

Using the Volume Rendering Plugin

- Accessed via Define Datamode ---> Plugins ---> Render [new]

The screenshot shows the Volume Rendering Plugin interface with the following callouts:

- Pick new underlay dataset**: Points to the 'Choose Underlay Dataset' button.
- Name of underlay dataset**: Points to the text field 'AFNI_data1/afni/strip+orig [spgr]'.
- Sub-brick to display**: Points to the 'Brick #0 #0' field.
- Open color overlay controls**: Points to the 'Overlay' button.
- Range of values in underlay**: Points to the 'Min=0 Max=435' field.
- Range of values to render**: Points to the 'Bot 40' and 'Top 150' fields.
- Change mapping from values in dataset to brightness in image**: Points to the 'Brightness' graph.
- Mapping from values to opacity**: Points to the 'Opacity' graph.
- Histogram of values in underlay dataset**: Points to the 'Sqrt Histogram' graph.
- Maximum voxel opacity**: Points to the 'Opacity Factor' field (set to 1).
- Menu to control scripting (control rendering from a file)**: Points to the 'Scripts' button.
- Cutout parts of 3D volume.**: Points to the 'Cutouts' field.
- Render new image immediately when a control is changed**: Points to the 'Compute' button.
- Accumulate a history of rendered images (can later save to an animation)**: Points to the 'Accumulate' checkbox.
- Compute many images in a row**: Points to the 'Frames' field (set to 5).
- Show 2D crosshairs**: Points to the 'See Xhairs' checkbox.
- Select incremental mode for angle changes**: Points to the 'I' checkbox.
- Control viewing angles**: Points to the 'Roll 70', 'Pitch 120', and 'Yaw 0' fields.
- Detailed instructions**: Points to the 'Help' button.
- Force a new image to be rendered**: Points to the 'Draw' button.
- Reload values from the dataset**: Points to the 'Reload' button.
- Close all rendering windows**: Points to the 'done' button.

- Volume rendering concepts:

- ✧ Goal is to create a 2D image consisting of pixels
- ✧ Each 2D pixel is obtained from data looking down line of sight into 3D volume:



If we looked directly from the subject's right to left, all the data along the white line would contribute to one image pixel

- ✧ Each 3D voxel contains one numerical value
- ✧ Voxel value determines the brightness (or color) of that voxel --- if it is visible
- ✧ Voxel value determines the opacity of that voxel:
 - ↳ Opacity = 0 \Rightarrow Transparent (brightness does not contribute to image)
 - ↳ Opacity = 1 \Rightarrow Opaque (nothing behind it along the line will be seen)
 - ↳ Intermediate values are translucent:
Opacity = 0.5 \Rightarrow 50% of voxel brightness is added to pixel; voxels farther down the line will contribute to other 50% of pixel result

- Opacity examples:
start with (remaining) opacity of 1, and apply a fraction of it at each new voxel

| | | | | | | | |
|----------------------|-----|------|-------|--------|---------|--|-------|
| Values encountered | 10 | 10 | 100 | 100 | 100 | | limit |
| | | | | | | | |
| Opacity used (@ 1.0) | 1 | | | | | | 0 |
| Value applied | 10 | | | | | | 0 |
| Opacity remaining | 0 | 0 | 0 | 0 | 0 | | 0 |
| Cumulative value | 10 | | | | | | 10 |
| | | | | | | | |
| Opacity used (@ 0.9) | 0.9 | 0.09 | 0.009 | 0.0009 | 0.00009 | | 0 |
| Value applied | 9 | 0.9 | 0.9 | 0.09 | 0.009 | | |
| Opacity remaining | 0.1 | 0.01 | 0.001 | 0.0001 | 0.00001 | | |
| Cumulative value | 9 | 9.9 | 10.8 | 10.89 | 10.899 | | 10.9 |
| | | | | | | | |
| Opacity used (@ 0.5) | 0.5 | 0.25 | 0.125 | 0.0625 | 0.03125 | | |
| Value applied | 5 | 2.5 | 12.5 | 6.25 | 3.125 | | |
| Opacity remaining | 0.5 | 0.25 | 0.125 | 0.0625 | 0.03125 | | |
| Cumulative value | 5 | 7.5 | 20 | 26.5 | 29.625 | | 32.5 |
| | | | | | | | |
| Opacity used (@ 0.2) | 0.2 | 0.16 | 0.128 | 0.1024 | 0.08192 | | |
| Value applied | 2 | 1.6 | 12.8 | 10.24 | 8.192 | | |
| Opacity remaining | 0.8 | 0.64 | 0.512 | 0.4096 | 0.32768 | | |
| Cumulative value | 2 | 3.6 | 16.4 | 26.64 | 34.832 | | 67.6 |

✧ 3D viewing angles:

Roll = angle about I-S axis

Pitch = angle about L-R axis (after roll rotation)

Yaw = angle about A-P axis (after roll and pitch)

✧ Rendering is CPU and memory intensive --- a fast computer is very desirable

- Utility program [3dIntracranial](#) can be used to strip the scalp off of a T1-weighted anatomical volume. In some cases, this may need to be done with the `orig` dataset, which may then be written out in Talairach coordinates.

✧ For example:

`3dIntracranial -anat anat+tlrc -prefix airstrip`

- AFNI can now render datasets that are stored with an arbitrary orientation and voxel size
 - ✧ Datasets are internally re-oriented (see `3dresample`) to axial slice order, so that cut directions make sense. This may take a few seconds, depending on the computer.
 - ✧ Note that axial slice order is the standard for ‘warped’ datasets written out to disk in `+acpc` or `+tlrc` coordinates.
 - ✧ The Overlay dataset may also be resampled, so that its grid spacing matches that of the Underlay dataset.

- In [Talairach view](#), open the rendering plugin, and choose `astrip+tlrc` as the underlay dataset
 - ✧ Plugin will load the voxel values, build the histogram, and then be ready to render
 - ↳ Press [Draw](#) to make your first image
 - ✧ Press [Accumulate](#), then [DynaDraw](#), then [Roll](#)  a few times
 - ↳ Will generate renderings from different angles (i.e., lines of sight)
 - If [DynaDraw](#) is off, then you must press [Draw](#) to get a new rendering
 - ↳ [Accumulate](#) on \Rightarrow rendered images are saved, and can be reviewed by using the image viewer slider
 - This slider does not move you through slices, as it does in the 2D image viewing windows
 - It just moves you backward and forward in the history of saved rendered images
 - If you turn [Accumulate](#) off, then creating the next rendered image will erase the history
 - By default, the plugin's controls ('widgets') do not change as you move around in the rendering history
 - Selecting [Script](#) \rightarrow [Load widgets](#) will make the widgets display the settings they had when the currently displayed image was rendered

- Controlling the mappings from voxel value to brightness and opacity:

Values below “Bot” are equivalent to “Bot” Values above “Top” are equivalent to “Top”

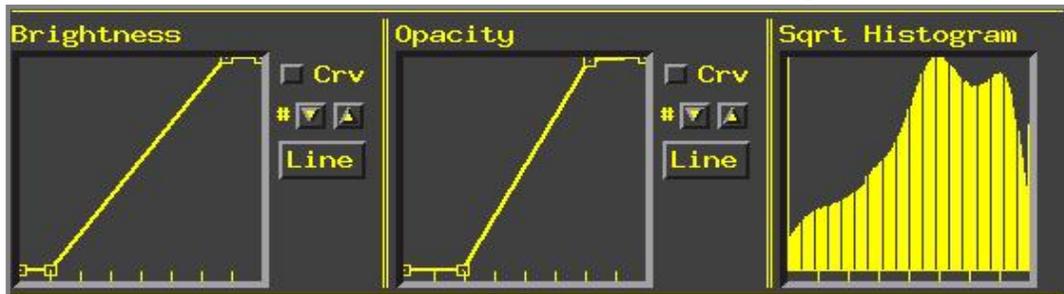
‘Handles’ that let you change the shape of the mapping functions

Gray and White matter peaks for this SPGR dataset

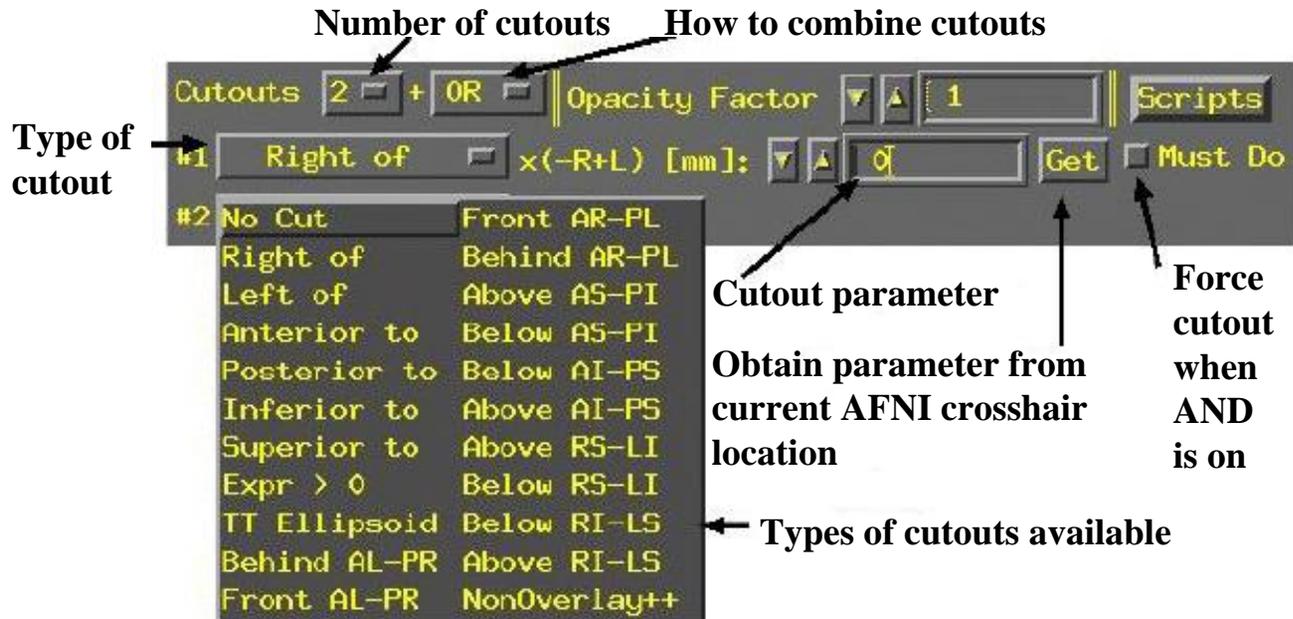
Horizontal axes correspond to voxel values between “Bot” and “Top”

Crv ⇒ curve between handles
⇒ changes number of handles
Line ⇒ restore straight line map

- Probably want to make white matter be fully white. Drag #3 Brightness handle up to top, over to white matter value
- Probably want to reduce Opacity to 0 for all low intensity voxels. Drag #2 Opacity handle to bottom, over to histogram trough value
 - Then press Draw to force a re-rendering



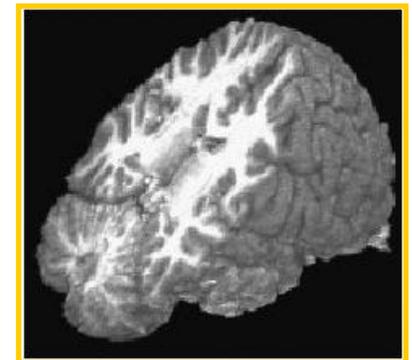
- Cutouts are for removing parts of the volume so you can see the parts you want:



- ✧ Each cutout specifies a sub-volume in space that will be removed from the dataset before rendering (done by setting voxel opacity to zero inside the cutout)
- ✧ Multiple cutouts can be combined in two different ways:
 - ↳ OR ⇒ all voxels in all cutouts will be removed
 - ↳ AND ⇒ only voxels that are in every cutout sub-volume will be removed
 - Must Do can be used to force the removal of cutout voxels even if AND is active
 - OR is equivalent to Must Do for all cutouts

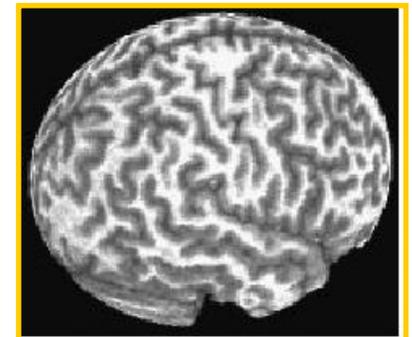
- ❖ Most cutout types are controlled by a single numerical parameter determining the position of the cutout
 - ↳ Right of 'x' means to cut out all voxels to the right of the given x-coordinates (-x is Right, +x is Left)
 - Similarly, can cutout everything Anterior to, Posterior to, or Superior to, Inferior to, or Left of a given coordinate position
 - ↳ Behind..., Below..., Front..., Above... cut out 45° diagonally slanted half-spaces, with respect to the listed planes:

For example, Above AS-PI is above a plane that slants from the Anterior-Superior front of the brain downwards to the Posterior-Inferior back of the brain -- that is, halfway between a coronal and axial slice



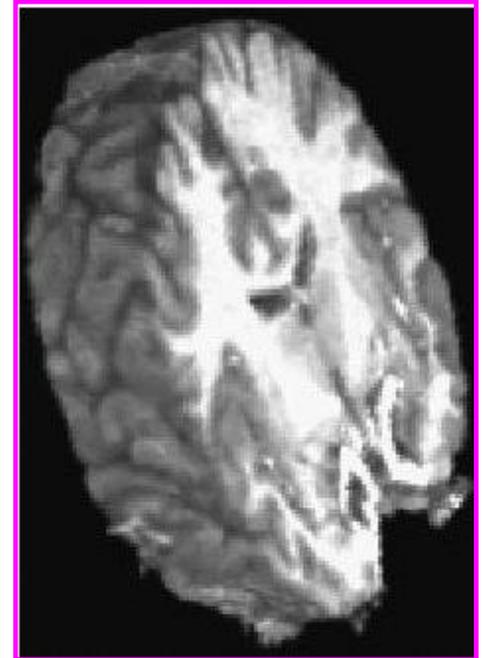
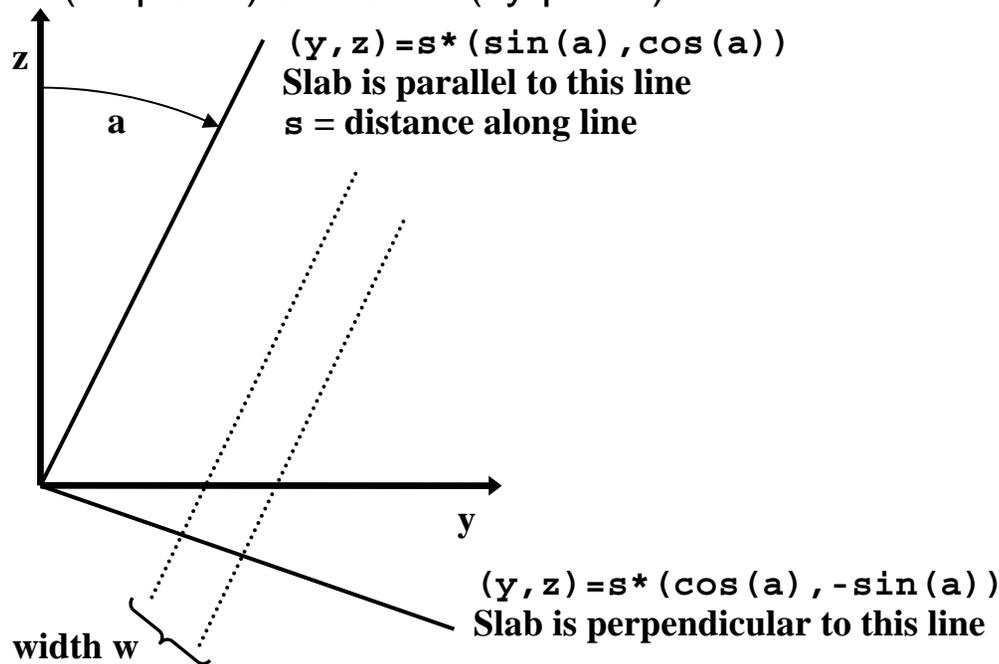
- ↳ TT Ellipsoid cuts out the region outside an ellipsoid with the same proportions as the Talairach-Tournoux Atlas brain

This is fun, but not much use



- ✧ Cutout type Expr > 0 defines the region to be removed by a general mathematical expression, rather than a single parameter
 - ↳ The expression uses the same syntax as 3dcalc
 - ↳ Variables that can be used are 'x', 'y', and 'z', corresponding to spatial coordinates in the dataset
 - When using Automate (infra), variable 't' can also be used
 - The (x, y, z) locations where the expression evaluates to a positive number will be cut out

↳ Example: rendering a slab tilted at an arbitrary angle between coronal (xz-plane) and axial (xy-plane):



- ⌞ The set of points within the slab is described by the inequality

$$|y \cdot \cos(a) - z \cdot \sin(a) - s| < 1/2w$$

for angle= a , slab center offset= s , and slab width= w . To render a slanted coronal slab 30 mm thick, tilted posteriorly from the vertical of 25°, we would use this for the cutout expression:

$$\underline{\text{abs}(y * \text{cosd}(25) - z * \text{sind}(25) - 20) - 15}$$

where the `sind()` and `cosd()` functions take arguments in degrees, and where the offset has been set to 20 mm (you will have to alter this offset to get the exact position you want)

- By using [Automate](#) and setting the angle (25 above) and/or the offset (20 above) to depend on 't', we can make a sequence of images where the slab rotates downwards and/or moves backwards

- [Automate](#) lets you create a large number of renderings at once
 - ✧ Note that most (but not all) number entry boxes have slightly raised borders:



- ✧ Such boxes can use an expression with the variable 't' when [Automate](#) is used:
 - ✧ Turn [Automate](#) on
 - ✧ Enter some small number in the [Frames](#) control (say 5)
 - ✧ Enter $70+5*t$ in the [Roll](#) control, then press [Compute](#)
 - ✧ The dataset will be rendered with the variable 't' set to 0, 1, 2, 3, 4 in turn
 - That is, t will run from 0 to one less than the number of Frames, and all the raised-border boxes that use expressions with 't' will be evaluated prior to each frame being rendered
 - ✧ In this example, this will result in a sequence of views of the dataset from different roll angles 70°, 75°, 80°, 85°, 90°
 - ✧ Can also use 't' in cutout parameters to make cutouts depend on 'time'
 - 2 cutouts, $\text{Left of}=10+3*t$ and $\text{Right of}=-10+3*t$ will produce a 20 mm thick slice that slides leftwards as t increases
 - ✧ Can use 't' in more than one raised-border box simultaneously to make complex animations (e.g., [Roll](#) and [Cutouts](#) together)
 - ✧ Put cursor in raised-border box and press [Enter](#) to have box reset to last numerical value used by [Automate](#)

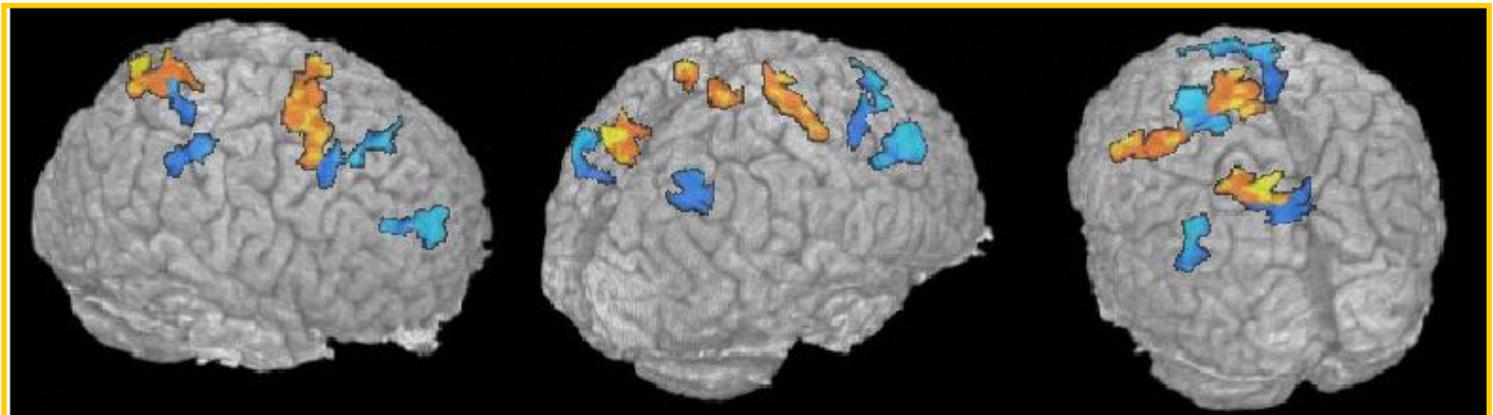
- Color overlays (e.g., of functional activation maps)
 - ✧ Press the [\[Overlay\]](#) button to open up the panel that controls how functional overlays are generated:

The screenshot shows the 'Choose Overlay Dataset' panel in AFNI. The title bar reads 'AFNI_data1/afni/func_slim+orig [fbuc:7] [* :'. The panel is divided into several sections:

- Choose Overlay Dataset:** A dropdown menu at the top.
- Thresh Color:** A vertical color bar on the left. The top is labeled '1.00' and the bottom '-1.00'. A slider is positioned at '.5000'. A label points to the bottom of the bar: 'Voxels below threshold will not be colored'. A label points to the top of the bar: 'Determines colors for above-threshold voxels'.
- Color and Thresh Controls:** Two rows of controls. The first row has 'Color #1 Actions[0] Coe' and a checkbox. The second row has 'Thr #2 Actions F-stat' and a checkbox. A label points to these two rows: 'Sub-bricks for color and thresholding'.
- Color Opacity:** A slider set to '0.5'. A label points to it: 'Opacity of colored voxels'.
- ShowThru Mode:** A dropdown menu set to '75%'. A label points to it: 'Select ShowThru mode and corresponding percentage'.
- Other Options:** Checkboxes for 'See Overlay TT Atlas', 'Cutout Overlay', and 'Remove Small Clusters'. A label points to the 'See Overlay TT Atlas' checkbox: 'Show regions from Talairach Atlas'. A label points to the 'Remove Small Clusters' checkbox: 'Turn overlay coloring on'.
- Advanced Settings:** Two rows of controls with up/down arrows and input fields. The first row has 'min' and '1'. The second row has 'max' and '200'. A label points to these rows: 'Apply cutouts to overlay'.
- Color Range and Thresholds:** A section with 'Color -2.805953: 1.730824' and 'Thr 0: 1000'. A label points to this section: 'Clean up small blobs of above-threshold voxels (as in 3dclust)'. Below this are 'autoRange: 2.805953', a checkbox, and a slider set to '10000' with 'Rota' controls.

- ✧ Controls are similar to [Define Function](#) for overlaying color on 2D image viewer windows
 - ↳ Will only discuss differences from 2D overlay control panel

- ✧ Color Opacity lets you select the opacity of colored voxels (those that are above the threshold)
 - ↳ Opacity of overlaid voxels is different from the opacity it would have from the underlay dataset at that location
 - ↳ Usually want this to be high (0.5 or above)
 - ↳ Two special values on this menu:
 - Underlay means that the colored voxel's opacity will be by the opacity that it would have from the underlay image
 - ShowThru means that colored voxels show through underlay voxels (the 'glass brain' effect), no matter how opaque the underlay is
 - Takes some practice to become accustomed to this type of image
 - But can be a very useful way to see lots of activation at once:



- Seeing this animated is especially useful (but hard to publish)