User's Guide

PMOD 3D Rendering Tool (P3D)

Version 3.9

PMOD Technologies

Printed on 30 October, 2017

Contents

PMOD 3D Rendering Tool Introduction

Chapter 1

Overview

Physicians trained in cross-sectional imaging are able to understand the spatial relationship between tissue structures by mere exploration of cross-sectional slice images. However, for communication purposes it is often helpful to generate simulated views of isolated objects of interest from different viewpoints. 3D image processing allows deriving such objects from slice images and calculating virtual reality scenes which can be explored interactively.

Surface Rendering (SR)

One way to derive a virtual object is to track (segment) an organ boundary in the slice images, build a 3D surface from the contours, and shade the surface. The P3D tool supports such segmentations by applying thresholds and/or region growing, optionally restricted within volumes-of-interest. As a special option, the object can be colored by projecting the information of matched images onto its surface (texturing), even animated in time. This feature allows, for instance, visualizing the concentration changes of the NH3 perfusion tracer at the myocardium surface throughout a dynamic acquisition.

Volume Rendering (VR)

A different way to derive a virtual object is to take a certain viewpoint in front of the object, cast rays through the object and record the ray values when the rays pass a plane behind the object, thereby producing an image. The ray value recorded in the image plane is a combination of the values in all the pixels met along the way from the viewpoint to the image plane, thus the name Volume Rendering. Typically, the combination is just the sum of the pixel values, each multiplied by a certain weighting function, called opacity. The result depends heavily on the image values and the opacity function. There are very successful configurations available for contrast enhanced CT examinations which provide the illusion of a three-dimensional representation, especially if the viewpoint is interactively changed.

Skeleton Rendering (Path)

An additional way to derive a virtual object is to extract the "center-lines" of a 3D binary image generated by a segmentation algorithm. These lines are known as "paths" or "skeletons" and can efficiently represent a 3D object, like a SR or a VR one. The curve skeletons are well suited to describe tube-like anatomical structures, e.g. vessels, nerves, and elongated muscles. When a selected path is bound with an oblique plane, the plane is automatically placed perpendicular to the path direction. This feature facilitates the placement and the cutting of a vessel with an oblique plane at 90 degree angle.

Scene Generation in P3D

The virtual reality scenes in P3D are constructed by segmenting tissue structures from images, and rendering them using Surface, Volume or Skeleton Rendering techniques. The scenes can be interactively explored to understand the spatial relationships of the segmented tissue structures. To this end the scene can be rotated in any direction, zoomed, and objects obstructing the view to deeper ones can be temporarily hidden or set to a high degree of transparency. Furthermore, planes showing the original image data can be added, or

volumes-of-interest (VOI). Meaningful renderings can be saved as a screen captures, or a movie of a rotating scene can be generated. Protocol files allow reconstructing a particular scene from the original data at any later time.

Combination of 3D objects from Matched Series

A unique feature of P3D is the ability to combine and manipulate different types of virtual reality objects (VRO) in one common scene, even when they stem from different studies. Once a scene has been created, it can be saved in a VRML format file (SR objects only). These files can later be loaded into P3D (or an external VRML-Browser) to continue scene exploration. It is in principle also possible to import VRML-scenes from some dedicated 3D rendering programs and combine them with P3D objects, but the import may sometimes fail. So far, AMIRA data has been successfully imported.

In addition, the curve skeletons can be saved as paths (*.vec). These files can later be loaded in P3D to recreate the skeleton scene. Note that the 3D binary image segment used initially to create the skeleton is not required anymore.

Requirements

Hardware Requirements and Performance

The P3D tool is based on the OpenGL implementation in the ***Java3D library*** (***http://java.sun.com/products/java-media/3D/index.jsp***). Therefore, using P3D requires a *graphics board which supports OpenGL*, preferably with at least *1GB* video RAM to accommodate high quality texture displays. As the 3D rendering implementation in Java3D is quite memory demanding (particularly with VR), a *64-Bit* operating system and a *RAM size of at least 8GB* is required when working with highly resolved or dynamic data. We strongly suggest testing P3D on your platform with your own data before purchasing this PMOD option.

Image Data Requirements

There are situations where objects are derived from a single data set, for example the boundary of a brain tumor and the outer brain contour. In this case the spatial alignment of the generated objects will be inherently correct.

However, if objects from different image series are rendered in a single scene, some requirements must be met to ensure that the objects are correctly positioned in the scene. The positioning rules in P3D are as follows:

1) The generated objects are positioned relative to the *center* of the scene volume (i.e. the center of the image volume is aligned with the center of the scene).

2) For the placement and the size of the objects the pixel size is taken into account.

So if the centers of two image volumes are aligned, objects derived from them can be combined, even if the pixel size is not the same.

CAUTION: When combining multiple studies in a single rendering, please match beforehand all images series in the PFUS tool and save the resampled images. Otherwise, unexpected shifts between the objects might occur in the scene.

When importing virtual reality (VRML) objects generated by other programs this behavior may not be adequate. In this case it can be changed in the P3D tool configuration.

Starting P3D

The P3D tool is started from the PMOD ToolBox by clicking the dedicated button, or by dragging image files onto the button.

An automatic **Acceptance test** is taking place when P3D is started for the first time. Several 3D scenes are presented for visual inspection and a dialog window opens for each scene. Please confirm whether the expected output is shown on the screen or not.

The following tests are performed:

1) Test of texture 3D support.

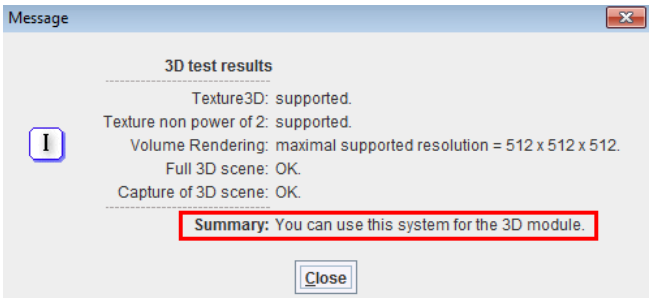
2) Test of a non power of 2 texture support.

3) Test of maximum resolution of Volume rendering (from 128x128x128 up to 512x512x512).

4) Test of a full 3D scene loaded from protocol.

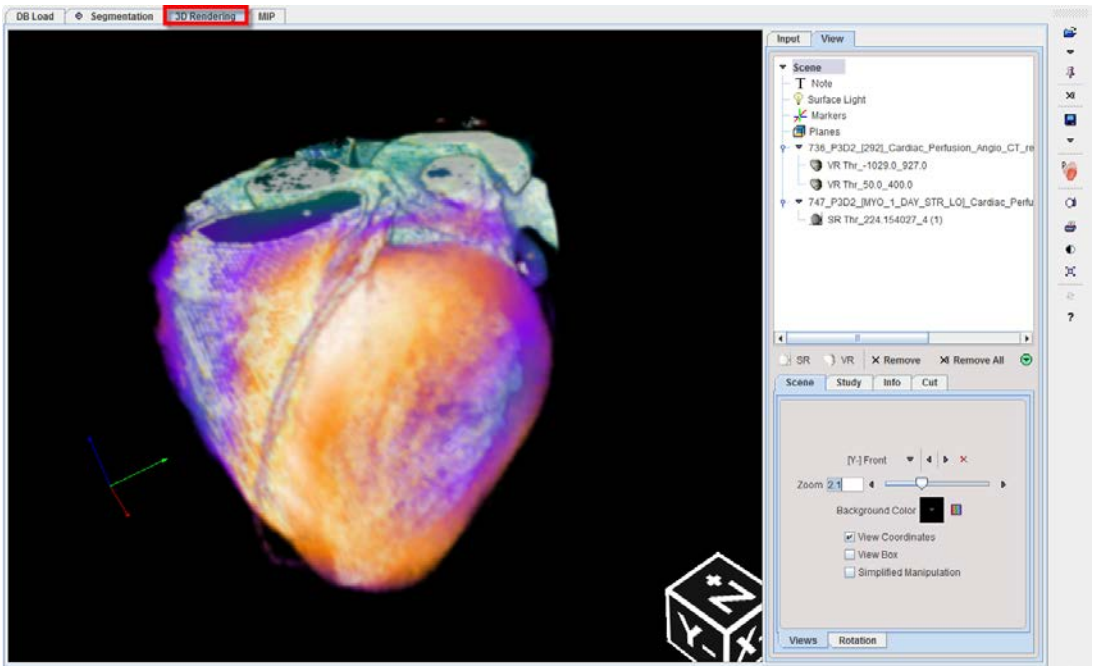
5) Test of a scene capture.

At the end of all steps the test results will be summarized in a message window. If the test succeeded the results should be similar like the one below:



User Interface

P3D organizes the functionality on four pages which can be selected by the upper tabs.



The detachable **taskbar** at the right side of the application window offers the following functions. Please also note the button tooltips which provide short explanations.

Load input image data. The arrow below the load button is used for switching among the available image formats.

If the pin is fixed, loaded studies will be appended to the list of present studie(s). Otherwise the prior studies will be overwritten.

Save results of the last segmentation. The arrow below the save button is used for switching among the image formats which can be used for saving.

Contains rendering definitions for some data combinations as well as example renderings.

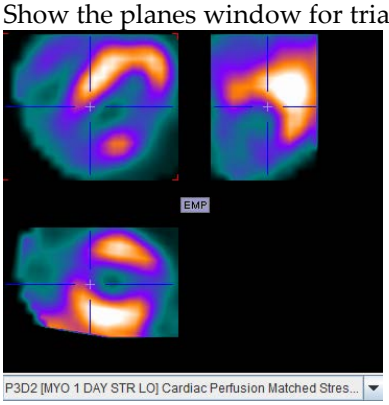
Allows capturing the 3D scene.

Allows printing a report of the 3D scene.

Allows hiding/showing image controls

Toggle button for enlarging/shrinking the image viewport in the **3D Rendering** page, **Input** tab.

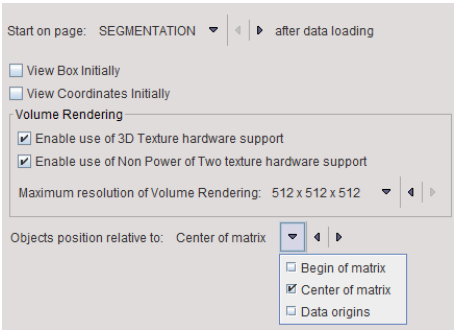
Show the planes window for triangulating a position



Explicit call to a scene refresh. This button is only enabled after changing VR properties.

Configuration Settings

There are a few specific configurations for the P3D tool. They can be accessed by the **Menu** entry **Settings/Modify**, or directly using the  button.



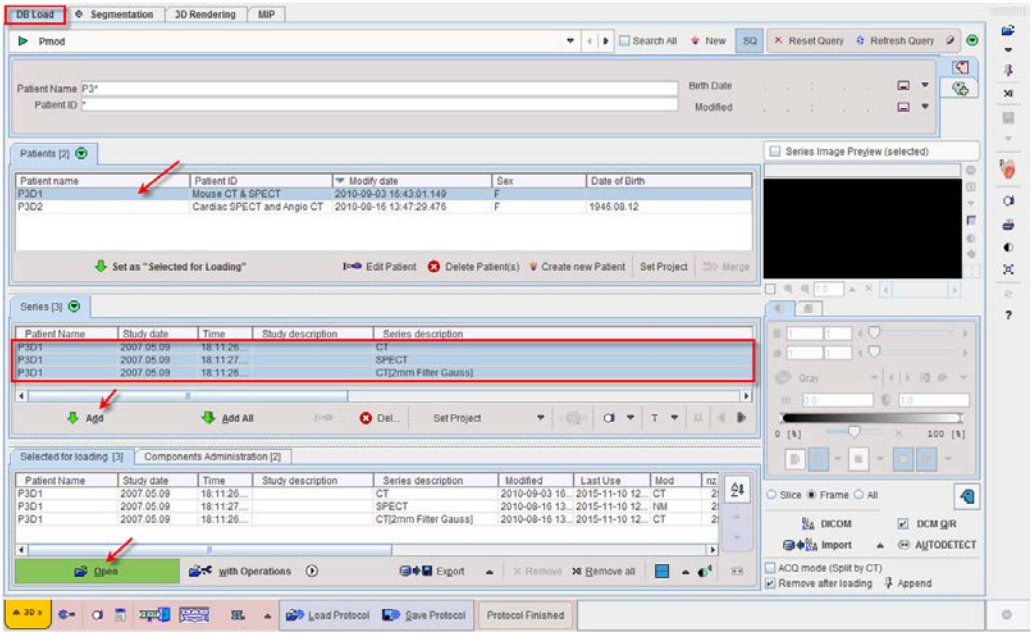
The **View Box Initially** check is for enabling the wire frame box, and **View Coordinates Initially** the orientation cube at start-up.

Some graphics boards may also support volume rendering textures in hardware. To exploit this acceleration feature, check the **Enable use of 3D Texture hardware support** box, and/or the **Enable use of Non Power OF Two Texture hardware support** box. **Maximum resolution of Volume Rendering** is identified during the **Acceptance test**. However it can be decrease selecting a smaller resolution available on the list.

Objects generated in P3D are positioned relative to the center of the scene volume (i.e. the center of the image volume is aligned with the center of the scene, **Center of matrix**). When importing virtual reality (VRML) objects generated by other programs this behavior may not be adequate. It can be modified by setting **Objects position relative to** to **Begin of matrix, Center of matrix** or **Data origins**.

Image Data Loading

The images can be loaded via **Load Input** in the **P3D** menu, the button in the taskbar to the right, or if the PMOD database functionality is enabled from the **DB Load** page as illustrated below.



Note that several images can be loaded at once. It is also possible to incrementally add images later on. To do so, the append pushpin button should be activated ( ) when loading.

Segmentation Page

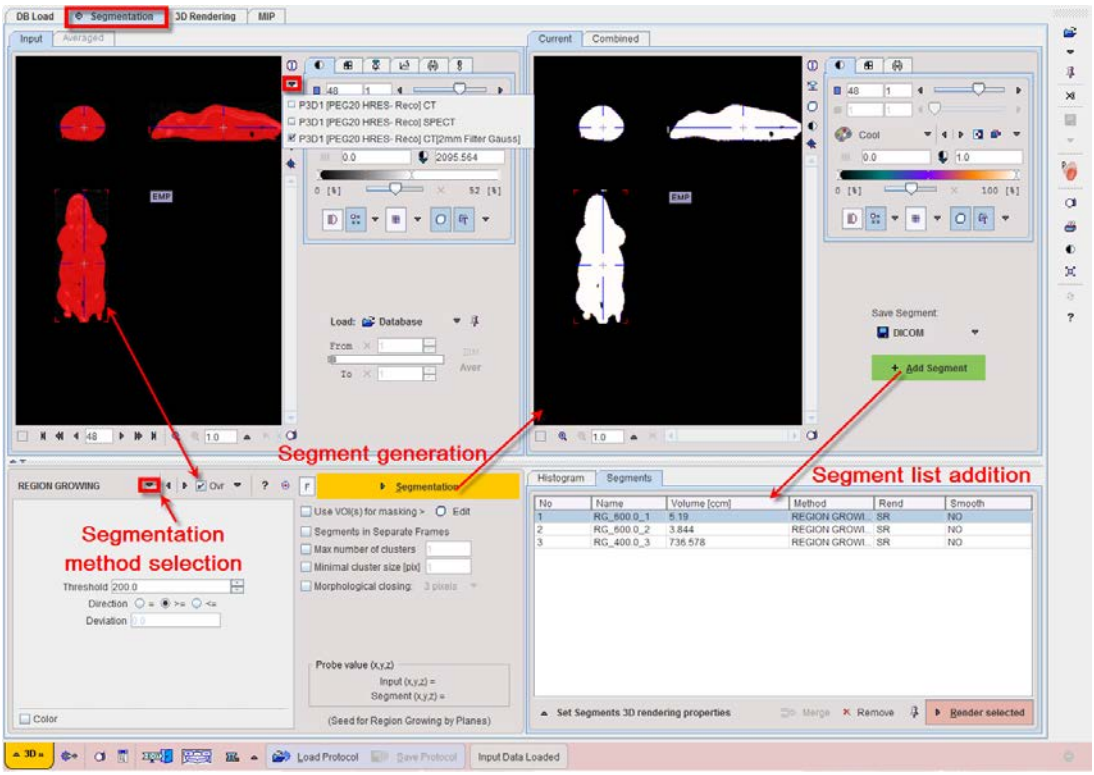
Please refer to the *PMOD Base Functionality Guide* for more information about image loading.

Segmentation is a crucial step of 3D rendering. In P3D this task is preferably performed in the **Segmentation** page. The **Input** tab on the **3D Rendering** page offers similar functionality, but is more difficult to use because of the compact size.

Segmentation Page Layout

The **Segmentation** page is organized as follows:

- The input images are shown in the upper left. An overly color is used to highlight the pixels which qualify for the current segmentation settings. This segmentation preview is only approximate.
- The segmentation method is configured in the lower left by a list selection. The parameters of the selected method are updated accordingly. P3D offers several general ***segmentation methods*** (on page 22) for finding the object contours. The contour finding process can further be restricted by an enclosing volume-of-interest (VOI).
- The actual segmentation is initiated with the **Segmentation** button. It produces a binary image in the upper left with a black background and the white object.
- Useful segmentations can be recorded by the **Add Segment** button and appear in the list in the lower right.
- **Render selected** converts the selected segments into objects on the **3D Rendering** page.

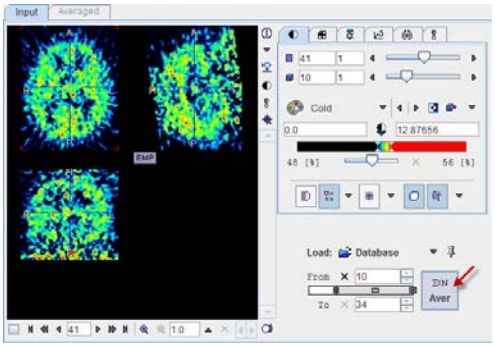


The different steps outlined above are described in the following sections.

Image Selection

The loaded images are stacked in the **Input** display in the upper left. The current image can be selected with the down arrow as illustrated above.

When working with dynamic series, the segmentation can be applied to the currently displayed frame, or to each frame separately as described below. Alternatively, a frame average image can be created with a range specified by the **From** and **To** numbers. The average is calculated by the **Aver** button and the result image is available on the **Averaged** pane.



Each **Input** study has its own segmentation definition which will be updated when switching between the studies. Therefore, to begin with a segmentation task, first select the appropriate **Input** study, and then adjust the segmentation method.

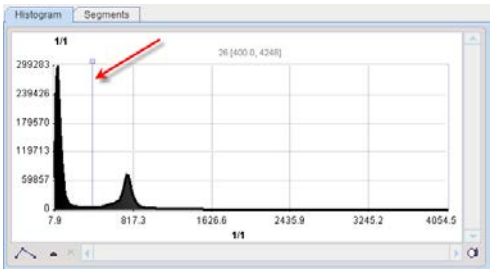
The segmentation will use the current image in the upper left.

Segments Creation

The next step consists of generating segments which represent tissues of interest.

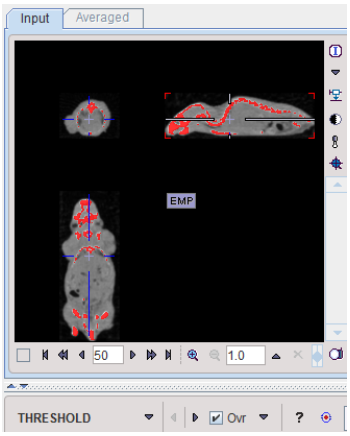
Pixel Value Distribution

The **Histogram** in the lower right provides an overview of the pixel value distribution in the current image volume. It may indicate a peak usable for setting a background threshold. The current threshold value is indicated by the vertical line in the histogram.



Segmentation Method Selection

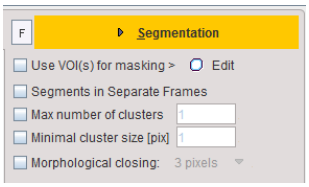
The segmentation method for defining the pixel inclusion criterion is selected by the list selection. Each method has its own parameters which are described ***below*** (on page 22). A preview facility is available to get an impression of the performance, although it is not effective for all methods. It is enabled by the **Ovr** box. Consequently, the pixels which satisfy the criterion highlighted in the image overlay. This works best if the image uses the **Gray** color table.



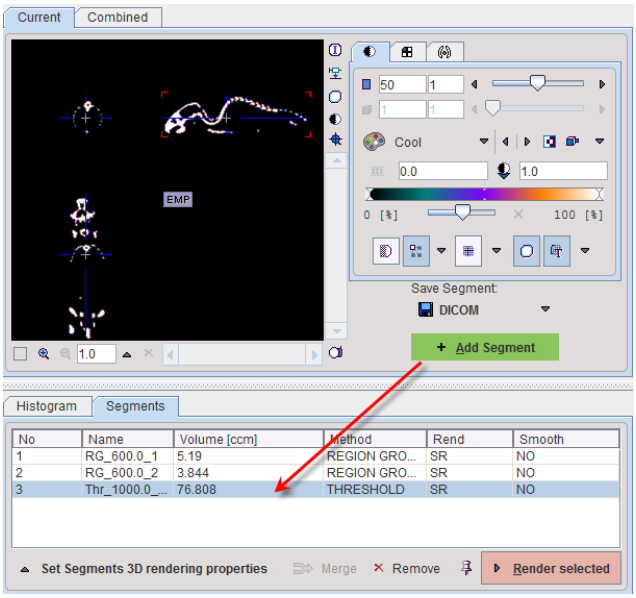
CAVEAT: Overlay updating might be slow for high-resolution data sets. In this case it is worthwhile switching the **Ovr** box off for most of the time.

Segmentation

The **Segmentation** button starts the actual segmentation on the selected image. The **F** toggle button in front of the **Segmentation** button is applicable for dynamic **Input** images. If it is enabled, only the currently shown frame is processed, otherwise all of the dynamic frames resulting in a "dynamic segment". The purpose of dynamic segmentation is to construct an object which can be animated over time.



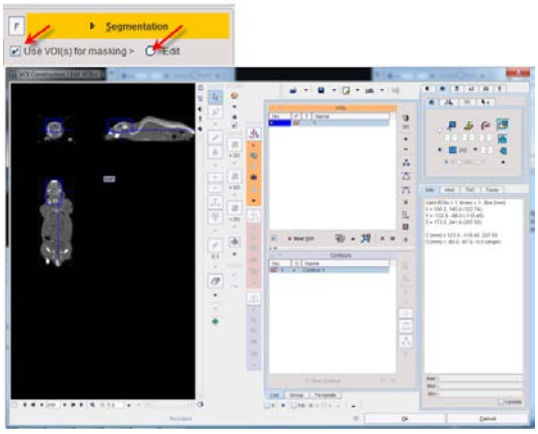
The result is shown in the **Current** tab to the right, overwriting any existing content. The **Add Segment** button appends the **Current** segment to the **Segments** list, an also adds it to the **Combined** display.



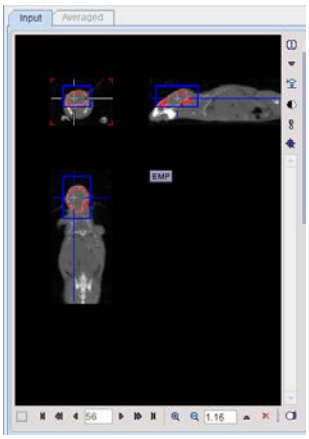
The **No** entry in the **Segments** list indicates the number by which a segment is identified in the **Combined** image, **Name** provides some descriptive information, **Volume** its physical volume, and **Method** identifies the applied segmentation method. Multiple segments in the list can be selected and transformed into a single segment by the **Merge** button. Hereby, the initially distinct values are replaced by a common value, and the original list entries deleted.

Segmentation with VOI Masking

In certain circumstances, the segmentation methods alone may not be sufficient for separating an object from other structures. If this happens, the user can define a VOI which prevents segmentation from leaving the area of main interest. To do so, the **Use VOI(s)** box has to be enabled and the **Edit** VOI button activated. The VOI tools interface appears and allows drawing a VOI. In the example below a cuboid object VOI was centered on the mouse head.

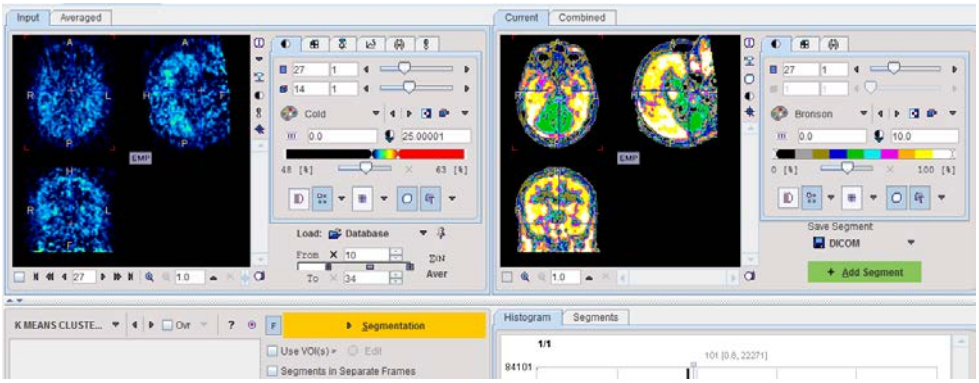


Closing the VOI tools with the **Ok** button confirms the VOI definition, and the criterion is only applied within the VOI.



Saving Segments

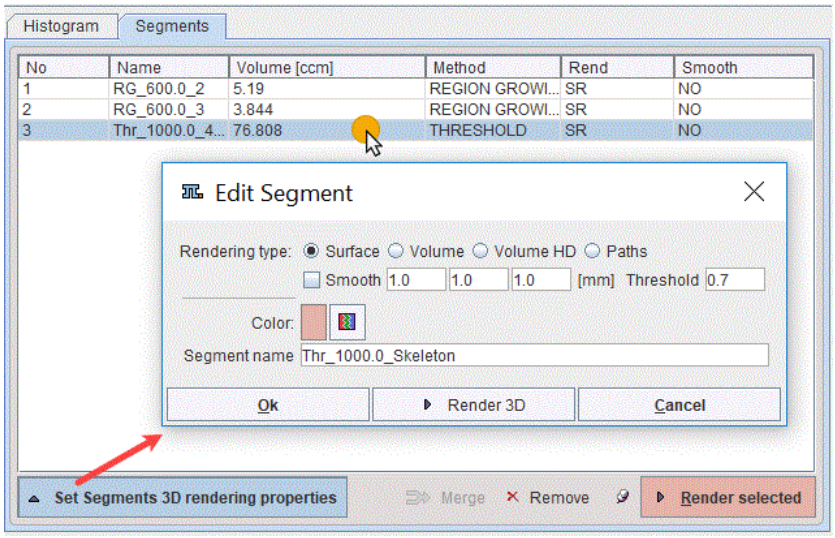
While standard segmentations create binary images with 0 (background) and 1 (segment) pixel values, there are clustering approaches which generate multiple segments in a single calculation. These segments are distinguished by increasing integer pixel values as illustrated below with the result of a k-means clustering of the dynamic uptake.

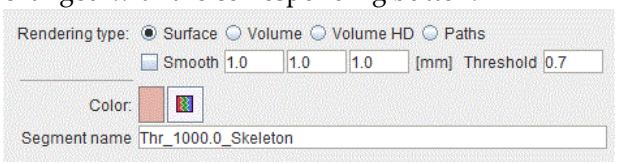
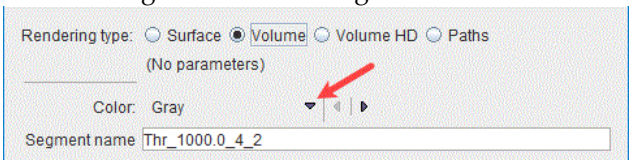


The **Current** segment or the **Combined** segments can be saved as images in any of the supported formats using the **Save segment** button.

Rendering Properties

Each entry in the **Segments** list has its own rendering properties which can easily be changed. **Set Segments 3D rendering properties** or double clicking a list entry brings up the dialog window below.



Rendering type	▸▸ **Surface** (default) generates a SR object. The surface **Color** can be changed with the corresponding button.  **Smooth**ing applies a Gaussian filter before the 3D surface mesh generation. This will change the apparent volume of the segments and is thus not recommended for small objects. ▸▸ **Volume** generates a VR object. The initial color table to be applied can be set as illustrated below. The other ***VR properties*** (on page 46) can be changed after rendering.  ▸▸ Volume HD is the same as **Volume**, but in order to achieve smooth animations the data is interpolated to the smallest pixel dimension. ▸▸ **Path** extracts the "center-lines" of a 3D binary image generated by a segmentation algorithm (SR or VR) and generates curve skeletons also known as path vectors:

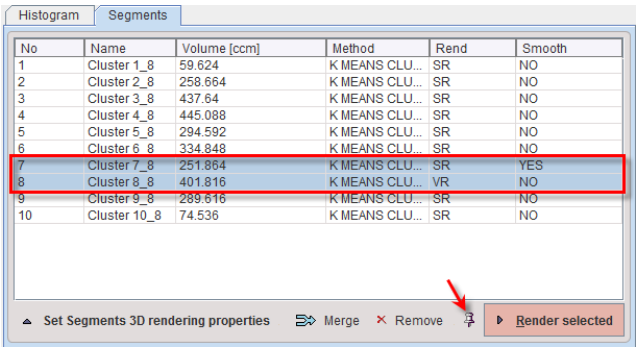
	
Segment name	Change the name to a meaningful string.
Ok	Close, accepting the changed properties.
Cancel	Close without changes.
Render 3D	Starts the 3D rendering based on the selected rendering properties. The created object appears in the **3D Rendering** scene and in the **View** tree.

Saving Segments

The **Current** segment or the **Combined** segments can be saved as images in any of the supported formats using the **Save segments** button.

Object Rendering

Rendering of the segments in the list is selective. A number of entries needs to be selected first, and then **Render selected** applied.



CAVEAT: As long as the "Append" pushpin indicated above is not activated, the 3D scene is cleared before each rendering. Please enable the pushpin for incrementally building up the scene.

The results are shown on the **3D Rendering** page.

Segmentation Methods

ITK-based Implementations

Some of the segmentation methods are based on the use of the ITK (Insight Toolkit) libraries. ITK can be used under the open-source BSD license which allows unrestricted use, including use in commercial products (see www.itk.org). The ITK-based methods are clearly denoted with ITK in brackets.

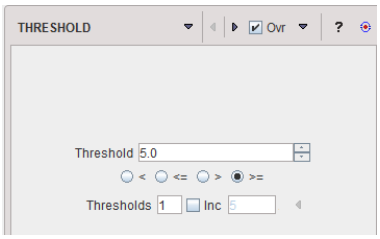
Note: PMOD Technologies cannot be held liable for permanent support of the ITK interface, nor for the performance of the provided libraries.

Region Growing Methods

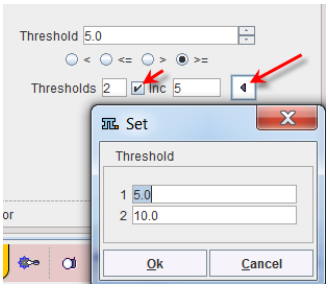
"Region growing algorithms have proven to be an effective approach for image segmentation. The basic approach of a region growing algorithm is to start from a seed region (typically one or more pixels) that are considered to be inside the object to be segmented. The pixels neighboring this region are evaluated to determine if they should also be considered part of the object. If so, they are added to the region and the process continues as long as new pixels are added to the region. Region growing algorithms vary depending on the criteria used to decide whether a pixel should be included in the region or not, the type of connectivity used to determine neighbors, and the strategy used to visit neighboring pixels." ***The ITK Software Guide.*** (***http://www.itk.org/ItkSoftwareGuide.pdf***)

THRESHOLD Segmentation

The **THRESHOLD** segmentation is conceptually simple. All pixels with value above or below the threshold are included in the segment, depending on the condition settings.



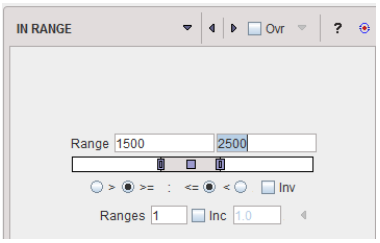
Sometimes it is helpful to segment at several threshold levels at once. This can easily be realized by setting the number in the **Thresholds** field accordingly, and entering the threshold values in the appearing **Set** dialog window.



Multiple threshold values can be incremented with a constant value enabling the **Inc** box and specifying the step in the dedicated text box.

IN RANGE Segmentation

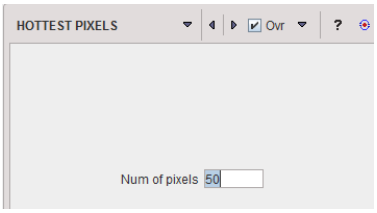
The **IN RANGE** segmentation is similar to the **THRESHOLD** method, except that two limits are defined. The boundary values are included or excluded, depending on the radio button settings. The complementary criterion is obtained by checking the **Invert** box.



Similar to the **THRESHOLD** segmentation, multiple **Ranges** can be defined and segmented at once.

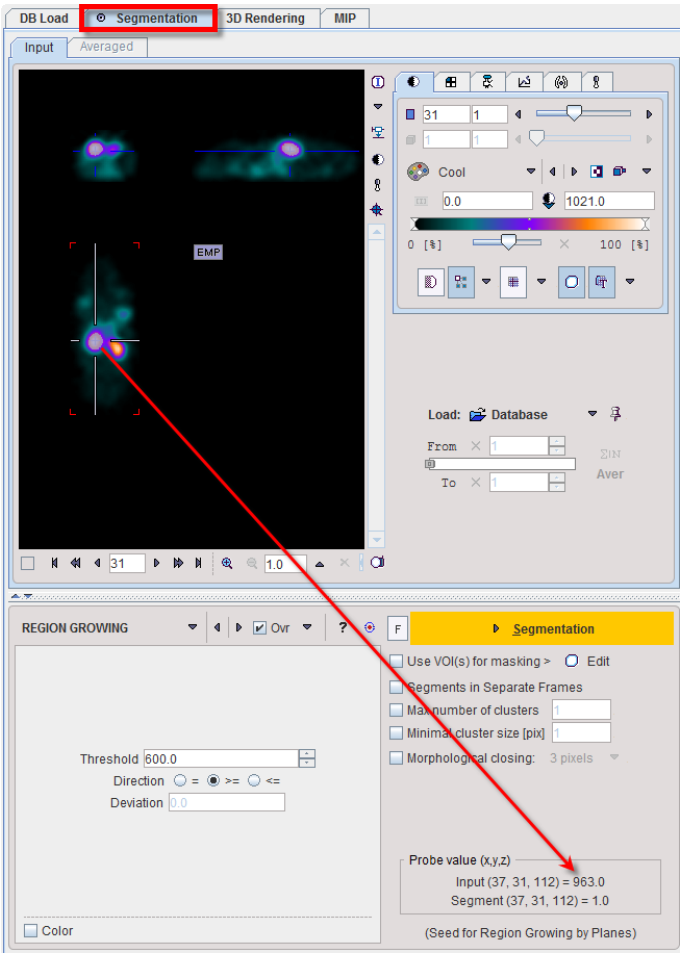
HOTTEST PIXELS Segmentation

The **HOTTEST PIXELS** segmentation allows obtaining the (often disconnected) pixels with the highest values. The number of included pixels can be specified in the **Num of pixels** field.



REGION GROWING Segmentation

REGION GROWING is a method by which the user defines a starting point (seed) within the object of interest, and the algorithm tries to find all connected pixels which fulfill a certain criterion. It is required to triangulate the seed point with the orthogonal plane layout as illustrated below.



The inclusion criterion is formed by the **Threshold** value and the **Direction** specification:

- = include all connected pixels with a value **Threshold** ± **Deviation**;
- >= include all connected pixels above the defined **Threshold** value;
- <= include all connected pixels below the defined **Threshold** value.

Note the **Probe value** which displays the image value at the position of the cursor, which is useful for finding appropriate **Threshold** setting.

Brain Extraction (G + W + CSF)

This segmentation can be used for

1) the extraction of the whole brain as a unique segment from a T1 weighted brain MRI. To this end the **Brain (MRI) Type** need to be selected.

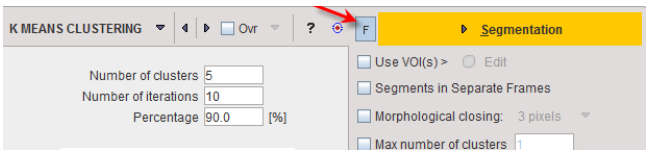
2) the classification of the pixels in a T1-weighted brain MRI into Grey Matter, White Matter and CSF when the **G+W+CSF** is selected as **Type**.

3) the extraction of the whole brain as a unique segment based on the **Mask** of the selected **Template,** with parameter **Sampling distance** in mm and the **Probability level** for the map discretization.

Note: The methodology variants from point 1 and 2 above is an implementation of the segmentation in SPM8 (**3 Probability Maps**) and SPM12 (**6 Probability Maps**) respectively, with parameter **Sampling distance** in mm and the **Probability level** for the map discretization.

K-MEANS CLUSTERING

K-Means is a ***popular http://en.wikipedia.org/wiki/K-means_clustering*** method which partitions N observations into K clusters. The parameters are the **Number of clusters** (=K), the **Number of iterations** of the heuristic procedure, and an assumed **Percentage** of background pixels.



The procedure can be applied to static and dynamic data. In the latter case, pixel with similar signal shapes will be clustered using the time-weighted Euclidean distance as the measurement of dissimilarity (or distance) between TACs. Note that the **F** button has to be enabled, because otherwise each frame of the dynamic series will be clustered separately.

The procedure for clustering based on the signal shape proceeds as follows:

1) Background pixels are removed by calculating the signal energy of the pixel-wise TACs (sum of squared TAC values), and considering only pixels above a specified percentile.

2) K non-background pixels are randomly selected as initial cluster centroids .

3) Each non-background pixel is assigned to the centroid with minimal distance between the TACs, thus forming K initial clusters.

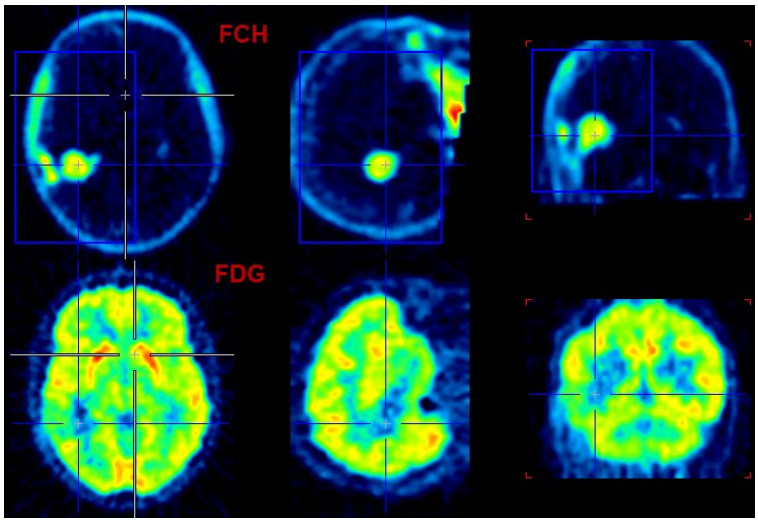
4) For each cluster a new centroid TAC is calculated as the average TAC of all pixels in the cluster.

5) An iterative process is started which repeats the following two steps:
(1) Each pixel TAC is compared with all centroid TACs and assigned to the cluster with minimal distance.
(2) All centroid TACs are recalculated to reflect the updated cluster population.
The iterations are repeated until no pixels are re-assigned to a different cluster, or a maximal number of iterations is exhausted.

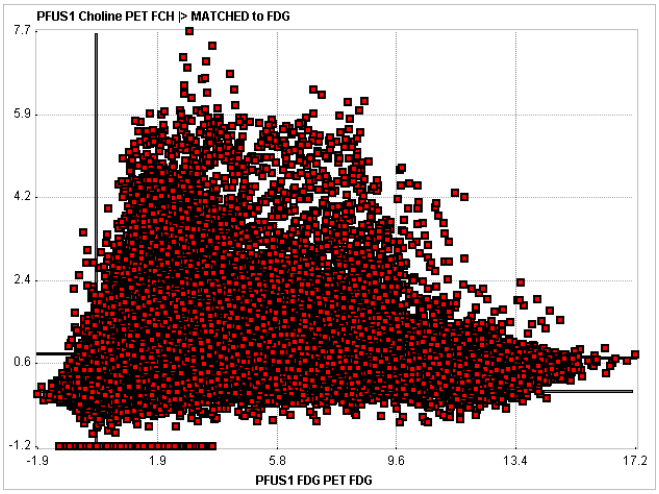
SCATTER IN VOI

Purpose

Scatter plots are a convenient method for exploring tissue function, if matched image series are available. For instance, with the brain PET example illustrated below,



the following scatter plot of the pixel values in the blue VOI box is obtained.



The FDG uptake of a pixel is plotted on the x-axis, and the corresponding FCH uptake on the y-axis. The purpose of the **SCATTER IN VOI** tool is to provide a means of mapping the position of points in the scatter plot back to the image. This is done by generating a segment image of the scatter plot points.

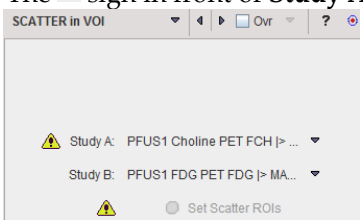
Segmentation Procedure

The following steps result in a segment image of a subset of points in the scatter plot:

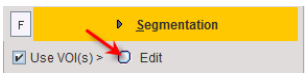
1) Load the two matched image series.

2) Select the **SCATTER in VOI** segmentation method.

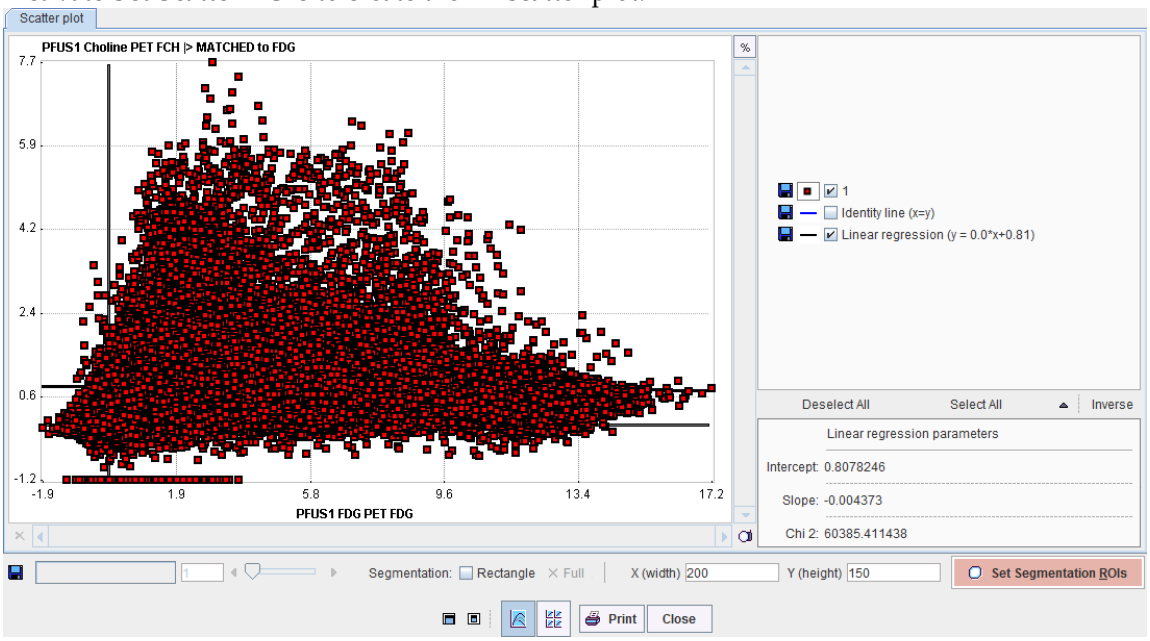
3) The ⚠ sign in front of **Study A** indicates that no VOI has been defined yet.



4) Define one or multiple VOIs on the first image series which enclose the tissue of interest, if needed the whole volume. To this end enable **Use VOI(s)** and then activate the **Edit** button.

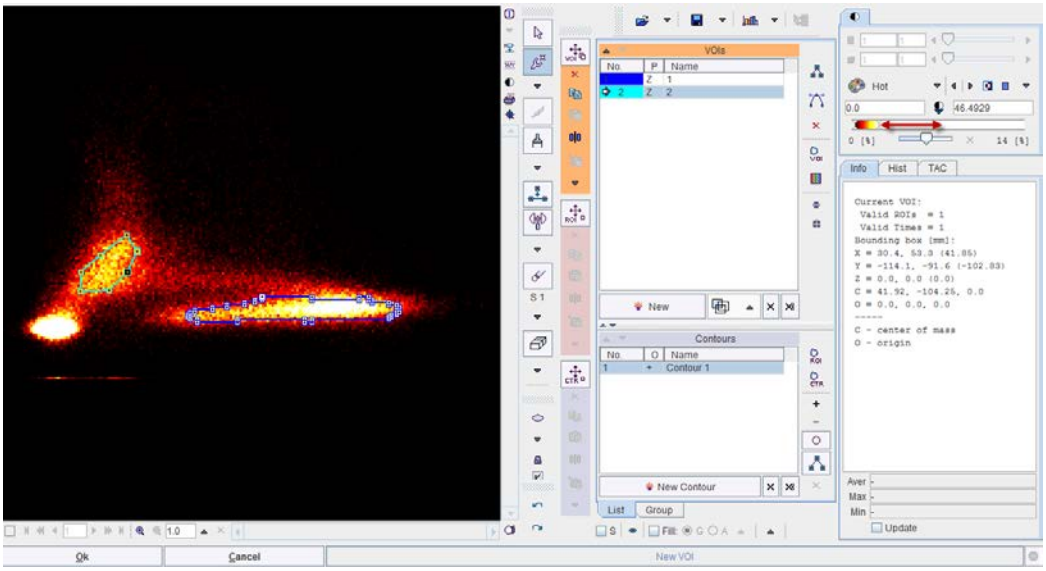


5) Activate **Set Scatter ROIs** to create the 2D scatter plot.

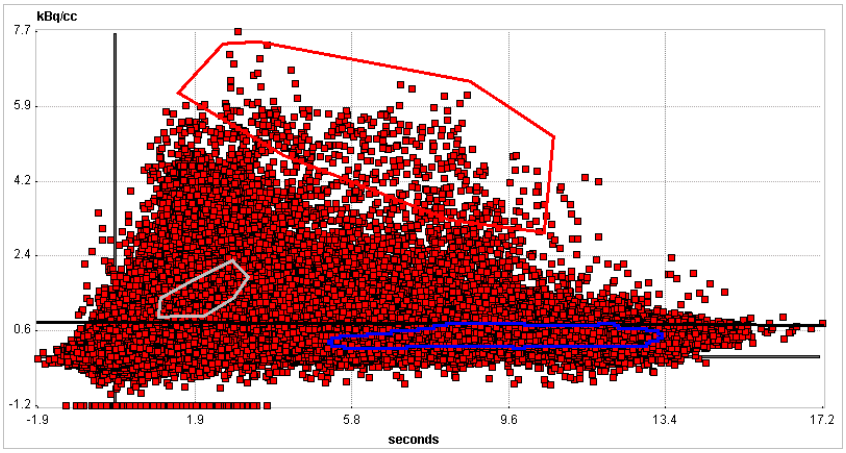


6) Convert the plot into an image by **Set Segmentation ROIs**. The **X(width)** and **Y(height)** define the number of image pixels in the two dimensions. The image value is given by the number of scatter points in the area of each image pixel.

7) A window appears showing the generated image together with the VOI definition interface. Adjust the color thresholds for localizing pixels with a specific uptake pattern and draw the ROIs.

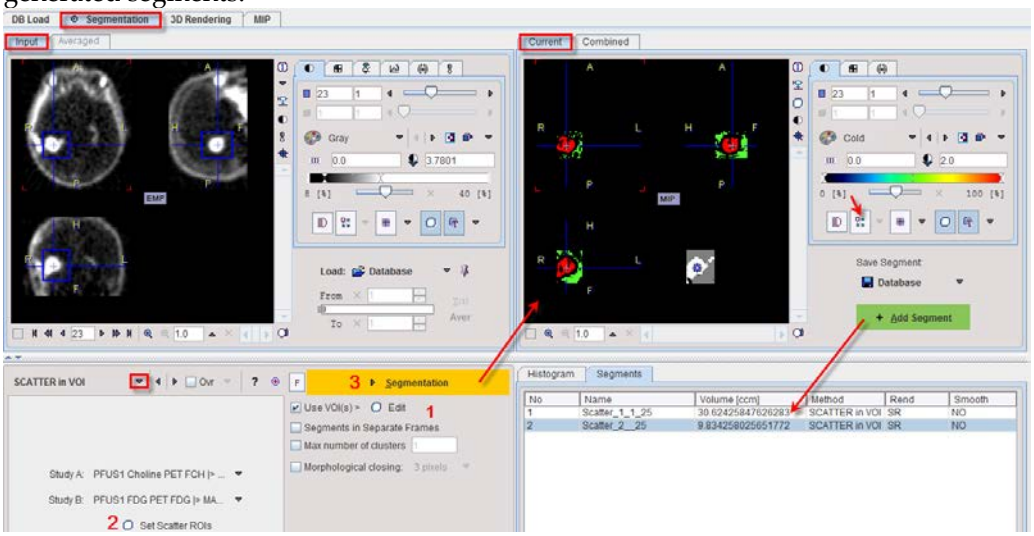


8) Closing of the VOI interface with **Ok** returns to the scatter plot which now also displays the ROIs.



9) Complete the definition by the **Close** button to return to the **Segmentation** tool.

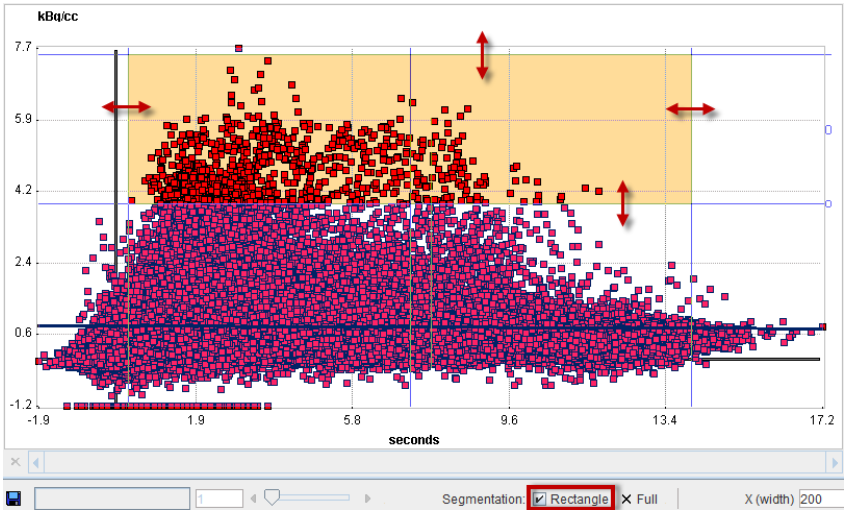
10) Now the scatter pixels within the ROIs in the scatter plot can be mapped with the **Segmentation** button. Note that the interpolation should be disabled to better see the generated segments.



11) Finally the segments can be saved as images to be fused with the original data, or rendered in 3D.

Rectangle ROI

A quick alternative to going through scatter plot rasterization and ROI outlining is to use a simple rectangular ROI. This functionality is enabled by the **Rectangle** box. A yellow-shaded area appears, which can be adjusted by dragging the edges. **Segmentation** will map the pixels in the defined rectangle.



ACTIVE MODELS

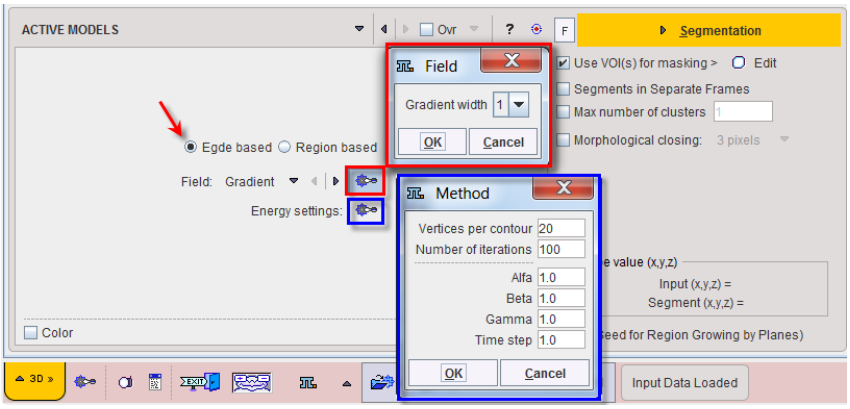
This segmentation method implements two main approaches for the ***active contours*** ***http://en.wikipedia.org/wiki/Active_contour_model*** models (or Snakes) methodology based on energy calculation:

1) **Edge based** models**:** utilize the image gradient to guide the evolving curve toward object boundary.
2) **Region based** models: the energy is treated as an inequality between regions inside and outside the model, therefore the model tends to move onto areas of similar feature (e.g. homogeneity).

Both methods request an initial VOI, which is modified iteratively to find the least energetic state.

1. Edge Based

This segmentation implements several variants of edge based active contours methodology:



There are two relevant parameters, **Field** and **Energy settings**.

The **Field** selection determines how the energy is calculated and has the choices:

- **Gradient**: calculates a simple image gradient using pixel values differences;
- **Sobel**: calculates the image gradient using the Sobel digital filter;
- **GVF**: gradient vector flow algorithm, which calculates image energy iteratively;
- **GGVF**: generalized gradient vector flow algorithm, with better convergence than GVF.

Each of the **Field** selections has its own parameters. For GVF and GGVF:

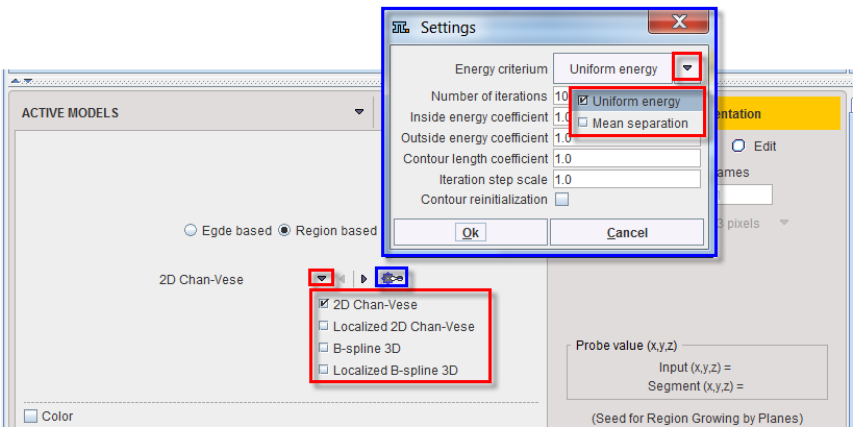
- **Iterations**: Number of iterations.
- **mu**: Smoothness of the energy field.
- **time step**: Speed of the algorithm convergence. Values bigger than one can cause algorithm instability.

The **Energy Settings** are

- **Vertices per contour**: Number of contour nodes.
- **Number of iterations**: Number of contour modifications before the algorithm stops.
- **Alfa**: Hooke's constant of contour tension.
- **Beta**: Contour rigidity/stiffness against bending.
- **Gamma**: Impact of the external force field.

2. Region Based

The region based models make use of statistical information of region instead of using image gradient, offering better performance in case of noise and weak boundaries or discountinuous boundaries.



The **2D Chan-Vese** model has been successful in handling images with homogeneous regions. It is a 2D algorithm. In case of MR, PET or CT images affected by shading artefact this method is not effective for segmenting objects with intensity inhomogeneity.

The **Localized** version of **2D Chan-Vese** algorithm takes into consideration just the neighbourhood of the analyzed node instead of a global image space. The algorithm is slower (energy is calculated NxM times instead of just once) but much more accurate.

In both **Chan-Vese** implementations the **Energy criterium** determines how the energy is calculated and has two choices:

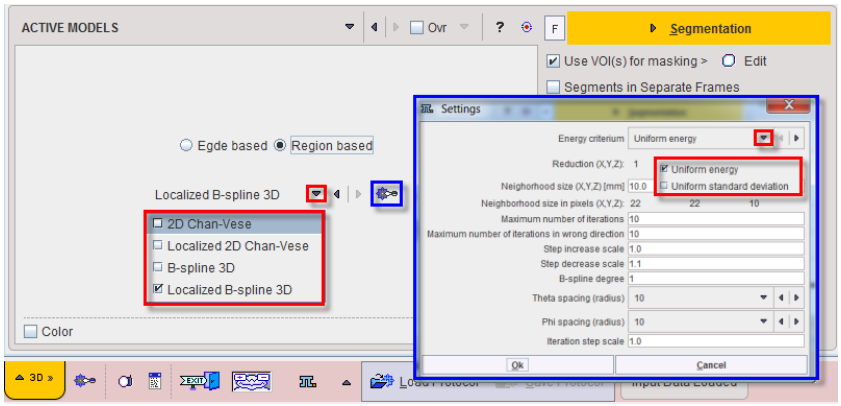
- **Uniform energy:** the energy models the foreground and background as constant intensities represented by their means. In particular, the modeling flow finds its minimum energy when the interior and exterior are best approximated by means. In the localized version, the minimum is obtained when each point on the curve has moved such that the local interior and exterior about every point along the curve is best approximated by local means.
- **Mean separation:** this energy relies on the assumption that foreground and background regions should have maximally separate mean intensities. Optimizing the energy causes the curve to move such that interior and exterior means have the largest difference possible. The optimum of this energy is obtained when local means are the most different at every point along the contour. In some cases, this is more desirable than attempting to fit a constant model. In the current implementation the local foreground and background means are encouraged to be different rather than constant. This allows

this energy to find image edges very well without being distracted when interior or exterior regions are not uniform.

For both type of energy criterium the below parameters can be set:

- **Number of iterations**- defines the maximum number of iteration at which the algorithm is stopped. It is an iterative algorithm: for each iteration the energy is calculated and compared to the one from the previous step.
- There are three energy components: energy outside the contour, energy inside the contour and energy resulting from the contour length. The **Inside energy coefficient**, **Outside energy coefficient** and **Contour length coefficient** define the impact of each energy type on the overall energy. The coefficient with the highest value has a greater impact in the energy calculation. Please note that only the ratio between the coefficients is taken into consideration, i.e. 1.0, 1.0, 1.0 returns the same result as 2.0, 2.0, 2.0.
- **Iteration step scale** defines the factor by which the calculated movement of the contour in each iteration is multiplied. For example, based on the energy, the contour should be moved to the left side by 2 millimeters. If the parameter is set to a value of 2.0, the contour will be moved to the left side by four millimeters.
- **Contour reinitialization** - if enabled, the contour is reinitialized in each iteration (i.e. its state after previous iteration is described using zero level set once again).

In case of the **B-spline 3D** model the surface/contour is only sampled at some points and approximated between them using b-splines. The energy formulations are similar to Chan-Vese basics. This framework allows real-time 3-D segmentation since it reduces the dimensionality of the segmentation problem.



The **Energy criterium** - defines the way of energy calculation. Two options are available:

- **Uniform energy:** prefers the state when the areas with similar pixel values are grouped at different size of the contour/surface.
- **Uniform standard deviation**: prefers the state when the standard deviation of pixels values grouped at the different sides of the surface is equal.

The following parameters are available for setting:

- **Reduction (X,Y,Z)**: allows creating simplified surface based on data simplification. The defaults value is 1 and is representing the smallest value to be defined. Initially, the program will use this number and multiply the pixel size of the data with this value. Finally, the surface will be generated on the new data. It is a faster procedure and the results will consist in surface being less precise.
- **Neighborhood size (X, Y, Z)**: allows defining the size of the neighborhood in each direction in mm. The values are used to automatically calculate the **Neighborhood size in pixels (X, Y, Z)**
- **Maximum number of iterations:** defines the maximum number of iterations at which the algorithm is stopped.
- **Maximum number of iterations in wrong direction** - defines the maximum number of interations allowed in case there is a number of subsequent iteration with energy increase. In such situation the algorithm is stopped and the current state is treated as a local minimum. The "wrong direction" is identified when after contour modification the overall energy is higher than previously.
- **Step increase scale** - factor by which iteration step scale is multiplied after each successful iteration.
- **Step decrease scale** - factor by which iteration step scale is divided after each "wrong direction" iteration
- **B-spline degree** - degree of a b-spline function approximating the surface.
- **Theta/Phi spacing (radius)** - a radial space between two neighbouring nodes of a surface (in degrees).
- **Iteration step scale** - defines the factor by which the calculated movement of the contour in each iteration is multiplied. For example, based on the energy, the contour should be moved to the left side by 2 millimeters. If the parameter is set to a value of 2.0, the contour will be moved to the left side by four millimeters.

Reference:

[1]. Mark A. Heidekker, Advanced Biomedical Image Processing. John Wiley & Sons, 29 mar 2011 - 528, Chapter 6: Active models.

[2]. Tony F. Chan, Luminita A. Vese, Active Contours Without Edges, IEEE Trans. Image Process. Feb 2001; 10(2):266-277.

[3]. Chenyang Xu, Jerry L. Prince. Snakes, Shapes, and Gradient Vector Flow. IEEE Trans. Image Process. March 1998; 7(3):359-369

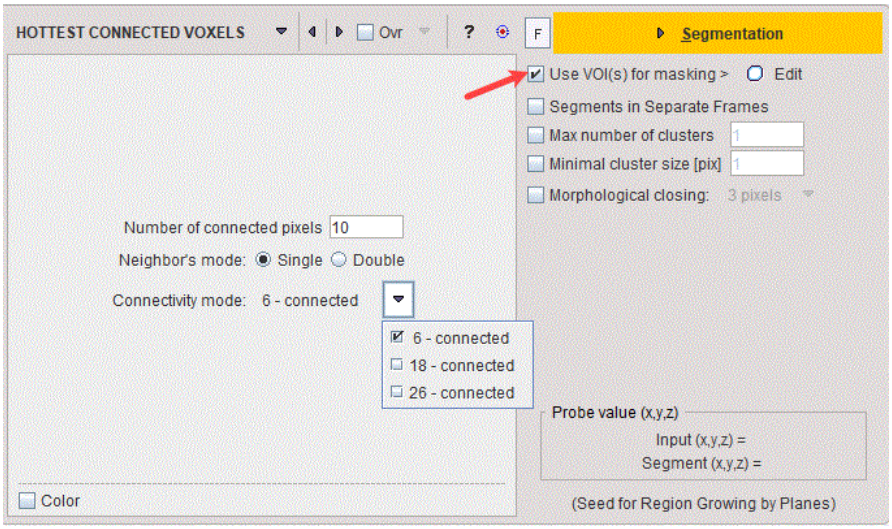
[4]. Barbossa et al, B-Spline Explicit Active Surfaces: An Efficient Framework for Real-Time 3-D Region-Based Segmentation. IEEE Trans. Image Process. Jan 2012; 21(1):241-251,

[5]. Shawn Lankton, Allen Tannenbaum. Localizing Region-Based Active Contours, IEEE Trans Image Process. Nov 2008; 17(11): 2029–2039. doi:10.1109/TIP.2008.2004611

Hottest Connected Pixels

The **Hottest Connected Pixels (HCP)** is a segmentation method used to objectively outlined brain anatomical structure like the locus coeruleus/subcoeruleus complex in T1 weighted MRI images [2]. Another potential application is the extraction of the image derive input function.

It is recommended to use a VOI as boundary restrictions for the algorithm as illustrated below:



The HCP method aims at identifying a well-defined number of connected pixels whose average value is maximal. The pixel connectivity is representing the way in which pixels in 2-dimensional (or voxels in 3-dimensional) images relate to their neighbors [1].

The actual **Number of connected pixels** can be specified in the interface.

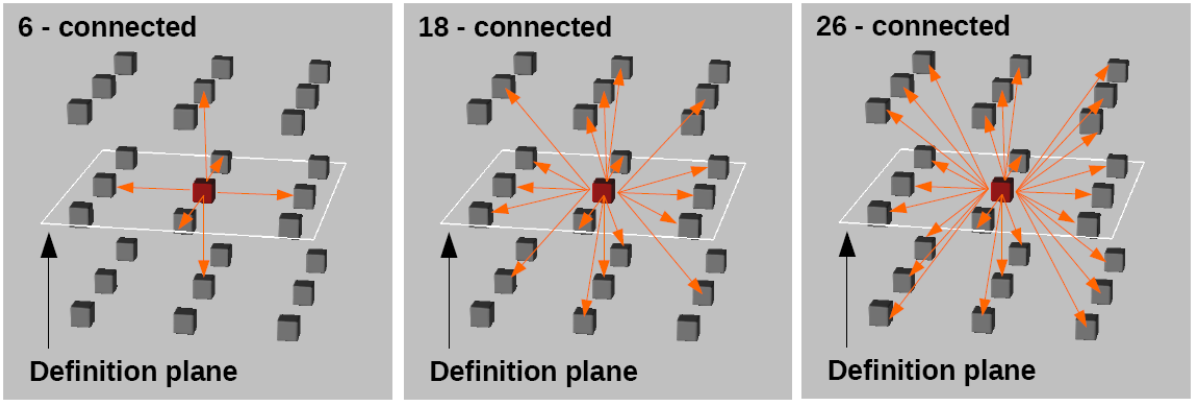
Initially, the average value of the activity in the whole VOI is calculated. The searching process of the neighbors, using either the **Single** or the **Double** mode, is repeated for each pixel with value greater than the average value in the parent VOI. The selected **Connectivity mode** is used to find the neighbors of the current pixel in the searching process.

With the **Single** mode enabled and based on the selected connectivity model, the algorithm is searching around the current voxel the voxel with the highest value to add it as part of the segment. Next, the voxel with the highest value is searched around the new segment. The operation is repeated until the pre-defined number of connected pixels is detected.

With the **Double** mode enabled and based on the selected connectivity model, the algorithm is searching around the current voxel the voxel which has the highest summed value with his next neighbor. When the highest summed value is identified the algorithm is adding to the segment the voxel and its next neighbor in one step. If the tested voxel has no neighbor, his value is considered by the algorithm. Note that once a voxel is added to the segment, its value is not considered anymore as the next neighbor. The process is repeated until the pre-defined number of connected pixels is detected.

The segment which average value is maximal will generate the 3D binary segment.

The tool implements three models of connectivity as illustrated below:



The 6-connected pixels are neighbors to every pixel that touches one of their faces (cross connection). These pixels are connected along one of the primary axes [1].

The 18-connected pixels are neighbors to every pixel that touches one of their faces (cross connection) or edges (oblique connection). These pixels are connected along either one or two of the primary axes[1].

The 26-connected pixels are neighbors to every pixel that touches one of their faces (cross connection), edges (oblique connection), or corners. These pixels are connected along either one, two, or all three of the primary axes [1].

References

[1] https://en.wikipedia.org/wiki/Pixel_connectivity

[2] Lorenzo DG, Longo-Dos Santos C., Ewenczyk C., Leu-Semenescu S., Gallea C., Quattrocchi G.,Pita Lobo P., Poupon C.,Benali H., Arnulf I., Vidailhet M., Lehericy S.: The coeruleus/subcoeruleus complex in rapid eye movement sleep behaviour disorders in Parkinson's disease. Brain 2013, 136:2120–2129. ***DOI http://www.***.

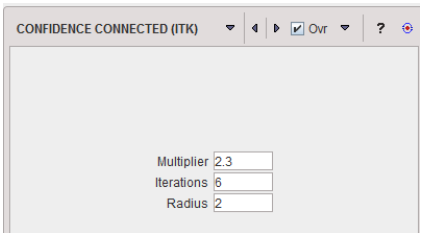
CONFIDENCE CONNECTED (ITK)

From the ***The ITK Software Guide*** (***http://www.itk.org/ItkSoftwareGuide.pdf***): "The criterion used by the **Confidence Connected** method is based on simple statistics of the current region. First, the algorithm computes the mean and standard deviation of intensity values for all the pixels currently included in the region. A user-provided factor (**Multiplier**)is used to multiply the standard deviation and define a range around the mean. Neighbor pixels whose intensity values fall inside the range are accepted and included in the region. When no more neighbor pixels are found that satisfy the criterion, the algorithm is considered to have finished its first iteration. At that point, the mean and standard deviation of the intensity levels are recomputed using all the pixels currently included in the region. This mean and standard deviation defines a new intensity range that is used to visit current region neighbors and evaluate whether their intensity falls inside the range. This iterative

process is repeated until no more pixels are added or the maximum number of iterations is reached.

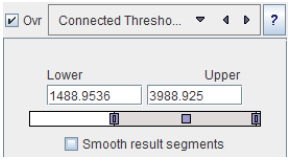
The number of **Iterations** is specified based on the homogeneity of the intensities of the anatomical structure to be segmented. Highly homogeneous regions may only require a couple of iterations. Regions with ramp effects, like MRI images with inhomogeneous fields, may require more iterations. In practice, it seems to be more important to carefully select the multiplier factor than the number of iterations. However, keep in mind that there is no reason to assume that this algorithm should converge to a stable region. It is possible that by letting the algorithm run for more iterations the region will end up engulfing the entire image."

The initialization of the algorithm requires the user to provide a **seed point**. It is convenient to select this point to be placed in a typical region of the anatomical structure to be segmented. A small initial neighborhood **Radius** around the seed point will be used to compute the initial mean and standard deviation for the inclusion criterion.

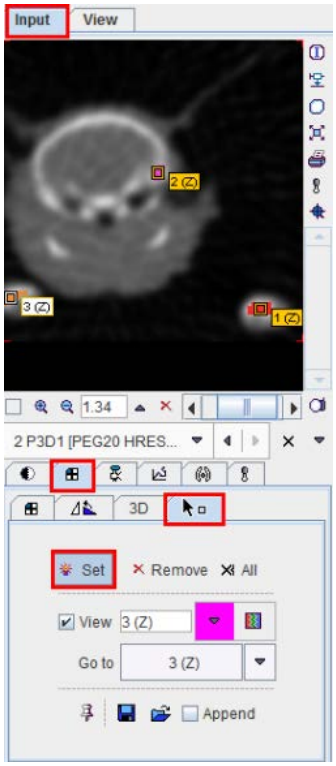


CONNECTED THRESHOLD (ITK)

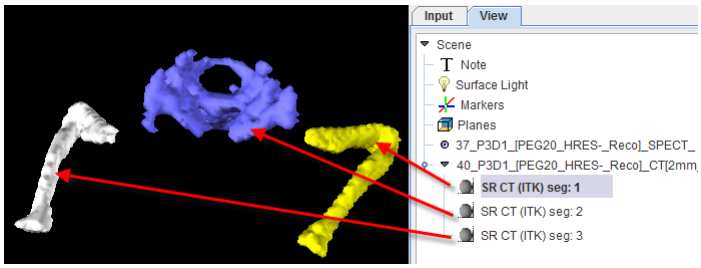
Connected Threshold is a region growing method with the criterion that (similar to the **IN RANGE** method) the included pixels must have values between a **Lower** and **Upper** threshold.



As usual with region growing methods, the user has to specify a seed point by clicking into the image. The ITK region growing methods are particular in that multiple seed points can be specified at once by the use of markers as illustrated below.



Once the markers tab has been activated and the **Set** button enabled, a marker is created as a seed point for each clicking into the image. When the segmentation is started, the region growing with the specified value range is performed from each marker.



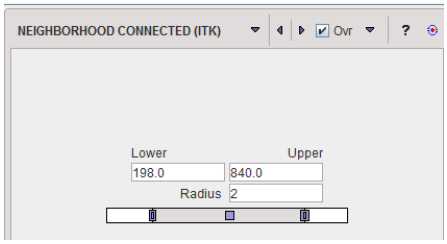
NEIGHBORHOOD CONNECTED (ITK)

NEIGHBORHOOD CONNECTED is a region growing method which applies two grouping criteria:

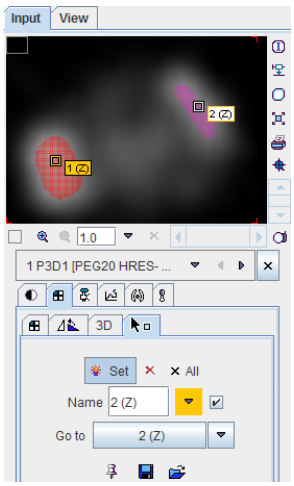
1) An included pixel must have a value in the range between the **Lower** and **Upper** threshold.

2) All pixels in its neighborhood within the specified pixel **Radius** must also be within the value range.

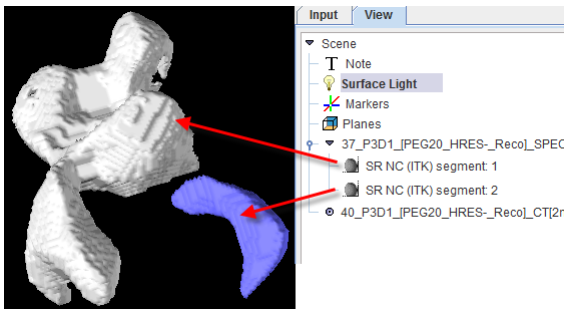
By the combination of these criteria small structures are less likely to be accepted in the region.



A single seed is provided by the triangulation point of the orthogonal planes. Multiple seed points can be specified by the use of markers.



The number of structures found depends on the number of seeds.

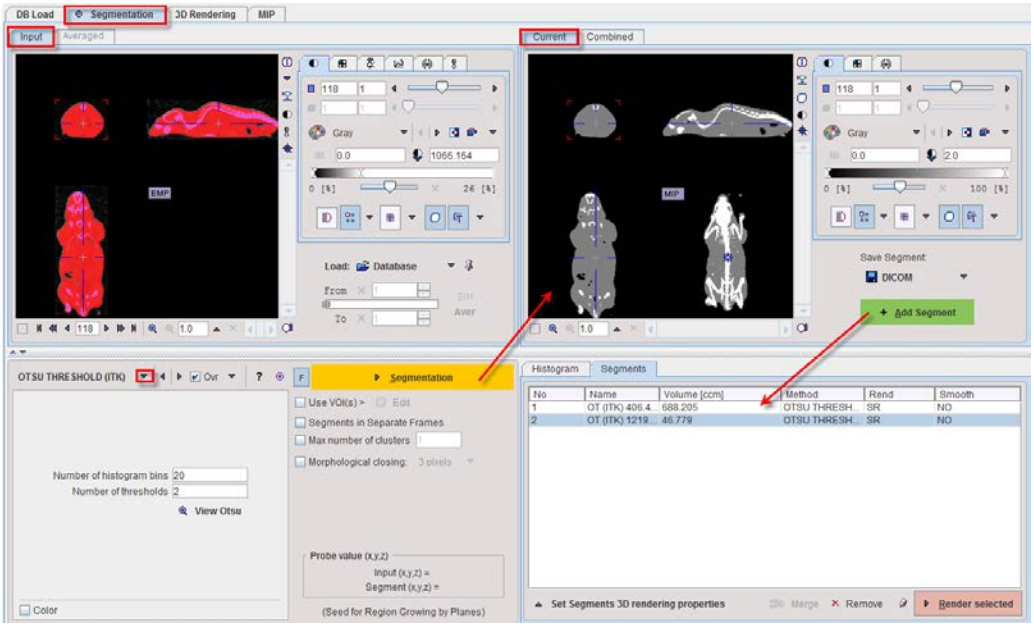


OTSU THRESHOLD (ITK)

The OTSU THRESHOLD method is based on the analysis of the pixel histogram and is described by ***The ITK Software Guide*** (***http://www.itk.org/ItkSoftwareGuide.pdf***) as follows: "Another criterion for classifying pixels is to minimize the error of misclassification. The goal is to find a threshold that classifies the image into two clusters such that we minimize the area under the histogram for one cluster that lies on the other cluster's side of the threshold. This is equivalent to minimizing the within class variance or equivalently maximizing the between class variance."

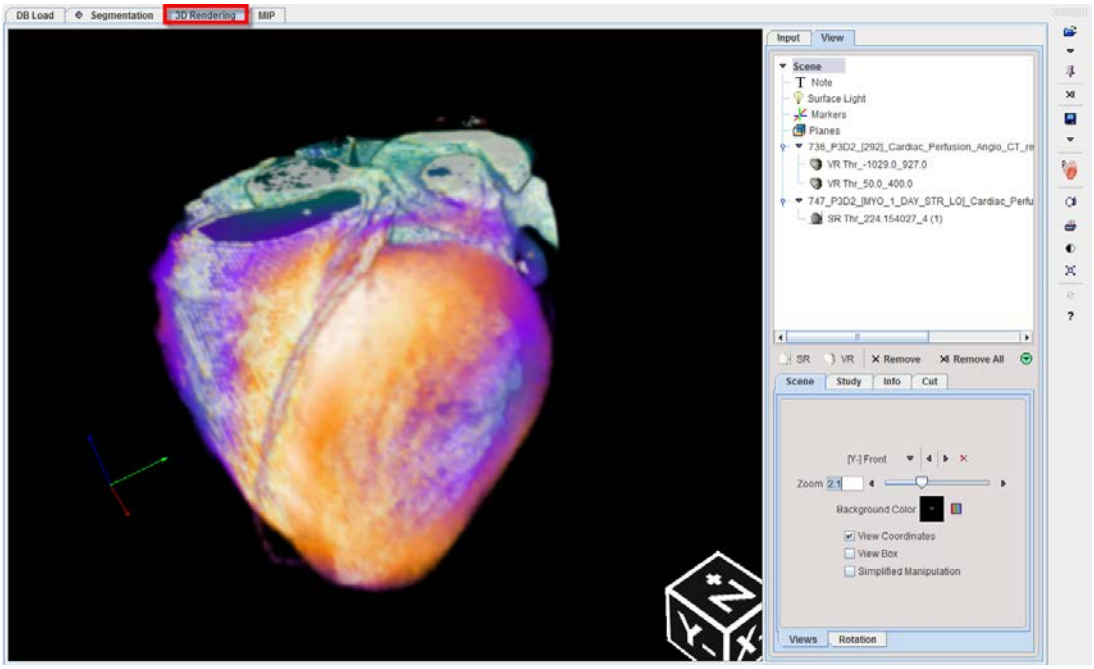
The **Number of histogram bins** defines the resolution of the histogram shape, whereas the **Number of thresholds** defines the number of distinct clusters. With the mouse CT, Otsu

with the settings below finds appropriate thresholds for the skeleton object and the soft tissue, as illustrated below. Note that the threshold values can be inspected only after the actual segmentation by the **View Otsu** button.



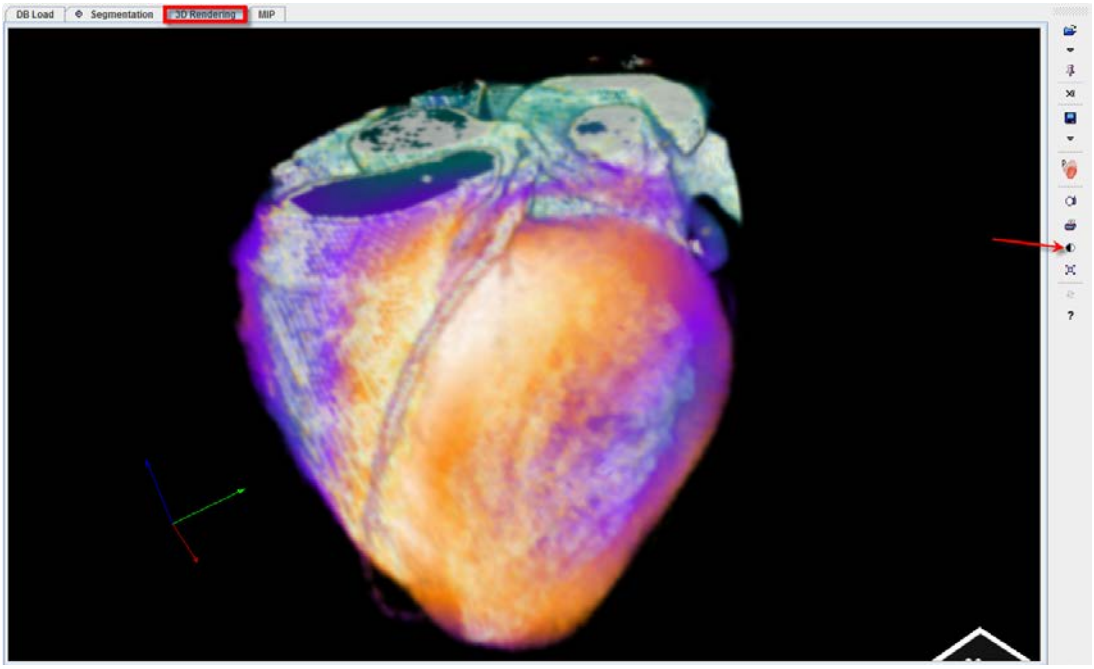
3D Rendering Page

The **3D Rendering** page has the layout illustrated below.



The 3D scene area to the left shows the current scene, whereas the properties of the visualized objects can be edited on the **View** panel to the right. The **Input** panel is mostly used to adjust the coloring of plane images and textures, although it also offers a small

segmentation interface. Note the toggle button in the taskbar which allows maximizing the visualization area.

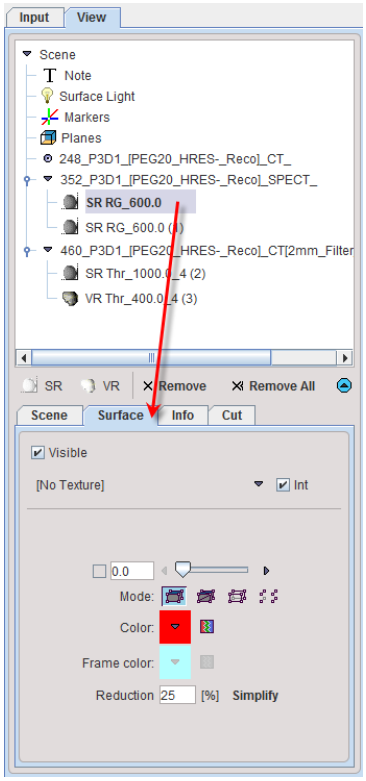


The following sections describe how the visualization properties of the SR and VR objects are changed, and how additional objects like image planes and object markers can be added.

Object Management and Adjustments of Properties

Object Tree

All segmentation operations result in virtual reality objects which are arranged as an object tree in the **View** tab as illustrated in the example below.



Entries with **SR** represent surface renderings, **VR** volume renderings, the **Surface Light** is the optional lightning source for SR objects, the **Markers** are synthetic objects usable for indicating points of interest, the **Planes** are sets of image slices, and **Note** is a text shown in a corner of the 3D scene. If an object is not needed any more, it can be removed from the tree by the **Remove** button.

The SR VR buttons are used to create *ghost* objects, which are empty SR or VR objects with defined visualization attributes. These attributes will be used for rendering the objects resulting from the next segmentation. The advantage of using ghost objects is avoiding lengthy rendering operations with inadequate default settings.

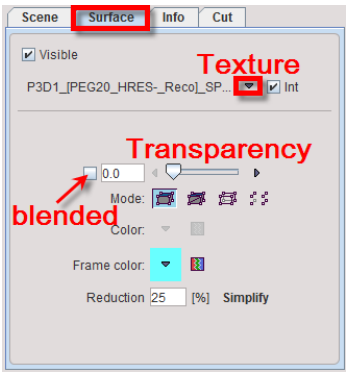
To delete a 3D object select the corresponding object in the tree and then activate **Remove**. The button collapses the tree, whereas expands it fully.

Object Properties

Every object type has a set of visualization attributes which are shown in the lower panel as soon as it is selected in the tree. Its tab name corresponds to the object type, **Surface**, **Volume**, **Planes** etc. The visualization attributes are described in the next sections. Note that if a sub-tree or several objects in the tree are selected, the common properties may be manipulated for all contained objects at once.

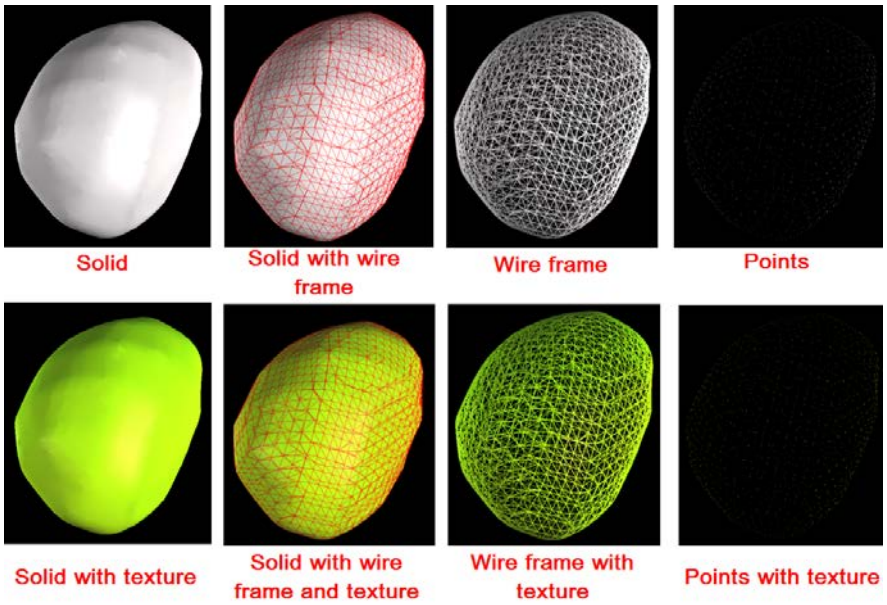
Viewing Options for Surface Rendering (SR) Objects

A **SR** object only consists of a surface. Its properties define how the surface is represented in the scene.



The **Visible** box determines whether or not the object is shown. The texture selection allows to switch off texturing (**No Texture**), or to select one of the loaded image series for coloring the surface. **Int** enables interpolation of the texture information. If no texture is active, the surface color can be defined using the **Color** selection.

There are four **Modes** how to render a surface: **solid**, **solid with wire frames** on it, as a **wire** frame, and by **points**. The wire frame and the points modes have the advantage that inner objects are visible. The examples below illustrate the different modes without and with (lower row) texturing.



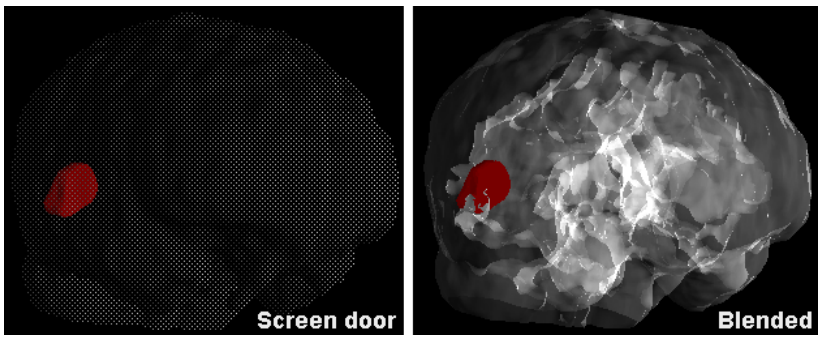
Transparency

Transparency is another option to render SR objects non-obstructing. There are two ways how transparency is implemented:

1) **Screen door** transparency (box not checked): This implementation punches holes into the surface. The higher the transparency, the bigger the holes.

2) **Blended** transparency (box checked): This implementation provides a smoother view, but *is only available as long as no VR objects have been generated.*

The check box next to the transparency slider allows switching the two modes. The effect is illustrated below.

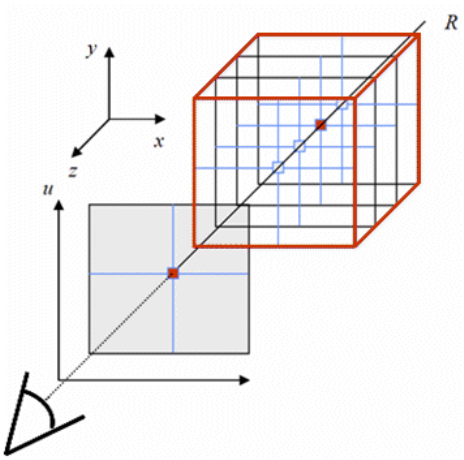


Note: Property changes of SR objects are immediately reflected in the scene.

Viewing Options for Volume Rendering (VR) Objects

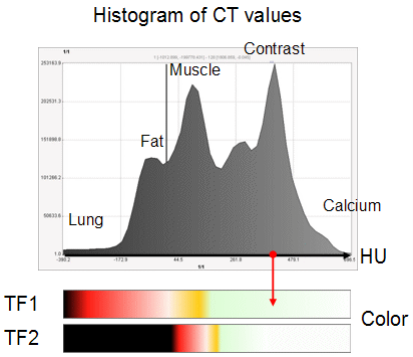
VR Principle

VR objects are computed from all the data, not just from a derived surface. The method is based on ray tracing from a certain viewpoint. Rays are cast through the image volume, and the *ray values* recorded when they pass a plane behind the object, thereby producing an image.

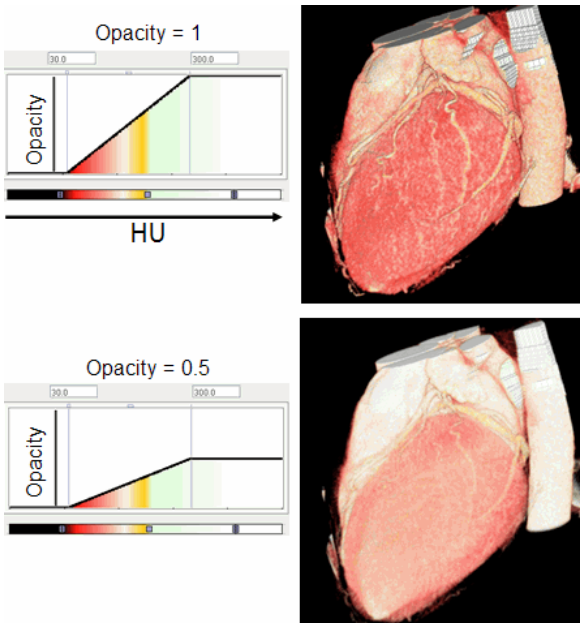


Each voxel is regarded as a "particle" that emits a certain color and absorbs a fraction of the passing light from the other voxels along the same ray.

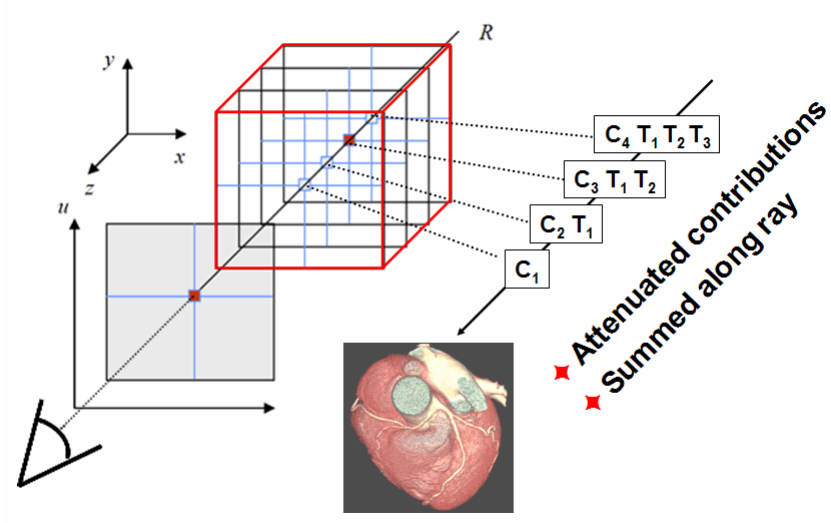
The *color transfer function* (TF) describes the voxel color as a function of the voxel value. For CT data, for example, the voxel values range from -1000 to 3000, and color mappings as illustrated below are often used.



The *opacity function* describes the voxel opacity for the ray as a function of the voxel value. Opacity values range from 0 to 1, whereby 0 means full transparency, and 1 full opacity. Often, ramp functions are applied as illustrated below. If lower opacity values are used as in the lower example, information from the inner of the object may become apparent such as the contrast agent in the ventricular cavity.

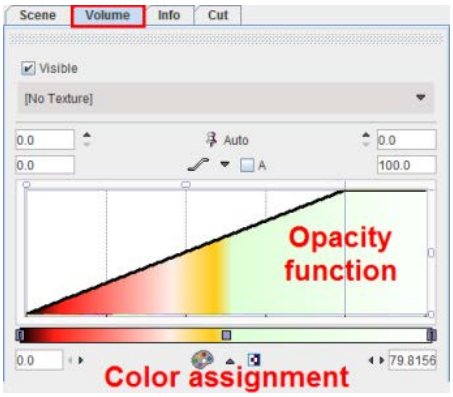


The ray value (the VR image pixel color) is calculated by summing up the contributions along a ray as illustrated below. C_i denotes the color of a voxel$_i$, and T_i the transparency (1-opacity) of a voxel. Contributions of voxels far from the surface will be attenuated by multiple intervening voxels and thus be faint, depending on the opacity function.



VR Implementation in P3D

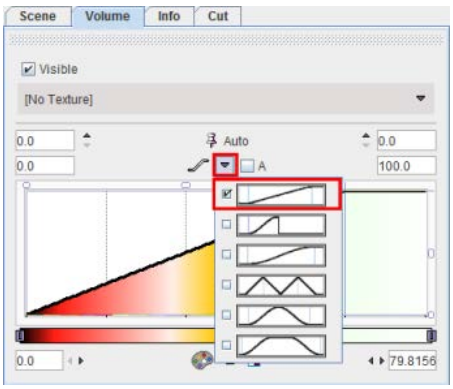
The properties tab **Volume** of VR objects shows several elements.



The **Visible** box serves for hiding the VR object altogether. The **Texture** selection allows combining volume rendering with color texturing from a matched study. The lower part of the tab contains the elements for defining the color and opacity transfer functions. The **Auto** refresh button serves for enabling/disabling automatic updating the display after any settings has been changed.

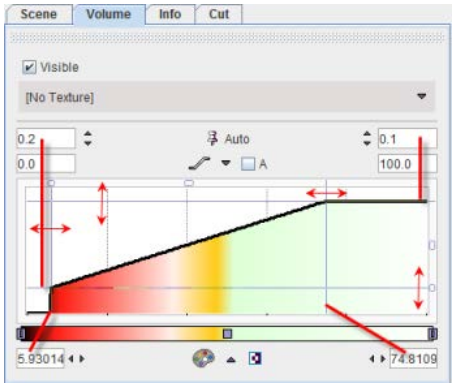
Opacity Functions

There are different shapes of opacity functions which can be selected using the arrow button highlighted below.



The purpose of the different shapes is to implement a variety of weighting functions, so that certain intensity ranges can be emphasized or suppressed. Usually, the ramp function is applied.

Once an opacity function has been selected, it can be manipulated in several ways as illustrated below:



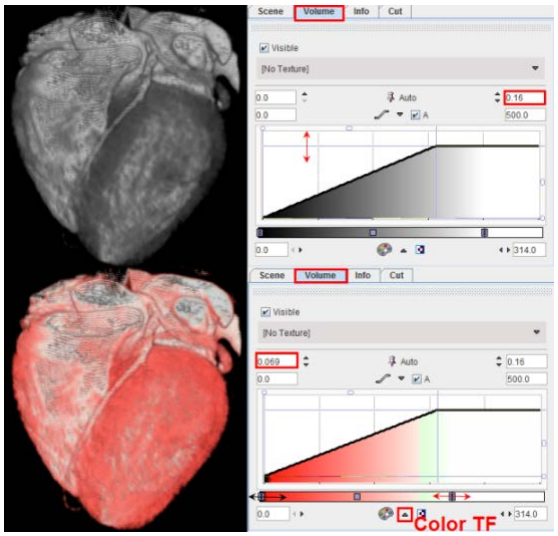
1) The opacity value is in the vertical direction. The minimal and the maximal values of the opacity function can be set numerically or by dragging the horizontal handle lines. Example: 0.2 (= minimal opacity), 0.1 (= transparency at highest opacity)

2) The pixel value is in the horizontal direction. The min/max range of the transfer function can be specified as absolute numbers (if the **A** box is checked), or in relative percent values (if the **A** box is not checked as in the example). This is only for the convenience of transfer function adjustments and has no impact on the rendering. The range can be specified by entering numbers directly (example: 5.9, 74.8), or by dragging the vertical handle lines.

3) Sometimes the transfer function should be defined in a small sub-range of the entire value range. In this case the x-range can be zoomed by changing the limits of the display range. The example above displays the whole range from 0% to 100%.

Color Transfer Functions

There are different color transfer functions available as illustrated in the example below.

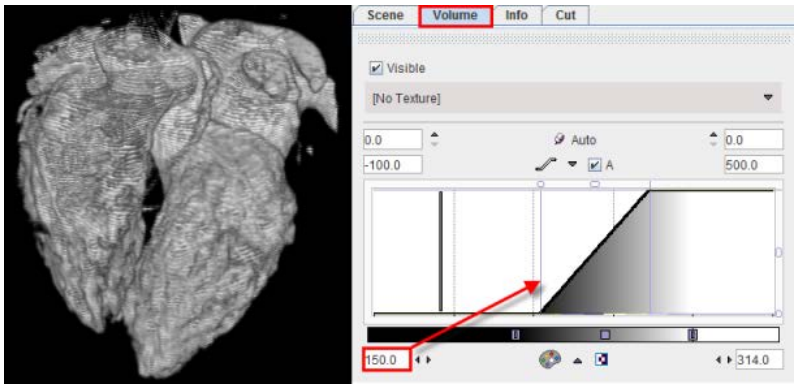


The upper rendering of the angio CT uses a gray color TF, and the lower rendering the **Heart VR** color TF. Note the indicated down arrow button which allows selecting among the available color TF. After a color TF has been chosen, the mapping of voxel values to colors

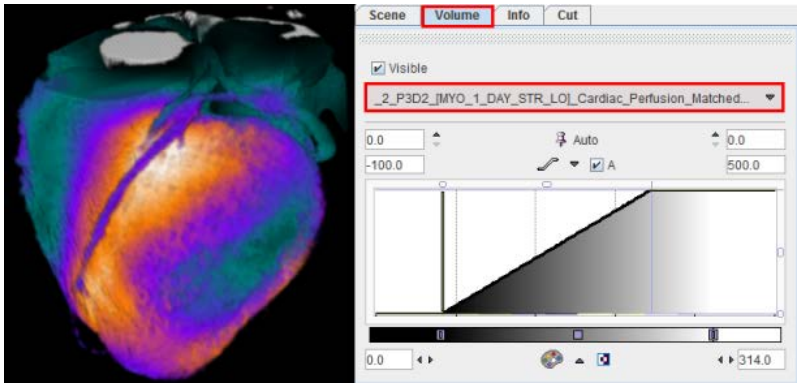
can be adjusted by moving the upper/lower color TF limits as indicated by the horizontal arrows above. The **Heart VR** table makes lower values look fleshy, and higher values white.

Note: With big data sets refreshing of scenes with VR objects may be quite time-consuming. In this case it is recommended to switch the **Auto** refreshing to off. As soon as the VR properties are changed, the alert icon appears in the taskbar to the right. Once all VR properties have been adjusted, the user can trigger scene refresh by clicking at .

Increasing the lower value of the opacity function allows cutting pixels with the density of myocardium, and the result only shows the contrast filled ventricles and the coronary arteries.



Volume rendering can also be combined with textures to introduce functional information into an anatomical rendering. The example below shows such a 3D fusion rendering of an angio CT combined with the texture color from a matched SPECT perfusion scan. The opacity function is linear and with the same range as above.



Notes:
1. The 3D fusion result also depends on the color and window level settings of the texture study which can be changed on the **Input** tab.
2. Voxels in the segment which have values below the lower or above the upper thresholds of the texture color table obtain the color of the selected color TF (Gray scale in the example above).

Viewing Options for Skeleton Rendering (Path) Objects

A skeleton object consists of a VOI (**VOICARD Skeleton**) and path vectors (**Path**). Each of the skeleton components has its own properties and define how the component is represented in the scene.

VOI Skeleton Properties

The **Visible** box determines whether or not the VOI object is shown. The texture selection allows switching off texturing (**No Texture**), or to select one of the loaded image series for coloring the VOI object. **Int** enables interpolation of the texture information. If no texture is active, the VOI color can be defined using the **Color** selection.

The **Modes** and the transparency option available for the VOI rendering are identical to the ones described for the ***surface rendering objects*** (on page 44).

The **Closed** button enables the closure of the VOI shape.

Path Properties

The **Visible** box determines whether or not the path vectors belonging to the **Paths** tree are shown. The **Selected Only** allows visualizing only the selected path in the tree:

When an image plane object is defined it can be tighten to the selected path line in the tree setting the **Bound Plane** selection from **None** to the plane. The selected plane will be placed perpendicular to the line path. The **Position** slider allows moving the selected plane along the active **Path** in the **View** tree.

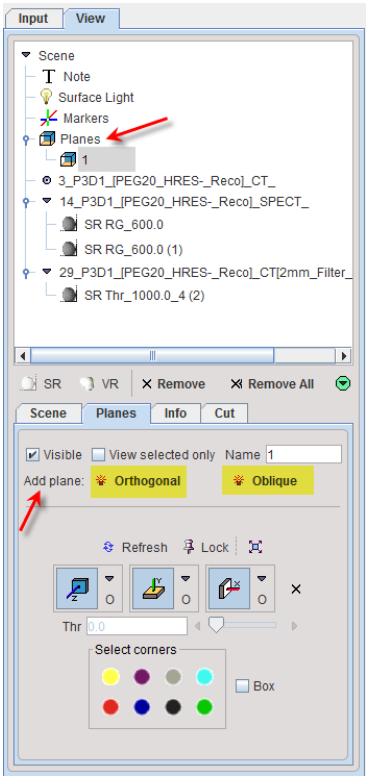
The **Next lines** option lists the closest neighbor paths available for the currently active one.

Note: It is recommended to turn off all the VOIs, SR and VR objects in the scene when selecting an active path for plane binding.

Image Plane Objects

Additional elements for composing a meaningful scene are the slice images from the different image series. Planes also serve for defining the octants to be cut out of VR or SR objects.

Plane objects can be created by selecting **Planes** in the object tree, and then on the appearing **Planes** tab activating one of the **Add planes** buttons: **Orthogonal** or **Oblique.**



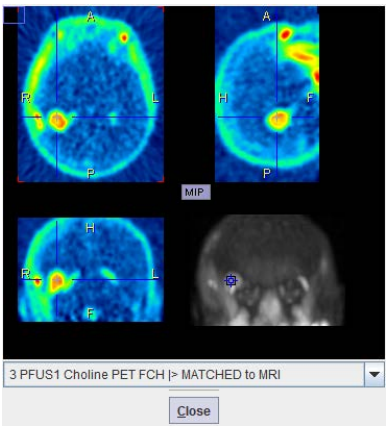
Orthogonal Planes

With the **Orthogonal** button a new planes object appears in the tree. Per default the first plane is named 1. This name can be edited in the **Name** field. Initially, all three orthogonal planes are displayed, but they can individually be switched on/off using the **Z**, **Y** and **X** toggle buttons. The **x** button switches off all three planes. The planes show the slice images of the study selected on the **Input** tab. There the image coloring can be adjusted.

Note: Because there might be several planes object in the tree, a plane object has to be selected when changing the coloring or the planes location.

Navigator Window

The position of the planes can best be changed using the planes navigator, which is started using the button. The appearing window shows three orthogonal slice images and a list selection to choose among the loaded series.



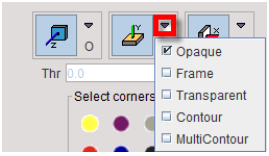
The triangulation point can be changed by simply clicking into the navigator images. The slice locations can be changed by selecting other slices in the **Input** series.

In alternative, the position of the planes can be changed in the **3D** scene by clicking with the left mouse button, holding and dragging the plane of interest.

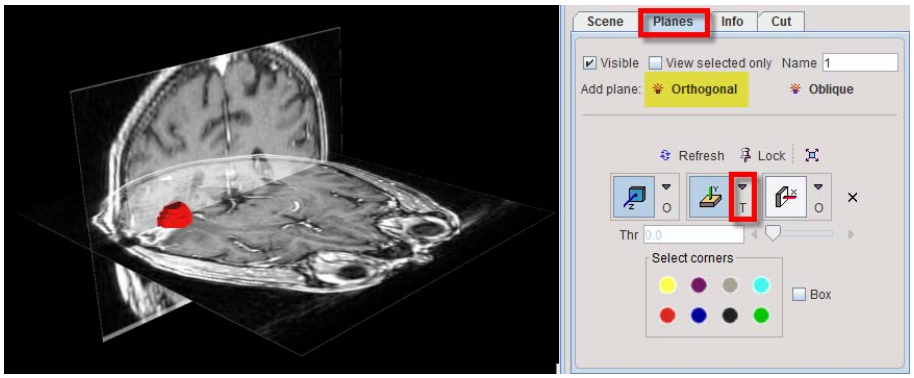
In case the plane locations of the **Planes** object should no more be changed, select the **Lock** pushpin button.

Planes Styles

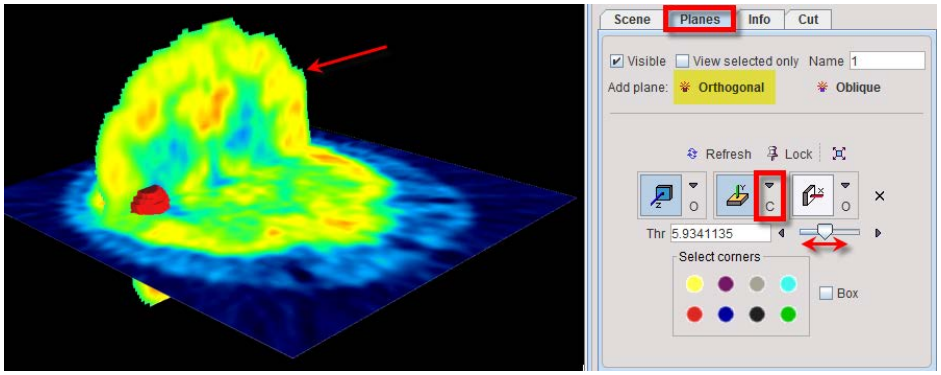
The planes can be rendered in different styles which can be selected from the list to the right of the plane buttons.



The axial slice in the example below is **Opaque**, while the coronal slice is **Transparent**. Transparent slices do not obstruct the view, but this style is *not applicable if a VR object has been rendered*.



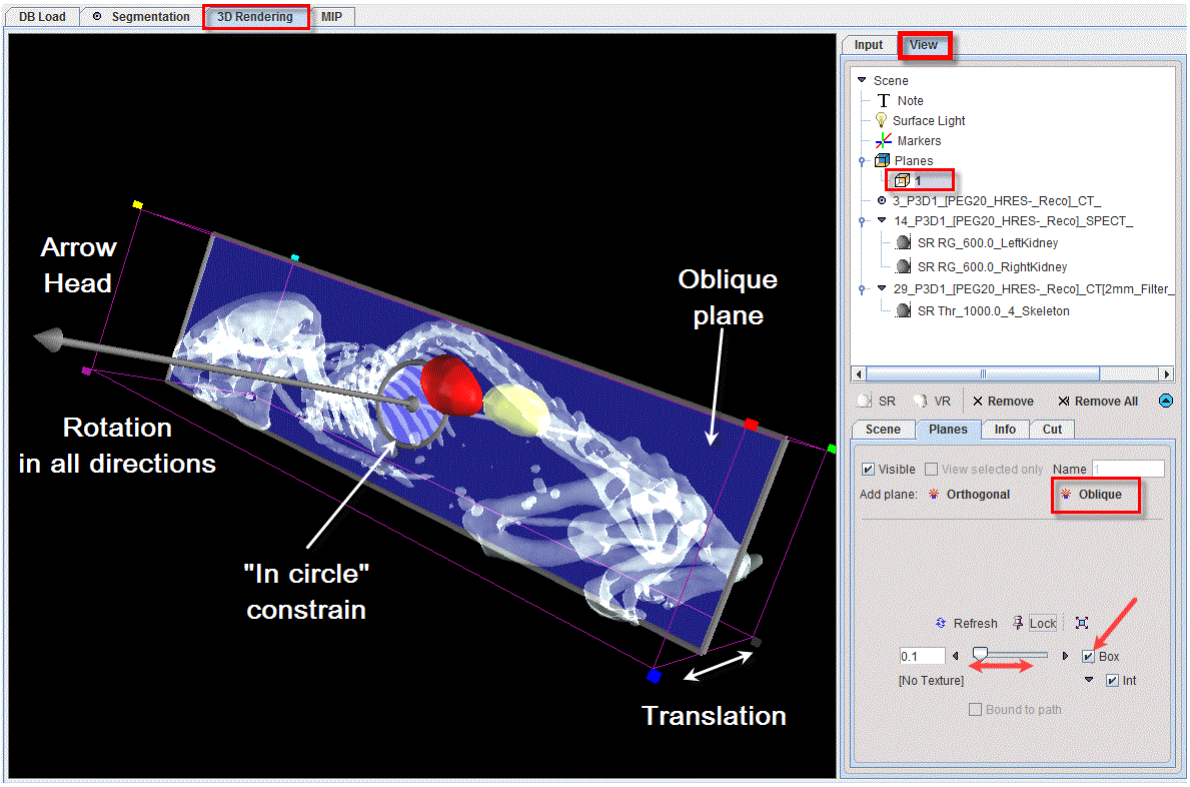
The **Contour** style suppresses the image information below a threshold which can be defined with the **Thr** slider. Only the biggest connected area will be shown. In contrast, **MultiContour**, shows all area above the threshold.



If multiple plane sets have been created**, View selected only** allows showing only the planes belonging to the object currently selected in the tree.

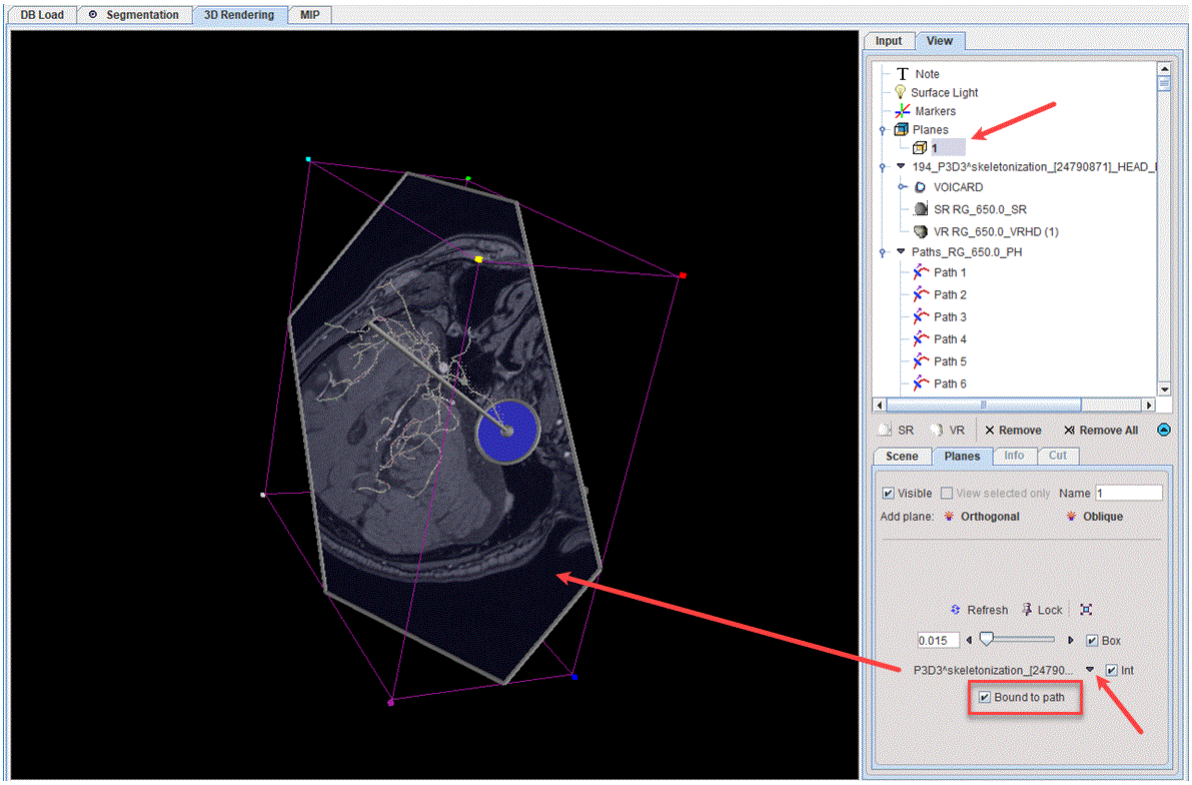
Oblique Planes

With the **Oblique** button a new plane object appears in the tree. This plane allows dividing the object in the scene at any user defined angle. The name can be edited in the **Name** field.



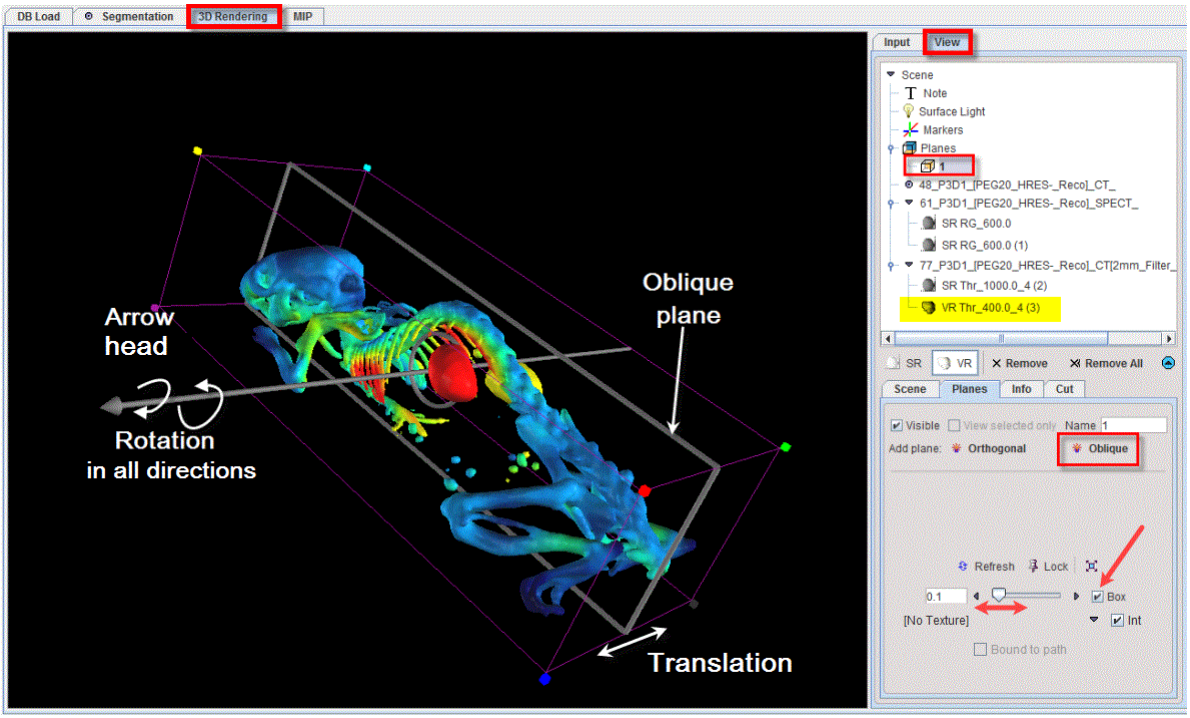
The oblique planes appear in the scene as blue transparent (blended transparency) as long as no VR object is available in the **View** tree.

The texture selection allows to switch off texturing (**No Texture**), or to select one of the loaded image series for coloring the plane surface. **Int** enables interpolation of the texture information. The **Bound to path** option becomes active when paths objects are available in the scene:



Note: When the oblique plane is **Bound** to a selected path, the plane is positioned perpendicular to the path direction.

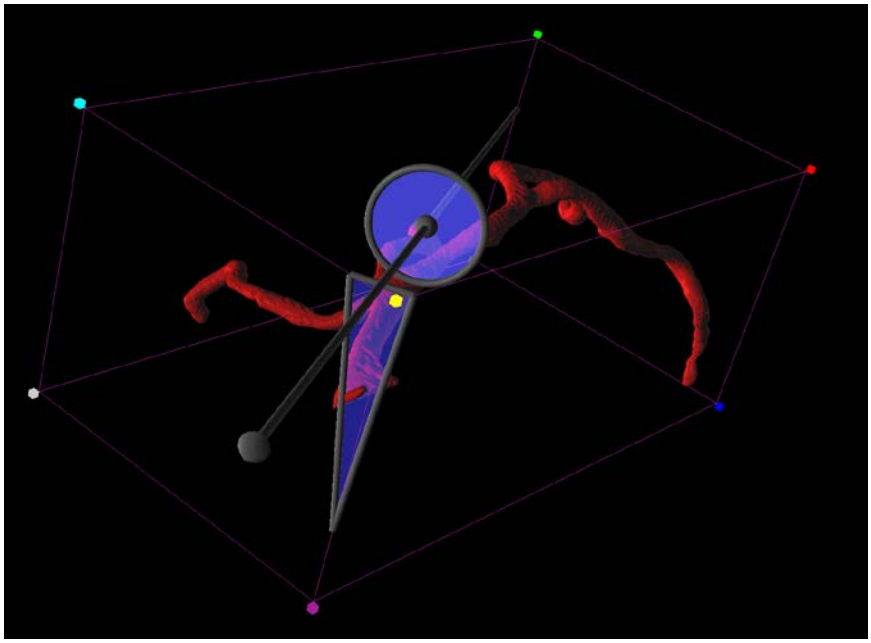
With a VR object in the **View** tree the aspect of the oblique plane is that of an *empty* frame:



It is recommended to enable the **Box** for a better visualization of the plane. By default, the oblique plane is positioned initially in the middle of the box, like an **X** plane.

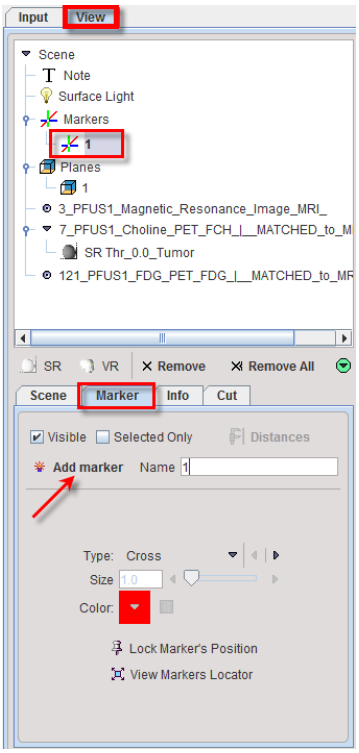
The plane can be moved and rotated in all direction. The arrow axis, initially positioned in the center of the plane, is used for the plane rotation. The origin of the rotation is represented by a circle which center location is represented by the arrow. Circle size can be changed using the **Increase**/**Decrease** black arrows or with the slider. The circle can be used to constrain the cutting within it. The sides of the plane can be holded and dragged in a certain direction for the plane translation.

Please note that the origin of the rotation, the circle with the arrow, can be completely displaced in respect to the plane. This functionality is useful when the position of the cutting plane has to be close to the corner of the bounding box, e.g. on the edge of a vessel:

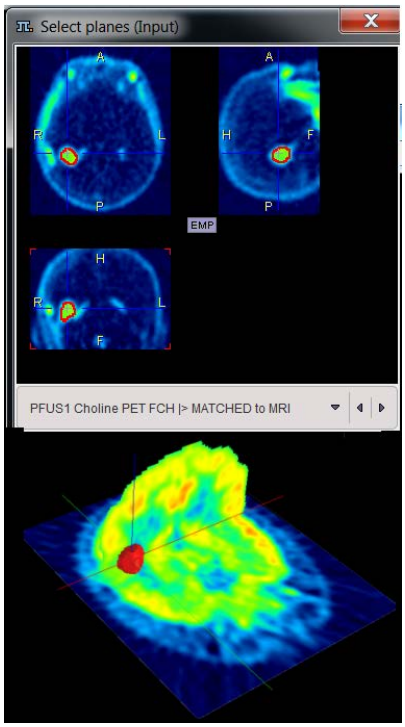


Marker Objects

Marker objects might be helpful for labeling points of interest in the scene. A marker object can be created by selecting **Markers** in the object tree, and then on the appearing **Marker** tab activating the **Add marker** button.

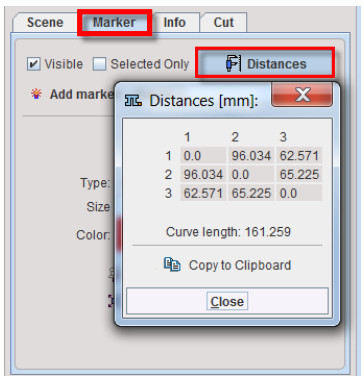


The marker is initially positioned at the plane intersection point of the **Input** series. Its position can be changed by changing the triangulation point there, or by opening the navigator window with and clicking to the point of interest. Once its final location has been found, activate **Lock Marker's Position** to prevent unintentionally moving the marker.



The marker can have the shape of a **Cross** as illustrated above, or a **Sphere**. If multiple markers have been created, **Selected only** allows showing only the marker belonging to the object currently selected in the tree.

If at least two markers have been created the **Distances** button become active. It allows calculating the distances between any two landmarks and shows them in a table:



The length of the curve that passes through the markers is also calculated. The values can be **Copy to Clipboard** and paste into Excel.

Cutting away Parts of the VR or SR Information

Sometimes it may be helpful to cut parts of the scene. In P3D it is possible to remove parts from VR or SR objects, limited by orthogonal planes or using oblique planes.

By Orthogonal Plane

Cutting Procedure

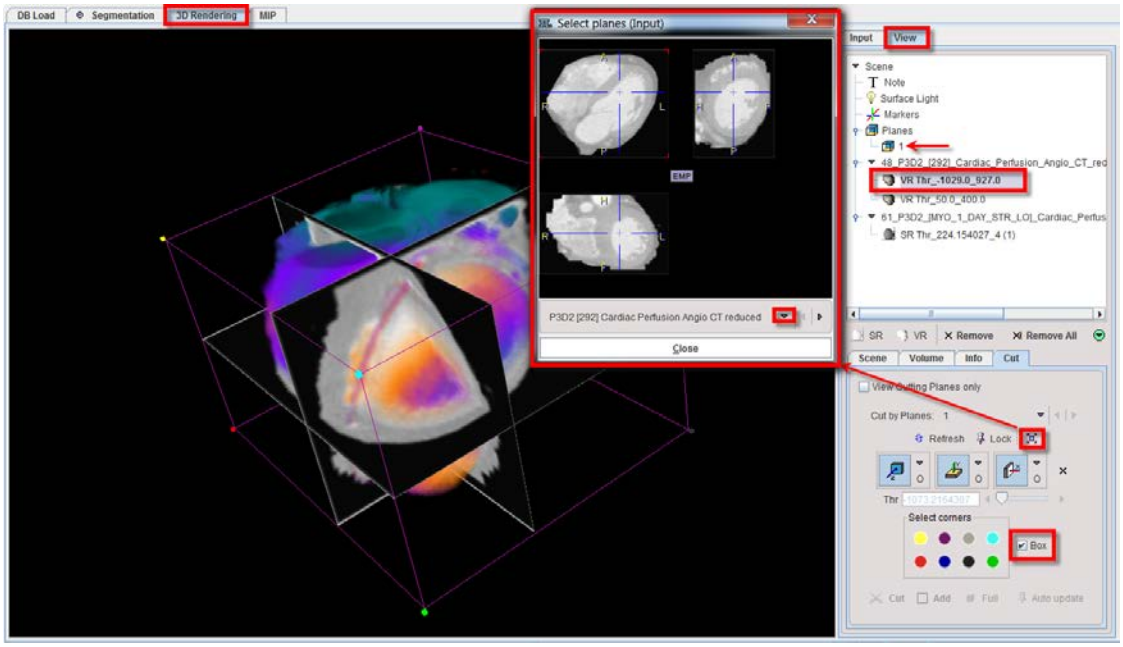
1) Make sure a **Planes** object is available. If not, then create one.

2) Select the VR or SR object(s) in the tree which should be cut. Multiple objects can be selected for cutting at the same time.

3) Select the **Cut** tab belonging to this object (NOT the Cut tab of the Planes object).

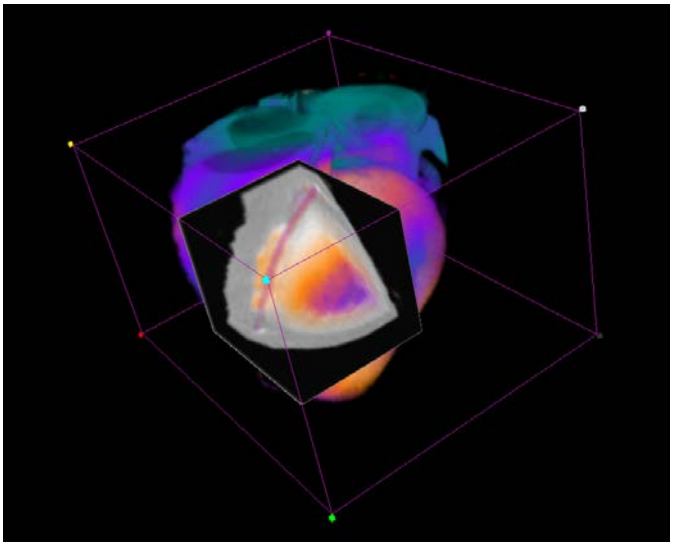
4) Enable the image planes display with the button.

5) Position the plane intersection point such that the three orthogonal planes enclose the area to be removed.

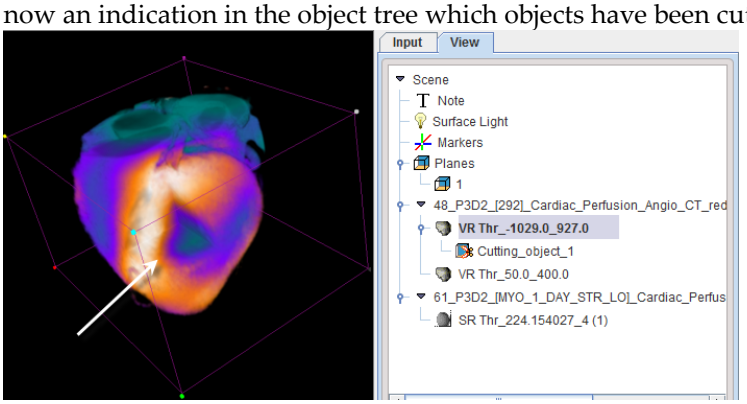


6) Enable **Box** to see the wire-frame box with the colored corners. Define the part to be removed by selecting color circles in the **Select corners** area. The color circles represent octants identified by the colored bullets in the corners of the wire box. After selecting the button(s), only the plane parts enclosing the parts to be removed are shown, and the **Cut**

button becomes active.

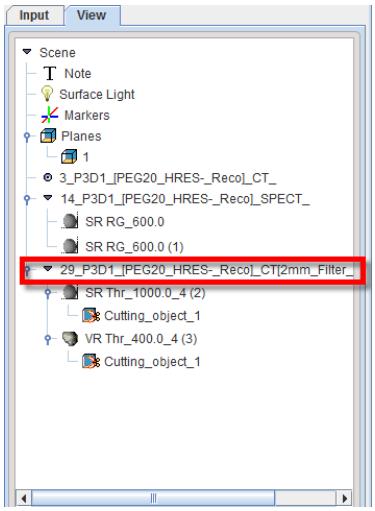


7) The **Cut** button starts a process with clears all information of the selected object(s) in the defined area and refreshes the rendering. Use the **x** button to remove the planes. There is now an indication in the object tree which objects have been cut.



8) The **Full** button undoes cutting and brings back the rendering of the full volume.

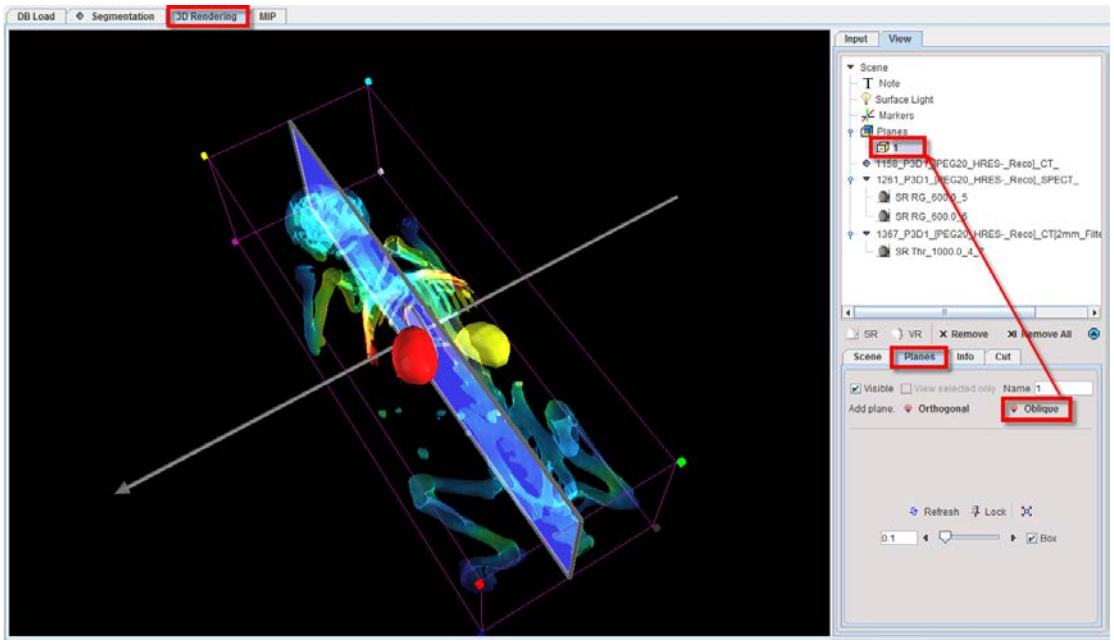
To apply the cutting procedure to all objects belonging to an image series the root object can be selected in the object tree as illustrated below.



By Oblique Planes

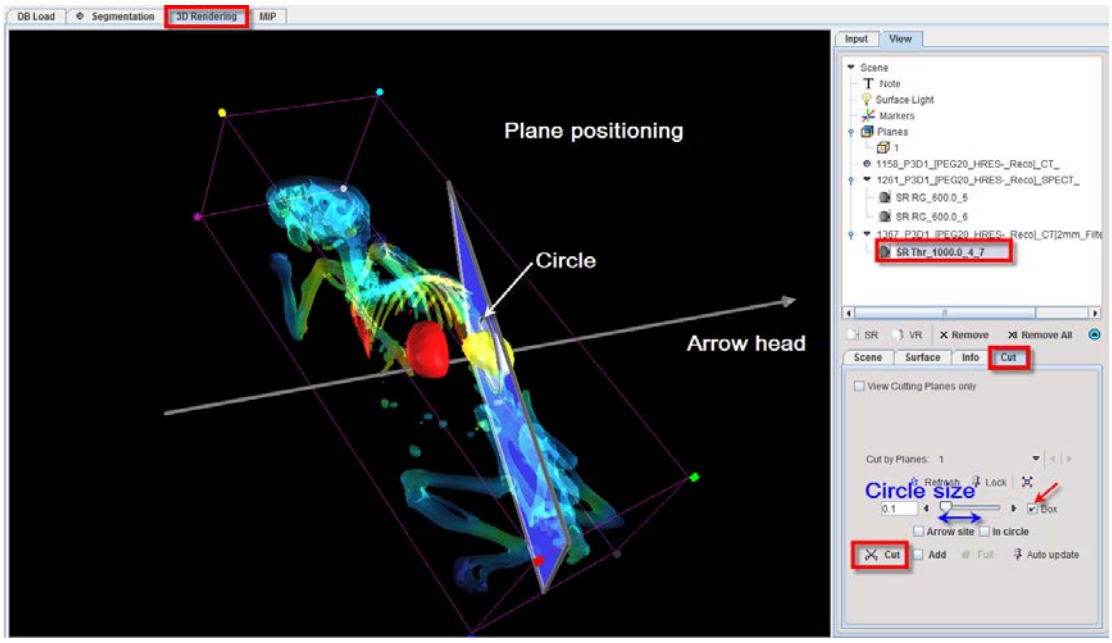
Cutting Procedure

1) Make sure an oblique **Planes** object is available. If not, then create one with the **Oblique** button in the **Planes** tab.



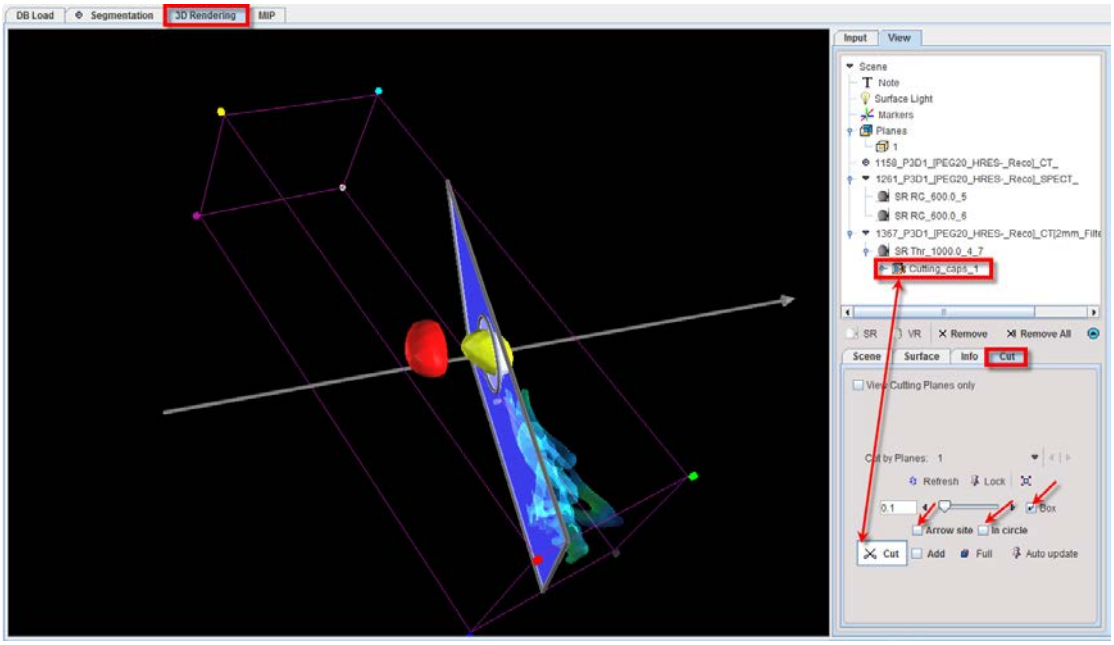
Enable **Box** to see the wire-frame box with the colored corners. The color circles represent octants identified by the colored bullets in the corners of the wire box.

2) Rotate and position the plane in the scene. Use the arrow head for rotation and the sides plane for translation of the plane. The circle can be further translated within the oblique plane. The circle is representing the origin of the rotation and its center is represented by the arrow. Select the VR or SR object(s) in the tree which should be cut.

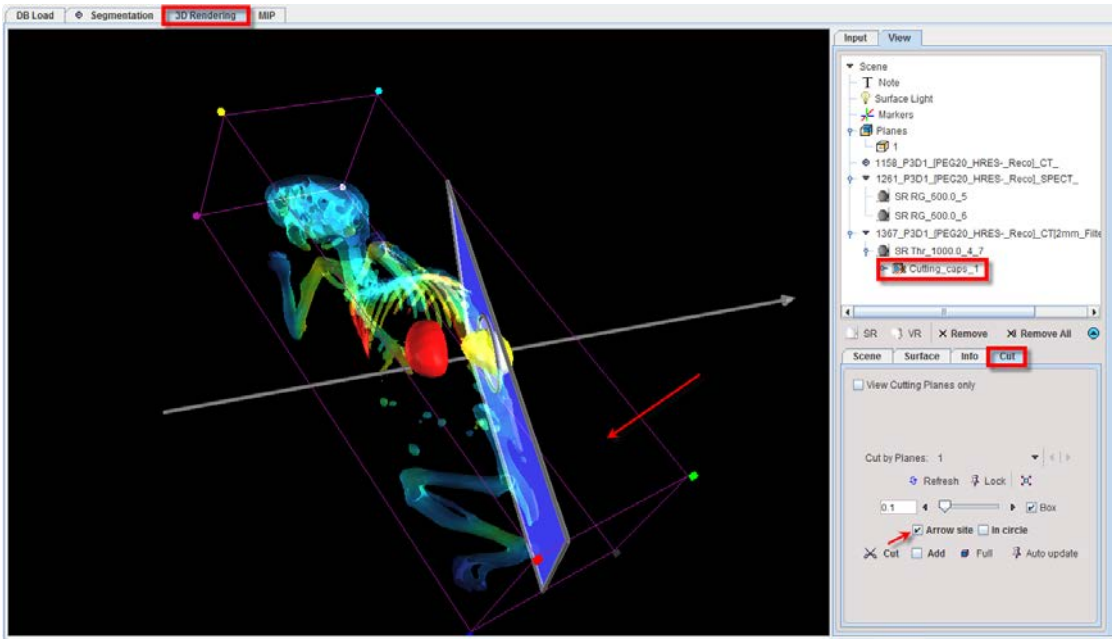


3) Select the **Cut** tab belonging to this object (NOT the Cut tab of the Planes object).

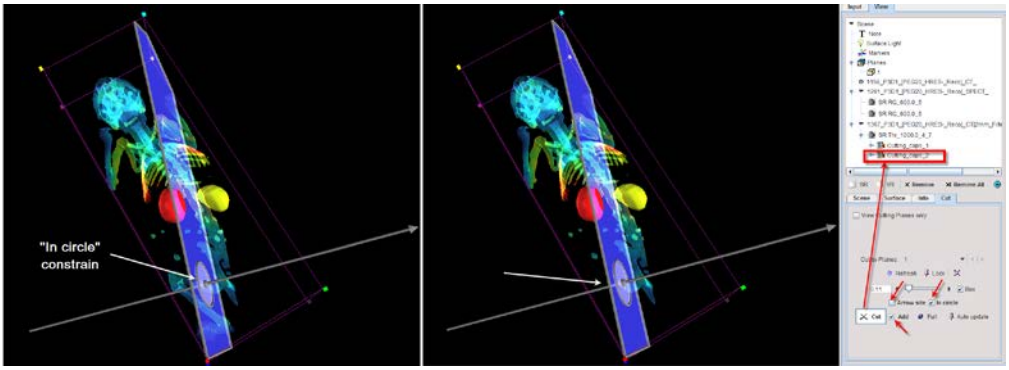
With the **Arrow site** and **In circle** boxes disabled the **Cut** button is activated. The **Cut** button starts a process which clears all information of the selected object on the defined side and area and refreshes the rendering scene. The part of the SR object (the skeleton) opposite to the arrow head is cut from the scene. There is now an indication in the object tree which objects have been cut:



Activating only the **Arrow site** box before cutting, the part of the SR object on the same side of the arrow head is cut out from the scene:



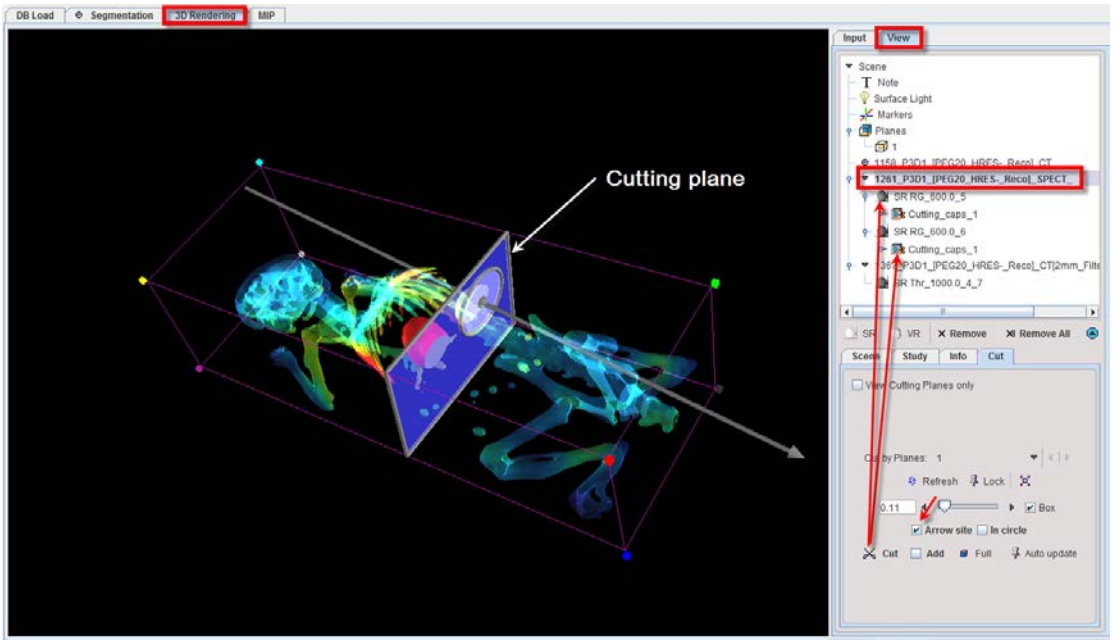
4) To add a cut for the same object in the scene the **Add** box has to be enabled. The cutting plane can be repositioned and, for example, the **In circle** box enabled can be used to constrain the cutting only within it. With the **Arrow site** box disabled the part of the object on the opposite side of the arrow head will be cut:



5) The **Full** button undoes cutting and brings back the rendering of the full volume.

6) Use the **x** button to remove the planes.

To apply the cutting procedure to all objects belonging to an image series the root image can be selected in the object tree as illustrated below. Please note that correct positioning of the oblique plane is mandatory for the cutting procedure.



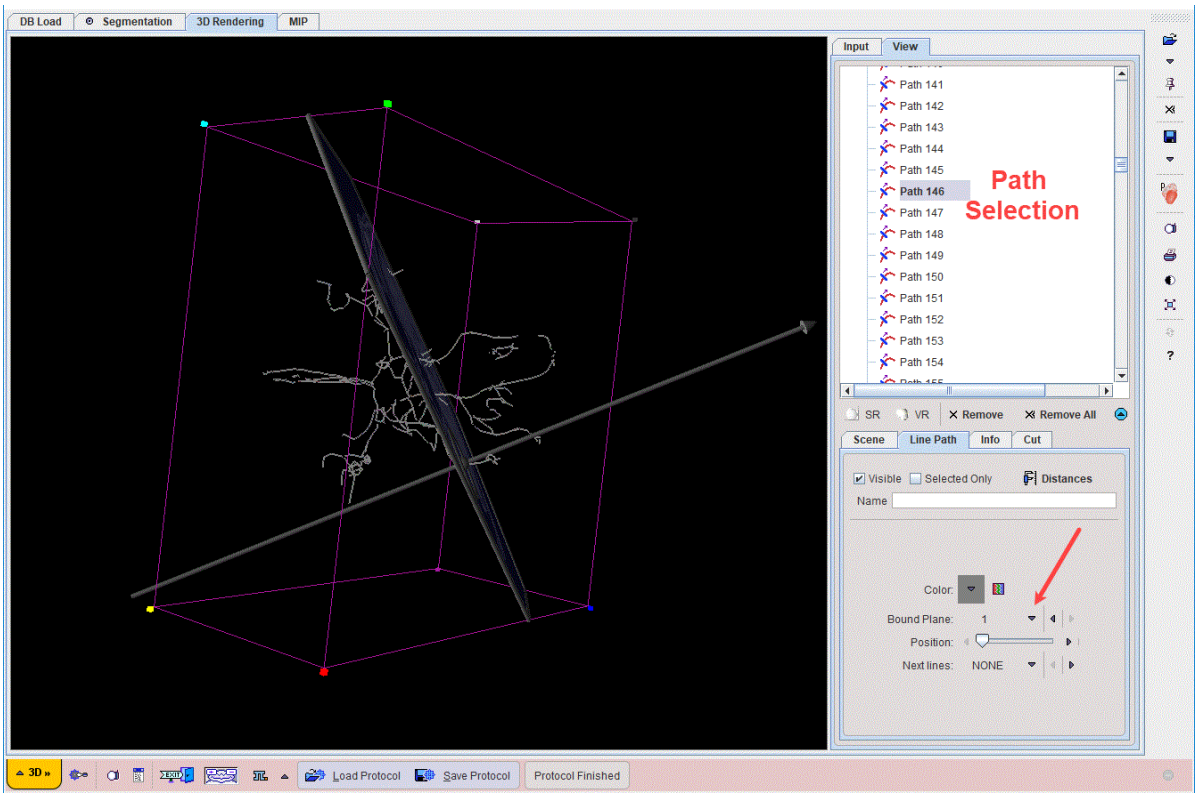
Using Skeletons (Paths)

Cutting Procedure

1) Make sure an oblique **Planes** object is available. If not, then create one with the **Oblique** button in the **Planes** tab.

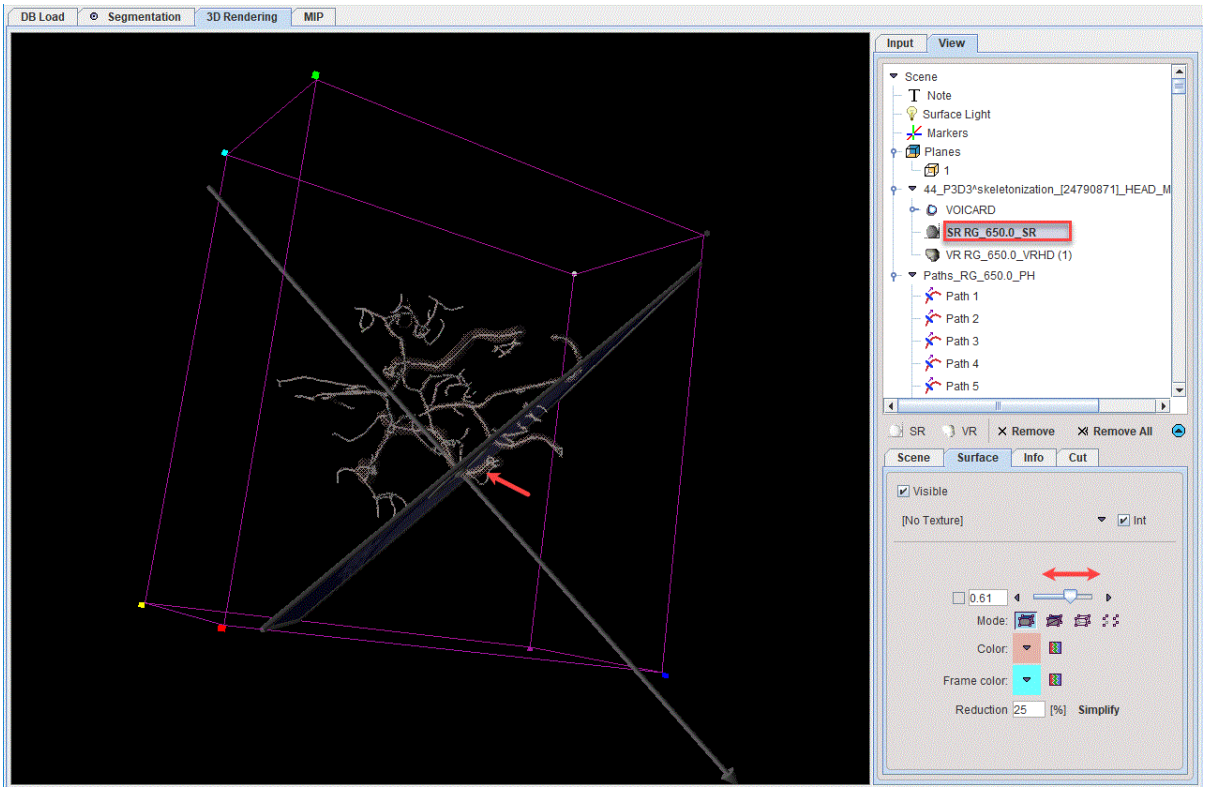
Enable **Box** to see the wire-frame box with the colored corners. The color circles represent octants identified by the colored bullets in the corners of the wire box. In the example above the input image was used to add a texture to the oblique plane.The **Int** box enables interpolation of the texture information.

2) Bind the plane with a path by selecting first the path in the tree and then selecting the oblique plane **1** as **Bound Plane:**

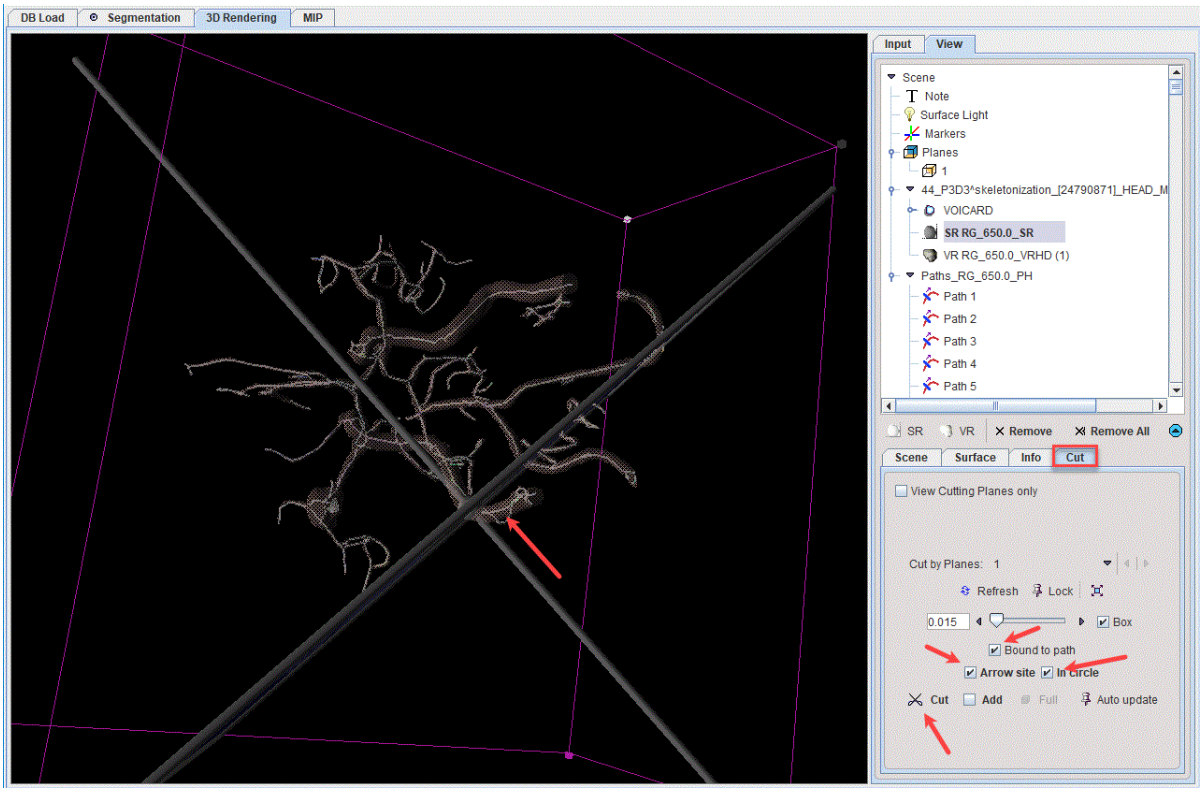


Note that the selected plane **1** is placed perpendicular to the path line. The **Position** slider allows moving the selected plane along the active **Path**. The SR object used to generate the skeleton (paths) is set to ful transparency.

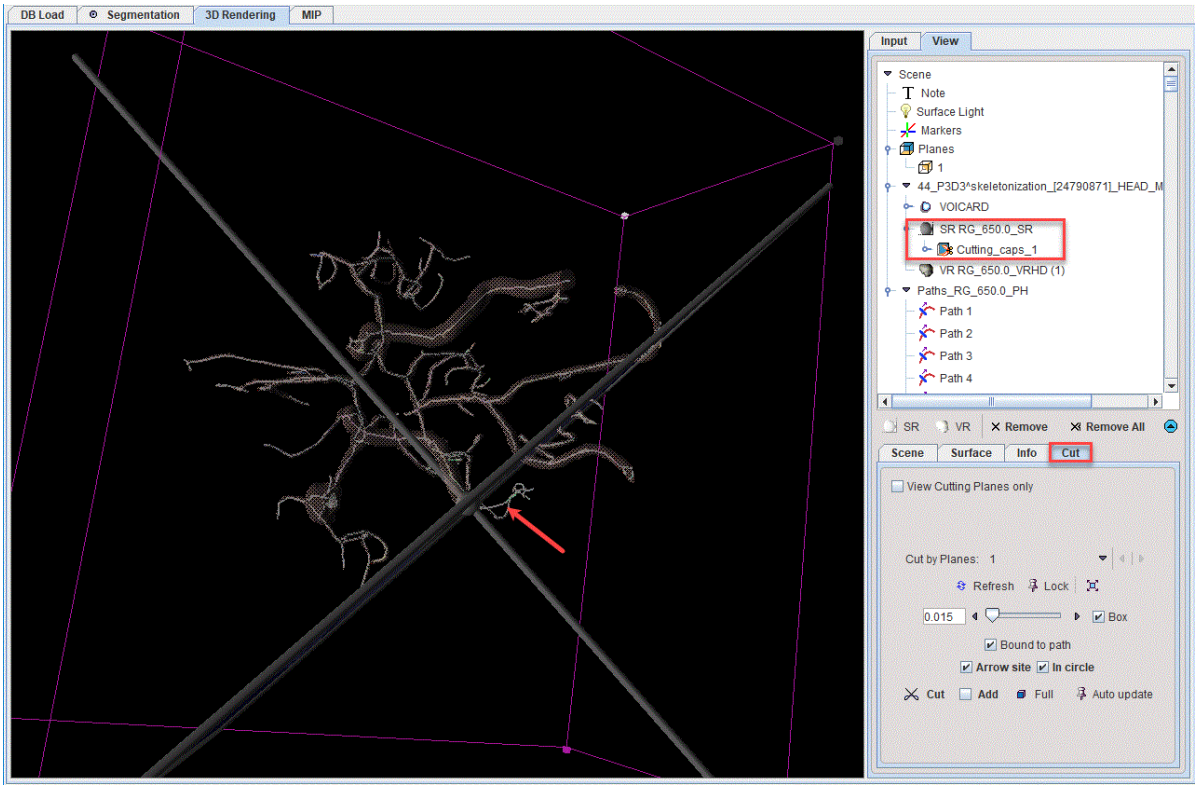
3) To cut a part from the SR object used for the skeletonization select the SR object in tree and adjust the transparency. In the example illustrated below a value of 0.61 transparency is used for the solid appearance.



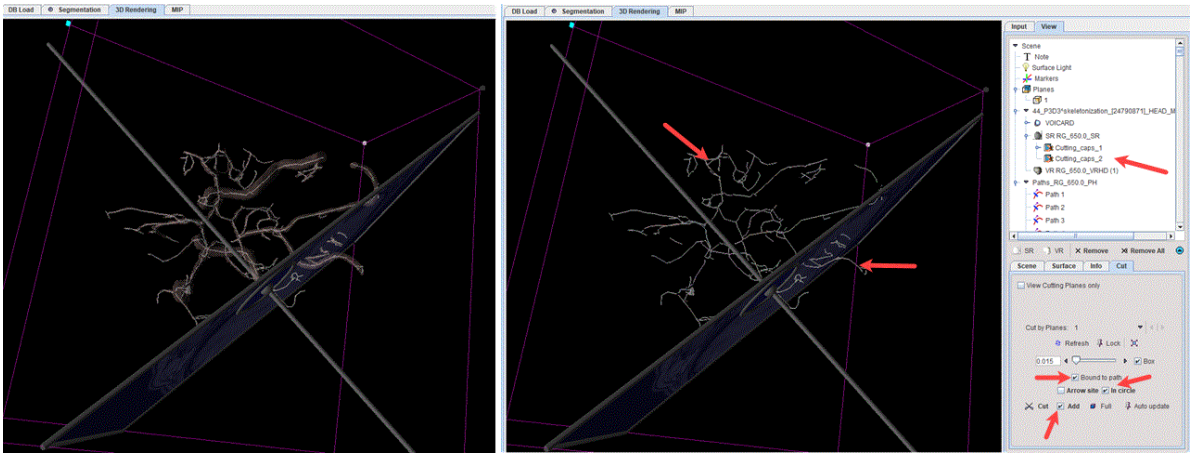
4) Select the **Cut** tab belonging to this object (NOT the Cut tab of the Planes object).



With the **Bound to path**, **Arrow site** and **In circle** boxes enabled the **Cut** button is activated. The **Cut** button starts a process which clears all information of the selected object on the defined side and area and refreshes the rendering scene. The part of the SR object on the same side as the arrow head is cut from the scene. There is now an indication in the object tree which objects have been cut:



5) To add a cut for the same object in the scene the **Add** box has to be enabled. With the **Arrow site** box disabled the part of the object, **In circle** and **Bound to path** on the opposite side of the arrow head will be cut:

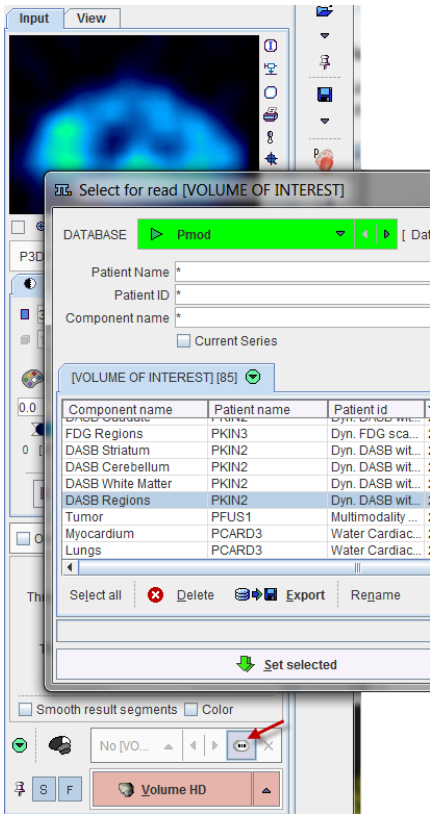


Please note that in the capture above parts on both sides of the cutting plane were cut. This occurs because the SR object was created using a region growing algorithm and the different paths of the skeleton are connected to each other. Particularly, the path 146 which is bound with the cutting plane is connected with all the other available paths.

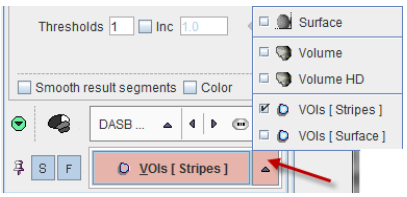
Rendering of VOIs

Volumes of Interest

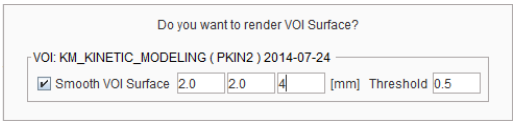
Volumes-of-interest can also be rendered by P3D. One application is the visualization of VOIs independent of an image study. To this end the VOI file is selected with the button indicated below on the **Input** tab.



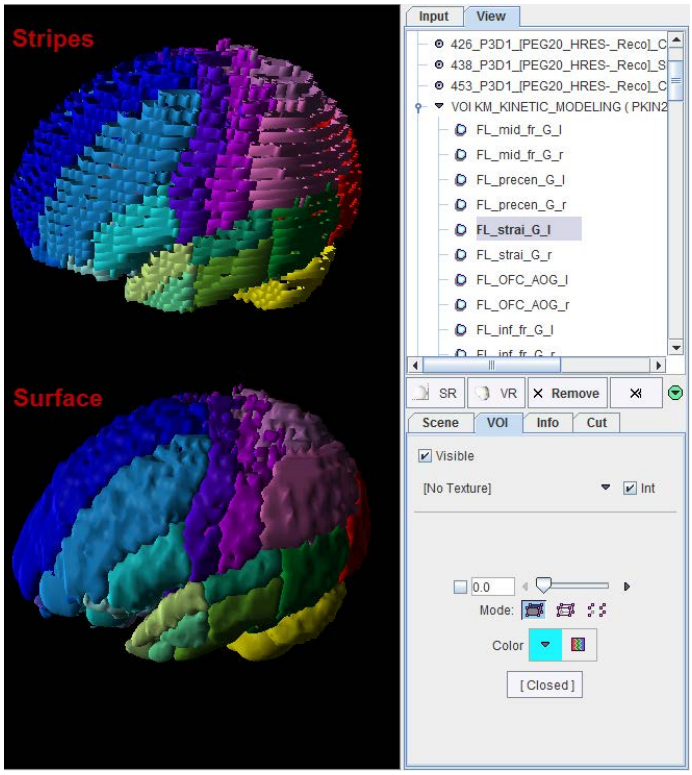
For rendering VOIs, there are two choices available, **VOIs [Stripes]** and **VOIs [Surface]**.



When starting the rendering, a dialog window is opened to enable smoothing of the VOI information.

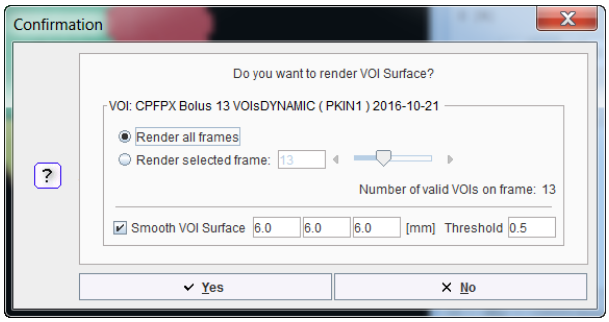


The results of both rendering types are illustrated below. Note that each VOI results in an object which can be selected in the tree and modified.



In case a VOI was used for restricting an image segmentation, it is rendered together with the segment and will also be available in the object tree.

In P3D there is the possibility of rendering dynamic VOIs. Select the VOI file. When no image is loaded, the activation of one of the VOIs rendering options opens a dialog window as shown below:



When the **Render all frames** radio button is enabled, the entire set of VOIs will be rendered. With **Render selected frame**, the user can select the frame which VOIs are going to be rendered. Additionally, the VOIs surface can be smoothed if the **Smooth VOI Surface** is checked

---

Note: When the P3D option has been purchased, the 3D rendering of the VOIs can also directly be initiated from the VOI definition tool.

Scene Operations

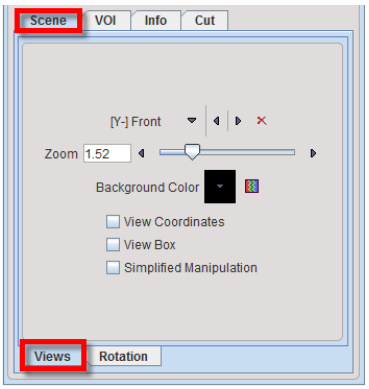
The main advantage of 3D renderings is that a user can interactively manipulate the scene to see it from different viewpoints. The scene can be explored in a mouse-operated way:

- To rotate the scene, click the left mouse button into the scene and drag the mouse.
- To zoom the scene, click the middle mouse button into the scene and drag the mouse. Note that this mouse-operated zooming changes the rotation center. In order to keep the rotation center in the center of the object, the zoom slider on the **Scene** pane should be used.

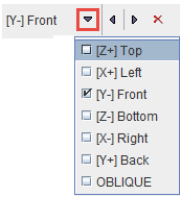
To shift the scene out of the center, for instance to examine a zoomed lesion, click the right mouse button into the scene and drag. To have a full view of the scene use the  button in the taskbar for hiding the controls to the right. Activating the same button again brings the controls back.

In the **Scene** panel of the **View** tab there are two sub-panes.

Scene Views



The list selection at the top



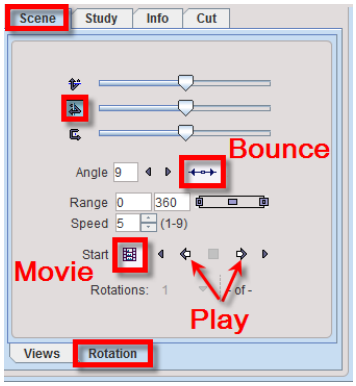
allows establishing well-defined viewpoints.

Zoom	Zoom factor for scene rendering.
Background Color	List selection for changing the background color.
View Coordinates	Show/Hide the box indicating the anatomical directions

View Box	Show the wire-frame box.
Simplified Manipulation	Show points instead of a surface during the scene rotation. It is recommended to be used with large SR object.

Scene Rotation and Cines

The **Rotation pane** contains multiple elements for creating cines or movies.



The sliders allow well-controlled rotations of the scene.

A cine is configured by the following properties:

1) the initial rotation angles (sliders),

2) the rotation axis (icons left to sliders),

3) the **Angle** increment in degrees (smaller increments produce smoother cines),

4) the rotation **Range** in degrees,

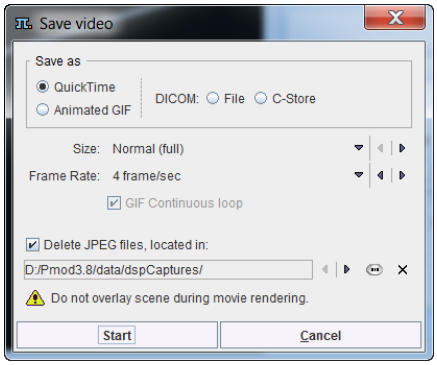
5) the behavior at the end of the rotation range (bouncing),

6) the **Speed.**

The cine can then be started with one of the play buttons, eg . Note that the rotation axis may be changed while the cine is playing, and scene manipulations by the mouse are still active.

Creating Movies

To record a cine as a movie first configure the cine loop appropriately and test it. Then enable the movie button  and start. A dialog appears

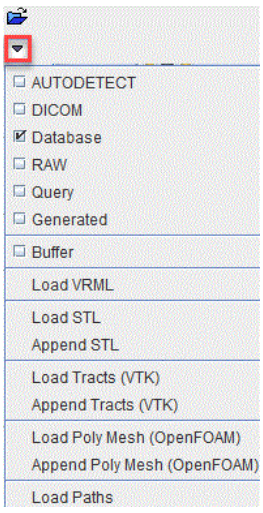


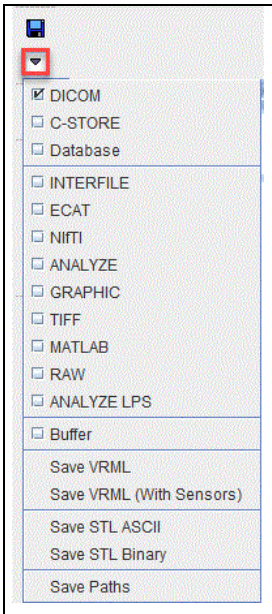
which allows defining the movie format (QuickTime, Animated GIF or DICOM), quality settings and the destination path.

After confirming with **Start**, the scene is stepped through the angles until the defined number of **Rotations** is covered, creating a jpeg file at each angle. Finally, the jpeg images are compiled into the movie, and the jpeg files (optionally) deleted.

Scene IO

The elements for documenting the scene and saving it are located in the lateral taskbar.

	Create a screen capture of the current scene for documentation purposes.
	Create a screen capture of the current scene and put it into a report page.
	The loading function has six parts. **1)** If an image format between **AUTODETECT** and **Buffer** is selected, an image is loaded which can be used for segmentation or texturing. **2)** The following part is for loading a 3D rendering definition in **VMRL**, or **STL** format. **3)** Tracking results saved as VTK files are loaded with **Tracts (VTK)** selection. **4)** The next part is for loading OpenFoam simulation results: **PolyMesh.** **5)** The last part is for loading skeletonization results **(Paths)** in **.vec** format.
	The saving function has the following parts:

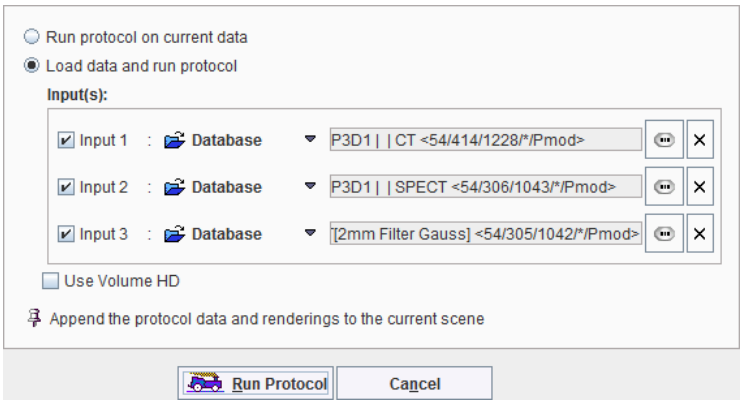


1) If an image format between **DICOM** and **Buffer** is selected, the last used segment image is saved.

2) The lower part is for saving the 3D rendering definitions in different formats: **VMRL**, **VMRL (With Sensors)**, **STL ASCII** and **STL Binary,** and **Paths**. Note that only the SR objects are saved, not the other scene parts such as the VR objects or the planes. The **STL Binary** file is the most compact format, but it is a proprietary implementation and not compatible with other systems. Recommended for data exchange is **STL ASCII**. Note that skeletonization results (Paths) should be saved as **Paths**.

Protocol Files

Saving a scene in VRML is limited to SR objects, and may create huge files. A more flexible alternative is to save a description of the P3D rendering in a protocol file using the **Save Protocol** button in the status line. When such a P3D protocol is loaded again, a dialog window appears which indicates the data to be loaded.



If the user wants to apply the same rendering to similar data, he can replace the input files with the buttons. If he wants to apply the procedure to data already loaded, the **Run protocol on current data** can be enabled.

Normally the scene will be cleared and replaced by the result of the protocol. To add to the current scene, the **Append the protocol data and renderings to the current scene** pushpin should be fixed. To create high definition VR objects the **Use Volume HD** option should be enabled.

The button shows a dialog window containing standard rendering definitions.

Currently, the following predefined protocols are supported:

1) Heart Atlas

2) MR-AAL

3) Angio CT renderings are offered with and without multi-modal fusion.

4) Bones (CT)

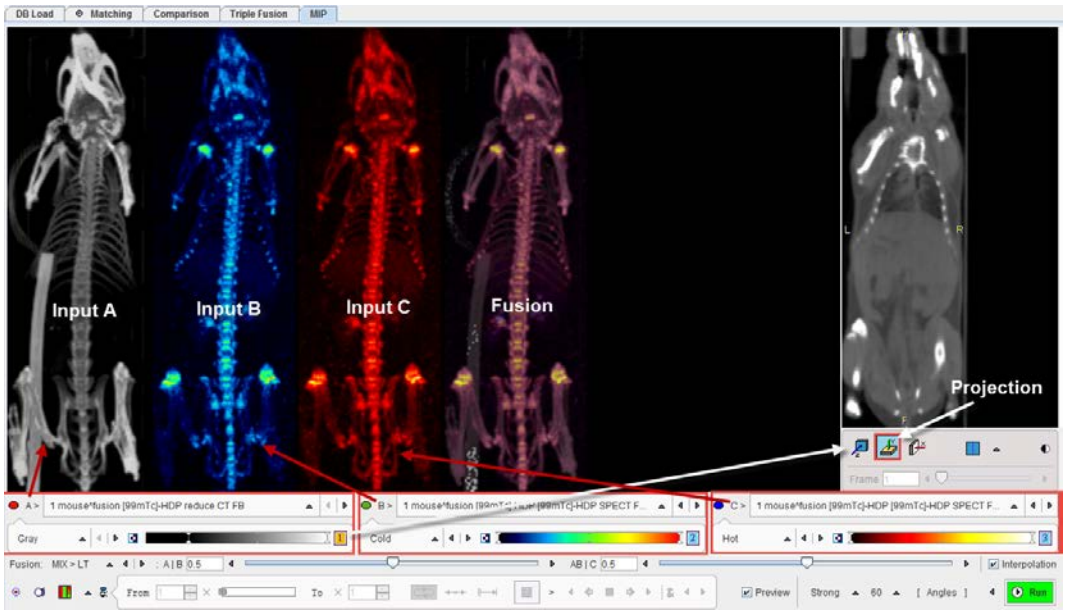
5) AAL VOI

6) Mouse (CT+NM)

Note that when a predefined protocol is selected, appropriate input files have to be set using the buttons. Alternatively, the predefined protocol can be applied to data already loaded by enabling the **Run protocol on current data** radio button.

MIP Page

The **MIP** main page for generating **M**aximum **I**ntensity **P**rojection cines has the layout illustrated below.

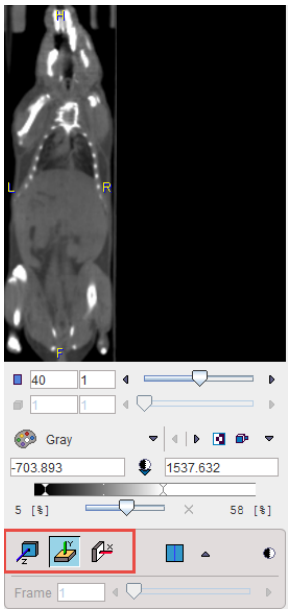


- The upper left image area shows a preview of the input images (**A**, **B**, **C**) and their fusion. Each input image has a color bar associated to adjust the image coloring. The upper right area serves for defining the projection direction, coronal in the example above.
- The fusion image is obtained by first fusing **A** and **B**, and then fusing the result with **C**. The mixing is defined by the two corresponding **Fusion** sliders **A|B** and **AB|C** below the colorbars.
- The control of the MIP characteristics and movie generation is located at the bottom.

Please proceed as described below.

Projection Direction

The MIP projection direction is set using the plane orientation buttons in the image area to the right. If necessary, the displayed image can be switched as illustrated above.



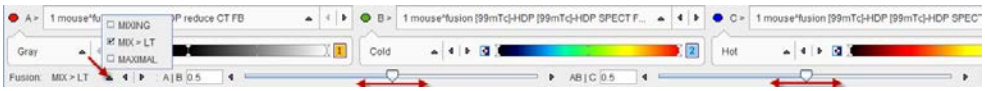
Input Image Selection

Per default, the reference image is arranged as series **A**, and the matched input images in sequentially as series **B** and **C**. This arrangement can be changed using the series selection as illustrated below.

Note the EMPTY entries which are only available for **B** and **C** which allow excluding those images from MIP generation.

Fusion Configuration

Each series has its own colorbar for adjusting the image presentation. The color choices should be such that the different image components can be distinguished in the fusion. As a default configuration, the reference is shown with **Gray** colors.



There are three fusion options available:

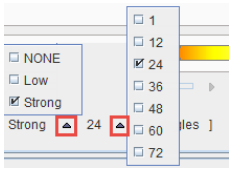
MIXING Simple weighted averaging of the RGB values, whereby the relative contributions are defined by the fusion slider.

MIX>LT Weighted RGB averaging considering only pixels which are above the respective lower thresholds, hereby removing the background. The relative contributions are defined by the fusion slider.

MAXIMAL With this setting no color averaging of the two inputs is performed. Rather, the bigger of the two contributions is selected.

MIP Configuration

The **MIP** calculation performs a ray tracing from different angles and selects the maximal value on a ray for display in the MIP image. There are two MIP calculation parameters in the lower right.



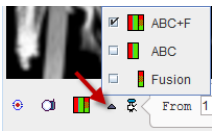
1) Distance waiting with the settings **NONE** (default), **Low**, **Strong**. This option emphasizes objects closer to the observer by multiplying the value with a factor which decreases with distance.

2) Number of projection angles. The selection ranges between **1** and **72** angles. The more angles are chosen, the smoother the rotation cine will appear, however at the cost of longer preparatory calculations.

Cine Control

Image Display Selection

Per default, all the input MIPs as well as the fusion MIP are rotated. However, there is a choice in the lower left which allows showing subsets of these images.



Show the three input MIPs as well as the fusion MIP (default).

Show only the three input MIPs.

Show only the fusion MIP.

Projection Calculation

Calculation of the configured projections is started with the 

button. After the calculation is completed, the rotation cine is immediately started.

Cine Controls

The direction of the cine, the speed and the behavior after a rotation can be configured with the usual cine control elements.

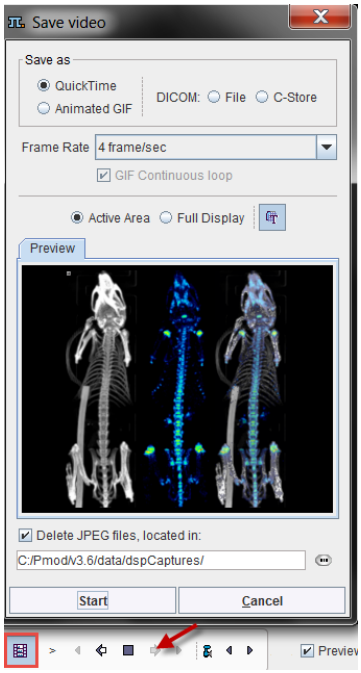
If any of the image presentation options is activated, the cine stops. However, in most cases a projection recalculation is not required, so the rotation can be simply restarted.

Maximizing the Display Area

In order to maximize the image area for watching the cine, the controls can be minimized with the show/hide taskbar button indicated below. They are recovered by activating the same button again.

Movie Generation

In order to create a movie file, the button has to be activated and then the cine started. A dialog window is shown for configuring the movie format **QuickTime**, **Animate GIF** or **DICOM**.

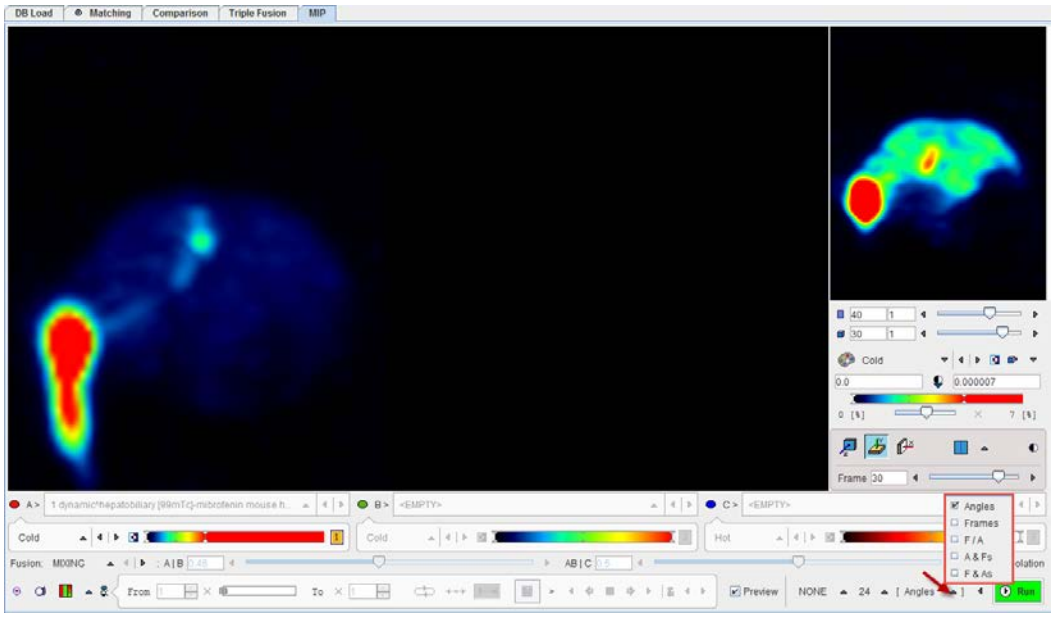


The movie will be assembled from JPEG files which are saved to a folder which is to be configured in the lower part. The JPEG images may be persistent, depending on the **Delete JPEG files** option.

After activating the **Start** button, the JPEG images corresponding to the different angles are written to disk and a dialog window opened defining a location and a name for the movie file.

MIPs using Dynamic Series

In the special case that an image series is dynamic, there are additional control options.



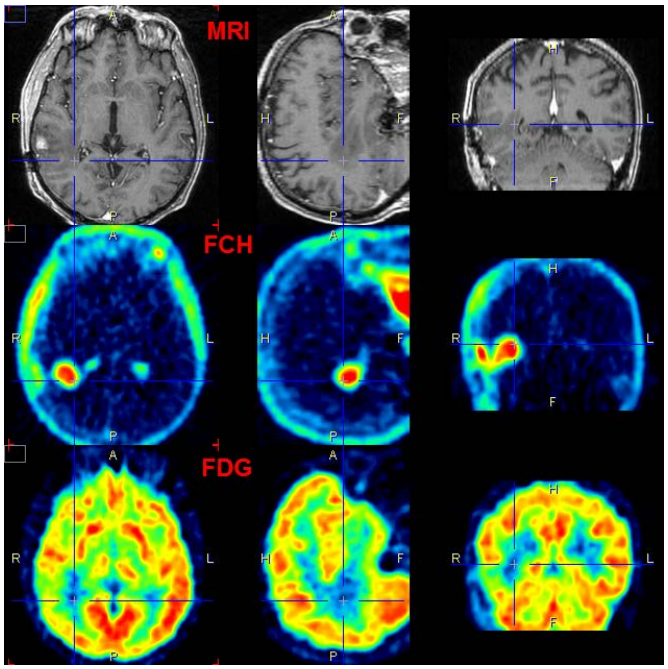
Angles	Standard rotating MIP cine of the current frame.
Frames	MIP across all dynamic frames in the selected projection direction.
F/A	Mode in which the angle and the frame are simultaneously incremented. As a consequence, the number of angles equals the number of frames. The effect is, that the image changes during the rotation.
A & Fs	In this mode, the projection angle is fixed, while all frames are MIP rendered. This rendering is sequentially performed for all angles.
F & As	In this mode, a full rotation MIP is generated for a fixed frame. This rendering is sequentially performed for all frames.

3D Rendering Examples

The main visualization capabilities of P3D are illustrated by some examples.

Example 1: Surface Rendering of a Brain Tumor

This example can be reproduced with data of patient **PFUS1** available in the example PMOD database after installation. Please note that the FCH and FDG PFUS1 series are rigidly matched beforehand to the MRI PFUS1 using the FuseIt module.



These three matched studies are loaded as **Input** series. The aim is to localize and visualize the brain tumor. A simple yet useful approach to be performed on the **Segmentation** page is the following:

FCH The tumor is well delineated in the FCH data but needs a restriction.

Set **Region Growing** segmentation with criterion >=0.

Enable **Use VOI** box and activate **Edit** button.

From PMOD database load the VOI **Tumor** .

Remove the *Contralat* VOI. Confirm with **OK** and click into the Tumor VOI.

Activate the **Segmentation** button

Add Segment to the **Segments** list.

Set Segments 3D rendering properties: set **Surface** as **Rendering Type** and surface **Color** to red.

Render the SR segment with the **Render selected** button

FDG The FDG data is used to obtain the outline shape of the brain.

First smooth the FDG data with a 6mm Gaussian. Save the smoothed FDG data.

Apply a **THRESHOLD** segmentation to the smoothed FDG study with a value of 7.9.

Activate the **Segmentation** button.

Add Segment to the **Segments** list.

Set Segments 3D rendering properties: set **Surface** as **Rendering Type** and surface **Color** to yellow.

Render the current segment in append mode by selecting the pushpin and subsequently the **Render selected** button.

Set blended transparency using a value of 0.7.

MRI MRI slices can be included for a better anatomical understanding.

On the **3D Rendering** page, **Input** tab select the MRI study.

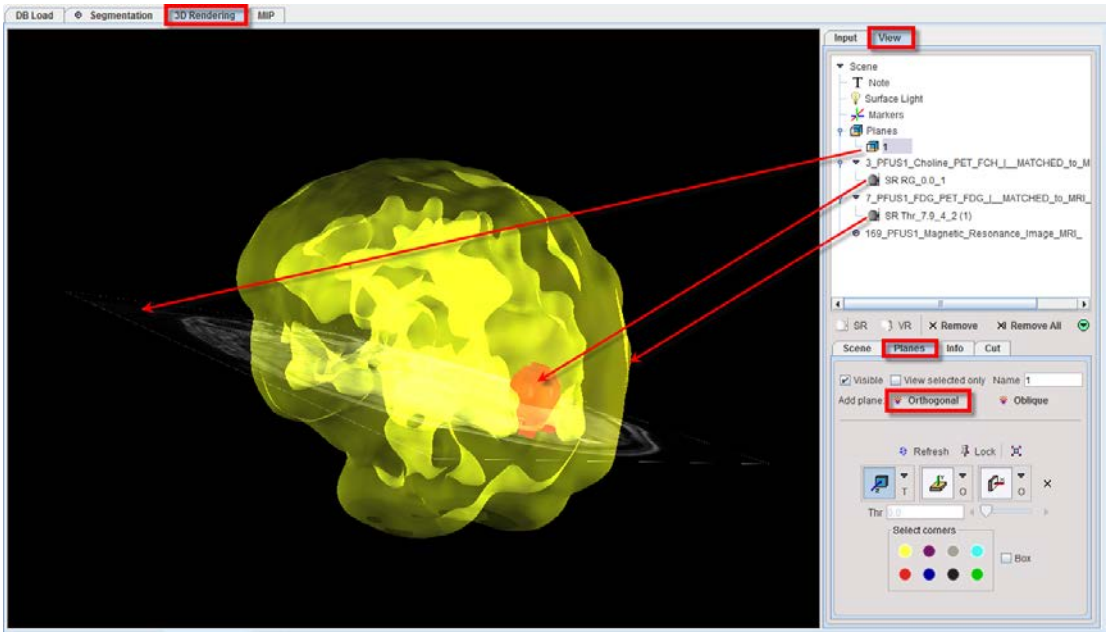
In the object tree select **Planes** and then activate **Orthogonal** for the **Add planes**. The three orthogonal MRI planes are shown.

Switch off the **y** and the **x** plane for display in the scene. Only axial MRI slice is shown.

Select **Transparent** mode (selection right to **z**-plane button)

Define the location of the plane using the navigator window, pointing at the location of interest. The scene gets updated accordingly.

The result from a specific viewpoint is shown below.



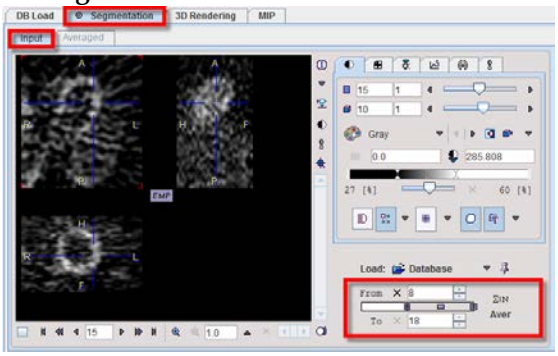
Example 2: Animated Texture on Cardiac Surface

This example illustrates the capability of P3D to project a dynamic texture onto a static surface using a dynamic NH_3 cardiac PET study. It can be reproduced with data of patient **PCARD1** available in the example PMOD database after installation.

Anatomy

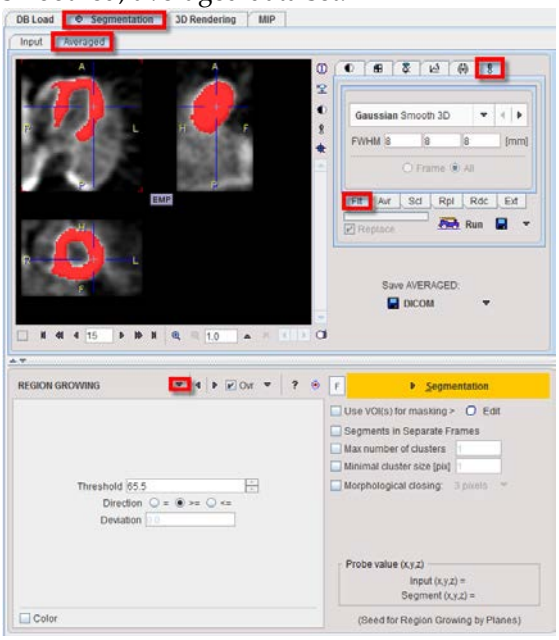
Load the dynamic **NH3, Stress** study of patient **PCARD1**.

On the **Segmentation** page set the average of the frames **From** 8 **To** 18. Activate the **Aver** button. The average image is available in the **Averaged** tab.



Smooth the generated average image with the external toolby applying an 8mm **Gaussian Smooth 3D** on the **Flt** tab. Save the smoothed average image.

Define **REGION GROWING** segmentation with threshold 65.5 for the smoothed, averaged data set.



Activate the **Segmentation** button and **Add Segment** to the **Segments** list.

keep the default settings for the rendering properties

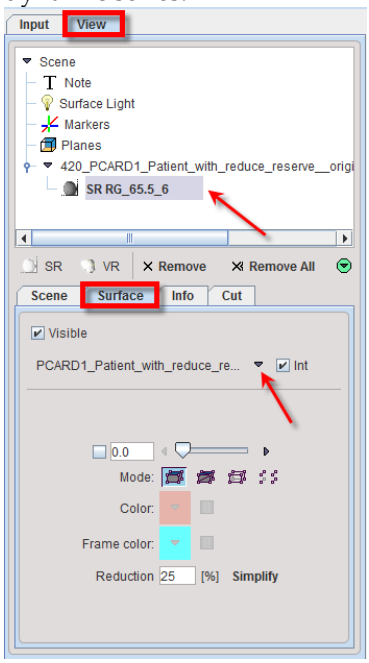
Render the SR segment with the **Render selected** button

A smooth shape of the myocardium is shown in the scene represented by the **SR RG_65.5** object in the tree.

Cine of Dynamic Uptake

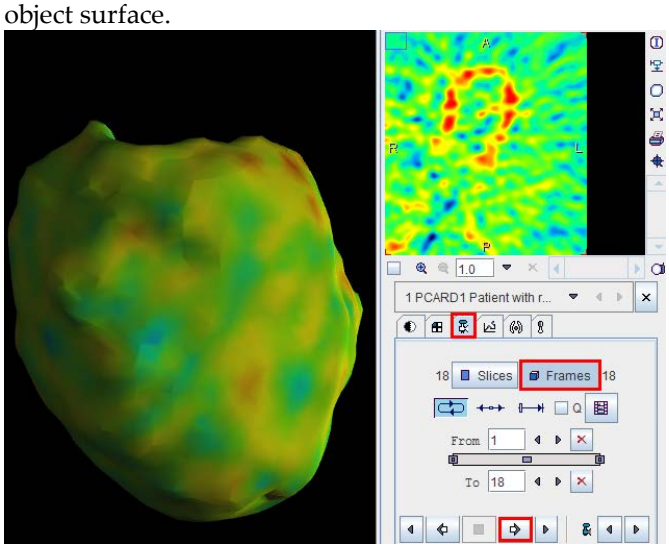
Set the dynamic **NH3, Stress** study of patient **PCARD1** into the **Texture** area.

Select the **SR RG_65.5** object in the tree, and switch the texturing to the dynamic series.



On the **Input** pane set the Input colortable to **Cold.**

Configure the cine player to show a movie in time (**Frames** mode), then start. The uptake over time is now shown as a dynamic texture on the object surface.

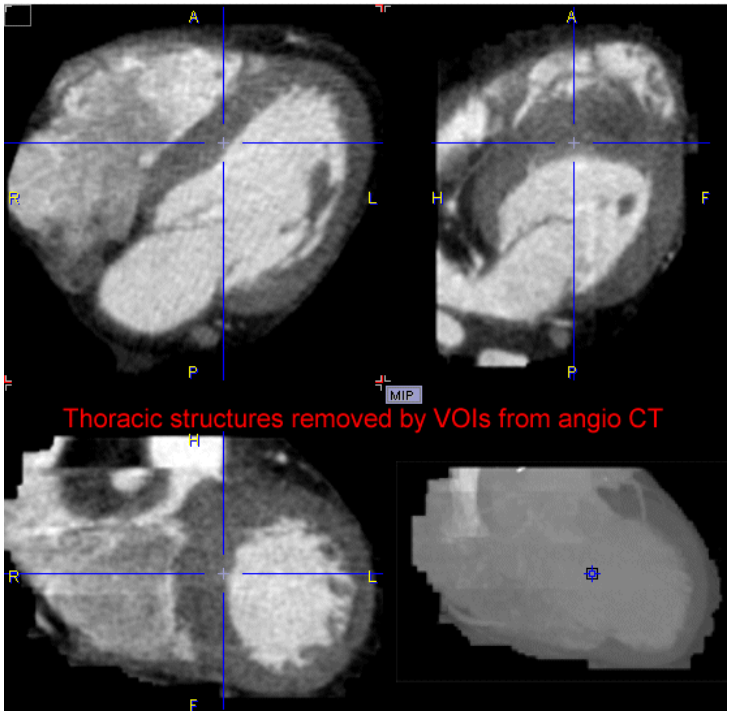


Note: When a dynamic study is segmented and the **F** button (single frame) is not selected, one object per time will be created. In this case a cine of the **Input** images can be run to show an animation of the object shape over time. An obvious application for this feature is the rendering of a beating heart from a gated acquisition.

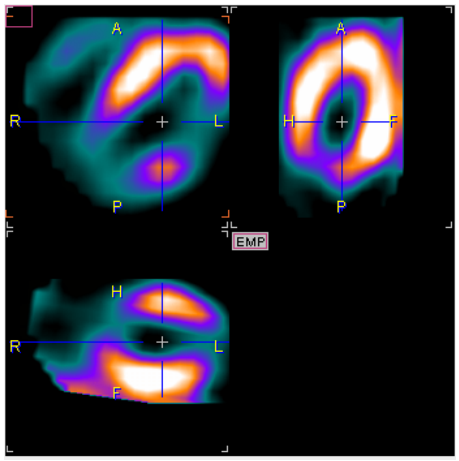
Example 3: Fused Volume Rendering for Angio CT

This example illustrates the use of the **Predefined** protocol for an angio-CT/SPECT fusion. It can be reproduced with data of patient **P3D2** available in the example PMOD database after installation.

First the CT data was prepared. A cubic sub-volume containing the heart is extracted from the data using the **Resize** function available in the external tools. The remaining thoracic structures are interactively removed by drawing VOIs and setting the values outside the VOI to -1024.



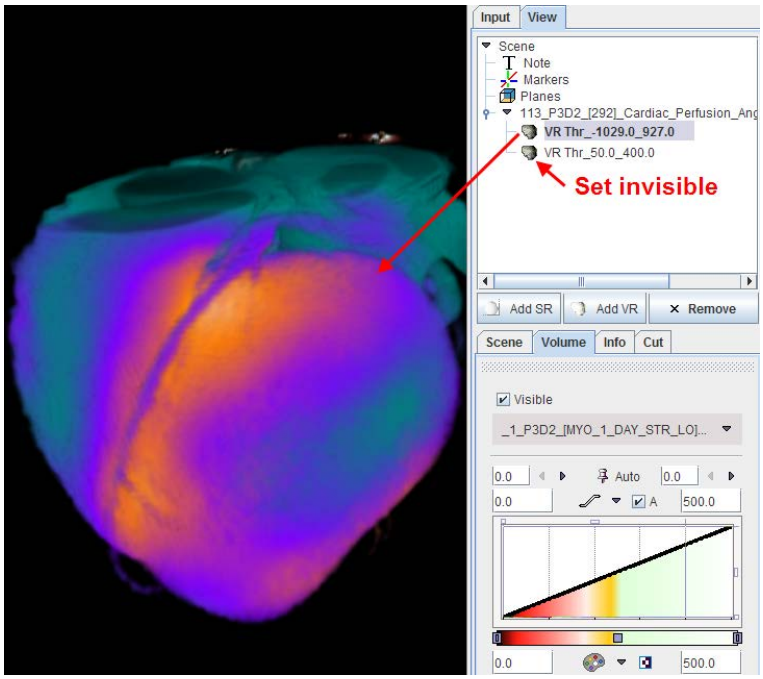
Next the SPECT data was matched to the CT in the PFUS tool, and the resliced images saved.



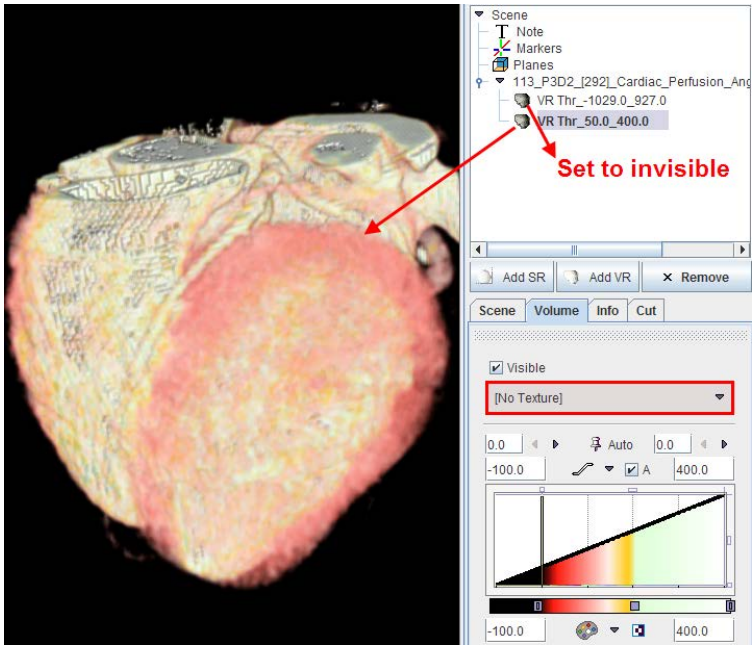
In P3D, load the prepared CT study **Angio CT Reduced** series of patient **P3D2** and the matched SPECT. Then, select **Heart (CT+ PET)** from the **Predefined** protocols. Select **Run protocol on current data** and than **Run Protocol** button.

The protocol performs a segmentation with two **RANGE** thresholds (-1029-927 HU, 50-400 HU), and performs a VR rendering for each of the resulting segments.

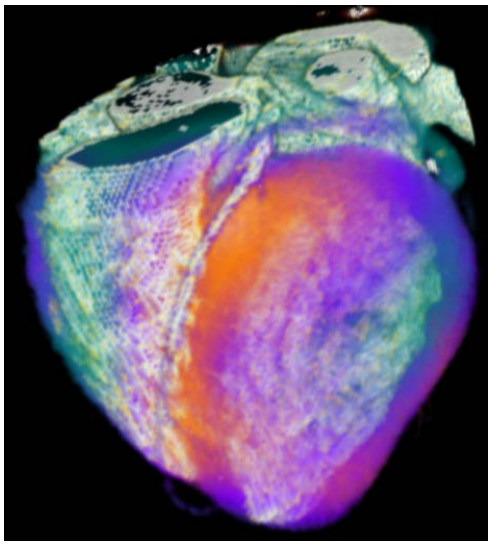
The first segment is textured using the SPECT colors, and results in a rendering as illustrated below. To see this segment only remove the **Visible** check of the second VR object. After adjusting the SPECT colors a bit the result looks like illustrated below:



Note that the coronary arteries are colored because the SPECT study is not sharply bonded. Therefore, the second segment is used for highlighting the coronary artery structure.



When both VR renderings are combined in a single scene (checking both **Visible** boxes), the following result is finally obtained.



This scene could be extended by additional objects. For instance, if an important coronary artery could be individually segmented, it could be added as a clearly visible SR object.

PMOD Copyright Notice

PMOD Technologies Ltd
Sumatrastrasse 25
8006 Zürich
Switzerland
+41 (44) 350 46 00
support@pmod.com
http://www.pmod.com

Index

V