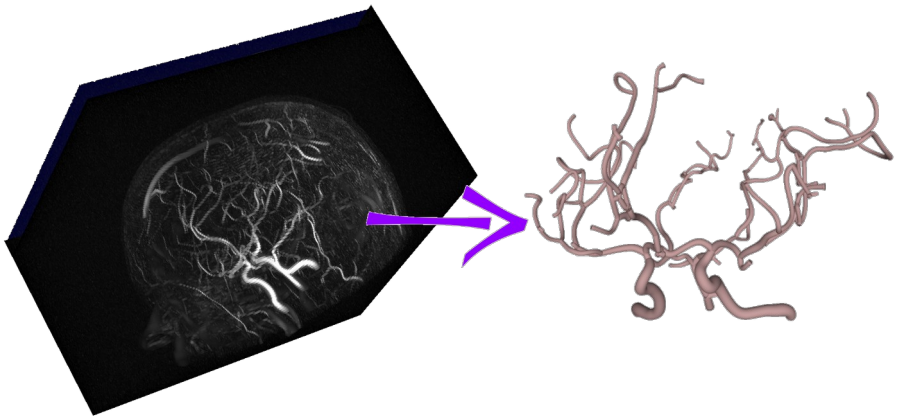
Piotr M. Szczypiński

vesselknife manual

2022-04-15

Introduction

One of the issues in supporting medical diagnosis is analysis of angiogram images to identify and determine the course and radius of blood vessels. The idea behind *VesselKnife* development is to create a tool that implements algorithms for angiogram segmentation and for converting raster data into vectorized vessel tree models. The program includes advanced raster image processing algorithms, such as vesselness function computation, and algorithms which enable vectorization of data: determination of center lines, bifurcation locations and radius estimation.



The algorithms, in part, results from original studies, and others are from Insight toolkit library. The multiscale vesselness function computation and application of skeletonization were proposed by Grzegorz Dwojakowski and Adam Sankowski (PhD students of the Institute of Electronics).

VesselKnife also includes the original algorithms for visualization of three-dimensional image data, including imaging with transparency (maximum intensity projection) and mixing data in RGB color channels. 3D images, resulting from various processing procedures, can be displayed with different colors, and combined for comparison. The visualization module is directly implemented in OpenGL. The *VesselKnife* package of programs is still under development. Software is provided under GPL3 license.

Source codes are available from GitLab repository: https://gitlab.com/vesselknife/vesselknife.

Binaries are available from http://www.eletel.p.lodz.pl/pms/SoftwareVesselKnife.html.

How to install

Precompiled binaries of the program are available from my webpage http://www.eletel.p.lodz.pl/pms/ProgramyVesselKnife.html.

Download appropriate package file for your system. Linux and Windows platforms are supported. Extract the files from the package to the folder of you choice. There are three executable files: *VesselKnife*, *vkconvert* and *vk4console*. Also, copy this manual file (*vesselknife.pdf*) to the same folder.

VesselKnife is a main program with a graphical user interface. Two others (*vkconvert* and *vk4console*) are programs that can be run from a console. Click on the files or run them in a console (bash or cmd) and verify if they work. Run the three programs one-by-one. If the system asks, confirm that you want to run them and they are from trusted source.

In Linux you may need to install Qt libraries. You may try this command to install the required libraries on Debian-like Linux distributions:

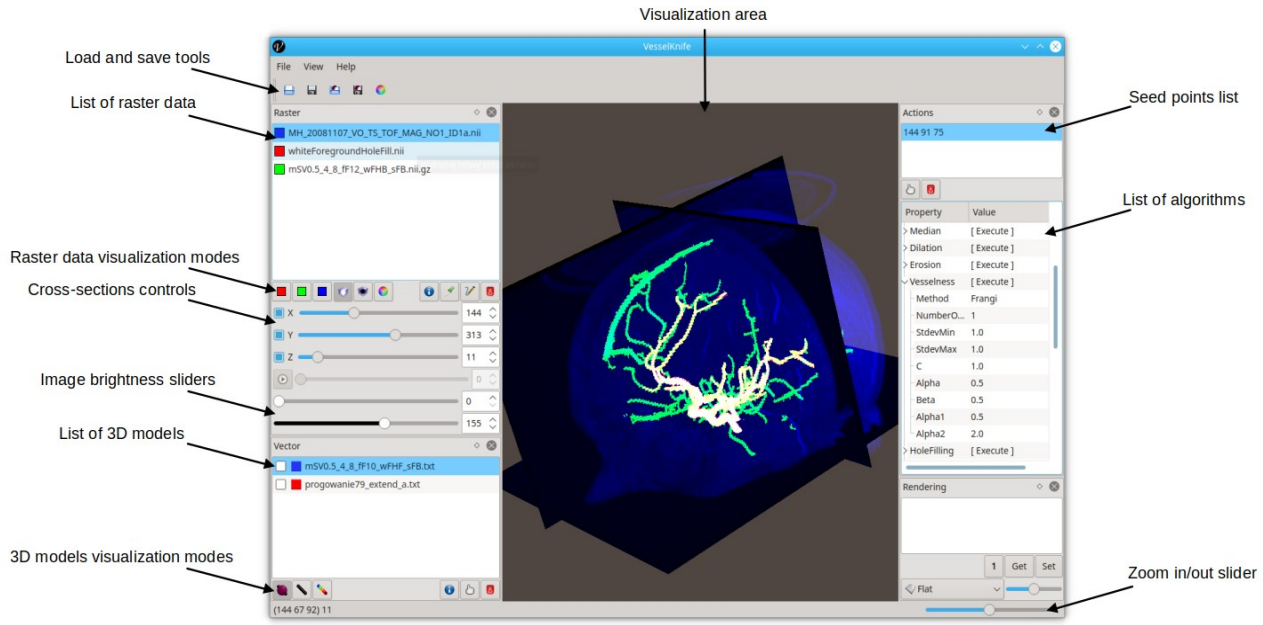
```
sudo apt install qt5-default
```

If you do not trust these binaries or the programs do not run on your system, you can compile them by yourself. *Vesselknife* is an open source project and can be compiled from sources. The sources are available under GNU General Public License from https://gitlab.com/vesselknife/vesselknife.

VesselKnife interface

Vesselknife's window contains several panels. The central panel is an area for visualization of three-dimensional raster data (images from tomographs, 3D angiograms) and 3D models produced by the program. By pressing the left mouse button and moving the mouse cursor over the area, you can rotate the view. If you press Shift button at the same time, you will be able to shift the view. If by chance you shifted away the whole thing and you cannot restore it, search for three buttons *1*, *Get* and *Set* at the right bottom region of the window. Press *1* and then *Set* to restore default size, position and orientation.

The top panel contains buttons for loading and saving raster images, loading and saving 3D models, and the last button for setting the background color for visualization.



On the left-hand-side there is a list of raster images. The images on the list can be loaded from files. However, all the images on the list need to have exactly the same size in voxels. As the user runs image processing algorithms the resulting images will be added to the list.

Panel with buttons for raster data visualization modes contains 6 + 4 buttons. The firs three represent visualization colors (red, green and blue) that can be associated with particular image from the list. Select the image first and then press one of the three color selection buttons. The image will be presented in the selected color. This mode enables to show three raster images at the same time with different colors. This mode is useful for image mixing and comparison. The

following two buttons turn on maximum and minimum intensity projection (MIP) viewing modes. The sixth one enables setting the background color. The following four buttons are for: showing some detailed information about the selected image, picking up a gray-level of the indicated voxel, filling in individual voxels with the picked gray-level, and for removing selected image from the list.

By default raster images are shown as three perpendicular cross-sectional planes. The panel with cross-section controls have check boxes to show or hide particular cross-sections, and sliders to select which cross section should be presented.

The next panel below contains two sliders to set minim and maximum grey-level values for visualization. Be aware that using these sliders may be choppy if you have MIP visualization turned on.

Below there is a panel with the list of centerlines and models produced by the program. This list gathers data in vector format. As you run algorithms for vector data processing, the resulting data are added to the list. Each element on the list comes together with check box and color box. Check boxes enable showing or hiding particular models. Clicking on the color box enables setting unique color for particular model.

Below the list there are 3 + 3 buttons related to the 3D models' visualization. The first three buttons switch presentation mode to vessel volumes, centerlines and color coding of radii. The other three buttons are to show detailed information on the selected data, indicate a specific node of the centerline or to remove selected model from the list.

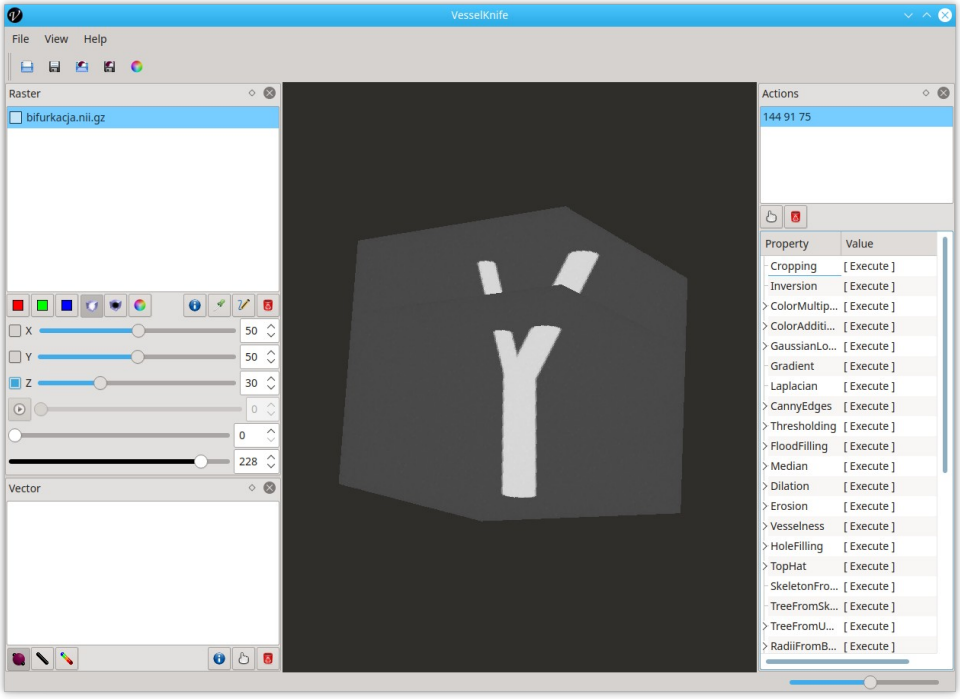
At the bottom of the window there is a status bar. It shows information and warnings about your actions. On the right-hand-side of the status bar there is a slider for zooming the view.

The middle panel on the right-hand-side lists the implemented algorithms which can by run. Each algorithm has a unique name and it has a drop down list of parameters. You can set (edit) appropriate values for the parameters of the algorithm. Then you should select an image or the model from the lists on the left-hand-side panels, which should be used as inputs for the algorithm. Finally to run the algorithm, click on the [Execute] "button" next to the algorithms name. Be aware that most of the algorithms take as input images or models which are highlighted on the lists, yet not necessarily visualized in the central panel.

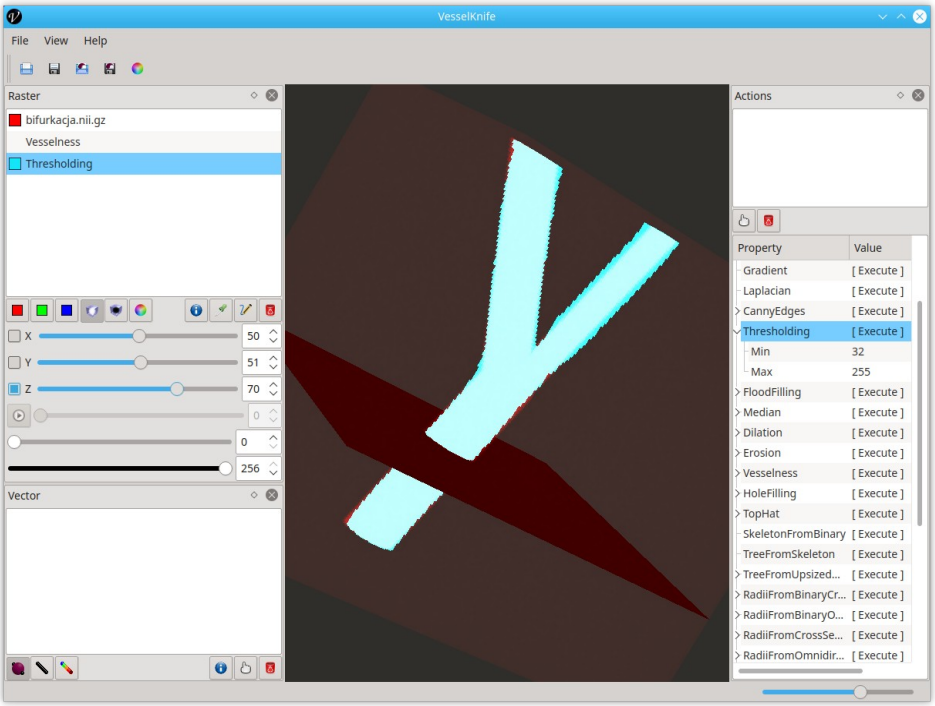
Some algorithms require additional information on image coordinates. For example the flood-fill algorithm needs a starting voxel (a seed point) as an input. You can define seed points by means of the top-right list of seed points. First click on the *hand* button below the list. Next click on the voxel you want to indicate within the visualization area. The coordinates of the indicated voxel will appear on the seed-points' list. If you are not satisfied with the coordinates, you can edit their values manually or remove the seed-point from the list by pressing the below button with the *trash basket*.

Step-by-step example

1. Start *VesselKnife* program.
2. Load the example file *bifurkacja.nii.gz* (Menu: File > Load raster…).
3. View the bifurcation image, try to use various viewing modes, including MIP.



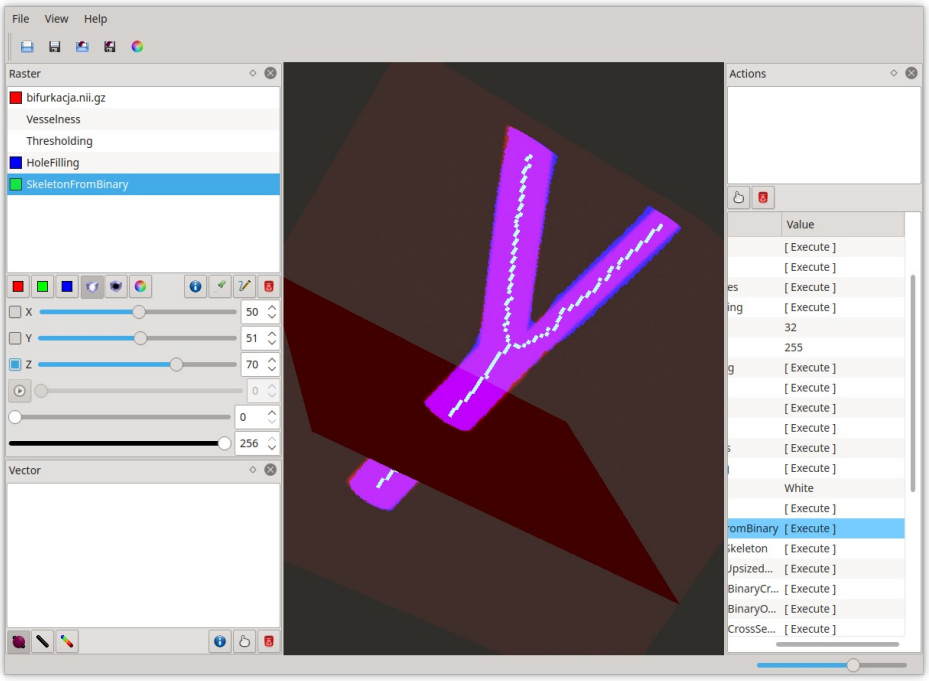
4. Find the *Vesselness* algorithm in the right-hand-side list, expand the list of its parameters.
5. Set: Method = Fragi, NumberOfScales = 5, StdevMin 3 and StdevMax = 10.
6. Press [Execute] next to the *Vesselness*.
7. Result of vesselness computation should appear in the left-hand-side panel list with the same name as the name of the applied algorithm.
8. Select (highlight) Vesselness image on the list.
9. Find *Thresholding* algorithm, set threshold Min = 32 and Max = 255, and press [Execute].
10. Compare the images on the list – use different colors for each of the images.



11. Make sure the Thresholding image name is highlighted. Find *HoleFilling* algorithm, set Object = white, and [Execute].

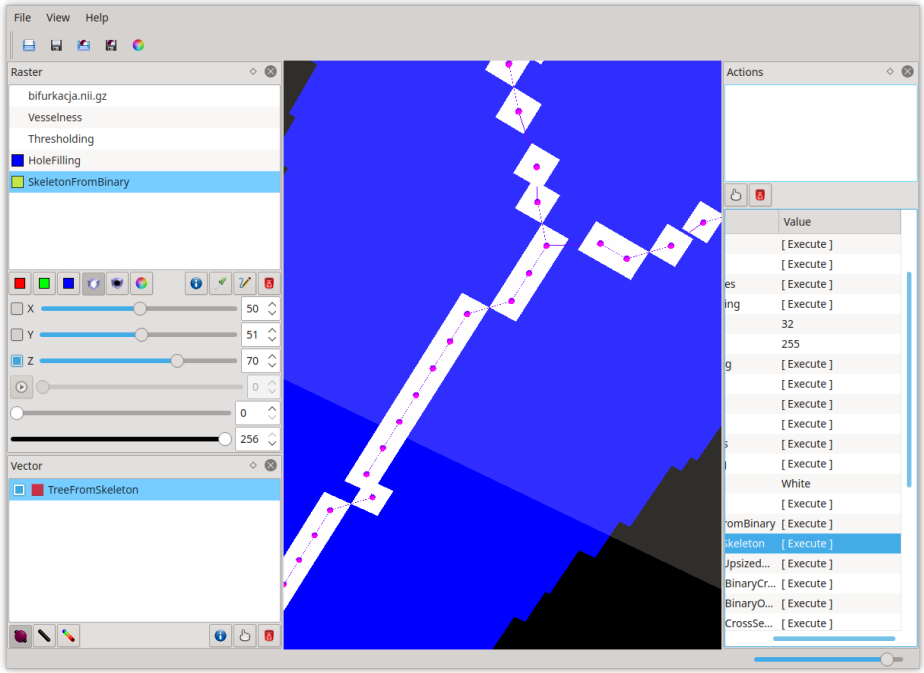
12. Now, make sure the HoleFilling image name is highlighted. Find *SkeletonFromBinary* algorithm and [Execute].

13. Compare original image, the result of hole filling and the skeleton with different colors, and in MIP mode.



14. Make sure the SkeletonFromBinary image name is selected on the list, find an algorithm *TreeFromSkeleton* and [Execute].

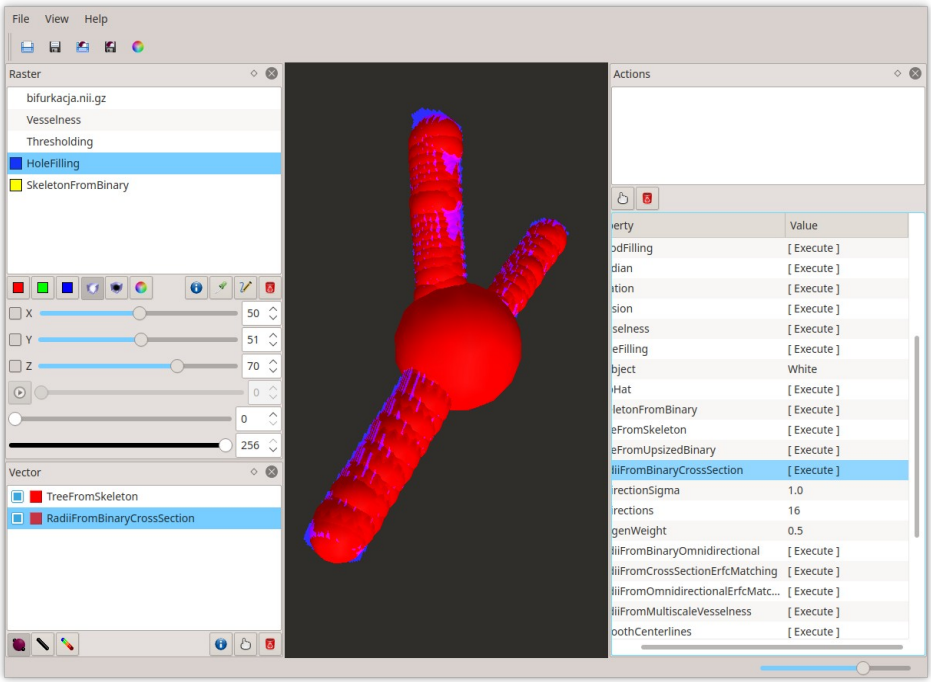
15. The new model, a centerline, shows up on the list of vector models. It is visible as a thin red line with nodes, that aligns with the skeleton.



16. Select the binary image HoleFilling from the list of raster data, and TreeFromSkeleton from vector data list.

17. Find the algorithm *RadiiFromBinaryCrossSection* and [Execute].

18. The model is presented with estimated radii shown as spheres. It can be noticed that radius at the bifurcation is overestimated.

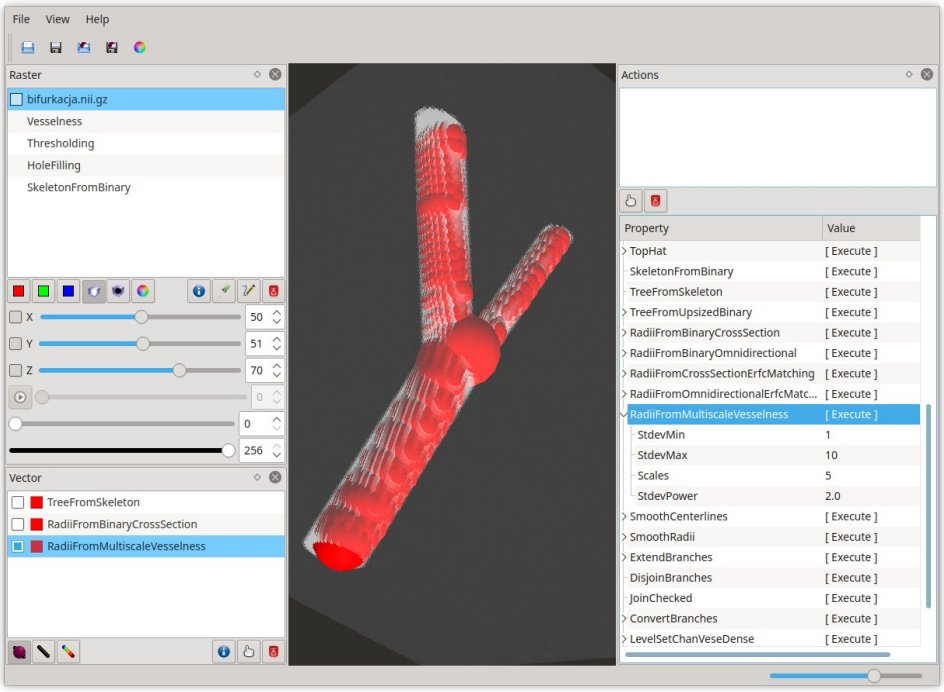


ALERNATIVELY:

16. Select the original grey-scale image bifurkacja.nii.gz on the list of images, and TreeFromSkeleton model on the list of models.

17. Find the algorithm *RadiiFromMultiscaleVesslness*, set Scales = 5, StdevMin = 1 and StdevMax = 10 and [Execute].

18. Uncheck 3D models except RadiiFromMultiscaleVesslness. The model is presented with quite correctly estimated radii.



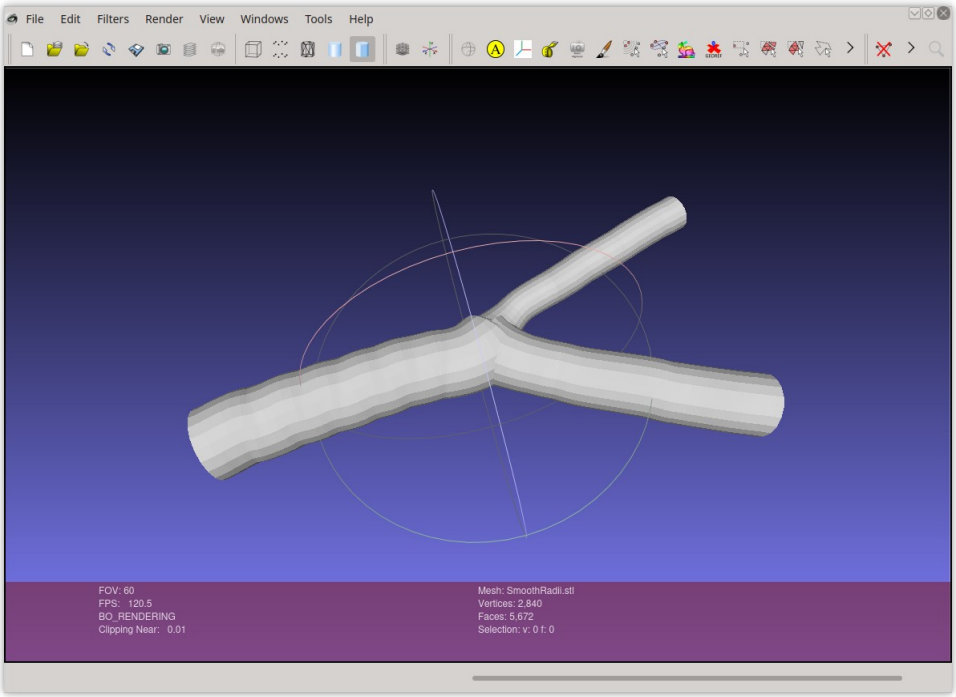
19. Select the resulting model from the list and apply *SmoothCenterlines* algorithm.

20. Select SmoothCenterlines model from the list and apply *SmoothRadii* algorithm.

21. Uncheck all the models except the SmoothRadii.

22. Select the SmoothRadii model from the list and save it in STL file format (Menu: File > Export > Tree to STL…).

20. Show the resulting model with the 3D data viewer (below MeshLab was applied).



Step-by-step with vk4console

Vk4console is a console program that implements the same image processing algorithms as Vesselknife. It enables to automatize actions available from Vesselknife as it can be easily run from the console or used in a script file.

If you run the program without any arguments it should print a short info, a list of available options and examples:

```
$ ./vk4console

Usage: vk4console [OPTION]...
VesselKnife for console and batch processing
Release 22.04
Compilation date and time: Apr 11 2022 18:09:38
Options:
  -a, --algorithm      Name of the algorithm
  -i, --input <file>   Input image
  -inr, --input-red    Red channel image
  -ing, --input-green  Green channel image
  -inb, --input-blue   Blue channel image
  -ina, --input-alpha  Alpha channel image
  -o, --output <file>  Output image or tree
  -t, --input-tree     Input tree
  -p, --set-parameter  Set parameter, e.g. -p Sigma=1
  -x, --fiducial-x     x coordinate in integer, voxel space
  -y, --fiducial-y     y coordinate in integer, voxel space
  -z, --fiducial-z     z coordinate in integer, voxel space
  -v, --verbose        Verbose mode
  -n, --num-threads    Number of ITK threads - defaults to maximum
available
  /?, --help           Display help and exit

Examples:
  vk4console -v -o dummy -a dummy
  vk4console -i image.nii -o thresholded.nii -a Thresholding -p Min=127
  vk4console -i thresholded.nii -o skelet.nii -a SkeletonFromBinary
  vk4console -i skelet.nii -o tree.txt -a TreeFromSkeleton
  vk4console -i image.nii -t tree.txt -o out.txt
             -a RadiiFromMultiscaleVesselness -p StdevMin=1 -p StdevMax=4
```

Calling the program without a proper algorithm name invokes a list of available algorithms, their parameter names and default values. The list of the algorithms should match the algorithms listed in the right-hand-side panel of *VesselKnife*.

```
$ ./vk4console -v -o ? -a ?

Initializing algorithms:
  Cropping
  Inversion
  ColorMultiplication
    GammaRed=1.0
    GammaGreen=1.0
    GammaBlue=1.0
    Multiplier=1.0
(...)

  RadiiFromMultiscaleVesselness
    StdevMin=0.5
    StdevMax=2.5
    Scales=5
    StdevPower=2.0
  SmoothCenterlines
    Tau1=2.0
    Tau2=1.0
    Iterations=1
  SmoothRadii
    Tau1=2.0
    Tau2=1.0
    Iterations=1
(...)

  LevelSetCurves
    Iterations=800
    SmoothingIterations=5
    InitialDistance=1.0
    PropagationScaling=1.0
    Sigma=2.0
    Alpha=1.0
    Beta=1.0
    ThresholdLower=-1000.0
    ThresholdUpper=0.0
    ThresholdOutside=0.0
    ThresholdInside=255.0
    SigmoidMin=0.0
    SigmoidMax=1.0
    CurvatureScaling=1.0
    AdvectionScaling=1.0
    Rms=0.02
    SmoothingTimeStep=0.125
    SmoothingConductance=9.0
```

Below there is a script that calls vk4console which reproduces actions from the previous section “Step-by-step example”. The script is written for Linux bash, however it can be easily adopted for Windows batch files.

The last call is to vkconvert program, which converts file with vessel tree saved in VesselKnife’s native format to the STL file.

```
#!/bin/bash

./vk4console -i bifurkacja.nii.gz -o vesselness.nii -a Vesselness
-p Method=Fragi -p NumberOfScales=5 -p StdevMin=3 -p StdevMax=10

./vk4console -i vesselness.nii -o thresholded.nii -a Thresholding
-p Thresholding -p Min=32

./vk4console -i thresholded.nii -o holefilled.nii -a HoleFilling
-p Object=white

./vk4console -i holefilled.nii -o skeleton.nii -a SkeletonFromBinary

./vk4console -i skeleton.nii -o centerline.txt -a TreeFromSkeleton

./vk4console -i bifurkacja.nii.gz -t centerline.txt -o radii.txt
-a RadiiFromMultiscaleVesselness -p Scales=5 -p StdevMin=1 -p StdevMax=10

./vk4console -t radii.txt -o smoothed1.txt -a SmoothCenterlines
-p Iterations=3

./vk4console -t smoothed1.txt -o smoothed2.txt -a SmoothRadii
-p Iterations=3

./vkconvert -m tree2stl -i smoothed2.txt -o smoothed.stl
```