

Drishti : Volume Exploration and Presentation Tool

Drishti stands for vision or insight in Sanskrit, an ancient Indian language.

Understanding the data set is important. Equally important is conveying that understanding to the research community or a lay person. Drishti is aiming for both. The central idea about Drishti is that the scientists should be able to use it not only for exploring volumetric datasets but also in presentations.

Towards Drishti v1.0

This new release of Drishti has many new changes. The software is being rewritten from scratch. The previous versions of Drishti v0.1.* employed wxWidgets (<http://www.wxwidgets.org>) for the user interface. The new releases are built using the Qt (<http://www.trolltech.com>) user interface toolkit. Most of the functionality from previous versions has been reimplemented in this new version. Users are advised not to remove the previous version of Drishti.

This new early release has following major improvements :

Prithvi

Prithvi provides direct volume rendering facility for volumetric data defined over spherical grids. The input volume for Prithvi has same format as Renderer i.e. .pvl.nc files. Prithvi supports all the features available in Renderer including simultaneous visualization of multiple volumes.

Source code for prithvi comes with drishti source distribution.

Drishti Import

The new import tool for conversion of volumetric data from user's own format to .pvl.nc is now ready for use. The tool comes as a separate program and can be enhanced to import volume data formats currently not available for conversion. Source code for import tool comes with drishti source distribution.

Drishti Paint

The new 2D paint tool for painting 2D slices of volumetric data in .pvl.nc is now ready for use. The tool comes as a separate program. The tool can handle current (.xml) project files or .pvl.nc files. Source code for paint tool comes with drishti source distribution.

Drishti Composite

This is a image touch up utility for making composites of images. The utility can be quite helpful in modifying an already generated image with parts from other similar images of the volume. Source code for import tool comes with drishti source distribution.

Large Images

Previously image sizes were restricted to the screen size. This restriction has now been removed. Users can now generate large sized (for e.g. 4K x 4K) images/movies.

Batch Mode Rendering

Image sequences and movies are now rendered in background buffer. This improvement means that users can now minimize Drishti window without corrupting the images. Drishti window does not need to be in foreground for these activities.

RGB/RGBA volumes

Drishti can now handle coloured volumes. Color RGB/RGBA volumes are generated by importing color images via Drishti Import tool. Users can use transfer functions to bring out different features in the dataset.

Animation Interface

Drishti now boasts one of the simplest yet extremely powerful animation interface. All that a user needs to do is open the keyframe editor and start saving the keyframes. New keyframes can be added in between the existing ones. Keyframes can be modified, removed or moved around very easily. Almost everything in Drishti can be animated. It is hard to imagine any more simpler yet powerful animation interface. Users can copy and paste keyframes as well as import keyframes from another project.

Volume Masks

Tagging voxels in Drishti has now become easier with 2D paint interface and 3D paintball interface. With 2d interface users can paint on volume surface as defined by transfer functions. Upto 255 different tag

values can be used. Tags are kept separate from the volume data are not linearly interpolated, so there is no smearing of tag values at the tag boundaries.

Drag-and-Drop Interface

Users can now drop their project files into the mainwindow of Drishti to load them. The .pvl.nc files too can be dropped in order to load them. This interface will be enhanced to handle various other file formats and image stacks.

Transfer Function Editor

The transfer function editor is more flexible. The editor window can now be made larger in size which make editing more easier. The transfer functions too are much more flexible. Transfer function sets, tranfer funcion editor and colour-opacity gradient editor have all now been consolidated into a single window.

Gallery

Users can save their views in gallery. Restoring a view will restore the exact settings that the view had.

Rendering

Users can now move the light source all around the volume. The shadows are now calculated properly. Soft and diffused colored shadows are now possible. Users can also tune the shadow colors. Diffused and specular highlights now take into consideration the homogenous regions where gradients are nearly zero.

Back Plane

Back plane can now be added to the scene. Whenever light is in front of the volume, backplane can be used to enhance the image by casting shadows onto the plane.

Handle 4 volumes series

Users can now explore and animate upto 4 volumetric timeseries data sets simultaneously.

Docked Windows

All the windows in Drishti are now dockable. These are therefore more manageable.

Probing Raw Data

Individual points can be moved around and can be used for extracting raw values. Paths can be used for getting raw value profile curves.

Clipping planes reloaded

Users can now draw captions or paste images on to clip planes. Clip planes can also be used as textured planes. Pasting images may become useful for data registration, as well as for embedding more information such as photographs of real objects into images/animations produced using Drishti.

Captions/Labels

Captions/labels can be placed as overlays on images. Users can also now place text on to paths and clip planes. The text is pasted as a texture when placed on paths and clip planes, enabling text to share 3d space with volumetric data.

Embedding Images

Images can be embedded within the scene, enabling images to share 3d space with volumetric data. Images can be textured onto clipping planes as well as paths. When embedded on path, image follows the path's spline.

Features still to be implemented in Drishti v1.0

There are a few features that need to be implemented in Drishti v1.0. Users are advised not to remove the older (wxWidgets) version of Drishti, if they are using the following features.

The features that are missing in Drishti v1.0 include :

- Vector volume handling
 - Enclosing surfaces
 - Legend facility
-

RAW Volume File Format

The datatype for volume data in the **RAW** file can be either :

- 1 byte : **byte, unsigned byte**
- 2 bytes : **short, unsigned short**
- 4 bytes : **int, float**

Format for **RAW** file is as follows :

[single byte][NX][NY][NZ][volume data]

single byte specifies the voxel type.

0 : unsigned byte - 1 byte per voxel

1 : signed byte - 1 byte per voxel

2 : unsigned short - 2 bytes per voxel

3 : signed short - 2 bytes per voxel

4 : integer - 4 bytes per voxel

8 : float - 4 bytes per voxel

NX, **NY** and **NZ** are the grid dimensions written as 4-byte unsigned integers and **volume data** is the complete volumetric data with **Z** as the fastest varying loop index. The file size for such a raw formatted file is $13 + NX * NY * NZ * \text{sizeof}(\text{datatype})$

For example for volume of size 128x128x128 with datatype as unsigned short (2-bytes per voxel) the file size would be $13 + 128 * 128 * 128 * 2 = 4194317$ bytes.

Another format for **RAW** file is :

[header] [volume data]

header the user defined header can be as big as the user wants. The header will be skipped while loading the volume data. The user will have to specify number of header bytes to skip.

volume data is the complete volumetric data with **Z** as the fastest varying loop index. The user will have to specify grid size (NX, NY, NZ) and voxel type in the dialog for reading raw files. The file size for such a raw formatted file is $NX * NY * NZ * \text{sizeof}(\text{voxel type})$

For example for volume of size 128x128x128 with datatype as unsigned short (2-bytes per voxel) the file size would be $128 * 128 * 128 * 2 = 4194304$ bytes.

Volume Masks

Masks allow users to tag individual voxels (akin to segmentation). This tagging allows users to render portions of the volume in a different way or erase the portion. Tagging does not in any way modify the original volume, it is only rendering that is affected. Users can also calculate volume and surface areas of tagged portions.

This tag information is stored in a separate file having the same name as the **.pvl.nc** file, but with **.mask** extension. For e.g. test.pvl.nc file will have mask information in test.mask. The mask file format is as follows :

single byte (value is 0)

followed by 12 bytes containing grid size as three 4-byte integers - [nX][nY][nZ]

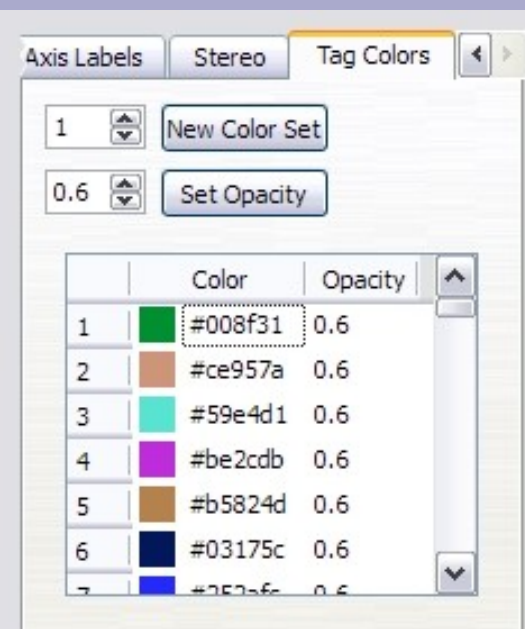
followed by the tag data which is 1-byte per voxel.

Volume masks are activated using **Enable Mask** switch under the **Toggle** menu. Volume masks can be used only for single volumes.

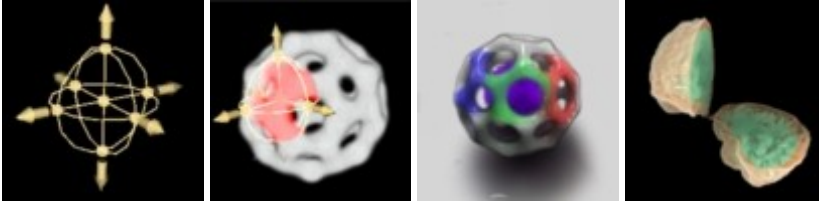
By default the tag value for all the voxels is 0. Tag value of 255 implies erase - i.e. the voxels having tag value of 255 will not be rendered, same as erasing them (the original volume is not touched, it is only the rendering that is affected). Voxels having tag values between 1 and 250 are tinged with a different color, when these are shown using transfer functions. Voxels that have tag values between 251 and 254 are rendered with the given tinge color even when no transfer function is applied - in essence these voxel are always shown (when the tinge opacity is greater than zero). The tinge color and opacity is taken from **Tag Colors** found under **Preferences** widget. Tag opacities are used to mix tinge color with the underlying transfer function color. Tinge colors and opacities can be changed and animated.

DrishTi provides two major ways for tagging the volumetric data - [2D paint interface](#) and [3D paintball interface](#). Users can also make use of paths (use *fill/fillempy*) for tagging regions.

Clipplanes can be used to restrict the masking process.

	Tag Value	Description
	0	Apply transfer function without change.
	1-250	Apply transfer function by mixing appropriate tag color with tag opacity. If the tag color is pure black i.e all components of tag color are 0, then tag opacity is used to modulate voxel opacity. This feature can be used for reducing opacity of the region occluding important feature in the data set. $\text{if (tag .gt. 0 .and. tag .lt. 251)}$ $\text{voxelColor} = \text{voxelColor} * (1 - \text{tagOpacity}) + \text{tagColor} * \text{tagOpacity}$ $\text{if (tagColor .eq. black)}$ $\text{voxelOpacity} = \text{voxelOpacity} * \text{tagOpacity}$
	251-254	Show voxel with appropriate tag color and opacity.

		<pre>if (tag .gt. 250 .and. tag .lt. 255) voxelColor = tagColor voxelOpacity = tagOpacity</pre>
	255	Do not show the voxel.



Mouse

Use the mouse to move the camera around the object. You can respectively revolve around, zoom and translate with the three mouse buttons. Left and middle buttons pressed together rotate around the camera view direction axis

Double clicks automates single click actions: A left button double click aligns the closer axis with the camera (if close enough). A middle button double click fits the zoom of the camera and the right button re-centers the scene.

A left button double click while holding right button pressed defines the camera Revolve Around Point.

Button(s)	Description
Wheel	Zooms camera
Left	Rotates camera
Left double click	Aligns camera
Right	Translates camera
Right double click	Centers scene
Middle	Zooms camera
Middle double click	Shows entire scene
Left & Middle	Rotates on screen camera
Shift+Left	Adds a point
Shift+Middle	Zooms on region for camera
Right double click with Left pressed	Resets revolve around point
Left double click with Right pressed	Sets revolve around point
Left double click with Middle pressed	Zooms on pixel
Right double click with Middle pressed	Zooms to fit scene

Keyboard

Uses single key stroke commands to add clipplanes, captions, change windows, change rendering mode and so on.

Key(s)	Description
ESC	Quit
DEL	Delete currently active point, path, clipplane.
F1	Help
F2	Toggle between lowres and hires windows
Home	Increase drag image quality
End	Decrease drag image quality
PgUp	Increase still image quality
PgDown	Decrease still image quality
1	Toggle shading mode in hires window
a	Toggle axis
b	Toggle bounding box
f	Toggle the display of FPS
s	Toggle stereo display (if stereo is enabled via command line)
t	Add captions Once the captions are added they can be edited, moved around and animated. Hover over a caption to activate it and press spacebar to edit it
?	Toggle the display of text
!	Toggle smoothing of subvolume when loading in hires window. By default, if the selected subvolume does not fit in texture memory, it is smoothed before subsampling. Users can toggle this smoothing by pressing ! and reloading the subvolume.
Ctrl+s	Save project

Alt+s	Save a screenshot
Alt+f	Change camera mode (revolve or fly)
Alt+Return	Toggle full screen display

Keyframe Editor

Keyframe Editor allows users to save keyframes for creating animations. Drishti interpolates the following properties between keyframes :

- transfer function information
- lighting information
- brick information
- clip information
- path information
- subvolume bounds
- mask tag colors
- background color
- captions
- tick information
- camera parameters

Users can save keyframes using **Set Keyframe**. Once saved, keyframes can be moved to change the frame number. Users can overwrite the keyframes.

Select a keyframe by clicking on the keyframe image. Once selected, a keyframe can be deleted by **Remove Keyframe**.

Button(s)	Description
Space bar	If a keyframe is selected, shows panel for selection of interpolation style for various properties. The selected style is valid only between selected keyframe and the next one.
Ctrl+C	If a keyframe is selected, copy that keyframe into an internal buffer.
Ctrl+V	Paste the copied keyframe at the specified position on the range. If there is already a keyframe at the specified position, user is shown a properties page to select properties to copy.
Left Click	If clicked on a keyframe image, that keyframe is selected. If clicked on the ruler and lies within the animation range, interpolate the frame.
Shift+Left Mouse	Press and drag to select a portion on the keyframe range. Once a portion on the keyframe range is selected, select and drag keyframe within the selected range to shrink/expand the animation.
Middle Mouse	Drag to move the selected sequence. If nothing is selected the entire animation sequence is moved.
Right Click	Remove the selection.



Clip Planes

Primary role of clip planes is for clipping volumes against a plane. Clipping by individual clip planes can be turned off via the **bricks editor**. Users can also use these as textured planes/slices or use for displaying captions in 3D or paste images and animated-gifs onto clip planes. Pasting image onto clip plane may help users when registering data from different modalities, especially helpful when used along with textured slice facility. Pasted images may also be used for conveying more information about the data. For example, users will be able to include photographs of real object into animations. Clip planes can be animated via **keyframe editor**.

Following keys can be used when not hovering over objects in the scene.

Key	Description
c	Add a clip plane. Clip plane is always added in the center of the volume.
v	Toggle visibility of clip planes (Affects all clip planes).

Hover over a clip plane and press single key commands or press **spacebar** to bring up the command input for natural language commands for that clip plane.

Key(s)	Description
DEL	Delete currently active clipplane.
Following keys affect movement of active clip plane.	
w	Remove restrictions on translations for clip plane
x	Restrict translations for clip plane along X-axis
y	Restrict translations for clip plane along Y-axis
z	Restrict translations for clip plane along Z-axis
Shift + w	Remove restrictions on rotations for clip plane
Shift + x	Restrict rotations for clip plane about X-axis
Shift + y	Restrict rotations for clip plane about Y-axis
Shift + z	Restrict rotations for clip plane about Z-axis

Following commands will affect the way in which clip plane is displayed.

Single Key Command	Command	Description
b		Toggle border for the clip plane. Border is always on whenever no image is pasted on the clip plane.
d		Toggle translation and rotation markers for the clip plane. Might be useful when using image on the clip plane for registration purposes.
f		Toggle image flipping on the clip plane. Might be useful when using image on the clip plane for registration purposes.
h		Whenever mouse is hovered over a clip plane that clip plane is shown with a larger scale. Toggle this behaviour.
i	image	Load image to be pasted on the clip plane. A file dialog will pop-up for image file name. Animated-gifs can be loaded and animated via keyframe editor by changing imageframe number. Users can change images on the clip plane at keyframes. Users can show either images or captions on a clipplane, but not both at the same time. Toggling visibility has not effect on images. Images will always be displayed.
	image no	Remove image (if any) pasted on the clip plane.
o		Toggle application of opacity parameter. Works only when pasting captions or images on the clipplane.
s		Toggle texturing of data volume on the clip plane. Slices can be used along with images.
t	caption	Use for displaying text captions on the clip plane. The captions dialog will pop-up where users can enter caption text and select color for the caption. The caption text or colour can be changed for different keyframes. opacity parameter of the clip plane will affect the opacity of the captions. Users can show either images or captions on a clipplane, but not both at the same time. Toggling visibility has not effect on captions. Captions will always be displayed.
	caption no	Remove caption (if any) pasted on the clip plane.
	imageframe	imageframe n Specify imageframe number n for animated-gifs. First image frame number is 0. Specify different frame numbers at different keyframes using keyframe editor for animations.
	tfset	tfset s

		Specify transfer function set to be used for texturing the clip plane. Default value is 1 (use transfer functions in set 1 for texturing the slice).
	scale	scale s scale sx sy Apply scaling parameter s or individually sx and sy to the clip plane. Might be useful when using image on the clip plane for registration purposes.
	opacity	opacity op Apply opacity op to the clip plane. The opacity parameter must be between 0 (transparent) and 1 (opaque).

Commands

Users can key in command strings into Drishti. These command strings can be typed in via the command text input box. The command text input box appears whenever the **spacebar** is pressed with focus in the mainwindow (where the images are displayed).

Command	Description
filter	filter no filter conservative filter mean filter median Whenever the subvolume doesn't entirely fit in the texture memory, it is subsampled. This command specifies the type of filter to be used while loading subvolume into texture memory. If no subsampling is required then no filter is employed. This command does not affect the already loaded texture, but comes into play for subsequent texture loads.
addpoint	addpoint x y z Add a point at the given x,y,z coordinates. Example : addpoint 10 20.5 123
addpath	addpath Add a path going through the selected set of points. If none of the points is selected, the path is created through all the available points. The order of selection of points affects the path. Once created a path can be edited.
loadimage	loadimage Load background image from a file. User will be asked for the image file. This image will be drawn for the background instead of background color.
resetimage	resetimage Reset background image. No background image will be drawn, instead the background color will be used to fill the background.
loadpoints	loadpoints Load points from a file. User will be asked for the text file name from which the points will be loaded. This file should specify number of points on a single line followed by all point coordinates with one point (i.e. 3 values) per line. For e.g. 4 0.5 1.0 4.0 3.5 1.0 1.0 2.5 1.0 4.0 0.5 1.5 6.0 Once loaded these points can be edited.
savepoints	savepoints Save points into a file. User will be asked for the text file name into which the

	points will be saved. This file will have number of points at the top followed by one point (i.e. 3 values) per line.
savepath	savepath Save all paths into a file. User will be asked for the text file name into which the path points will be saved. For each path this file will have number of coordinates at the top followed by point coordinates.
loadpath	loadpath Load a path from a file. User will be asked for the text file name from which the points will be loaded. This file must be a text file with number of points at the top followed by one point (i.e. 3 values) per line. The file may contain multiple paths, as shown in the example below. Once created a path can be edited. For e.g. 4 0.5 1.0 4.0 3.5 1.0 1.0 2.5 1.0 4.0 0.5 1.5 6.0 3 1.5 19.0 21.0 2.5 10.0 42.0 0.5 12.5 16.0
setfov	setfov fov Set (vertical) field of view, fov, in degrees. Default value is 45 degrees.
resetfov	resetfov Reset the (vertical) field of view to default value. Default value is 45 degrees.
saveopacityvolume	Save the opacity of voxels as volume data. Voxels in the selected subvolume are tested for clipping. For voxels that pass this test, opacity is calculated using the current transfer function. This opacity is then saved to a raw file.
maskrawvolume	Save original raw volume into another raw file using opacity of voxels to mask regions in the volume data. Voxels in the selected subvolume are tested for clipping. For voxels that pass this test, opacity is calculated using the current transfer function. This opacity is then used for masking voxels in the original raw volume. If the opacity is zero - i.e. voxel is transparent - voxel value in the new raw volume will be set to mask value (which is supplied by the user at the start of this process). Voxels that have non-zero opacity - i.e. visible voxels - are stored at their original voxel value in the new raw file.
getvolume	getvolume getvolume tag Calculate volume by counting voxels that have non-zero opacity, i.e. count voxels that are shown. When getvolume is supplied with tag, only those voxels that have the given tag value are counted.
getsurfacearea	getsurfacearea

	<p>getsurfacearea tag</p> <p>Calculate surface area by counting surface voxels (similar to getvolume). When getsurfacearea is supplied with tag, only those surface voxels that have the given tag value are counted.</p>
getangle	<p>getangle</p> <p>Given three points calculate angle in degrees.</p>
translate	<p>translate x y z</p> <p>Translate camera to the x,y,z position. Example : translate 200 250 200</p>
translatex	<p>translatex x</p> <p>Translate camera to the x,0,0 position. Example : translatex 200</p>
translatey	<p>translatey y</p> <p>Translate camera to the 0,y,0 position.</p>
translatez	<p>translatez z</p> <p>Translate camera to the 0,0,z position.</p>
move	<p>move x y z</p> <p>Move camera from the current position by x,y,z units. Example : move 50 20 20</p>
movex	<p>movex x</p> <p>Move camera from the current position by x,0,0 units. Example : movex 50</p>
movey	<p>movey y</p> <p>Move camera from the current position by 0,y,0 units.</p>
movez	<p>movez z</p> <p>Move camera from the current position by 0,0,z units.</p>
movescreenx	<p>movescreenx x</p> <p>Move camera horizontally as seen by the viewer from the current position by x units. Example : movescreenx 10</p>
movescreeny	<p>movescreeny y</p> <p>Move camera vertically as seen by the viewer from the current position by y units.</p>
movescreenz	<p>movescreenz z</p> <p>Move camera in/out as seen by the viewer from the current position by z units.</p>

rotate	rotate x y z a Rotate camera by a degrees about the axis defined by vector x,y,z . The vector x,y,z is internally normalized . Example : rotate 0.1 1.0 0.5 40
rotatex	rotatex a Rotate camera by a degrees about X-axis. Example : rotatex 30
rotatey	rotatey a Rotate camera by a degrees about Y-axis.
rotatez	rotatez a Rotate camera by a degrees about Z-axis.
addrotation	addrotation x y z a Rotate camera by a degrees about the axis defined by vector x,y,z from its current orientation. The vector x,y,z is internally normalized . Example : addrotation 0.1 1.0 0.5 40
addrotationx	addrotationx a Rotate camera by a degrees about X-axis from its current orientation. Example : addrotationx 30
addrotationy	addrotationy a Rotate camera by a degrees about Y-axis from its current orientation.
addrotationz	addrotationz a Rotate camera by a degrees about Z-axis from its current orientation.
rotatescreenx	rotatescreenx a Rotate camera by a degrees about horizontal screen axis from its current orientation. Example : rotatescreenx 30
rotatescreeny	rotatescreeny a Rotate camera by a degrees about vertical screen axis from its current orientation.
rotatescreenz	rotatescreenz a Rotate camera by a degrees about axis perpendicular to screen from its current orientation.

Points

Points can be added by pressing **Shift** and **Left** mouse button. Then can also be added using **addpoint** command.

Points are normally shown as yellow dots. Whenever mouse hovers over a point, the point is shown as a green dot, signifying that it is active. Once activated a point can be moved around. This movement can be constrained to one of the X/Y/Z axes or can be made free to move in all directions. Points can be selected by clicking them. Selected points are shown as red dots. Whenever any point is active, press **spacebar** to bring up the command input.

Following are the commands for points.

Command	Description
addpoint	addpoint x y z Add a point at the given x,y,z coordinates. Example : addpoint 10 20.5 123
setpoint	setpoint x y z Set point coordinates to the given x,y,z coordinates. Example : setpoint 10 20 123
deselectall	deselect Deselect all points.
removepoints	removepoints [all selected] Removes all points or the selected set of points. Example : removepoints all removepoints selected
normalize	Convert coordinates for all points into integers.

Single Key Command	Description
n	Toggle display of point numbers.
r	Toggle display raw values extracted from the RAW file.
t	Toggle display tag values extracted from the mask file.
c	Toggle display point coordinates.
DEL	Delete point.

x/y/z/w	x y z w Constrain movement of point in x/y/z direction or make it free in all directions with w .

Paths

Paths are created using **addpath** command from the points. Paths can be created using all the points or the selected set of points. Once the path is created the points used to create the path are removed from the scene. The path can be modified after it has been created - points can be added, moved and removed from the path. **Left** click on the curve will add a point and **right** click on a point will remove the point. Path consists of at least 2 points.

Hover over the path and press single key commands or press **spacebar** to bring up the command input for natural language commands for the path.

The **Path Parameters** dialog allows modification for the following parameters.

Parameter	Description
smoothness	Set path curve smoothness parameter. Higher the value, smoother will be the curve. Default (minimum) value is 1 - straight lines joining the path points.
sections	Set number of sections in the cross-sectional ellipse for the path. The number of sections are in multiples of 4. Higher the value smoother will be the cross-sectional ellipse. Its effect is visible only for when path is shown as tube. Default (minimum) value is 4 - four vertices making the cross-section.
color	Bring up a color dialog for changing color of the path.
opacity	Changing opacity of the path.
cap style	Set cap style for tube rendering.
arrow direction	Set arrow direction, when cap style is arrow.
arrow for all	When checked, sets arrow at all the points in the path, otherwise, draws arrow at one end. This is applicable only when cap style is arrow.
same for all	When checked, set the same radius and angle settings for all points, otherwise, set individual settings at each point.

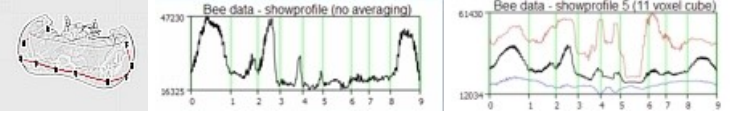
Following are key commands for paths.

Single Key Command	Description
DEL	Delete path.
t	Toggle display of path as caption/image/tube or line. If caption is enabled, then caption text is rendered. If image is enabled, then image texture is rendered. If neither is enabled then a tube is drawn.

Shift + t	Bring up caption dialog. Caption is enabled. When caption is enabled, pressing t will render caption text instead of tube.
c	Toggle connection of the two ends of path - making a closed or open path.
p	Toggle display of path points as big dots on the path.
n	Toggle display of path points numbers.
l	Toggle display of path length.
s Shift + s	Only when a point of the path is pressed, change radius S of cross-sectional ellipse for tube.
t Shift + t	Only when a point of the path is pressed, change radius T of cross-sectional ellipse for tube.
a Shift + a	Only when a point of the path is pressed, change rotation angle of cross-sectional ellipse for tube.
x/y/z/w	x y z w Constrain movement of point in x/y/z direction or make it free in all directions with w .

Following are the commands for paths.

Command	Description
moves [-]	moves Move the path in the direction of S-axis (red). moves - Move the path in the opposite direction of S-axis (red).
movet [-]	movet Move the path in the direction of T-axis (green). movet - Move the path in the opposite direction of T-axis (green).
caption [no]	caption Bring up caption dialog. Caption is enabled. When caption is enabled, pressing t will render caption text instead of tube. caption no Turn off caption for this path.
image [no]	image Image texturing is enabled. User is asked for the image file name. When image is enabled, pressing t will render image texture instead of tube.

	<p>image no Turn off image for this path.</p>
save	Save path coordinates to file.
normalize	Convert coordinates for all points into integers.
fill	<p>fill rad tag Tag the visible voxels within the cylindrical region around the path spline with the given tag value. rad parameter defines radius of the cylinder around the spline. Only those voxels that are connected to the spline will be tagged.</p>
fillempty	<p>fillempty rad tag Tag the non-visible voxels within the cylindrical region around the path spline with the given tag value. rad parameter defines radius of the cylinder around the spline. Only those non-visible voxels that are connected to the spline will be tagged.</p>
showprofile	<p>showprofile showprofile v Get raw data values along the path. These values are shown as a profile curve (shown below). When only showprofile is used, raw data values are extracted for voxels on the path. When showprofile is used with an integer value v, this value is used for placing a cube of size $2*v+1$ centered on each voxel on the path. Raw values are extracted for all voxels in the cube; average, minimum and maximum values are found at each point on the path. Three profile curves are drawn - one for each average value, minimum value and maximum value.</p> 
height	<p>height h Set height for the caption text or image texture.</p>
radius	<p>radius r Set radius for the cross-sectional circle to the given value r for all the points in the path - i.e. set radius S and T to be same.</p>
rads	<p>rads s Set radius S for the cross-sectional ellipse to the given value s for all the points in the path.</p>
radt	<p>rad t Set radius T for the cross-sectional ellipse to the given value t for all the points in the path.</p>
angle	<p>angle a Set rotation angle for the cross-sectional ellipse to the given value a for all the points in the path. This also affects the angle for caption and image.</p>

Following are the commands to extract raw data using paths.

Command	Description
---------	-------------

extractrawfast	<p>extractrawfast Extract raw volume along the path using nearest neighbour interpolation. This volume extraction facility allows users to unroll/unwrap regions of interest. Convoluted regions can be easily straightened using this technique. The opacity of voxels is used to mask regions in the volume data. Voxels in the selected region are also tested for clipping. For voxels that pass this test, opacity is calculated using the current transfer function. This opacity is then used for masking voxels in the original raw volume. If the opacity is zero - i.e. voxel is transparent - voxel value in the new raw volume will be set to mask value (which is supplied by the user at the start of this process). Voxels that have non-zero opacity - i.e. visible voxels - are stored at their original voxel value in the new raw file.</p> <p>Number of slices in the extracted volume are number of voxels along the path. The height and width of the slices is taken from the two radii parameters (rads and radt) of the first point on the path. The volume dimensions will be (no. of points along path) X (2*rads+1) X (2*radt+1)</p>
extractraw	<p>extractraw This command is very similar to extractrawfast, except it extracts raw volume along the path using linear interpolation. Linear interpolation is significantly slower compared to nearest neighbour interpolation, but gives better results.</p>
extractrawhalfpatchfast	<p>extractrawhalfpatchfast depth Extract raw volume using the patch defined by top half of skin for the given path. Patch that will be used for extraction can be seen by showing path as tube and switching on half section parameter. Nearest neighbour interpolation is used for extraction. Number of slices in the extracted volume is specified by the value of depth parameter. The height of the slices is the number of voxels along the path and width is half the circumference of the extrusion ellipse defined by two radii rads and radt.</p> <p>Opacity and clipping information is used as described in extractrawfast.</p>
extractrawhalfpatch	<p>extractrawhalfpatch depth This command is very similar to extractrawhalfpatchfast, except it extracts raw volume along the path using linear interpolation. Linear interpolation is significantly slower compared to nearest neighbour interpolation, but gives better results.</p>
extractrawfullpatchfast	<p>extractrawfullpatchfast depth Extract raw volume using the patch defined by skin for the given path. The skin patch that will be used for extraction can be seen by showing path as tube. Nearest neighbour interpolation is used for extraction. Number of slices in the extracted volume is specified by the value of depth parameter. The height of the slices is the number of voxels along the path and width is the circumference of the extrusion ellipse defined by two radii rads and radt.</p> <p>Opacity and clipping information is used as described in extractrawfast.</p>
extractrawfullpatch	<p>extractrawfullpatch depth This command is very similar to extractrawfullpatchfast, except it extracts raw volume along the path using linear interpolation. Linear interpolation is significantly slower compared to nearest neighbour interpolation, but gives</p>

	better results.
--	-----------------

Volume Masks

Masks allow users to tag individual voxels (akin to segmentation). This tagging allows users to render portions of the volume in a different way or erase the portion. Tagging does not in any way modify the original volume, it is only rendering that is affected. Users can also calculate volume and surface areas of tagged portions.

This tag information is stored in a separate file having the same name as the **.pvl.nc** file, but with **.mask** extension. For e.g. test.pvl.nc file will have mask information in test.mask. The mask file format is as follows :

single byte (value is 0)

followed by 12 bytes containing grid size as three 4-byte integers - [nX][nY][nZ]

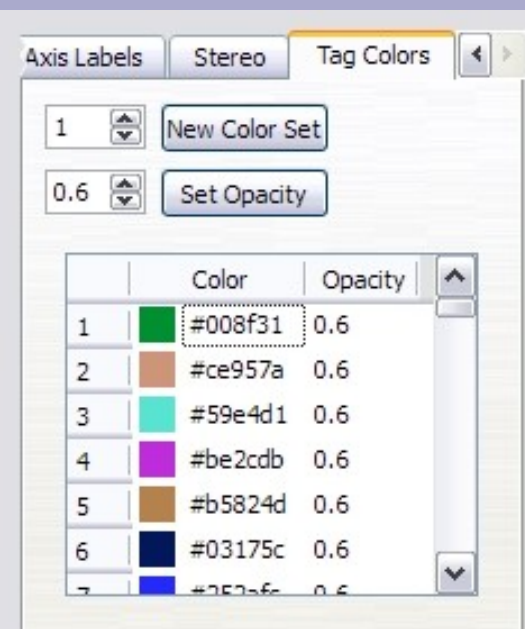
followed by the tag data which is 1-byte per voxel.

Volume masks are activated using **Enable Mask** switch under the **Toggle** menu. Volume masks can be used only for single volumes.

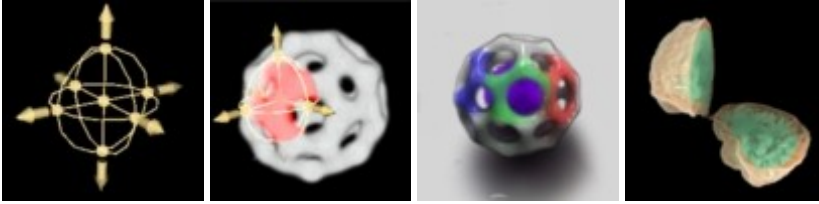
By default the tag value for all the voxels is 0. Tag value of 255 implies erase - i.e. the voxels having tag value of 255 will not be rendered, same as erasing them (the original volume is not touched, it is only the rendering that is affected). Voxels having tag values between 1 and 250 are tinged with a different color, when these are shown using transfer functions. Voxels that have tag values between 251 and 254 are rendered with the given tinge color even when no transfer function is applied - in essence these voxel are always shown (when the tinge opacity is greater than zero). The tinge color and opacity is taken from **Tag Colors** found under **Preferences** widget. Tag opacities are used to mix tinge color with the underlying transfer function color. Tinge colors and opacities can be changed and animated.

DrishTi provides two major ways for tagging the volumetric data - [2D paint interface](#) and [3D paintball interface](#). Users can also make use of paths (use *fill/fillempy*) for tagging regions.

Clipplanes can be used to restrict the masking process.

	Tag Value	Description
	0	Apply transfer function without change.
	1-250	Apply transfer function by mixing appropriate tag color with tag opacity. If the tag color is pure black i.e all components of tag color are 0, then tag opacity is used to modulate voxel opacity. This feature can be used for reducing opacity of the region occluding important feature in the data set. if (tag .gt. 0 .and. tag .lt. 251) voxelColor = voxelColor*(1-tagOpacity) + tagColor*tagOpacity if (tagColor .eq. black) voxelOpacity = voxelOpacity*tagOpacity
	251-254	Show voxel with appropriate tag color and opacity.

		<pre>if (tag .gt. 250 .and. tag .lt. 255) voxelColor = tagColor voxelOpacity = tagOpacity</pre>
	255	Do not show the voxel.

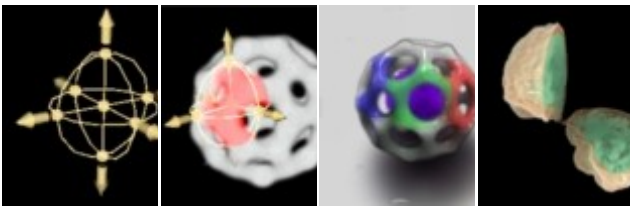


Volume Masks - 3D Paintball Interface

Once the [volume mask](#) is activated 3D paintball can be used for tagging the voxels. Tagging via paintball is enabled only when mask is enabled. By default paintball is not shown. Use **p** to show the 3D paintball. The paintball is shown as a wireframe ellipsoid. It can be repositioned by dragging the center dot. Size can be changed by dragging the boundary dots. Whenever the paintball intersects visible region of the volume, this region is tinged with red color.

Paintball is activated by hovering mouse over the center of the paintball. Active paintball is shown in green. Once activated users can give single keystroke commands or by pressing **spacebar** can bring up input.

Clippplanes can be used to restrict the masking process.



Single Key Command	Description
p	Toggle visibility of paint ball.
t	Tag the visible voxels within the ellipsoid with current tag value.
Shift+t	Tag the invisible voxels within the ellipsoid with current tag value. If the tag values are between 251 and 254 the effects will be immediately visible, otherwise the user will have to change the transfer function to see the effect.
f	Tag the visible voxels that are connected to the center of the ellipsoid and are within the ellipsoid with current tag value. This is essentially flood-fill with seed point as the center of the ellipsoid and bounded by the ellipsoid.
Shift+f	Tag the invisible voxels that are connected to the center of the ellipsoid and are within the ellipsoid with current tag value. This is essentially flood-fill invisible portion of the volume with seed point as the center of the ellipsoid and bounded by the ellipsoid. If the tag values are between 251 and 254 the effects will be immediately visible, otherwise the user will have to change the transfer function to see the effect.
s	Tag the visible voxels forming the surface boundary within the ellipsoid with current tag value. The thickness parameter defines how deep the surface goes.

Shift+s	Tag the invisible voxels forming the surface boundary within the ellipsoid with current tag value. The thickness parameter defines how deep the surface goes. If the tag values are between 251 and 254 the effects will be immediately visible, otherwise the user will have to change the transfer function to see the effect.
c	This is similar to flood-fill with seed point as the center of the ellipsoid and bounded by the ellipsoid, but in this case tag only the surface voxels.
Shift+c	This is similar to flood-fill invisible portion of the volume with seed point as the center of the ellipsoid and bounded by the ellipsoid, but in this case tag only the surface voxels. If the tag values are between 251 and 254 the effects will be immediately visible, otherwise the user will have to change the transfer function to see the effect.
x/y/z/w	Constrain movement of paintball in x/y/z direction or make it free in all directions with w .

Single Key Command	Description
d	Apply dilate operation using the tag and thickness value. This operations will affect only the visible voxels that lie within the ellipsoid. The region tagged with current tag value will be dilated by atmost thickness voxels on all sides.
Shift+d	Apply dilate operation using the tag and thickness value. This operations will affect only the invisible voxels that lie within the ellipsoid. The region tagged with current tag value will be dilated by atmost thickness voxels on all sides. If the tag values are between 251 and 254 the effects will be immediately visible, otherwise the user will have to change the transfer function to see the effect.
e	Apply erode operation using the tag and thickness value. This operations will affect only the visible voxels that lie within the ellipsoid. The region tagged with current tag value will be eroded by atmost thickness voxels on all sides.
Shift+e	Apply erode operation using the tag and thickness value. This operations will affect only the invisible voxels that lie within the ellipsoid. The region tagged with current tag value will be eroded by atmost thickness voxels on all sides. If the tag values are between 251 and 254 the effects will be immediately visible, otherwise the user will have to change the transfer function to see the effect.

Command	Description
tag	tag t Change current tag value to t. Current tag value is used for all subsequent tagging operations.

thickness	thickness t Change current thickness value to t. Current thickness value is used for all subsequent surface operations.
------------------	--

Volume Masks - 2D Paint Interface

Once the [volume mask](#) is activated 2D paint interface can be used for tagging the voxels. Tagging via 2D paint interface is enabled only when mask is enabled. Compared to the [3D paintball interface](#), the 2D paint interface is easier to operate. This interface allows users to paint on the surface of the volume. Surface is defined by the transfer functions. Users can specify how deep the tagging process can proceed. Voxels beneath the surface can be tagged in this way.

Tab key is used to toggle 2D paint interface. When 2D interface is enabled the last image is frozen on the screen and user cannot reorient the image. The program captures depth information for that image, which is used for enabling painting on the surfaces. Blue line are drawn in the high gradient areas. Areas where users cannot paint are colored in deep red. These are the areas where volume data is not rendered - it could be because of transfer functions or because of the limits of the data.

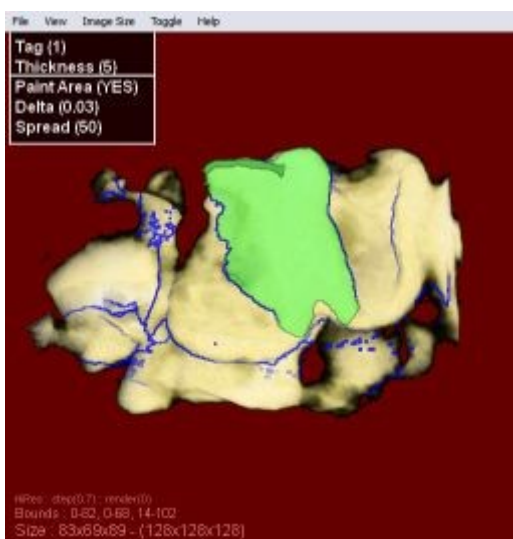
Turn off any geometry rendering - such as clip planes, bounding boxes etc.- before enabling 2D paint interface.

Once enabled a small menu will appear on the top-left corner of the image. The menu will list current *Tag*, *Thickness* (how far deep the tagging process can go), *Paint Area* : (Yes/No) (to paint using circular patch - *Paint Area* : Yes or to draw lines on the surface - *Paint Area* : No). When *Paint Area* is enabled *Delta* (depth difference threshold value) and *Spread* (radius of circular patch) are also displayed. Values of these variables can be changed by placing mouse on them and typing in the required values.

When *Paint Area* is "yes" a small green circular patch will follow the mouse, whenever it is on the surface as defined by transfer functions. The shape of this patch is dictated by difference between the depth values of point on the surface and the depth of point directly below the mouse, *Delta* (depth difference threshold value) and *Spread* (radius of the circular patch).

Use *Left mouse* to paint a patch or draw lines. *Right mouse* is used for deselected the patch. Users can color many patches/lines before deciding to proceed with tagging voxels. Use *Escape* key to clear all painted areas.

Clipplanes can be used to restrict the masking process.



Single Key Command	Description
--------------------	-------------

t (Tag)	<p>Tag the voxels with current <i>Tag</i> value. How deep the tagging process goes is defined by the <i>Thickness</i> parameter. Voxels that are tagged are those that :</p> <ol style="list-style-type: none"> 1. lie directly under the shaded region. 2. have opacity greater than zero. 3. are connected to the surface painted voxels. 4. no deeper from surface than the thickness value. <p>Interior voxels are said to be connected to the surface voxels if there is a connection that lies entirely within the shaded region.</p>
d (Drill)	<p>This is flood-fill with seed points as the surface painted voxels. Tag the voxels with current <i>Tag</i> value. <i>Thickness</i> parameter has no bearing on the tagging process. Voxels that are tagged are those that :</p> <ol style="list-style-type: none"> 1. lie directly under the shaded region. 2. have opacity greater than zero. 3. are connected to the surface painted voxels. <p>Interior voxels are said to be connected to the surface voxels if there is a connection that lies entirely within the shaded region.</p>
f (Fill)	<p>This is flood-fill with seed points as the surface painted voxels. Tag the voxels with current <i>Tag</i> value. <i>Thickness</i> parameter has no bearing on the tagging process. All the voxels connected to the surface painted voxels are tagged. Surface painting is used only for defining seed points for flood fill process and not for restricting the process to those voxels that lie directly underneath the shaded region.</p>

Networks (Graphs)

Users can view networks (graphs) with or without volumetric data. The networks are shown as balls (vertices) and sticks (edges). Hover over the centroid of the network and press space bar to bring up the panel to change various visualization parameters for that network. Multiple networks can be viewed simultaneously.

Users can drag-and-drop network files into Drishti provided the filenames are of the form ***porethroat.nc**.

Network (graph) is assumed to be stored in netCDF files with **.nc** extension. The network file should contain the following :

global attribute **gridsize**

variable **vertex_centers**

variable **edge_neighbours**

The **gridsize** global attribute has 3 integers specifying the grid size. X is the slowest varying axis and Z is fastest varying axis.

vertex_centers are 3 floats per vertex.

edge_neighbours are 2 integers per edge. These integers are indices of the two centers that the edge joins.

The netCDF file can also contain other variables. The scalar variables which have names starting with **vertex_** will be treated as properties of the vertices. The dimension of these variables must be the same as the number of vertices (i.e. the size of first dimension of **vertex_centers**). The scalar variables which have names starting with **edge_** will be treated as properties of the edges. The dimension of these variables must be the same as the number of edges (i.e. the size of first dimension of **edge_neighbours**). **vertex_radius** if present has a special meaning - the value will be used for radius of sphere depicting vertex, otherwise vertex is shown as a point.

edge_radius if present has a special meaning - the value will be used for radius of cylinder depicting edge, otherwise edge is shown as a line.

Following is an example of a network (graph) netCDF file (obtained using ncdump) :

```
netcdf pt80 {
dimensions:
nvertices = 11 ;
coords = 3 ;
nedges = 3 ;
connections = 2 ;
variables:
float vertex_centers(nvertices, coords) ;
float vertex_radius(nvertices) ;
float vertex_volume(nvertices) ;
int edge_neighbours(nedges, connections) ;
float edge_radius(nedges) ;
float edge_net_length(nedges) ;
// global attributes:
:gridsize = 80, 80, 80 ;
data:
vertex_centers =
4.829041, 76.55481, 16.07257,
7.333344, 78.66669, 53.83331,
11.83334, 64.66669, 23.66669,
12.66666, 67.66663, 19,
```

```
13.5, 68.5, 27,  
14.5, 79, 1,  
23.76926, 78.15387, 0.2307739,  
26.7973, 48.77142, 46.80756,  
38, 44, 42,  
39.40677, 66.16608, 61.58075,  
51.37497, 60.37494, 43.25 ;  
vertex_radius = 4.472136, 2.236068, 2.44949, 2.236068, 2.44949, 1.732051,  
1.414214, 9.899495, 4.898979, 4.690415, 1.414214 ;  
vertex_volume = 619.9999, 6, 6, 3, 2, 2, 13, 21150, 1, 3788, 8 ;  
edge_neighbours =  
1, 3,  
7, 9,  
8, 10 ;  
edge_radius = 0, 3.605551, 0 ;  
edge_net_length = 4.368e-005, 4.368e-005, 4.368e-005 ;  
}
```

Triangular Mesh

Users can view triangular mesh (trisets) with or without volumetric data. The trisets can be shown as dots or as triangles. Hover over the centroid of the triset and press space bar to bring up the panel to change various visualization parameters for that triset. Multiple trisets can be viewed simultaneously.

Users can drag-and-drop triangular mesh files into Drishti provided the filenames are of the form ***.triset**.

Triangle sets (triangular mesh) are assumed to be stored in files with **.triset** extension. The format of triset file is as follows :

```
[TAG]
[NX][NY][NZ][NVERT][NTRI]
[Coordinates]
[Normals]
[Triangle Indices]
```

TAG is a single byte of value 0.

NX, NY, NZ are 3 integers specifying the grid size. X is the slowest varying axis and Z is fastest varying axis.

NVERT is total number of vertices.

NTRI is total number of triangles.

Coordinates are vertex coordinates as triplets of x,y,z values as floats. They are NVERT in number.

Normals are normals as triplets of x,y,z values as floats. They are NVERT in number.

Triangle Indices are triangle indices coordinates as triplets of integers. They are NTRI in number.

Following code is used to load the triset file.

```
fread(&tag, sizeof(unsigned char), 1, fd);
if (tag != 0) -- wrong format - flag error
fread(&nX, sizeof(int), 1, fd);
fread(&nY, sizeof(int), 1, fd);
fread(&nZ, sizeof(int), 1, fd);
fread(&nvert, sizeof(int), 1, fd);
fread(&ntri, sizeof(int), 1, fd);
coord = new float[nvert*3];
fread(coord, sizeof(float), nvert*3, fd);
normal = new float[nvert*3];
fread(normal, sizeof(float), nvert*3, fd);
tri = new unsigned int[ntri*3];
fread(tri, sizeof(unsigned int), ntri*3, fd);
```

Transfer Functions

Transfer functions map voxel information to optical properties. Drishti implements transfer function as 256x256 lookup table of colors and opacities. Users are presented with a plotting interface in the transfer function widget. Horizontal axis represents the voxel intensities and vertical axis represents the rate of change of voxel intensity (gradient magnitude).

Transfer Function Editor has 3 sections - manager, plotting interface and color-opacity gradient interface.

Plotting Interface

The horizontal axis represents voxel intensities and vertical axis represents gradient magnitude.

Buttons(s)	Description
Left	Add transfer function spline point. Move an existing spline point. Move spline-normal independently.
Middle	Move entire transfer function spline.
Right	Remove transfer function spline point. If clicked above the interface box, then right button can be used to toggle overlay values.
Shift+Left	Move an existing spline point vertically. Move spline-normal points together.
Ctrl+Left	Rotate spline-normal.
Alt+Left	Move an existing spline point horizontally.

Color-Opacity Gradient Interface

The vertical axis in this interface represents the opacity. Higher values represent more opacity. The color and opacities are mapped across the transfer function spline.

Buttons(s)	Description
Left	Add gradient point. Move an existing gradient point. The two end gradient points can only be moved vertically.
Right	Remove gradient point. The two end gradient points cannot be removed.
Shift+Left	Move an existing gradient point vertically.
Alt+Left	Move an existing gradient point horizontally.
Double Click	Open colour editor.

Transfer Function Editor

New Refresh Remove

Replace existing transfer functions at keyframes

Morph transfer functions during keyframe animation

Name	0	1	2	3
TF 0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

1D 2D

RGB/RGBA Volume

RGB/RGBA volumes or photographic volumes can be rendered using Drishti. Such type of volumes present an interesting challenge for volume rendering. As colors are already known, transfer functions may seem unnecessary. But in order to bring out various features in the data, user may need to modify the opacity. This allows exploration of photographic volumes without having to commit to an a priori segmentation that might mask fine variations for interest.

The transfer function for photographic volume is employed to determine opacity for a voxel. Compared to this, for gray-scale volumes it is used to determine both color and opacity.

Ideally one would like define an opacity transfer function in the RGB space, which would allow for full exploration in color space. But this would result in an extremely complicated user interface for defining the opacity transfer functions. Drishti solves this problem by employing 3 opacity transfer functions defined on 3 faces of the color cube whose axes span red(R), green(G) and blue(B). Users define opacity transfer functions on RG, GB and BR faces respectively. Drishti then takes intersection of these 3 transfer functions to get the final opacity transfer function to be applied to the color data. Shown below is rendering of visible human leg along with the 3 opacity transfer functions.

The image displays three panels for defining opacity transfer functions (TF) on different faces of the color cube. Each panel consists of a control table, a 2D grid visualization, and a 1D line graph.

Left Panel (RG Face):

Name	0	1	2	3	4	5	6	7
TF1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TF2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TF3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table is a 2D grid visualization showing a grayscale gradient across the RG face. The '2D' radio button is selected. Below the grid are radio buttons for '1D', '2D', 'BR', 'GB', and 'RG', with '2D' and 'RG' selected.

At the bottom is a 1D line graph on a checkerboard background, showing a curve connecting three points.

Middle Panel (GB Face):

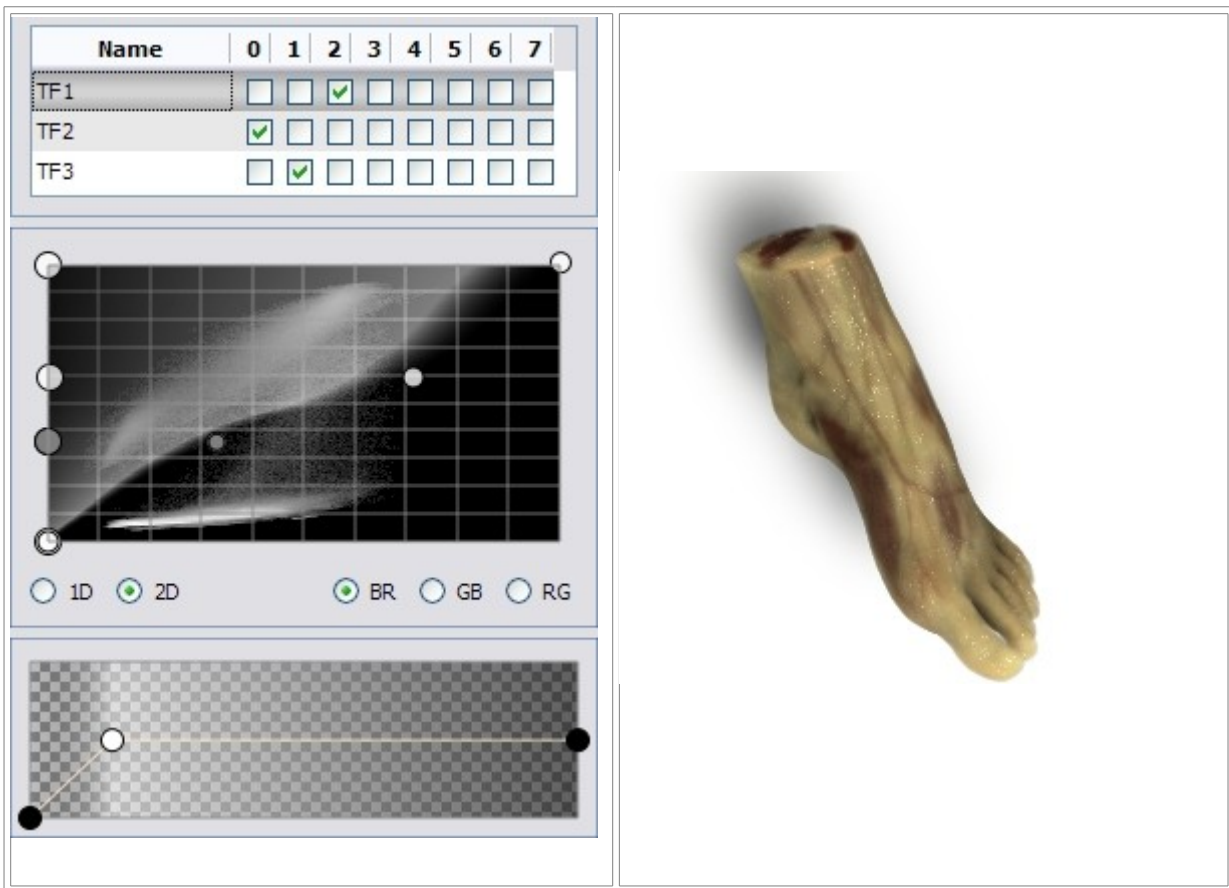
Name	0	1	2	3	4	5	6	7
TF1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TF2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TF3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table is a 2D grid visualization showing a grayscale gradient across the GB face. The '2D' radio button is selected. Below the grid are radio buttons for '1D', '2D', 'BR', 'GB', and 'RG', with '2D' and 'GB' selected.

Right Panel (BR Face):

Name	0	1	2	3	4	5	6	7
TF1	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TF2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
TF3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table is a 2D grid visualization showing a grayscale gradient across the BR face. The '2D' radio button is selected. Below the grid are radio buttons for '1D', '2D', 'BR', 'GB', and 'RG', with '2D' and 'BR' selected.



A transfer function must be defined for each of the first 3 sets. Same transfer function may be supplied to multiple sets. Compared to gray-scale volumes, transfer functions for photographic volumes are pretty complicated. Color for the transfer functions do not matter in this case, it is only the opacity that is used.

The transfer functions defined for photographic volume are still 2D transfer functions, except in this case the horizontal and vertical axes have different meanings. Depending on the transfer function set involved the horizontal and vertical axes are R-G, G-B or B-R respectively.

Last three sets may be used for defining transfer function for emissive lighting. In the case of emissive lighting, though, the color of transfer functions come into play.
