

Installation

Obtaining

ImageVis3D is available on the memory stick provided to you as a workshop attendee. Beyond this tutorial, however, you can download ImageVis3D from <http://www.imagevis3d.org/>.

Windows

ImageVis3D supports Windows Vista and later. When installing ImageVis3D on Windows a setup program will guide you through installation on your computer. Simply double-click the installer and follow the on-screen instructions.

Mac

ImageVis3D supports Macs running OS X 10.7 (Lion) through 10.9 (Mavericks). Please make sure your Mac is fully updated; Apple frequently bundles graphics driver updates into system updates, and these can be important for the proper functioning of ImageVis3D.

Mac binaries of ImageVis3D are provided via the familiar “dmg” mechanism. Open up the DMG to mount the volume. You can then copy the ImageVis3D application anywhere you’d like; to install it, drag it to your ‘Applications’ directory.

Linux

There are a large set of Linux distributions and we do not have the resources to try every one of them.

Debian

ImageVis3D is part of Debian. Install it using the command `apt-get install -y imagevis3d`.

Ubuntu

ImageVis3D is part of recent versions of Ubuntu. Simply open the Software Center and search for “imagevis3d”, then install as you would any other package.

Other

The alternative method to installing ImageVis3D on Linux (required if you are not using Ubuntu) is the tarball method. Simply obtain the tarball and extract it with the 'tar' command:

```
tar zxvf ImageVis3D-3.1.0-Linux64.tar.gz
```

This will create an *ImageVis3D-3.1.0* directory and place the binary and supporting files in that same directory. Navigate to that directory and run **ImageVis3D** to start the software.

Loading and Converting Data

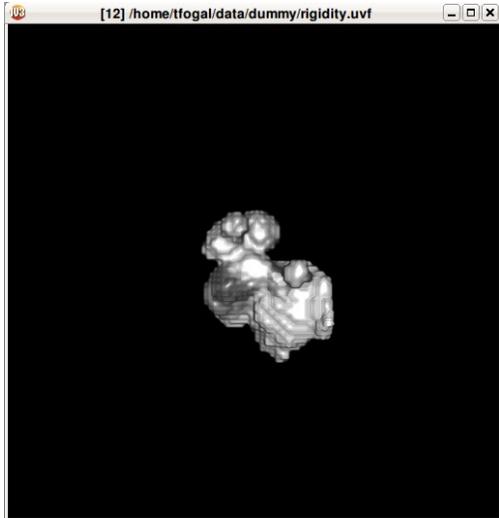
ImageVis3D was created to render very large data sets. It uses a multiresolution, level-of-detail representation of your data -- that is, it produces subsampled, low-resolution versions of your data. It packs these new resolutions together with your native resolution data into a format known as 'UVF'. When a rendering option is changed and ImageVis3D wants to give you fast feedback, it will use one of the lower resolution versions. During idle periods, ImageVis3D will render your native resolution data.

Display of lower resolutions is useful in that it allows you to immediately see the effect an adjustment has on your visualization, but has the unfortunate side effect that ImageVis3D can work only with UVF data. As such, all data must be converted to UVF before it can be rendered. ImageVis3D includes a large variety of built-in converters, and makes the process of adding new ones simple if you know C++. Look for the *Getting Data Into ImageVis3D* manual, included with the ImageVis3D binaries, if you are interested in learning more about the details.

For now, let's just load some data. Look in your CIBC data pack for the *rigidity.uvf* data set. It should be in the *software/ImageVis3D/data/* directory. To load it, you can drag and drop it into ImageVis3D, or navigate to it using the dialog provided by the *File | Load Dataset from File* menu.

Basic UI Elements

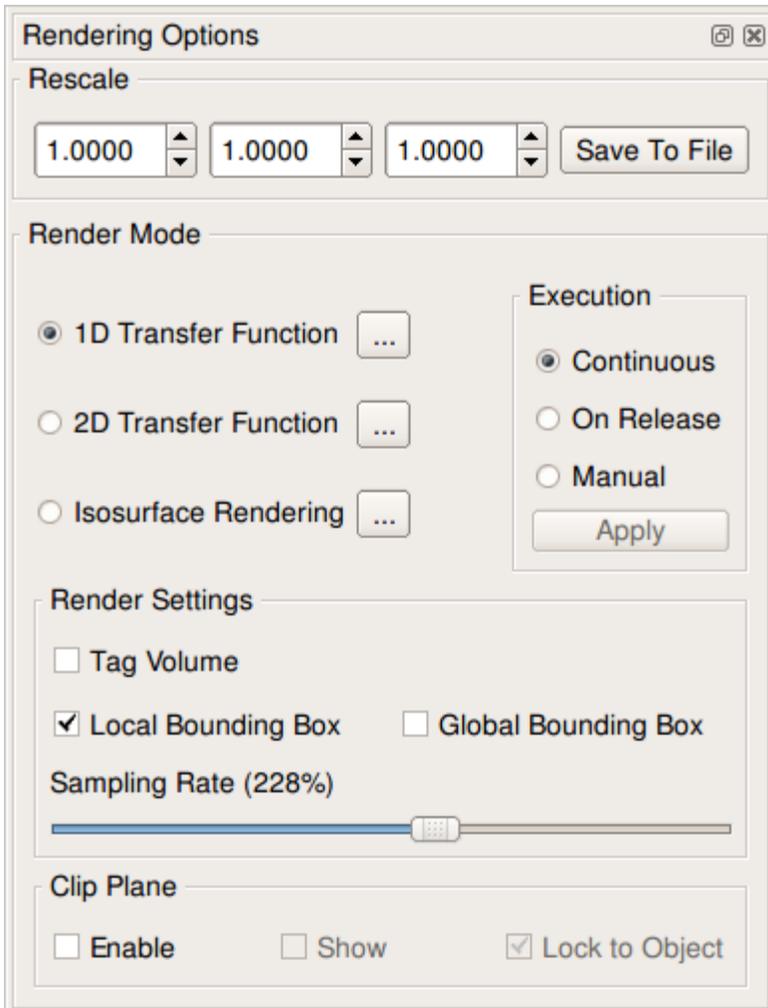
Now that we have some data loaded, ImageVis3D should look something like this:



Most interaction is controlled via the mouse. Hold the left mouse button to rotate the volume. Hold the right mouse button to translate the volume. The mouse wheel lets you zoom in and out, but you can use page up and page down to control the zoom level on a finer scale.

With a data set focused, hit the space bar to switch to a 2x2 view mode. Here we see the 3D view at the top left, the sagittal plane at the top right, the axial plane at the bottom left, and the coronal plane in the bottom right. You can left click the white lines in the middle to reorient these windows in any way you'd like. Try hitting the 'p' key now to bring up white boxes which

detail the plane of interest within the 3D visualization. When you are finished with the 2x2 mode, move your mouse over the view you would like and hit the space bar again.



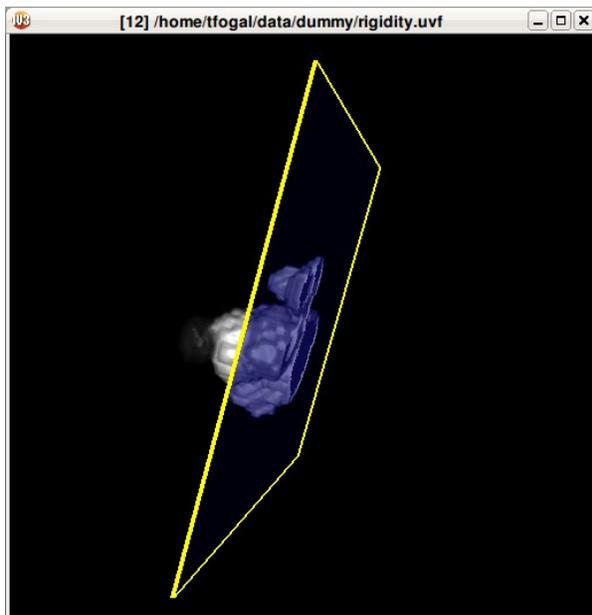
Rendering Options

Most of the UI is accessible from the "Workspace" menu. Any of the Ctrl-Alt-# entries (from "Rendering Options" and below in the 'Workspace' menu) open what we call a "Tool Widget": a tiny window which contains parameters relevant to your visualization. The first one we'll introduce is the "Rendering Options" widget you see directly to the left.

This widget controls the global properties of the visualization. At the top we have the ability to rescale the data in X, Y, and Z. This is useful when loading image stacks for which data typically have very high resolution in the XY plane but very little resolution in the Z direction; by cranking up the last number, you can adjust the anisotropy so that the data has realistic dimensions. In the middle we have the three primary render modes: two transfer function-based methods and an isosurfacing mode. We'll explore those in depth below. The execution model on the right is normally unimportant; leave it on "Continuous" for now.

Below that we have the "Render Settings". "Tag Volume" tells ImageVis3D not to interpolate, which is useful when you are loading a series of tags, as would be generated by a segmentation. The "Global Bounding Box" produces a red box around your data, which is particularly useful when you are looking at a small subset of your data, as it informs you where that feature lies relative to the overall volume. Generally you would leave the "Sampling Rate" at 100%, but when generating an image for a publication it can look nice to oversample a bit -- here I'm sampling the volume at about 2.3x the Nyquist rate.

Finally, ImageVis3D allows you to insert a clip plane to provide an easier method to look inside your data. Check "Enable" to turn it on; in your visualization, you'll immediately see a subset of a blue plane which details where the clip plane is:

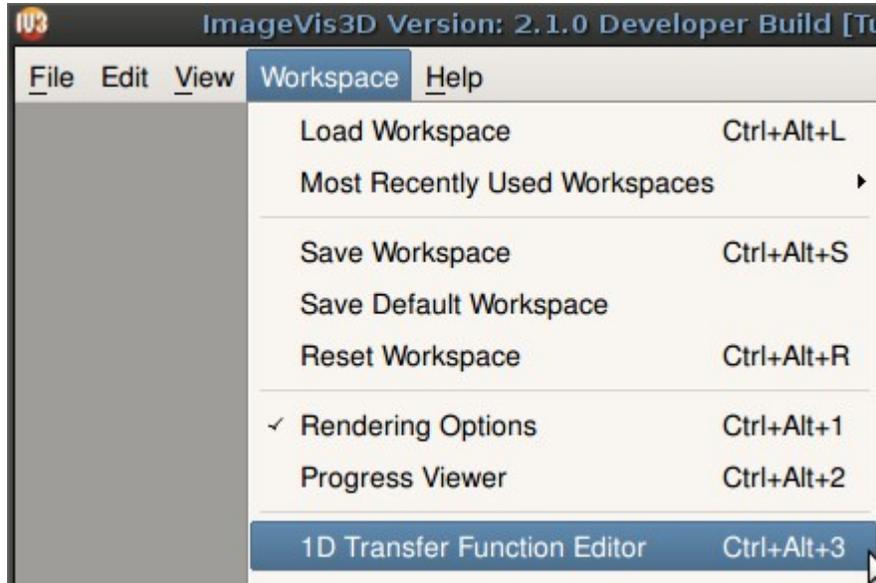


The clip plane is controlled via the same interactions as the data set, except that you must hold "Control" (on Mac: command) while you do so. Thus, to rotate the clip plane you could hold control + left mouse button, and move the mouse.

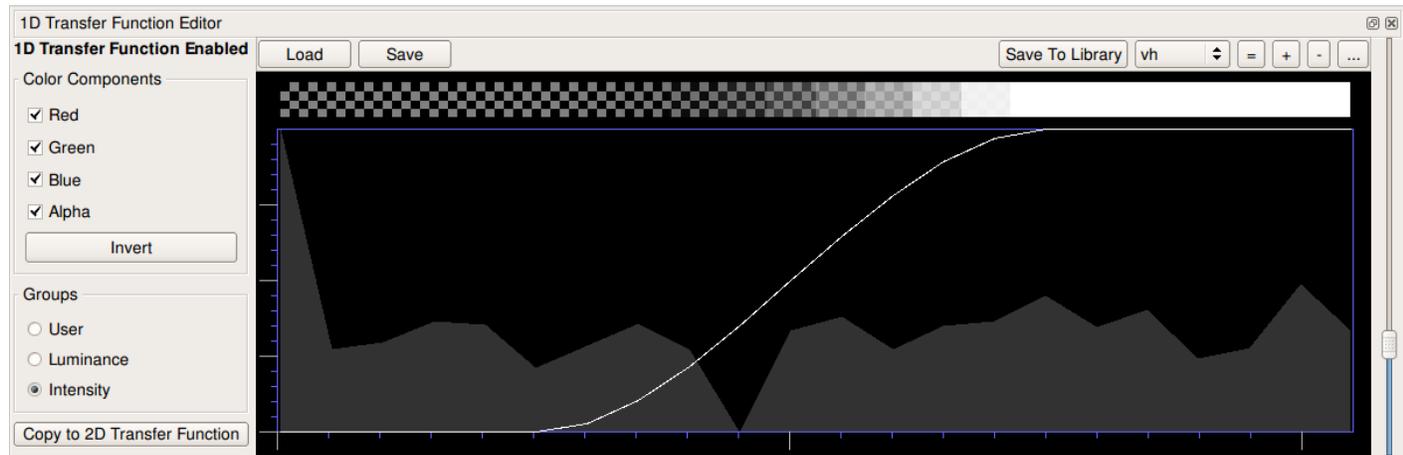
You can apply the clip plane without actually seeing its blue planar representation by unchecking the "Show" checkbox.

1D Transfer Function Editor

The 1D transfer function is the default method for visualizing your data in ImageVis3D. The method is based on ascribing *colors* to *data values*. Using the 1D transfer function editor, you can tell ImageVis3D such things as, “I want all of the 19s in my data set to appear blue” and “I want the range 42-108 to appear red”. By iterating over this process and inspecting the resulting visualizations, the goal is that you can highlight the features which are interesting in your data.



Let's open up the 1D transfer function editor now. To access it, you can either use the workspace menu or click the small  button in the “Rendering Options” widget that we just explored. When you do, you'll see a new window which has a gray-on-black background with some UI elements surrounding it:



The most important components of this window are what we call the “transfer function” -- the checkerboard-to-white pattern at the top of the dialog which is controlled by the white line in the middle. Try left clicking and dragging your mouse in that area to change the transfer function. You can use the right mouse button to recreate the smooth line of the default transfer function.

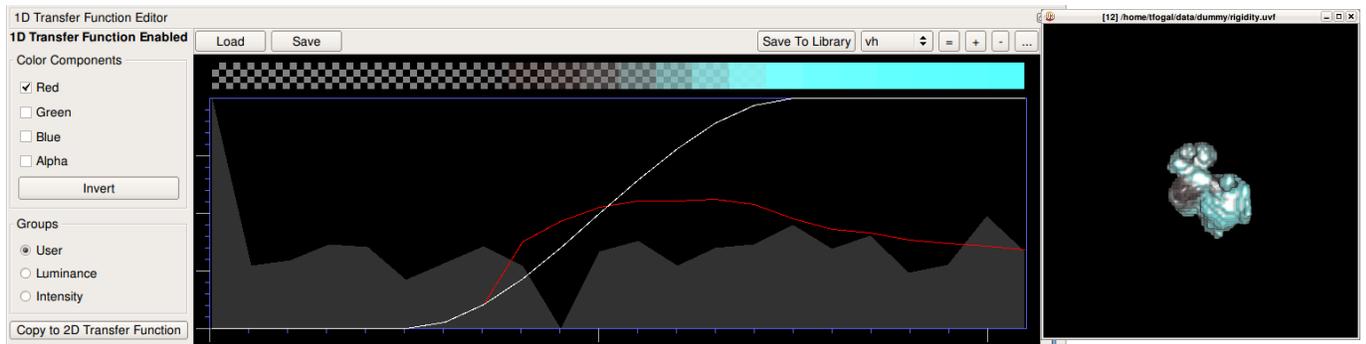
Notice that as you change the transfer function, your visualization is immediately updated as well. If you change the white line in an area where it hugs the bottom of the window, you'll see that new components of your data become visible. If you move the line from the top to the bottom, you'll notice aspects of your data disappear. In between those two extremes, you'll get a gradient of transparency -- near the top will create visualizations that are hard to see through,



and near the bottom will create tenuous visualizations with considerably more depth.

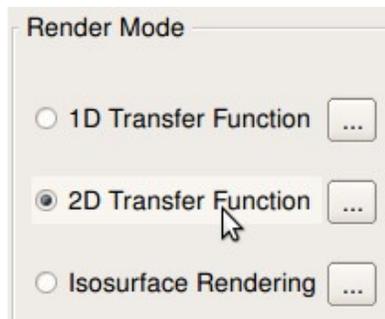
Up to now, we have been changing all of our color channels in concert. The line we have been modifying appears white because the channels we are modifying includes the "Alpha" channel. On the left of the dialog, however, you can select which color channels are 'active' individually. By changing the channels to be independent of one another we can add color to our visualization. Try selecting only the "Red" channel now, by

unchecking the "Green", "Blue", and "Alpha" channels. Now click and drag in the transfer function to create something that looks like this:

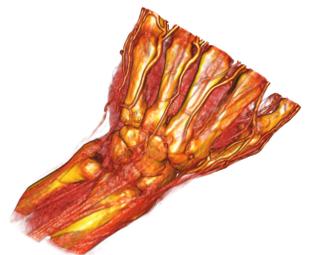


Note that the visualization takes on a cyan hue, just as the transfer function does at the top of the 1D transfer function editor. This is because the "Red" channel was *underneath* the "Alpha" channel. By dragging it halfway down, we have changed the 'red' in that area from 100% to 50%. However, the 'blue' and 'green' in those regions is still up at 100%, underneath the "Alpha", and the combination of those two colors produces the cyan you see now.

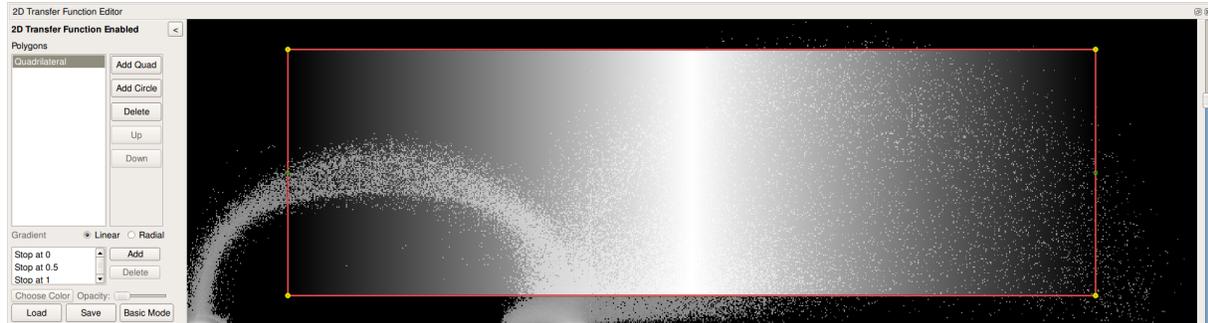
2D Transfer Function Editor



Let's try out the 2D transfer function editor now. Close any data you have open, including the 1D Transfer Function editor, and then navigate on your USB stick to find and load *hand16.uvf* (pictured on the right). In the "Rendering Options" widget, switch to using the 2D transfer function by clicking the radio button next to the "2D Transfer



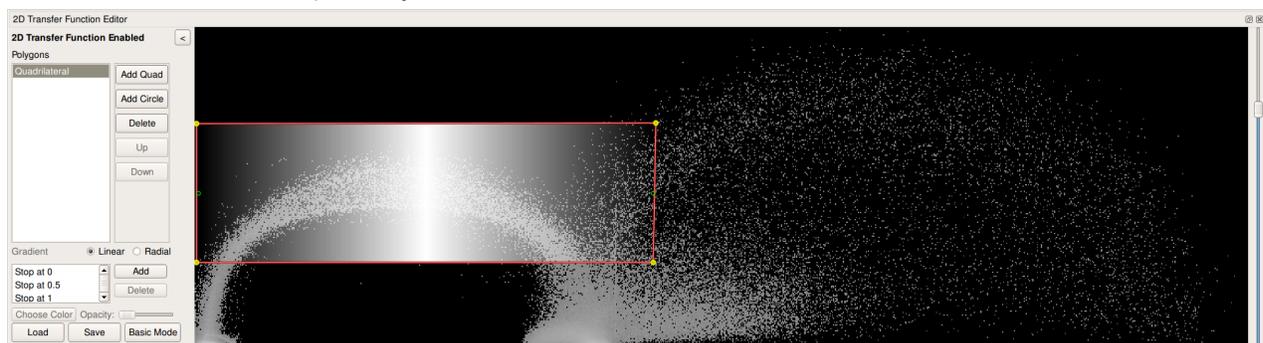
Function” render mode. Next, click the  button on the right of “2D Transfer Function” to bring up the 2D transfer function editor. Somewhat similarly to the 1D Transfer Function editor, you’ll be presented with a window that shows the transfer function on the right and some UI widgets on the left if you are in expert mode. You will see UI widgets on the bottom of the window if you are in basic mode. If you are in basic mode, click on the 'expert mode' mode button at the bottom of the 2D Transfer Function window. Once expert mode is enabled you will see the following:



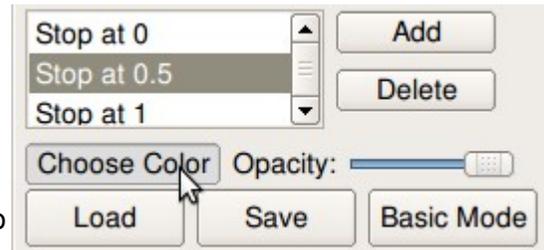
The default transfer function is simply the large white box you see in the middle, outlined by the red lines. As a backdrop, you are looking at a histogram of your data just like in the 1D case, except now we have the added dimension of *gradient magnitude* on the Y axis. This magnitude measures the difference between neighboring voxels. The theory is that high gradient data values will cluster around a boundary in the data, and low gradients will indicate a single material. The red-outlined box is the transfer function itself: all the data values it covers will appear in the visualization.

First let’s learn how to manipulate the widget we see there. A simple left click selects the nearest vertex (yellow dot) and allows you to change the shape of the polygon. Right clicking near the red line will add a new vertex, and right clicking on one of the yellow vertices will remove it from the polygon. With just these interaction techniques, you can shape the polygon in any way you see fit.

You’ll quickly find that it will be quite difficult to generate an appropriate transfer function just by moving things around in this manner. Hold shift and then left click to move the entire polygon. Hold control (Mac: command) and move the mouse up and down to scale the polygon. Using these interaction techniques, try to create this transfer function:



Note that this highlights the skin. Let's try changing the color from white to something similar to skin tone. Highlight the "Stop at 0.5" list item on the left, and then click the "Choose Color" button. This will bring up a color selection dialog and allow us to change the color of this particular area in the transfer function. Choose a color for the skin.

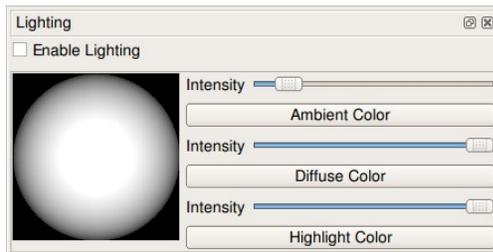


Now move up from this area and click the "Add Quad" button to add a new quadrilateral to our transfer function. A new (white) quadrilateral will appear in the center. Try to move this one around to highlight the bone before moving on, creating this visualization:



Lighting

Lighting can be 'hit and miss' when applied to volume rendering. In most instances, it can really improve the understanding of surface curvature, but in some cases it can



distract from a feature of interest. Generally this occurs when there is little depth in a rendering, or a boundary within the volume is rough or noisy, causing inconsistent lighting.

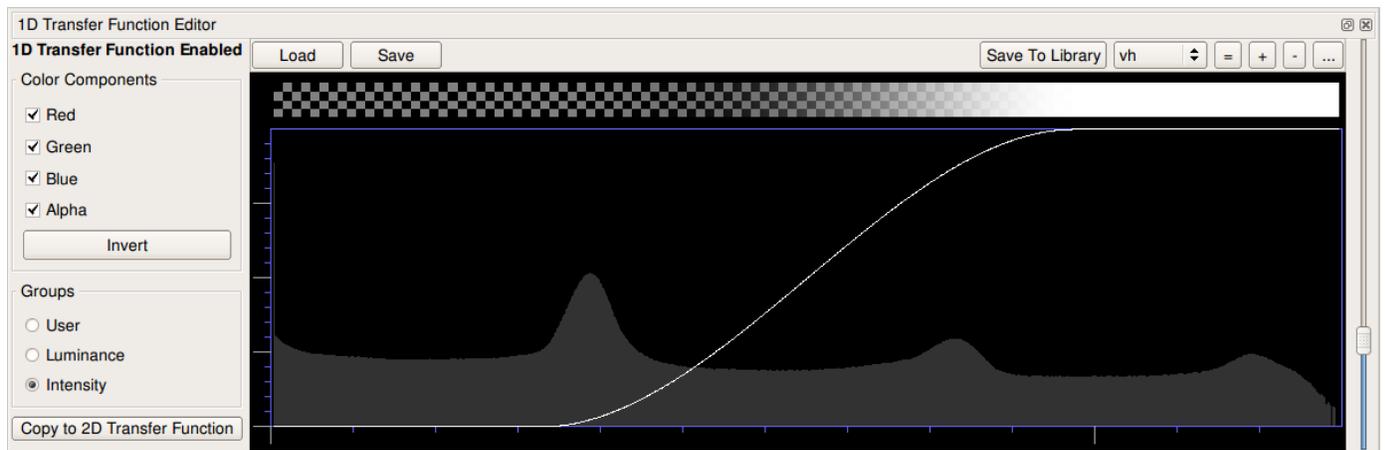
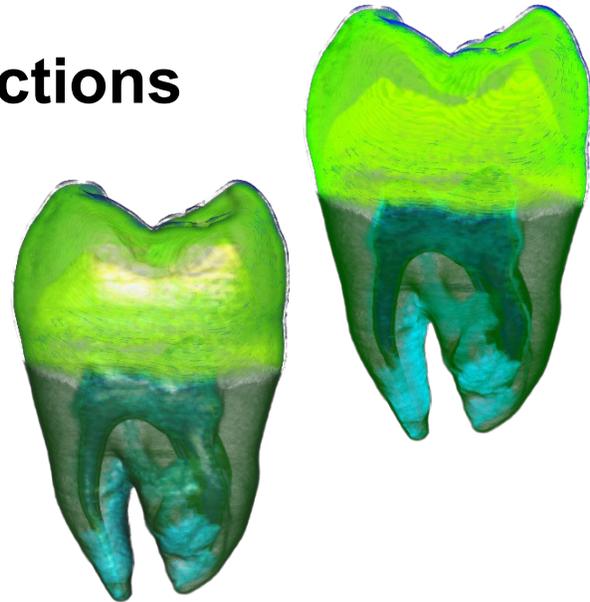
The lighting dialog allows one to control these interactions. In most cases, you will find that just disabling lighting entirely (via the check box at the top left) is what you are looking for. However, it can be useful to play with the other lighting parameters in some instances as well. One 'trick' is to turn off the 'Highlight Color' by dragging the bottom-most slider all the way to the left. This has the effect of disabling the highlights one

receives when a light hits a surface “just right”. One can also drag the light position around the ball on the left to reorient it with respect to the data. Both of these techniques can help to alleviate the issue of a highlight drowning out the feature you want to convey to someone else.

Designing 1D Transfer Functions

Designing an informative transfer function is a mixture between an art and a science. Most users find it difficult to do after just the introduction to ImageVis3D you are receiving today. With continued use and experience with different kinds of data, you will find this process becomes easier. Here we will go over some of the ‘tricks of the trade’ that will help you get a hang of what’s going on more quickly.

Let’s start by opening up the tooth.uvf data set and then opening the 1D Transfer Function editor:



The gray on top of the black represents the *distribution* of data values, but says nothing of the *range*. For the design of a transfer function the range is usually unimportant, but it will be a useful guide for our discussion of this data set, so the range of these data is from 0 to 1300¹. That means that we have a square block of data split into tiny bins of space (us Computer Scientists call those tiny bits ‘voxels’) based on the sampling rate. Think of each one of those bins of space as a house for a singular data value. Every one of those homes is going to have a number that is either 0, 1300, or a number in between.

¹You can identify the range of any data set loaded in ImageVis3D by using the *Dataset Explorer*, available from the *Workspace* menu.

Based on this knowledge, we can say a few things about this data set. First, there are few data values near 1300. We know this because at the right side of the histogram, we can see the gray makes a downward slope and tapers off quickly. Thus, values between say 1200 and 1300 occur very rarely. Another point of interest is the very left of the histogram: if you look very closely, you should be able to see a single gray spike at the leftmost column. This is the 0 column, and the fact that there is such a large spike here indicates that there are a *lot* of voxels (houses) which contain the number 0. Finally, because we can see four peaks in the data -- at 0, around 300, another around 650, and another just below 1200 or so -- we can *guess* that there are going to be four major 'components', or features of interest, within this data set.

Features of Interest

Be careful with that last assumption. It relies on intuition which is commonly, but not always, true. MRI or CT scanners generally acquire a range of data values for a given 'thing' in the data set. Even though muscle is basically of a uniform radiodensity, not every voxel which represents muscle is going to have the value 40 (for example). Rather, we'll see some 43s, 38s, 42s, even some 17s. In reality, a better model is to consider data values for a feature to be normally distributed, with the mean of the distribution within the range of the type of tissue or object being scanned.



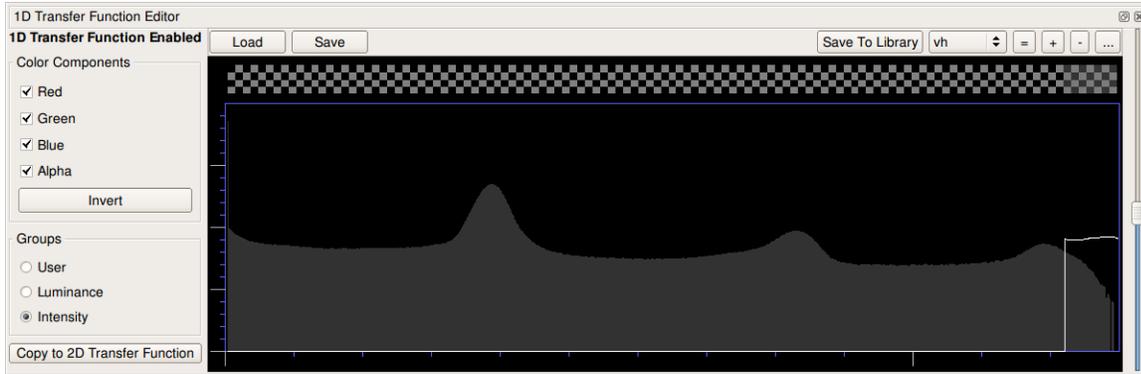
Independent normal distributions. If these were distributions for two features, the histogram would show peaks at the same locations as the features' mean values.

Thus, when we see peaks in the histogram like we do for the tooth, above, we gain some knowledge about the data. It could be the case that each peak represents a particular feature: the smooth slope details the standard deviation of how that feature is defined. It could also be the case that we have two features which are adjacent to the peak and both of their tails -- the width of the falloff, or the standard deviation -- are fairly large, and the combination of those tails translates into a great many data values.

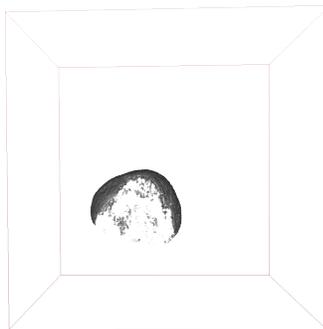


Overlapping normal distributions. If these distributions defined two features, the histogram in ImageVis3D would peak between the two features.

Why does it matter how many voxels have a particular feature? It affects the kind of transfer function we will need to highlight that feature, and even if such a transfer function is possible. Let's demonstrate with an example. Try changing your tooth's transfer function to be non-zero only in the 1200-1300 range:



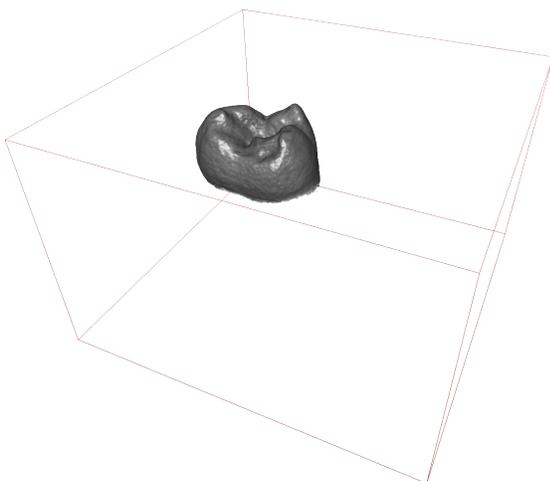
and note the resulting visualization you get:



It is the top of the tooth (the enamel), but it is broken and incomplete -- note that there is an outlining peninsula of structure, but the very top of the tooth is barely implied by, really, just a series of dots which coalesce in only some regions. First, we can see that the histogram tails off in this area; this implies that there are few voxels which have these data values. If we have a constant region of space (defined by the red box) and we 'turn on' a small subset of the voxels in that space, statistically it is unlikely those voxels will be adjacent. Second, we can see a peak off to the left of the defined transfer function. It is generally true that peaks occur because a feature has a

range of data values associated with it, and in our example here we have selected ("turned on") only a subset of that feature. Therefore, we should expect to see only a subset of the feature that this peak represents (the enamel).

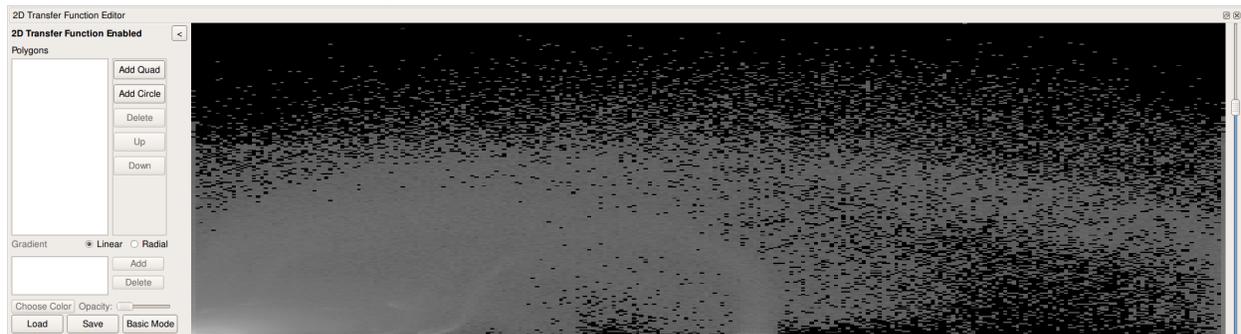
Try expanding that transfer function a bit to include the entire peak. Rotate the data set so you can see it from the side as well. Aim to produce this visualization before moving on:



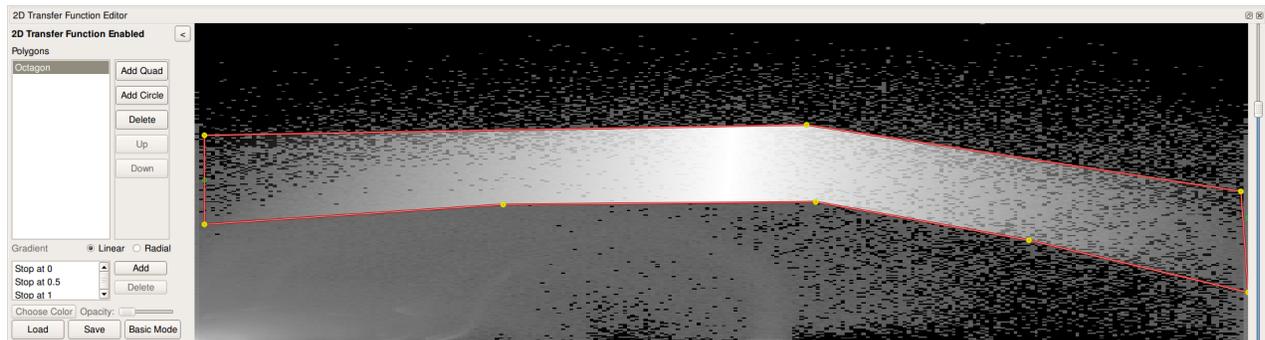
Designing 2D Transfer Functions

For example, if the voxels for bone are in the range [400, 572], and some muscle surrounding that bone are in the range [-30,45], there will be high gradient at the interface between the two. Within the muscle, the gradient will be low; at the interface between the bone and muscle, the gradient will be comparatively high. These areas will therefore be separated in Y within the 2D transfer function, allowing us to isolate the interface from the material.

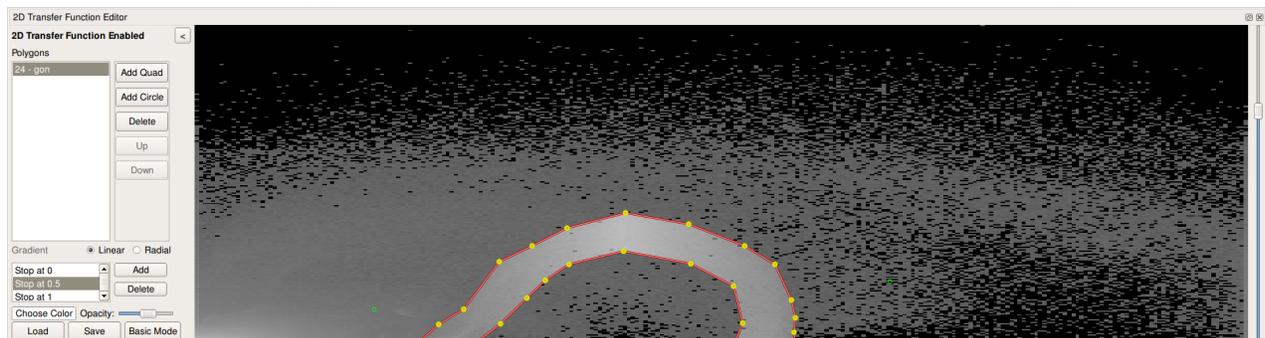
Let's examine a particular case in detail. Close any data sets you may have open, and then open the *backpack8.uvf* data set from your USB stick. Open up the 2D transfer function editor, and you should see something like this:



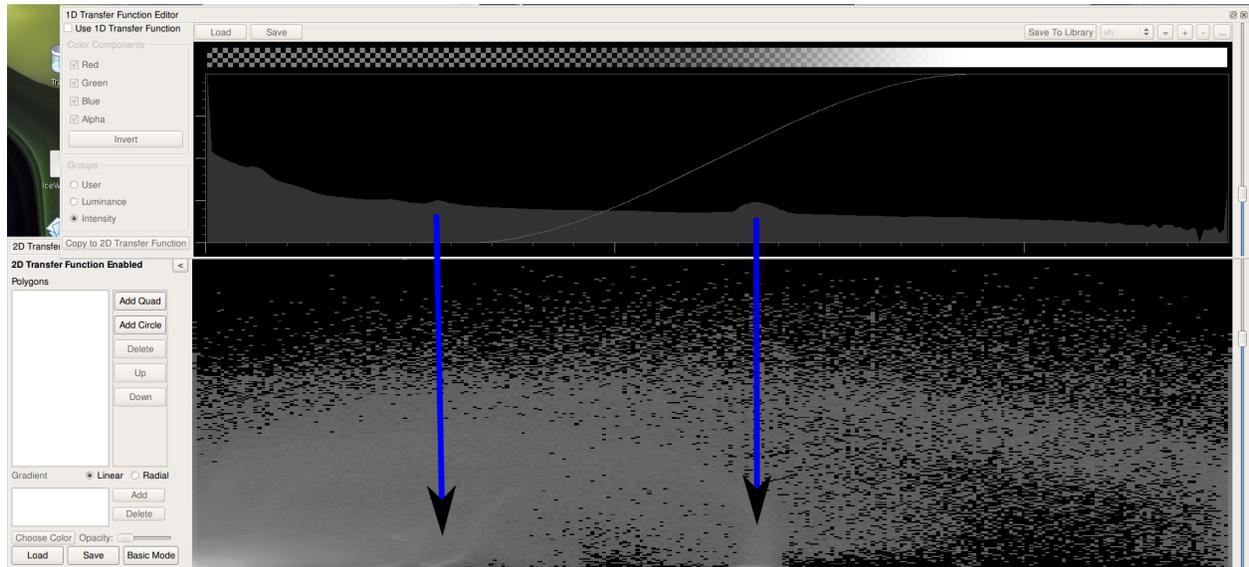
From this histogram we can infer a few things. The bottom left is gray and white, with no black interspersed -- this tells us that most of the data values are concentrated here, so if we place a polygon over this area, it will produce a visualization which depicts nearly every voxel in the data set (try it!). Secondly, we can see two major arches: a wide one which spans the entire data range and is higher in Y:



and a lower band that appears toward the middle and curves back down before it gets that wide:

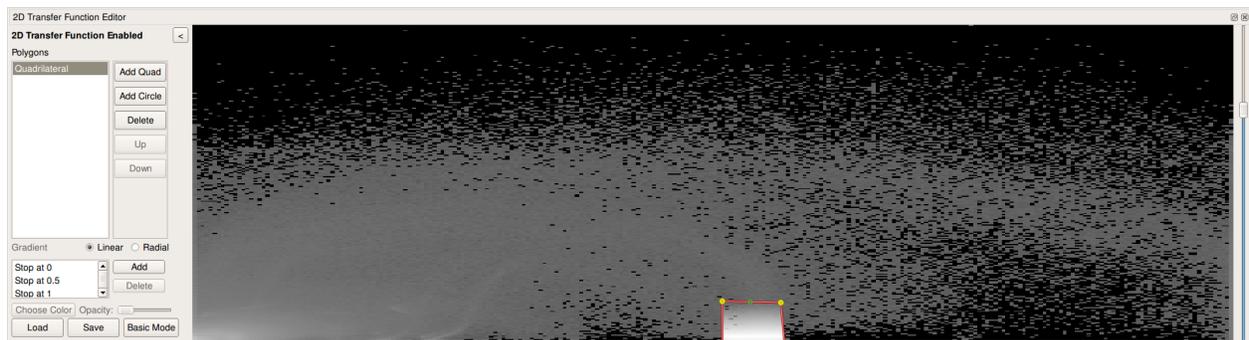


These can be a bit hard to see. It can help if you open up the 1D transfer function editor and align the histogram so that it is even with the 2D transfer function editor's histogram. By doing so we can see that the peaks in 2D space correspond to peaks in 1D space:



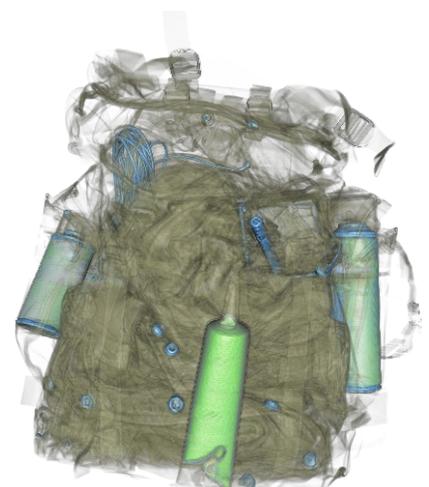
Note that left peak appears simply as higher contrast, whereas the right peak is more apparent simply because it produces a 'band' within which no black area appears.

Structure such as what is visible here does not necessarily imply that there is a single visualizable feature within that structure. In this case, the bottom loop contains a whole series of objects which are apparent beneath the backpack's exterior, but you can somewhat isolate some of the individual objects depending on the area you utilize within that curve. Try this transfer function for example:



Note that it mostly highlights the squeezable tube near the bottom of the bag.

Now that you have a basic idea of how to develop 2D transfer functions, try to create the visualization on the right. We have already found the squeeze tube; after you isolate that, try to get the blue of the canisters and the rope. Finally, move on to isolating the brown parts of the outer bag. When designing a transfer function, generally we want to start from the inside of our data and work outward.



While generating this transfer function you may notice that the approximation ImageVis3D gives you while interacting with the transfer function is less than stellar. Find the *Progress Viewer* from the *Workspace* menu. Open it up and note the sliders which control resolution on the bottom of the tool widget. You can drag the bottom slider all the way to the right to force ImageVis3D to avoid those earlier approximations. Feedback will be slower, but in cases like this one that is probably desirable.

If you get completely stuck, use the “Load” button at the bottom left of the 2D transfer function widget and find the *backpack8-solution.2dt* file in the same directory that the data file existed in.

Creating Compelling Visualizations

At this point, you probably know everything you need to in order to enable you to understand what features exist in your data. However, not all visualization tasks are *exploratory*; sometimes we produce visualizations to *convey* an interesting feature to a colleague. Here we’ll go over the process by which SCI Institute personnel produce some of their visualizations.

Before you really begin: consider the process by which you acquire and process data. In particular, try to take a quick look at your data before you apply any filtering or processing to it. In many cases, while data filtering techniques may be useful for analysis, they do not actually help one to generate a transfer function. So, try things both ways and be sure the input to your visualization step is the input you really want.

After loading your data, try creating a 1D transfer function using the techniques you have learned in this tutorial. If the 1D transfer function fails, try isolating features using the gradient available in the 2D transfer function.

In some cases, neither of those will be enough. If, for example, the bone you want to highlight has the values [30, 45], and there is a bunch of noise external to the region of interest which takes on values between 20 and 40, then it is likely you will not be able to visualize the bone without occluding it with the noise. In this case, we have hit the fundamental problem of volume rendering: sometimes it is not possible to segment data features based solely on data values. If this happens with your data, you will need to consider some type of processing.

The first thing to try is cropping your data. Are there areas external to a region of interest? Try to rewrite your data to crop out everything but that region. This will also benefit you by removing data values which are not relevant, which makes it easier to generate a transfer function.

The next thing to try is to artificially quantize your data. Set all data values above a particular threshold to 4096; all data values below a threshold to 0. ImageVis3D quantizes your data down to 12 bits when it loads your data, and if you have special knowledge of how your data values are distributed, you can do a better job than it can. Furthermore, having fewer data

values means that each bin in the transfer function histogram appears wider, which can make it easier to set a transfer function which finely distinguishes between data values.

If things still aren't working out, you'll probably need to segment your data. Thankfully Seg3D is another tool covered in this workshop. Learn how to perform a manual or semi-automatic segmentation on your data, and then apply that segmentation as a mask on your original data.

Once you have an expressive transfer function, test to see if lighting helps. Toggle it on and off and see if that improves the visualization. If turning it off generally helps, you might need to go back and make subtle modifications to your transfer function, usually to brighten or darken particular areas. You may want to consider trying to fake 'lighting' by making some data values black. In the 1D editor: give some data values non-zero "Alpha" settings, but 0 in red, green, and blue channels. In the 2D editor, you can make a polygon black.

When you have something you like, save a high quality, high resolution image with the *Recorder*, available from the workspace menu. Maximize ImageVis3D and your data set's window to get the highest resolution possible. Use floating tool widgets (instead of docking them to the main ImageVis3D window) to allow a larger window. In the "Rendering Options" dialog, crank up the "Sampling Rate" to 400%. Finally, select a path for the output image and click "Single" in the *Recorder* to save it.

In summary,

1. Pick the right data to visualize
2. Try to generate a good visualization using a 1D transfer function
3. Switch to 2D if required
4. Preprocess your data if needed
5. If you *still* can't isolate it: perform a segmentation.
6. Open up the lighting dialog and try with and without lighting.
7. Maximize both ImageVis3D and the window with the data set you care about. Make widgets like the "Rendering Options" and "Recorder" float, instead of being docked into ImageVis3D, so that you can get a bigger window.
8. In the "Rendering Options", crank the sampling rate up to 400%.

Appendix - Installations

Ubuntu PPA

We make a Launchpad Personal Package Archive (PPA) available for those who are interested in staying on the bleeding edge of development. Please keep in mind that these updates can come through daily and do not receive the same rigorous testing that the release versions do. Launchpad PPAs integrate with the Ubuntu package management system, so you will receive updates just like you receive them for other system components.

Details on how to configure this are available on the PPA's website:

<https://launchpad.net/~tfogal/+archive/ppa>

Source (Mac/Linux)

ImageVis3D Linux binaries are compiled statically on the latest stable release of a 32-bit Debian installation. These seem to work on most Linux systems with few issues. However, if they do not work out of the box for you, you will need to compile ImageVis3D yourself.

There isn't really a good reason to compile ImageVis3D on Mac, but the instructions are very similar to the Linux build process.

The only reason to compile ImageVis3D yourself on Windows is if you intend to modify it in some way. Such use is far beyond the scope of this tutorial, but you may pull Tom or James aside if you are interested in getting involved with ImageVis3D development.

You will need a basic development environment (on many Linux distributions: the "build-essential" package), flex, bison, and subversion. You also need Qt; the best way to acquire this for use with ImageVis3D is to use our provided scripts, however.

First download the ImageVis3D sources:

```
svn checkout https://gforge.sci.utah.edu/svn/imagevis3d
```

There are a variety of scripts under *Scripts/* which can be used to both acquire and compile Qt. Run the one most appropriate for your platform:

```
bash imagevis3d/Scripts/Linux-StaticQt.sh
```

This will require a network connection. After twenty or so minutes, it will have installed Qt into `${HOME}/sw/`. Now you just need to compile ImageVis3D:

```
cd imagevis3d  
${HOME}/sw/bin/qmake QMAKE_CONFIG="release" -recursive  
make
```

That's it! You can run *Build/ImageVis3D* to start ImageVis3D immediately, or you can copy *ImageVis3D* and *Tuvok/Shaders* into a directory of your choosing to 'install' it.