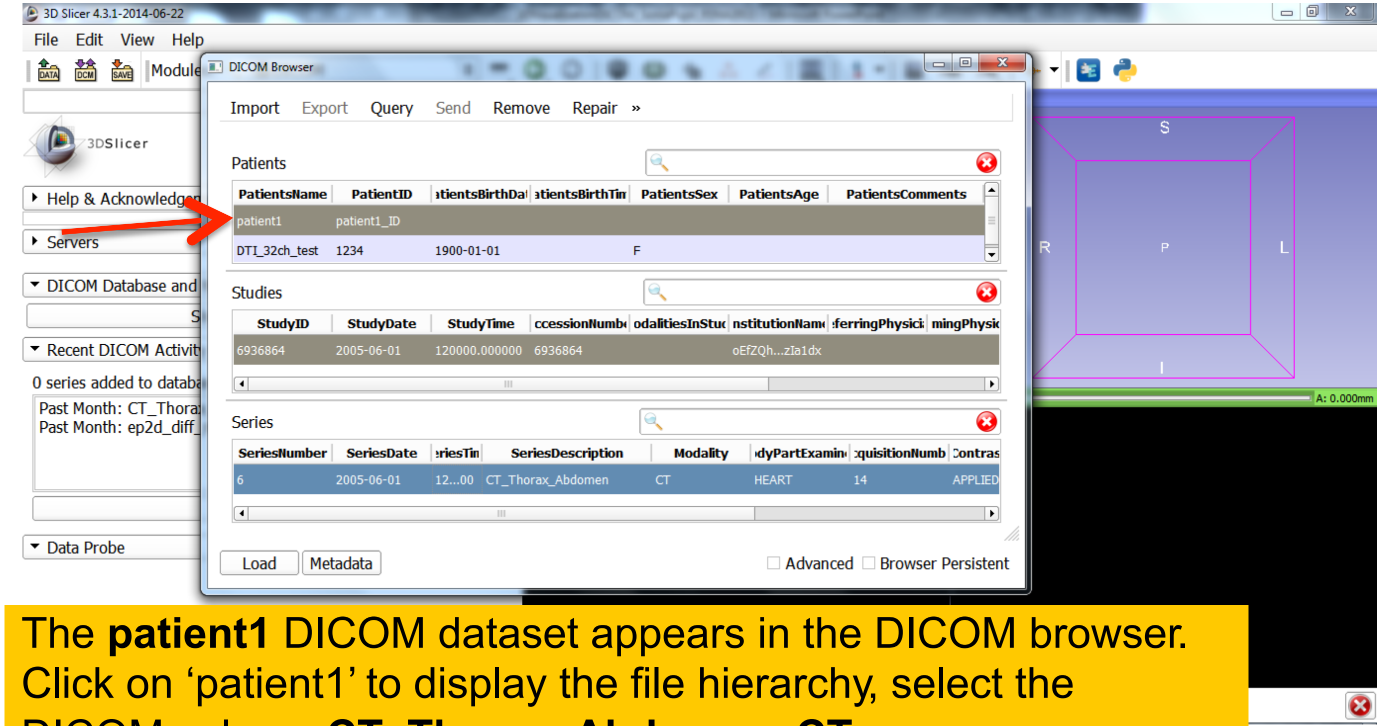
DICOM Directory Import

Directory import completed.

1 New Patients
266 New Studies
266 New Series
266 New Instances

Click on **OK** once the directory import is completed

Loading a DICOM volume



0 series added to databa

Past Month: CT_Thora
Past Month: ep2d_diff_

PatientsName	PatientID	atientsBirthDa	atientsBirthTin	PatientsSex	PatientsAge	PatientsComments
patient1	patient1_ID					
DTI_32ch_test	1234	1900-01-01		F		

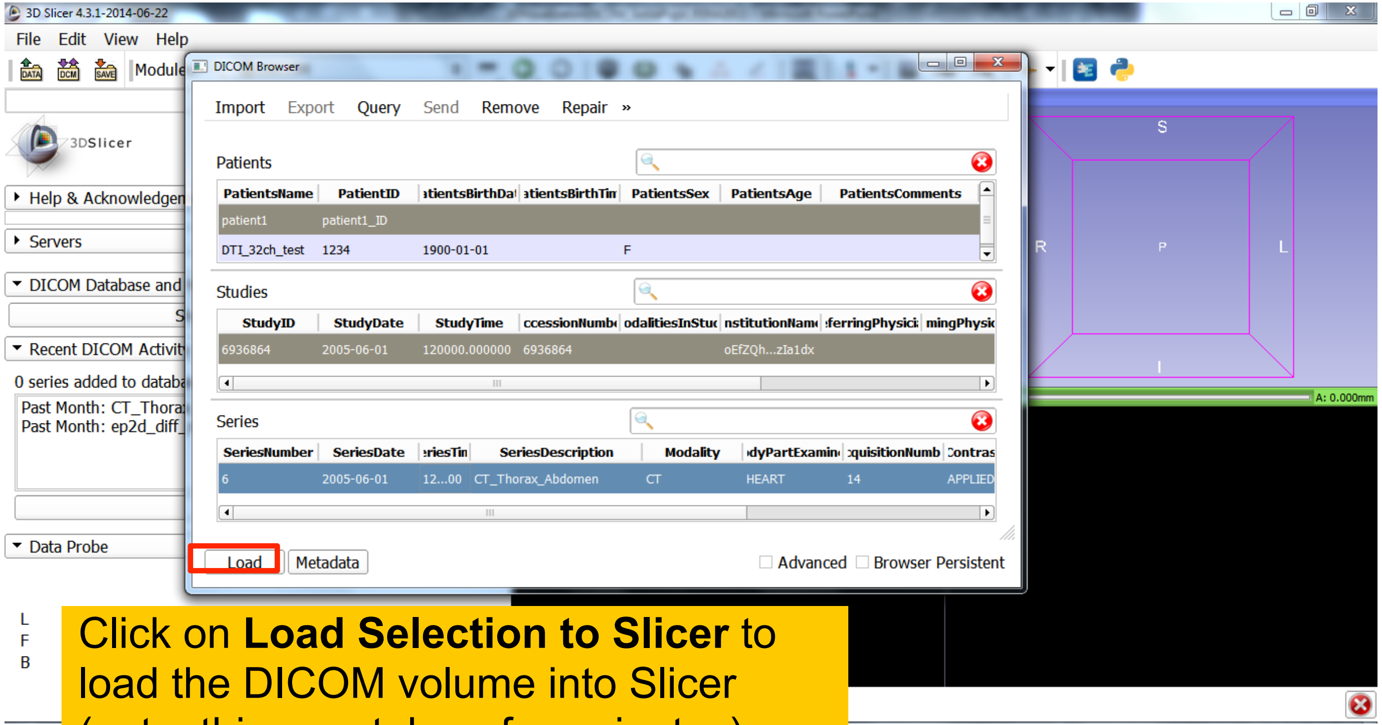
StudyID	StudyDate	StudyTime	ccessionNumb	odalitiesInStu	nstitutionNam	ferringPhysici	mingPhysic
6936864	2005-06-01	120000.000000	6936864		oEfZQh...zIa1dx		

SeriesNumber	SeriesDate	eriesTin	SeriesDescription	Modality	dyPartExamin	quisitionNumb	Contras
6	2005-06-01	12...00	CT_Thorax_Abdomen	CT	HEART	14	APPLIED

The **patient1** DICOM dataset appears in the DICOM browser. Click on ‘patient1’ to display the file hierarchy, select the DICOM volume **CT_Thorax_Abdomen_CT**

Slide 26

Loading a DICOM volume



Patients

PatientsName	PatientID	atientsBirthDa	atientsBirthTin	PatientsSex	PatientsAge	PatientsComments
patient1	patient1_ID					
DTI_32ch_test	1234	1900-01-01		F		

Studies

StudyID	StudyDate	StudyTime	ccessionNumb	odalitiesInStu	nstitutionNam	ferringPhysici	mingPhysic
6936864	2005-06-01	120000.000000	6936864		oEfZQh...zIa1dx		

Series

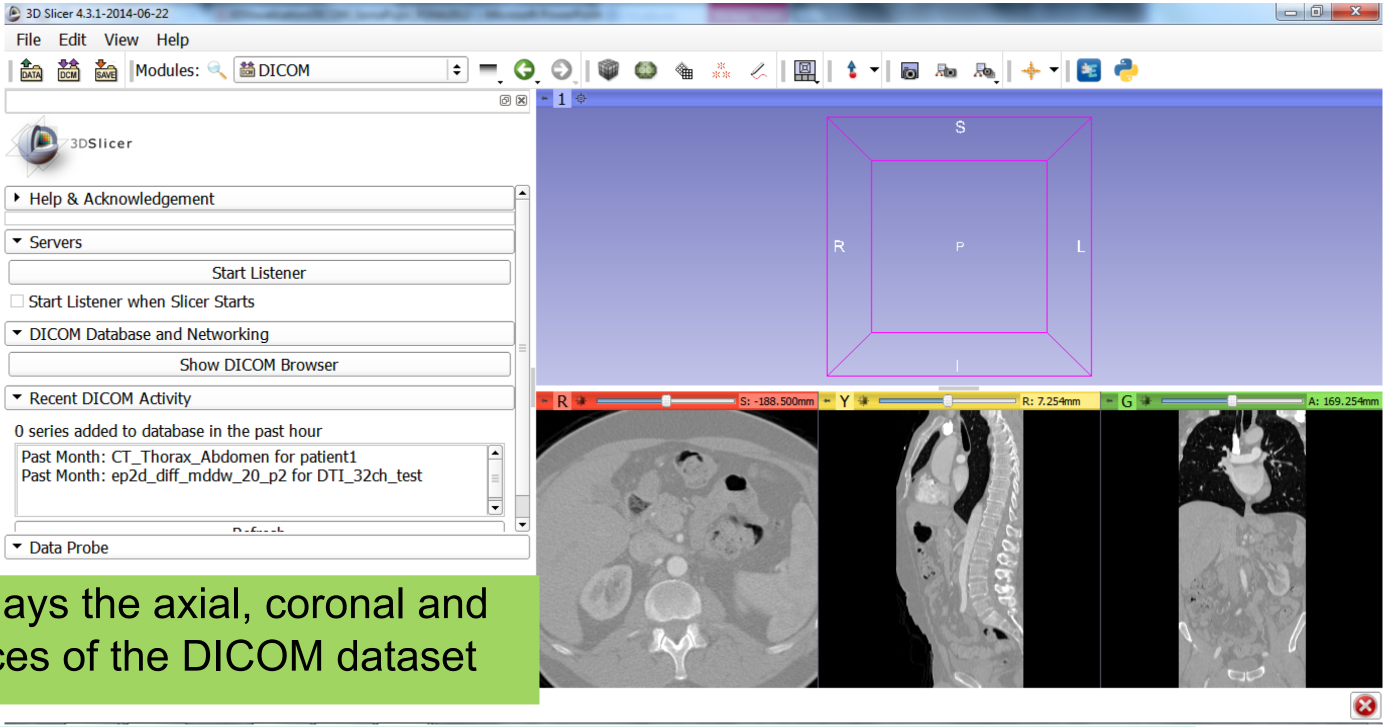
SeriesNumber	SeriesDate	eriesTin	SeriesDescription	Modality	dyPartExamin	quisitionNumb	Contras
6	2005-06-01	12...00	CT_Thorax_Abdomen	CT	HEART	14	APPLIED

Click on **Load Selection to Slicer** to load the DICOM volume into Slicer (note: this may take a few minutes)

©20

Slide 27

Loading a DICOM volume



Help & Acknowledgement

Servers

Start Listener

Start Listener when Slicer Starts

DICOM Database and Networking

Show DICOM Browser

Recent DICOM Activity

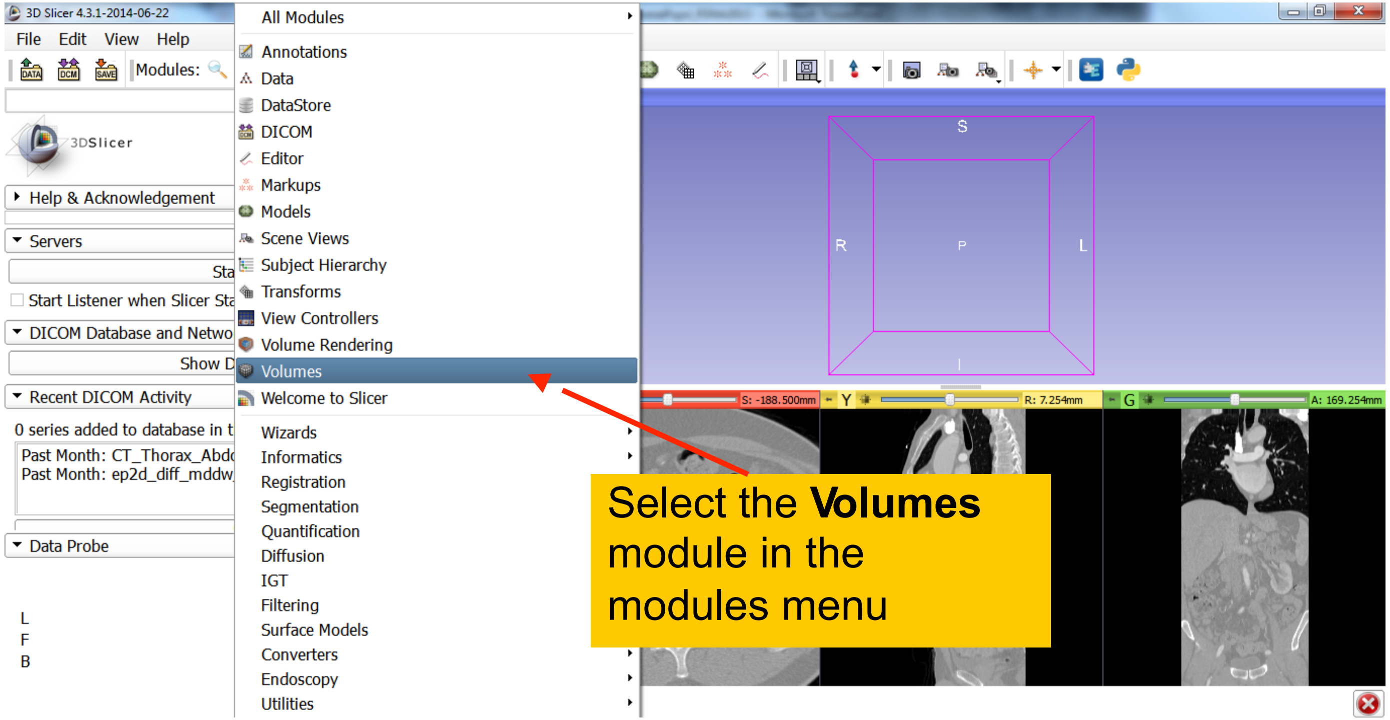
0 series added to database in the past hour

Past Month: CT_Thorax_Abdomen for patient1
Past Month: ep2d_diff_mddw_20_p2 for DTI_32ch_test

Data Probe

Slicer displays the axial, coronal and sagittal slices of the DICOM dataset

Loading a DICOM volume

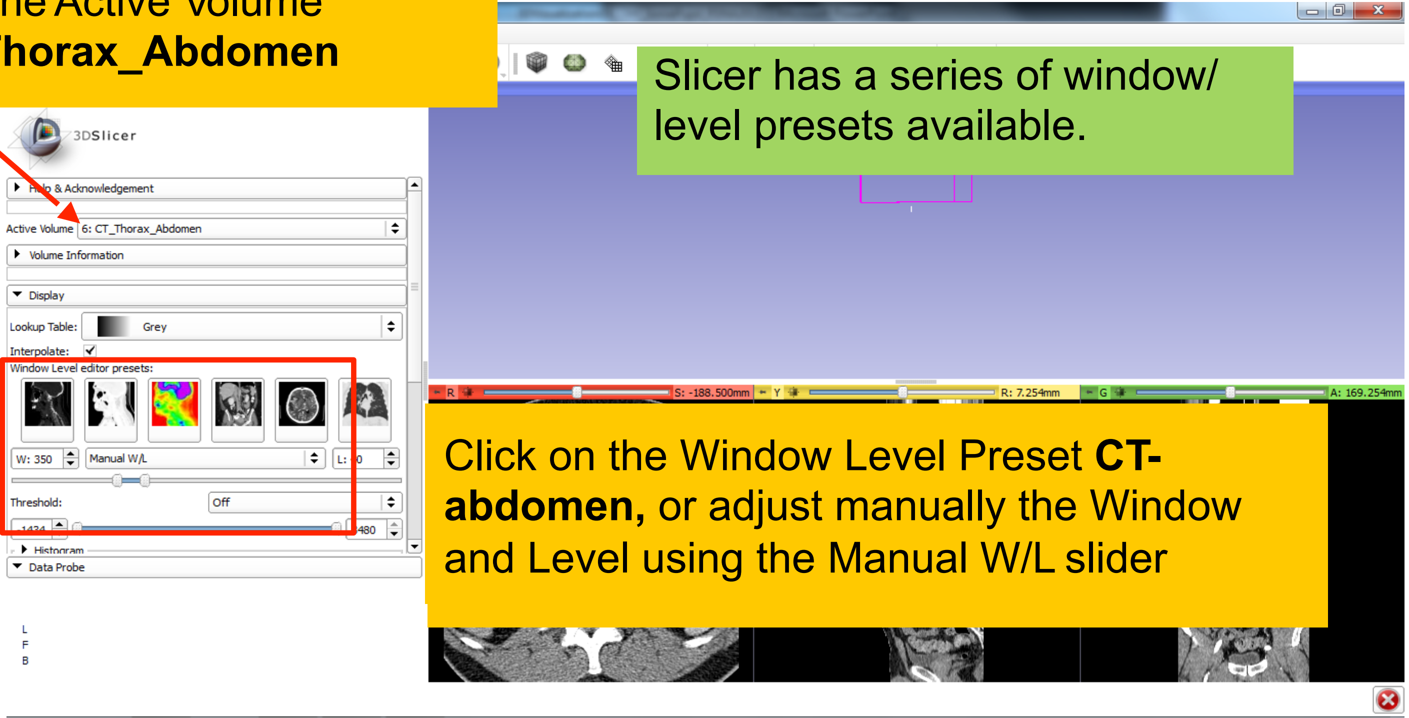


Select the **Volumes** module in the modules menu

Loading a DICOM volume

Select the Active Volume
6:CT_Thorax_Abdomen

Slicer has a series of window/ level presets available.

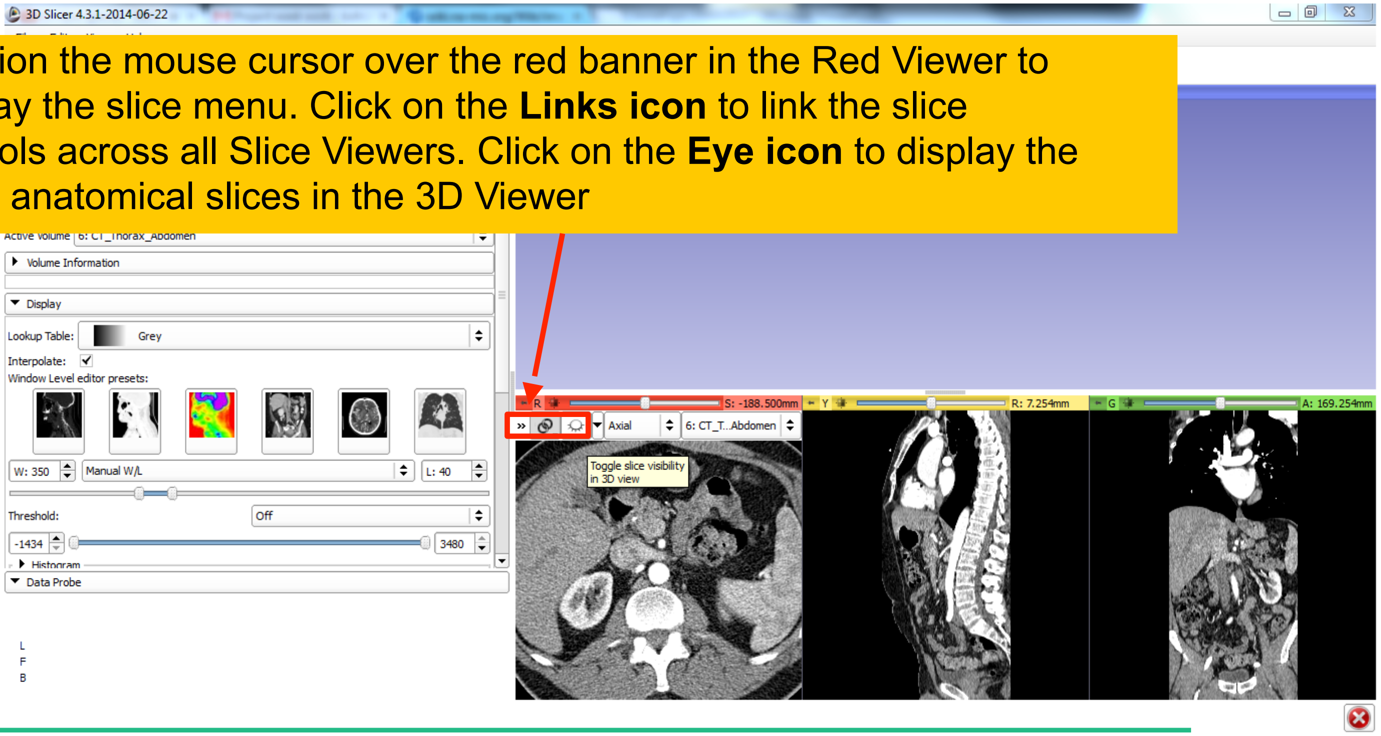


Active Volume 6: CT_Thorax_Abdomen

W: 350 Manual W/L

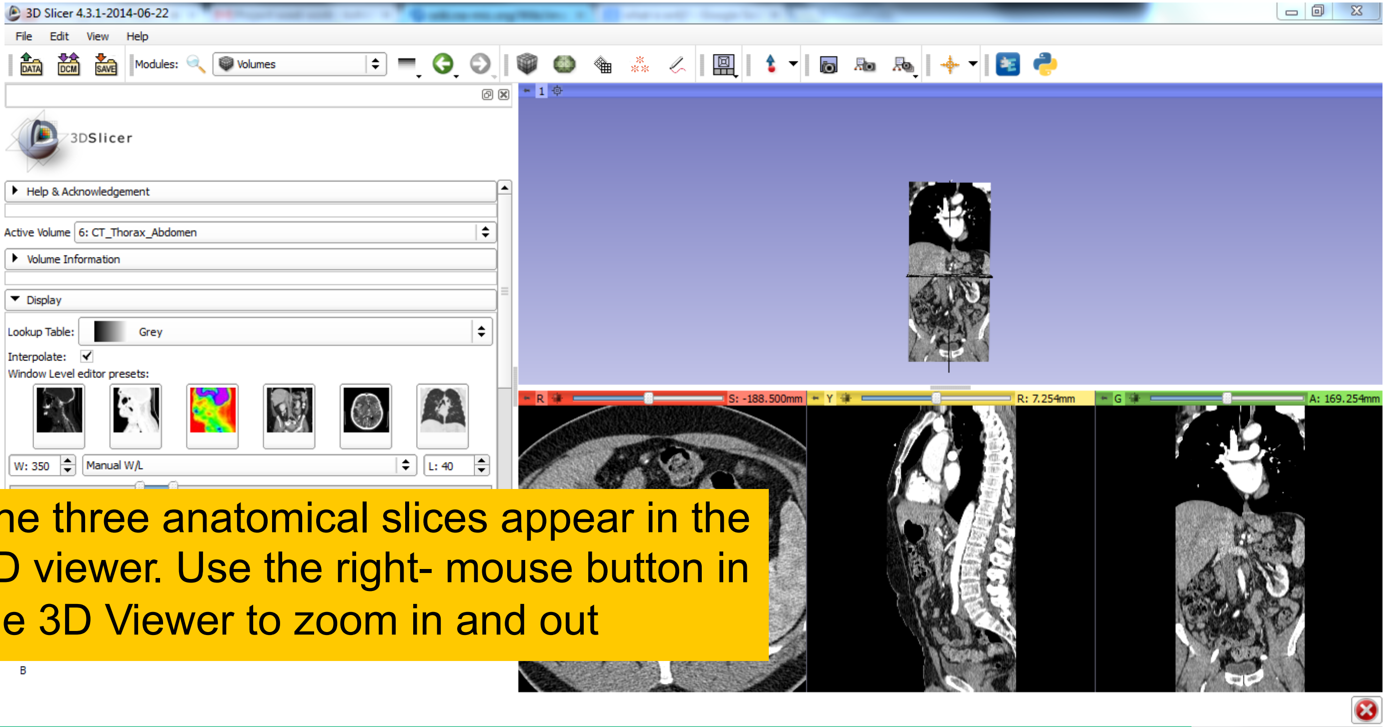
Click on the Window Level Preset **CT-abdomen,** or adjust manually the Window and Level using the Manual W/L slider

Loading a DICOM volume



Position the mouse cursor over the red banner in the Red Viewer to display the slice menu. Click on the **Links icon** to link the slice controls across all Slice Viewers. Click on the **Eye icon** to display the three anatomical slices in the 3D Viewer

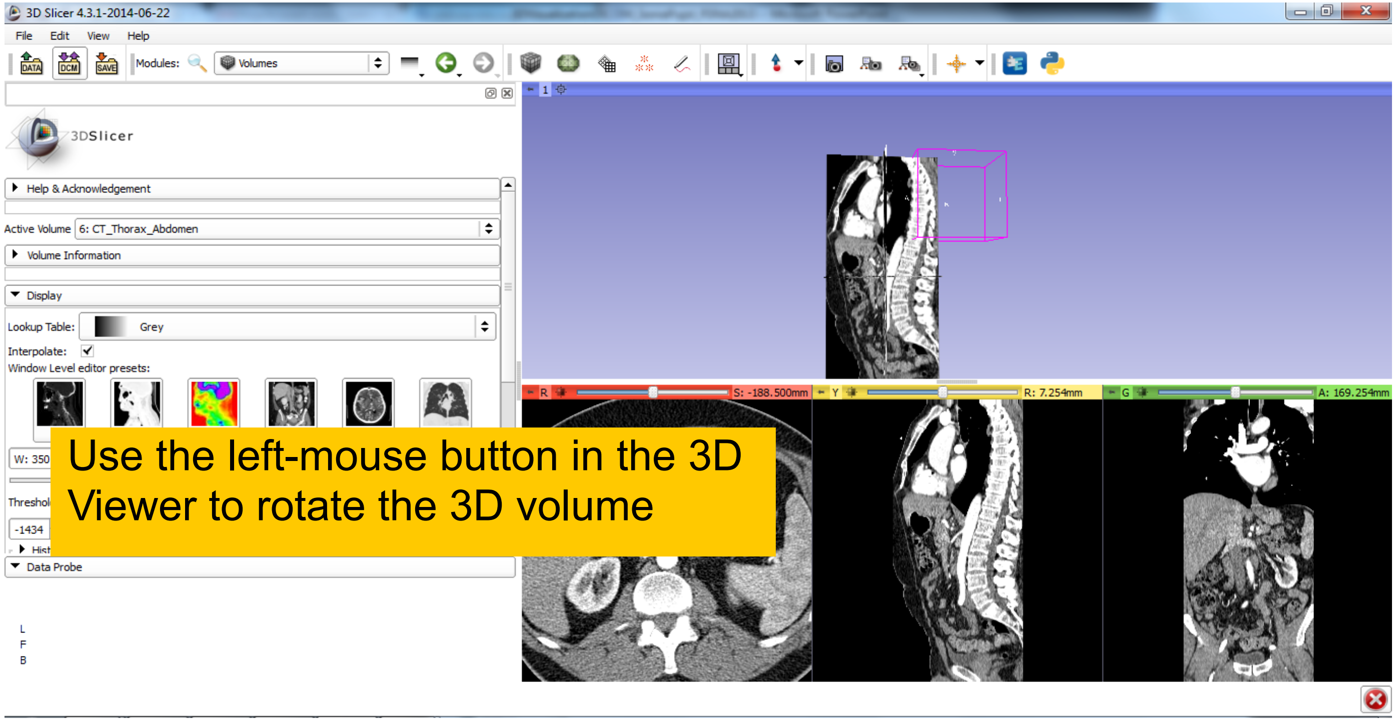
Loading a DICOM volume



Active Volume: 6: CT_Thorax_Abdomen

The three anatomical slices appear in the 3D viewer. Use the right- mouse button in the 3D Viewer to zoom in and out

Loading a DICOM volume

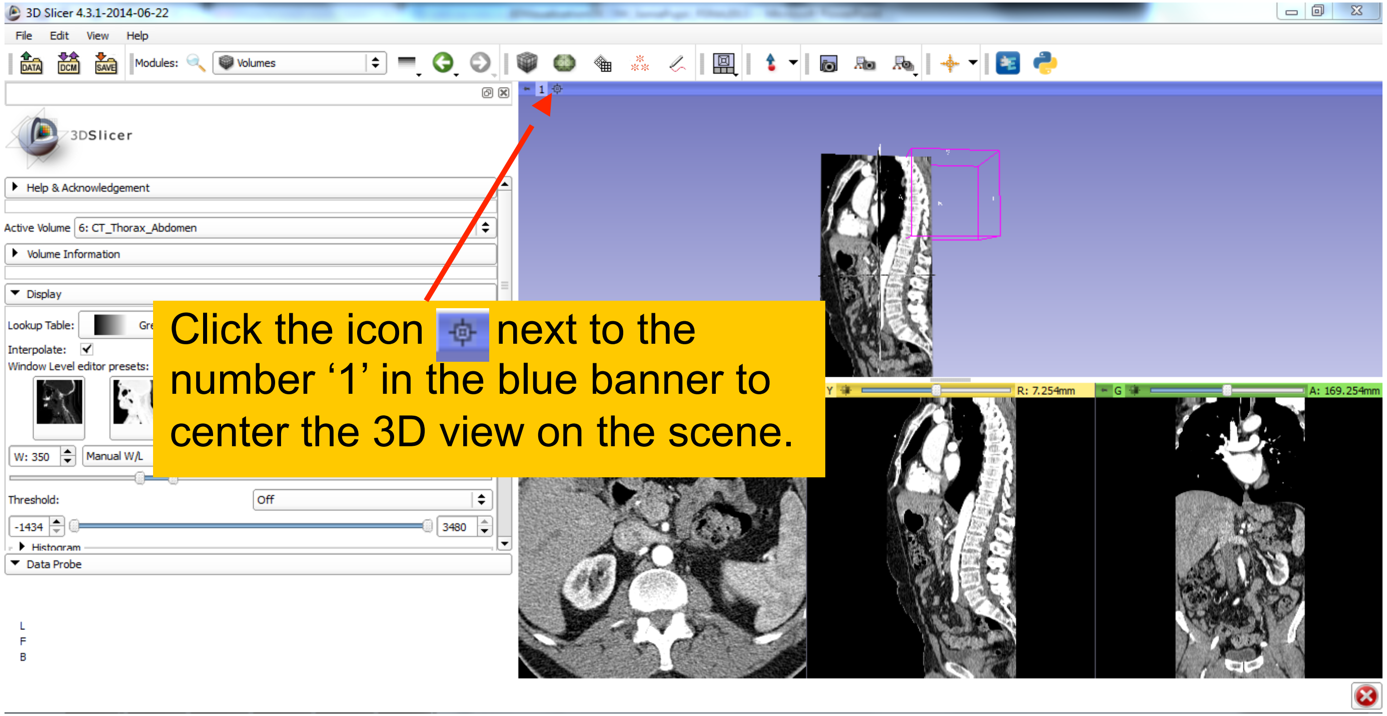


Active Volume: 6: CT_Thorax_Abdomen

Lookup Table: Grey

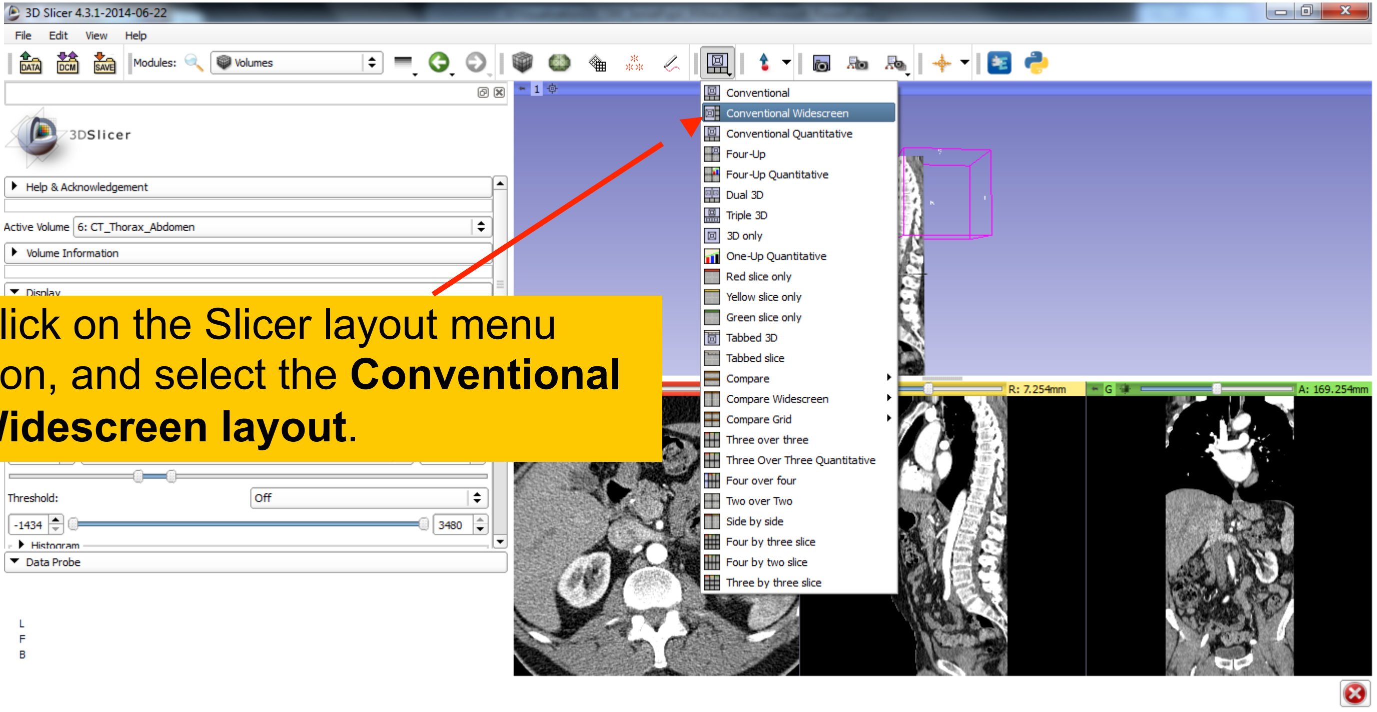
Use the left-mouse button in the 3D Viewer to rotate the 3D volume

Loading a DICOM volume



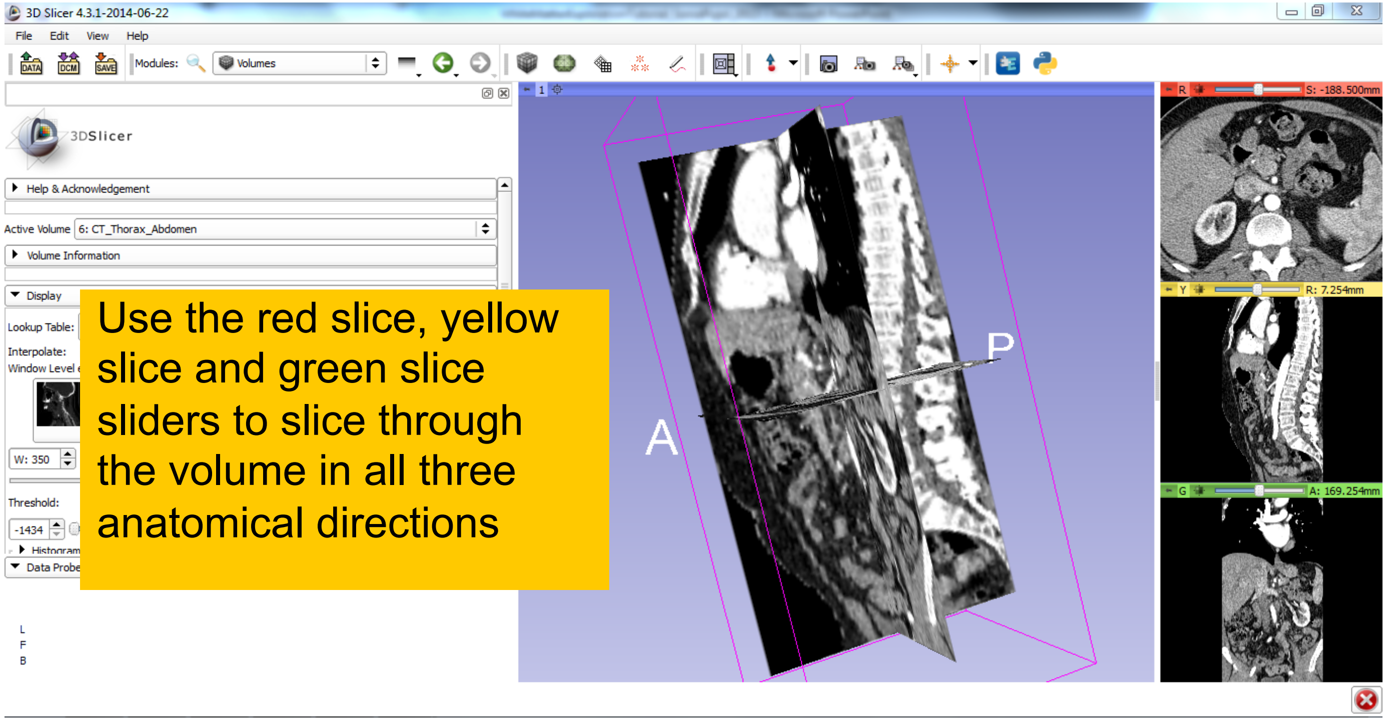
Click the icon ⌖ next to the number '1' in the blue banner to center the 3D view on the scene.

Loading a DICOM volume



Click on the Slicer layout menu icon, and select the **Conventional Widescreen layout**.

Loading a DICOM volume



Use the red slice, yellow slice and green slice sliders to slice through the volume in all three anatomical directions