

TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Obtaining geometries from CT scans of arteries to simulate the fluid flow of blood

TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació

AUTOR: Marta Lorente Muñoz

DIRECTOR: Gerber van der Graaf

DATA: 21 de novembre de 2011

Títol: Obtaining geometries from CT scans of arteries to simulate the fluid flow of blood

Autor: Marta Lorente Muñoz

Director: Gerber van der Graaf

Data: 21 de novembre de 2011

Resum

Aquest document conté informació sobre la tècnica VMTK, la qual s'utilitza per obtenir geometries de segments vasculars a partir d'imatges 3D creades amb aparells mèdics en els diferents hospitals i centres de radiologia. La tècnica VMTK es basa en programació lliure i permet tenir la màxima transparència pels científics que treballen amb aquests tipus de geometries per tal d'aconseguir estudis i resultats de més qualitat. Tot el procés desenvolupat serveix per avançar en àrees de programació, medicina i computació de fluids, entre altres.

Title: Obtaining geometries from CT scans of arteries to simulate the fluid flow of blood

Author: Marta Lorente Muñoz

Director: Gerber van der Graaf

Date: November, 21st 2011

Overview

This document contains information about the technical VMTK, which is used to obtain geometries of vascular segments from 3D images created with various medical devices in hospitals and radiology centers. The technique is available under Free and Open Source Software (FOSS) licences and provides a maximum transparency to the scientists who work with that to get studies and results of higher quality. The whole process serves to advance in areas such as programming, medicine, computation fluid, among others.

ÍNDEX

VOLUM 1

INTRODUCCIÓ.....	1
CAPÍTOL 1. INFORMACIÓ HOSPITALÀRIA I RADIOLÒGICA.....	2
 1.1. Sistema d'Informació Hospitalària.....	2
 1.2. Sistema d'Informació Radiològica.....	2
1.2.1. Angiografia.....	2
1.2.1.1. Angiografia ionitzant.....	3
1.2.1.1.1. Obtenció d'imatges amb raigs x.....	3
1.2.1.1.1.1. Tomografia Axial Computada.....	3
1.2.1.1.2. Obtenció d'imatges amb raigs γ.....	4
1.2.1.1.2.1. Tomografia per Emissió de Positrons.....	4
1.2.1.1.2.2. Tomografia per Emissió de Fotons Individuals.....	5
1.2.1.2. Angiografia no ionitzant.....	5
1.2.1.2.1. Obtenció d'imatges amb ultrasons.....	5
1.2.1.2.2. Obtenció d'imatges amb ressonància magnètica.....	6
 1.3. Sistema de Comunicació i Arxiu d'Imatges.....	6
 1.4. Interfases de comunicació.....	6
1.4.1. HL7.....	6
1.4.2. DICOM.....	6

CAPÍTOL 2. MODELATGE DE SEGMENTS VASCULARS.....**8**

2.1. Introducció a la tècnica VMTK.....	8
2.1.1. Mòduls de visualització.....	8
2.1.1.1. Mòdul vmtkimageviewer.....	9
2.1.1.2. Mòdul vmtksurfaceviewer.....	9
2.1.1.3. Mòdul vmtkcenterlineviewer.....	10
2.1.1.4. Mòdul vmtkmeshviewer.....	11
2.1.1.5. Mòdul vmtkrenderer.....	12
2.1.2. Reproducció d'imatges 3D.....	12
2.1.2.1. Mòdul vmtkimagereader.....	12
2.1.2.2. Mòdul vmtkimagevoiselector.....	13
2.1.3. Selecció i extracció d'un segment vascular amb branques.....	13
2.1.3.1. Mòdul vmtklevelsetsegmentation.....	13
2.1.3.2. Mòdul vmtkmarchingcubes.....	19
2.1.3.3. Mòdul vmtkcenterlines.....	19
2.1.4. Realització de millores en l'extracció.....	20
2.1.4.1. Mòdul vmtksurfacesmoothing.....	20
2.1.4.2. Mòdul vmtksurfaceclipper.....	20
2.1.4.3. Mòdul vmtksubdivision.....	21
2.1.4.4. Mòdul vmtkflowextensions.....	21
2.1.5. Divisió del segment vascular.....	22
2.1.5.1. Mòdul vmtkbranchextractor.....	22
2.1.5.2. Mòdul vmtkbranchclipper.....	22
2.1.5.3. Mòdul vmtkcenterlinelabeler.....	23
2.1.6. Anàlisi geomètric de l'extracció.....	23
2.1.6.1. Mòdul vmtkcenterlineattributes.....	23
2.1.6.2. Mòdul vmtkbifurcationreferencesystems.....	24
2.1.6.3. Mòdul vmtkcenterlineoffsetattributes.....	24
2.1.6.4. Mòdul vmtkbifurcationvectors.....	24
2.1.6.5. Mòdul vmtkcenterlinegeometry.....	25

2.1.6.6. Mòdul vmtkbranchgeometry.....	26
2.1.7. Anàlisi.....	26
2.1.7.1. Mòdul vmtkbranchmetrics.....	26
2.1.7.2. Mòdul vmtkbranchmapping.....	28
2.1.7.3. Mòdul vmtkbranchpatching.....	28
2.2. Introducció al mallatge.....	29
2.2.1. Mallatge a partir de vmtk.....	30
2.2.1.1. Mòdul vmtkmeshgenerator.....	30
2.2.1.2. Mòdul vmtklineartoquadratic.....	31
2.2.1.3. Mòdul vmtkmeshscaling.....	31
2.2.1.4. Mòdul vmtkboundaryinspector.....	31
2.2.1.5. Mòdul vmtkmeshtetrahedralize.....	31
2.2.1.6. Mòdul vmtkmeshlinearize.....	31
2.2.2. Mallatge a partir del format .stl.....	32
2.3. Interconnexió entre els mòduls.....	35
CAPÍTOL 3. ACTUALITAT.....	37
3.1. L'actualitat de la tècnica VMTK i el mallatge.....	37
CONCLUSIÓ.....	41
BIBLIOGRAFIA.....	42
VOLUM 2	
ANNEXOS	
ANNEX 1: DICOM	
ANNEX 2: VMTKIMAGEREADER	
ANNEX 3: VMTKVOISELECTOR	
ANNEX 4: VMTKLEVELSETSEGMENTATION	
ANNEX 5: VMTKMARCHINGCUBES	
ANNEX 6: VMTKCENTERLINES	
ANNEX 7: VMTKSURFACESSMOOTHING	
ANNEX 8: VMTKSUBDIVISION	
ANNEX 9: VMTKSURFACLIPPER	
ANNEX 10: VMTKFLOWEXTENSIONS	
ANNEX 11: VMTKBRANCHEXTRACTOR	
ANNEX 12: VMTKBRANCHCLIPPER	
ANNEX 13: VMTKBRANCHCLIPPER (ELIMINAR BRANQUES)	
ANNEX 14: VMTKCENTERLINELABELER	
ANNEX 15: VMTKCENTERLINEATTRIBUTES	
ANNEX 16: VMTKCENTERLINEATTRIBUTES + VMTKBRANCHEXTRACTOR	
ANNEX 17: VMTKBIFURCATIONREFERENCESYSTEMS	
ANNEX 18: VMTKCENTERLINEOFFSETATTRIBUTES	
ANNEX 19: VMTKBIFURCATIONVECTORS	
ANNEX 20: VMTKCENTERLINEGEOMETRY	
ANNEX 21: VMTKBRANCHGEOMETRY	
ANNEX 22: VMTKMESHGENERATOR	
ANNEX 23: VMTKLINEARTOQUADRATIC	
ANNEX 24: VMTKMESHSCALING	
ANNEX 25: VMTKBOUNDARYINSPECTOR	
ANNEX 26: VMTKMESHTETRAHEDRALIZE	
ANNEX 27: VMTKMESHLINEARIZE	
ANNEX 28: MALLATGE A PARTIR DEL FORMAT STL	
ANNEX 29: HEXAHEDRAL MESHES	

INTRODUCCIÓ

L'objectiu d'aquest document és l'estudi de l'obtenció de geometries a partir d'imatges 3D de segments vasculars. Per arribar a l'objectiu, són diversos els aspectes a mirar.

En el primer capítol, es troba com es gestiona avui en dia la informació dels hospitals, com a primer lloc de contacte en l'obtenció de les imatges 3D de pacients. Per tant, també s'explica el tipus d'imatges que es poden aconseguir a partir de l'equip que les produceix. I, finalment, les característiques de format que han de mantenir les imatges per poder ser visualitzades en la majoria d'equips i programes.

En el segon capítol, i un cop amb les imatges a la mà, es troba l'estudi de la tècnica VMTK com a primer pas de tractament d'imatges i extracció del segment vascular escollit. A més a més, es tracta la creació de mallatges a partir de les geometries.

Al tercer capítol, i final, s'explica l'actualitat de la creació de mallatges i l'estudi de segments vasculars parlant en termes de precisió i qualitat.

CAPÍTOL 1. INFORMACIÓ HOSPITALÀRIA I RADIOLÒGICA

1.1. Sistema d'Informació Hospitalària

El Sistema d'Informació Hospitalària (*Hospital Information System- HIS*) es considera un dels principals sistemes financers, administratius i de registre mèdic dels hospitals.

Aquest sistema administra les operacions de l'hospital i les dades de cada pacient. A més a més, és l'encarregat de distribuir la informació de cada un d'ells pels diferents departaments dins un hospital.

1.2. Sistema d'Informació Radiològica

El Sistema d'Informació de Radiologia (*Radiology Information System- RIS*) és l'altre sistema financer, administratiu i de registre corresponent al departament de Radiologia dels hospitals. La seva funció és emmagatzemar les dades sobre exàmens i dels pacients a examinar.

1.2.1. Angiografia

La paraula angiografia prové dels termes *angio-* (vas sanguini) i *-grafía* (representació gràfica). El significat d'angiografia és la tècnica mèdica d'obtenció d'imatges de vasos sanguinis a partir d'un estudi radiogràfic.

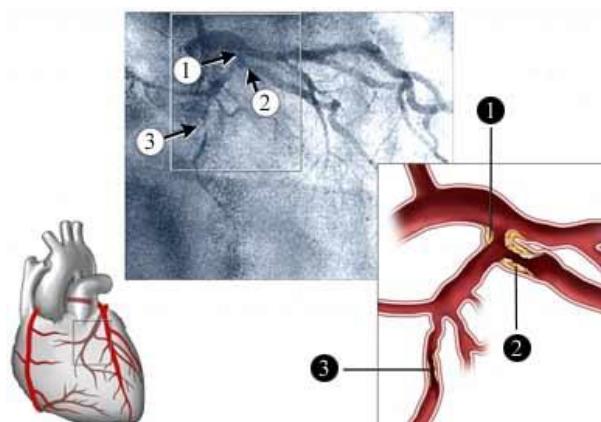


Fig. 1.1 Examen d'angiografia

Segons la part que s'estudia, la tècnica rep un nom més específic. L'estudi de les artèries s'anomena arteriografia, el de les venes es diu flebografia i el del sistema limfàtic, limfografia. No obstant aquesta diferenciació, el terme angiografia sortirà en tot el treball com a terme global.

La tècnica de l'angiografia pot ser invasiva o no invasiva pel cos humà. La modalitat invasiva requereix la introducció d'un catèter en un vas perifèric per poder afegir un colorant de contrast que permeti visualitzar el vas sanguini que es vol estudiar. En canvi, en la no invasiva no és necessària la col·locació d'un catèter ni de cap colorant. Aquesta modalitat és la millor alternativa.

Existeixen diferents mètodes dins la modalitat no invasiva per obtenir les imatges mèdiques, a continuació es fa un seguiment dels que fan possible obtindre imatges en 3D, ja que, a partir d'elles, es podran fer estudis de volum de vasos sanguinis.

Aquests mètodes es divideixen dependent de la radiació que radien. La classificació més directa és la que separa els de radiació ionitzant i els de radiació no ionitzant.

1.2.1.1. Angiografia ionitzant

Dins la angiografia ionitzant es troben els raigs x i els raigs γ .

1.2.1.1.1. Obtenció d'imatges amb raigs x

Els raigs X són una radiació electromagnètica d'origen l'òrbita electrònica, fonamentalment produïts per la desacceleració d'electrons.

La radiació de raigs X s'utilitza en la Tomografia Axial Computada (TAC) o (CT Scan en anglès) que permet l'obtenció d'imatges 3D.

1.2.1.1.1.1. Tomografia Axial Computada

La TAC és una tècnica desenvolupada per produir imatges mitjançant un tros de teixit en diverses direccions gràcies a un feix estret en forma de ventall. Cada direcció compren una dimensió i la imatge pot crear-se a partir de múltiples projeccions en diferents direccions.

Per poder captar bé els teixits, la TAC estableix uns nivells de densitat. Els equips moderns posseeixen una capacitat de 4096 tons de gris (12 bits) que

representen els nivells en UHs (Unitats Hounsfield). Un monitor pot representar fins a 256 tons de gris i l'ull humà en pot detectar 20. Donat que les intensitats dels teixits s'estenen dins un rang bastant estret de l'espectre total, és possible seleccionar una determinada finestra per representar el teixit d'interès.

L'aparell de TAC ha anat millorant amb el pas del temps. Existeixen les TAC de primera, segona, tercera, quarta, cinquena i sisena generació. Les diverses generacions depenen de la translació i rotació i són capaces de realitzar per obtenir les imatges. La més utilitzada és la TAC Helicoïdal- *multislice* que permet disminuir el temps d'exploració i augmentar la resolució per poder detectar lesions de menor grau.



Fig. 1.2 Tomografia Axial Computada Helicoïdal- *multislice*

1.2.1.1.2. Obtenció d'imatges amb raigs γ

Els raigs γ corresponen a una radiació electromagnètica d'origen nuclear produïts pel canvi d'un nucleó d'un nivell excitat a un altre de menor energia i en la desintegració de radioisòtops a partir de processos subatòmics com l'aniquilació d'un par positró- electró.

La radiació de raigs γ s'utilitza en la Tomografia per Emissió de Positrons (PET) i en la Tomografia Computada per Emissió de Fotons Individuals (SPECT).

1.2.1.1.2.1. Tomografia per Emissió de Positrons

La PET és una tècnica que combina la TAC junt amb un radioisòtop emissor de positrons que s'incorpora en una molècula metabòlicament activa i s'injecta al pacient. Quan el positró està injectat viatja uns mil·límetres mentre perd la seva energia cinètica. D'aquesta manera, quan xoca amb un electró es converteixen

en fotons d'aniquilació i es propaguen en direccions contràries. Un cop detectats es crea la imatge.

1.2.1.1.2.2. Tomografia Computada per Emissió de Fotons Individuals

La SPECT és una tècnica que també combina la TAC, però, en aquest cas, no es fa ús de positrons.

1.2.1.2. Angiografia no ionitzant

Dins l'angiografia no ionitzant es troben els ultrasons i la ressonància magnètica.

1.2.1.2.1. Obtenció d'imatges amb ultrasons

Els ultrasons són ones de so la freqüència de les quals es troba per damunt de la freqüència de l'oïda humana, és a dir, 20KHz. Les ones són creades gràcies a un transductor piezoelèctric contingut dins una sonda amb cristalls de quars. Al aplicar un corrent elèctric, els cristalls vibren i les produeixen.

Els ultrasons s'utilitzen en un equip portàtil amb un o diversos transductors. Les ones de so viatgen pel transductor fins a xocar amb el límit entre teixits. Un cop xoca, part de l'ona és reflectida i torna al transductor, mentre una altra part continua xocant amb el següent teixit i, així, successivament.

L'equip calcula la distància del transductor als diferents teixits gràcies a la velocitat del so i al temps que triga l'ona en tornar al seu punt d'origen. Durant el mateix procés, s'envien i es reben multituds d'ones.

Hi ha diversos equips dependent de com es vol obtenir la imatge. El Mode A està format només per un transductor que es mou però no capta cap moviment en el pacient; només hi ha un canvi en amplitud d'ona. El Mode B està format per dos transductors en moviment però tampoc capten el moviment del pacient; no obstant, afegeixen una escala de grisos per donar qualitat a la imatge. El Mode M es basa en la unificació de dos equips Mode B per poder captar moviment. El Mode Doppler utilitza l'efecte Doppler, és a dir, es rep un canvi en la freqüència ja que tan el transductor com la part a observar es mouen.

1.2.1.2.2. Obtenció d'imatges amb ressonància magnètica

La ressonància magnètica es crea quan s'alineen nuclis atòmics amb un camp magnètic constant i després es pertorben amb una altre camp magnètic extern.

La ressonància magnètica s'utilitza en un equip el qual s'anomena igual. Per tant, s'obtenen imatges per ressonància magnètica.

1.3. Sistema de Comunicació i Arxiu d'Imatges

El Sistema de Comunicació i Arxiu d'Imatges (*Picture archiving and Communication System- PACS*) consisteix en el funcionament d'un servei d'integració d'imatges mèdiques junt amb la informació clínica corresponen. Es pot definir com a un conjunt d'equips informàtics dedicats a l'adquisició, emmagatzematge, processat i comunicació d'imatges mèdiques digitals.

L'emmagatzematge de tota la informació es fa a dos nivells, un a curt termini i l'altre a llarg termini. Pels de curt termini s'utilitzen cabines de discs magnètics amb tolerància a falles. La capacitat mínima dels discs han de mantenir les dades almenys durant quinze dies. Per als de llarg termini, s'utilitzen sistemes òptics i de cintes magnètiques.

Generalment, la xarxa per a la transmissió de dades és una xarxa SAN que es basa en l'estàndard Fiber Channel, és a dir, d'alta velocitat i amb connectivitat entre servidors i dispositius d'emmagatzematge.

1.4. Interfases de comunicació

1.4.1. HL7

HL7 (Health Level Seven) és una interfase creada especialment per a la indústria de la salut. És capaç de connectar legalment els sistemes entre hospitals gràcies a un protocol estandarditzat de missatges. Tot va sorgir als EEUU quan tenien problemes a l'hora de comunicar-se i obtenir informació clínica de pacients entre diferents hospitals i entitats. D'aquesta manera, les dades es poden compartir, enviar i integrar amb un sol motor.

HL7 treballa amb un nombre d'estàndards: Conceptual, Documentació, Aplicació i Missatge. Aquest últim explica com la informació s'empaqueta i s'envia d'un lloc a un altre.

1.4.2. DICOM

DICOM (*Digital Imaging and Communications in Medicine*) és un protocol format per estàndards que permeten posar en comú un mateix format digital d'imatge mèdica i uns diàlegs normalitzats per poder intercanviar dades entre aparells de diferents fabricants.

El format DICOM permet unificar la imatge amb totes les dades del pacient com a atributs dins la capçalera del mateix arxiu [veure Annex 1].

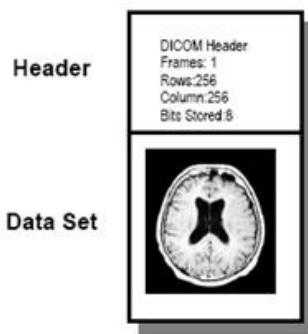


Fig. 1.3 Imatge DICOM (capçalera i dades d'imatge)

A partir d'una Declaració de Conformatitat, dos equips poden ser aptes per utilitzar el format DICOM.

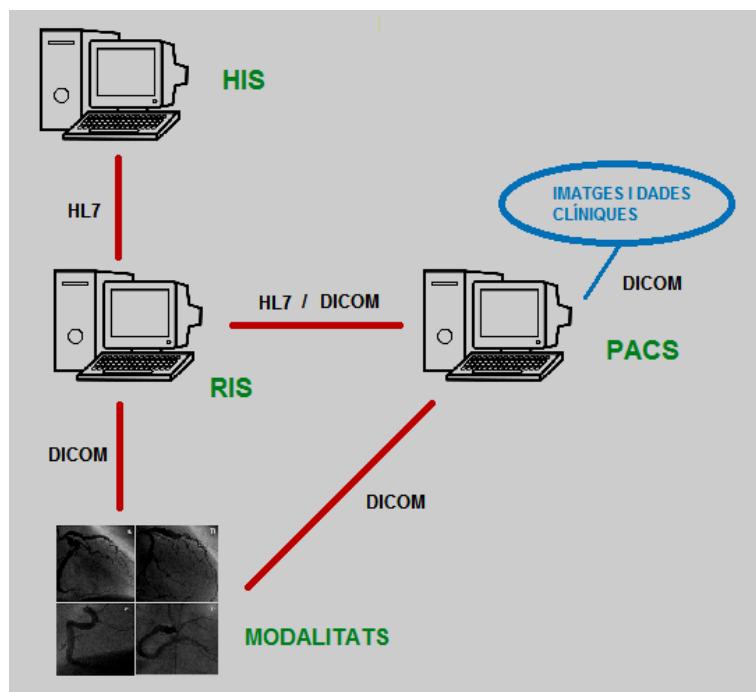


Fig. 1.4 Comunicació entre sistemes

CAPÍTOL 2. MODELATGE DE SEGMENTS VASCULARS

2.1. Introducció a la tècnica VMTK

Les sigles VMTK fan referència a unes eines de modelatge vascular: *Vascular Modeling Toolkit*, en anglès.

Aquesta tècnica consta d'un conjunt de llibreries i eines per reconstruir, analitzar geomètricament i generar mallatges a partir d'una imatge en 3D de segments vasculars.

VMTK es basa en software lliure i està compost per algoritmes implementats amb C++, com el VTK i l'ITK, i codis específics com el Python.

VMTK funciona com a un paquet oficial Debian, per aquest motiu s'executa en un sistema operatiu Linux- Debian.

VMTK és un pas inicial que al final de tot dóna lloc a que altres programes especialitzats en la reconstrucció i visualització de imatges en 3D facin més feines encara més precises i realistes. Per tant, s'ha d'entendre la tècnica VMTK com la inicialització per a futures aplicacions de modelatge.

Dins la tècnica VMTK s'utilitzen diferents mòduls per poder executar-la correctament. Aquests mòduls es troben dividits en el document depenen de la funció que realitzen. En el cas que s'executin dos o més, es fa ús d'un pype. Un pype fa d'unió entre ells, ja que comparteixen el mateix format de mòdul. L'escriptura del pype és - - pipe.

A continuació hi ha un llistat dels diferents mòduls separats per les funcions que duen a terme en general. Alguns d'ells necessiten altres programes de reconstrucció i visualització per poder estimar el resultat, per tant, només tenen codi escrit de referència, i no figures.

2.1.1. Mòduls de visualització

Els mòduls de visualització creen una finestra on es veuen les imatges i els volums extrets junt amb totes aquelles característiques produïdes en ells.

S'utilitzen quan és necessari visualitzar resultats o verificar que la feina s'està fent correctament, per tant no hi ha un ordre determinat d'ús. En qualsevol moment es pot afegir un mòdul de visualització i no implica cap canvi en el procediment que s'estigui realitzant.

Els mòduls més importants són: el vmtkimageviewer, el vmtksurfaceviewer, el vmtkcenterlineviewer, i el vmtkrenderer.

2.1.1.1. Mòdul vmtkimageviewer

El mòdul vmtkimageviewer permet visualitzar imatges en 3D a partir del format *vti*. Aquest format és en el que es transforma el format DICOM general per poder utilitzar les imatges amb la tècnica vmtk.

Dins els seus arguments d'entrada que pot mostrar, és important l'argument *array* ja que identifica la mostra.

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
disabled	Disabled	bool	1		0	disable execution and piping
i	Image	vtkImageData	1			the input image
ifile	ImageInputFileName	str	1			filename for the default Image reader
array	ArrayName	str	1			name of the array to display
renderer	vmtkRenderer	vmtkRenderer	1			external renderer
windowlevel	WindowLevel	float	2	[0.0, 0.0]		the window/level for displaying the image
display	Display	bool	1		1	toggle rendering
textureinterpolation	TextureInterpolation	bool	1		1	toggle interpolation of graylevels on image planes
continuouscursor	ContinuousCursor	bool	1		0	toggle use of physical continuous coordinates for the cursor
ofile	ImageOutputFileName	str	1			filename for the default Image writer

Fig. 2.1 Arguments d'entrada del mòdul vmtkimageviewer

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
o	Image	vtkImageData	1			the output image
xplane	PlaneWidgetX	vtkImagePlaneWidget	1			the X image plane widget
yplane	PlaneWidgetY	vtkImagePlaneWidget	1			the Y image plane widget
zplane	PlaneWidgetZ	vtkImagePlaneWidget	1			the Z image plane widget

Fig. 2.2 Arguments de sortida del mòdul vmtkimageviewer

2.1.1.2. Mòdul vmtksurfaceviewer

El mòdul vmtksurfaceviewer permet visualitzar imatges amb el format *vtp*, que corresponen a imatges on hi ha una superfície d'un volum escollit.

En aquest mòdul també és important l'argument *array* per visualitzar parts específiques de la superfície.

Quan s'utilitzen dos mòduls consecutius però només es vol visualitzar el resultat superficial d'un d'ells, es pot cridar aquest resultat. Per fer-ho és necessari escriure `-i @elnomdelsòlmoduldelresultat.o`, per exemple, `-i @vmtkbranchclipper.o`.

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
disabled	Disabled	bool	1		0	disable execution and piping
i	Surface	vtkPolyData	1			the input surface
ifile	SurfaceInputFileName	str	1			filename for the default Surface reader
renderer	vmtkRenderer	vmtkRenderer	1			external renderer
display	Display	bool	1		1	toggle rendering
opacity	Opacity	float	1	(0.0,1.0)	1.0	object opacity in the scene
array	ArrayName	str	1			name of the array where the scalars to be displayed are stored
scalarrange	ScalarRange	float	2		[0.0, 0.0]	range of the scalar map
legend	Legend	bool	1		0	toggle scalar bar
grayscale	Grayscale	bool	1		0	toggle color or grayscale
flat	FlatInterpolation	bool	1		0	toggle flat or shaded surface display
celldata	DisplayCellData	bool	1		0	toggle display of point or cell data
color	Color	float	3		[-1.0, -1.0, -1.0]	RGB color of the object in the scene
linewidth	LineWidth	int	1	(0.0,)	1	width of line objects in the scene
legendtitle	LegendTitle	str	1			title of the scalar bar
ofile	SurfaceOutputFileName	str	1			filename for the default Surface writer

Fig. 2.3 Arguments d'entrada del mòdul vmtksurfaceviewer

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
o	Surface	vtkPolyData	1			the output surface
oactor	Actor	vtkActor	1			the output actor

Fig. 2.4 Arguments de sortida del mòdul vmtksurfaceviewer

2.1.1.3. Mòdul vmtkcenterlineviewer

El mòdul vmtkcenterlineviewer permet visualitzar imatges amb el format *vtp*, que es basen en parts internes del volum escollit.

Els arguments *cellarray* i *pointarray* són els arguments clau per poder visualitzar les parts internes.

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
disabled	Disabled	bool	1		0	disable execution and piping
i	Centerlines	vtkPolyData	1			the input surface
ifile	CenterlinesInputFileName	str	1			filename for the default Centerlines reader
pointarray	PointDataArrayName	str	1			
cellarray	CellDataArrayName	str	1			
legend	Legend	bool	1		1	
renderer	vmtkRenderer	vmtkRenderer	1			external renderer
ofile	CenterlinesOutputFileName	str	1			filename for the default Centerlines writer

Fig. 2.5 Arguments d'entrada del mòdul vmtkcenterlineviewer

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
o	Centerlines	vtkPolyData	1			the output centerlines

Fig. 2.6 Arguments de sortida del mòdul vmtkcenterlineviewer

2.1.1.4. Mòdul vmtkmeshviewer

El mòdul vmtkmeshviewer permet visualitzar el mallatge de la superfície del volum en format *stl*.

Dins els seus arguments d'entrada que pot mostrar, és important l'argument *array* ja que identifica la mostra.

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
disabled	Disabled	bool	1		0	disable execution and piping
i	Mesh	vtkUnstructuredGrid	1			the input mesh
ifile	MeshInputFileName	str	1			filename for the default Mesh reader
renderer	vmtkRenderer	vmtkRenderer	1			external renderer
display	Display	bool	1		1	toggle rendering
opacity	Opacity	float	1	(0.0,1.0)	1.0	object opacity in the scene
array	ArrayName	str	1			name of the array where the scalars to be displayed are stored
scalarmrange	ScalarRange	float	2		[0.0, 0.0]	range of the scalar map
legend	Legend	bool	1		0	toggle scalar bar
grayscale	Grayscale	bool	1		0	toggle color or grayscale
flat	FlatInterpolation	bool	1		0	toggle flat or shaded surface display
ofile	MeshOutputFileName	str	1			filename for the default Mesh writer

Fig. 2.7 Arguments d'entrada del mòdul vmtkmeshviewer

Argument	Variable	Type	Length	Range	Default	Description
id	Id	int	1		0	script id
handle	Self	self	1			handle to self
o	Mesh	vtkUnstructuredGrid	1			the output mesh

Fig. 2.8 Arguments de sortida del mòdul vmtkmeshviewer

2.1.1.5. Mòdul vmtkrenderer

El mòdul vmtkrenderer permet visualitzar més d'un resultat en una mateixa finestra.

En aquest mòdul no hi ha arguments que caracteritzin la mostra, ja que és complementari als altres visualitzadors. Per tant, només es pot utilitzar si ja es fa servir un altre.

2.1.2. Reproducció d'imatges 3D

La reproducció d'imatges 3D és el principi de tot procediment. En aquesta part, les imatges es llegeixen d'un directori comú on es troben totes en format DICOM. Només és necessari nomenar la primera de les imatges i automàticament es llegeixen totes.

2.1.2.1. Mòdul vmtkimagereader

Aquest mòdul llegeix una imatge i les seves consecutives, totes en format DICOM (*dcm*), i les emmagatzema en un argument de tipus *vmtkImageData*. Aquest objecte fa referència a les dades d'imatge, les quals passaran a un argument de tipus *vmtkPolyData* un cop processada com a superfície d'un volum.

La visualització es mostra a l'Annex 2.

En primer lloc es detecta la tècnica vmtk i el mòdul. Després s'obre el directori i s'anomena la primera imatge. Finalment, es guarden les imatges en format *vti* i s'uneix tot el codi (amb la paraula *pipe*) amb el mòdul visualitzador.

2.1.2.2. Mòdul *vmtkimagevoiselector*

Amb aquest mòdul és possible l'elecció d'una part de la imatge en 3D. La seva funció és molt important ja que dóna l'opció d'escol·lir la part on es troba el volum d'interès que es vol extraure.

Per triar la regió s'utilitza la tecla *i* i s'arrossega el cub que apareix a la finestra. Un cop escollit, s'accepta amb la tecla *e*.

En aquest cas, s'obre l'arxiu anterior i es torna a guardar amb el mateix format, ja que encara no s'ha extret el volum.

La visualització es mostra a l'Annex 3.

2.1.3. Selecció i extracció d'un segment vascular amb branques

Els mòduls següents s'utilitzen per triar el volum desitjat i poder separar-lo de la imatge 3D inicial com a superfície.

2.1.3.1. Mòdul *vmtklevelsetsegmentation*

Aquest mòdul permet seleccionar del volum que es vol estudiar. Conté uns paràmetres per poder escollir-lo de la manera més acurada possible. Tot i ser triat, quan acaba el procediment es continua tenint una imatge, i no una superfície.

En primer lloc, el que fa el mòdul es pot iniciar de cinc maneres diferents: Fronts de xoc, Marxa ràpida, Llindar, isosuperfície i Punt.

Amb Fronts de xoc es tria un punt d'origen i un altre de destí per tal que el volum aparegui en aquesta direcció. Aquest mètode inicial detecta el primer canvi d'intensitat d'imatge i mostra la regió dins d'aquesta intensitat.

La Marxa ràpida és un procediment igual que Fronts de xoc, però detecta més intensitats i, per tant, l'elecció no és perfecta a l'hora de delimitar la part interessada.

El Llindar, la isosuperfície i el Punt, són mètodes que es basen en els píxel i subpíxels entre dos llindars; no obstant, el volum desitjat no apareix amb tan bones condicions com amb Fronts de xoc.

Per tant, Fronts de xoc és la base del procés d'obtenció de volums.

Finalment, existeixen també quatre paràmetres que depenen del valor numèric que se'ls hi assigna produueixen més canvis en la superfície del volum o menys. Aquests paràmetres apareixen per la meitat del curs del mòdul una vegada ja està escollit el volum. Són el Nombre d'Iteracions, l'Escala de Curvatura, l'Escala de Propagació i l'Escala d'Advecció.

El Nombre d'Iteracions correspon al número de passos que executarà el mòdul per tal que la superfície del volum es vegi més o menys definida. El valor per defecte és 300, augmentant cap a més definició.

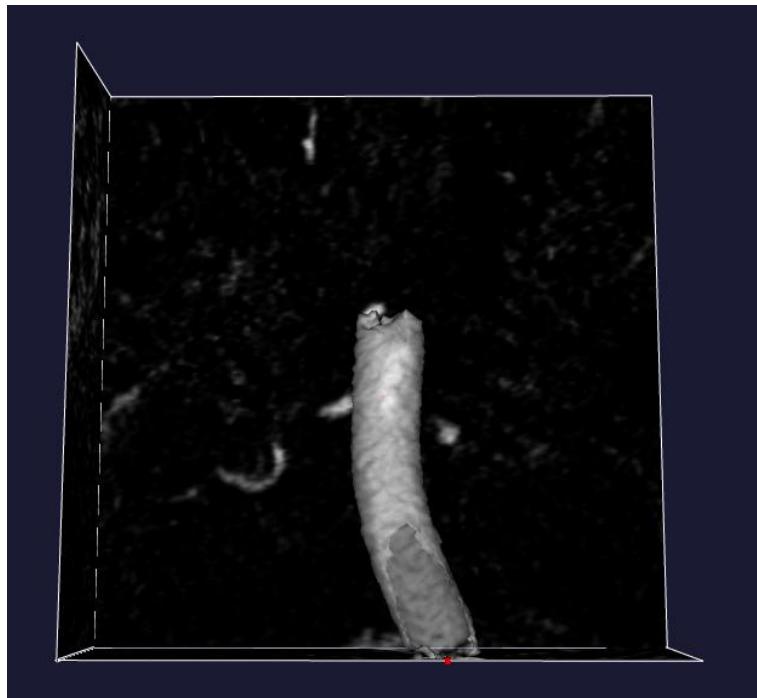


Fig. 2.11 Iteracions 300



Fig. 2.12 Iteracions 900

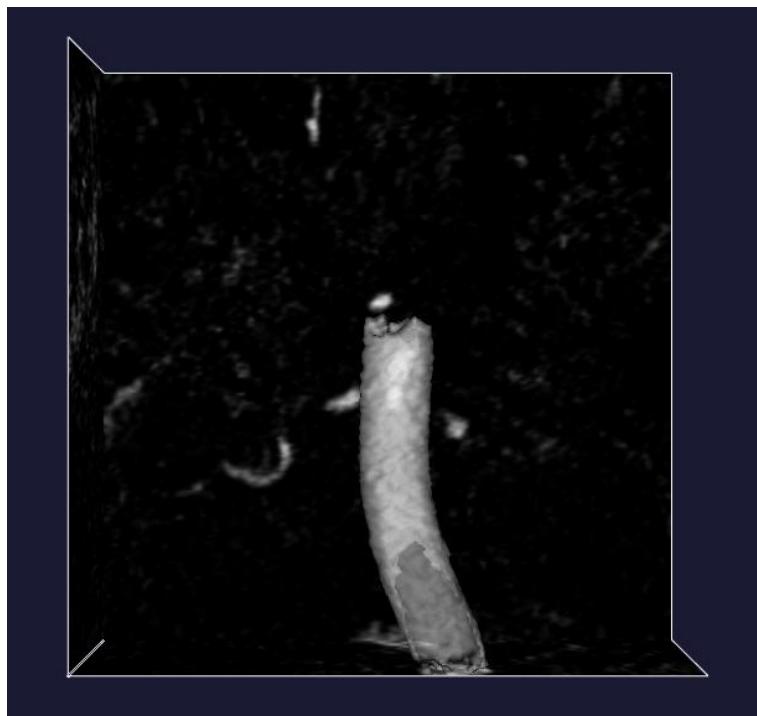


Fig. 2.13 Iteracions 1500

L'Escala de Curvatura elimina les zones corbes de la superfície del volum. Aquest paràmetre és 0 per defecte, el que vol dir que no elimina cap.

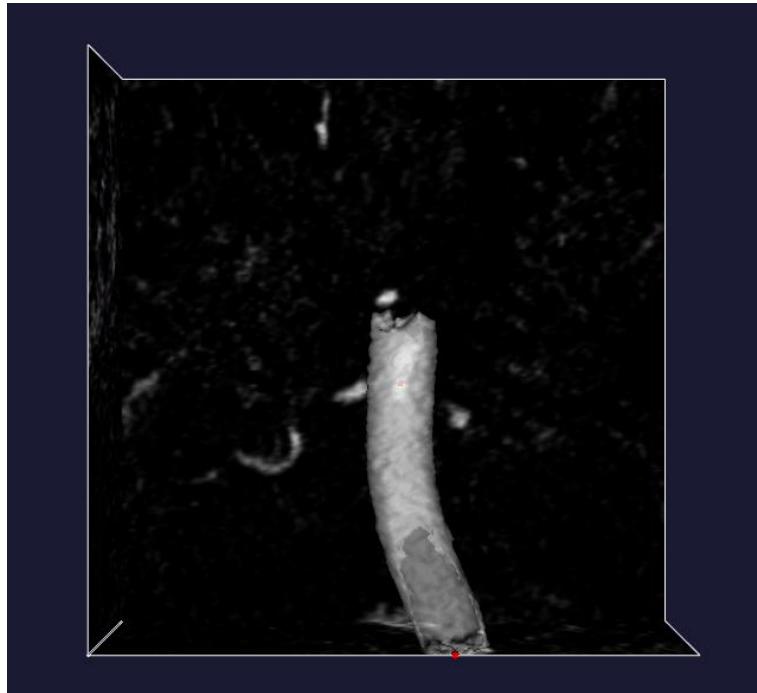


Fig. 2.14 Escala de Curvatura 0

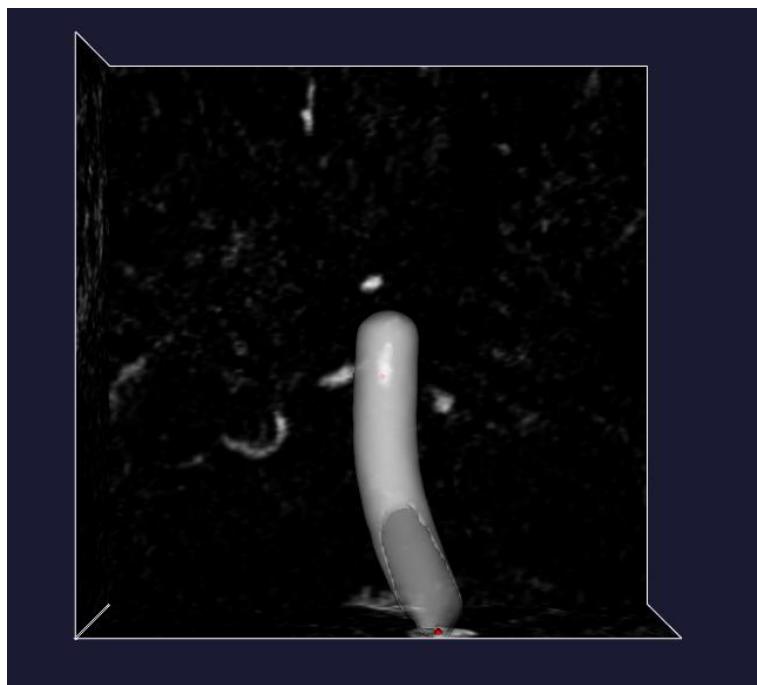


Fig. 2.15 Escala de Curvatura 10

L'Escala de Propagació correspon a la inflació que se l'hi vol donar al volum. El número per defecte és el 0.

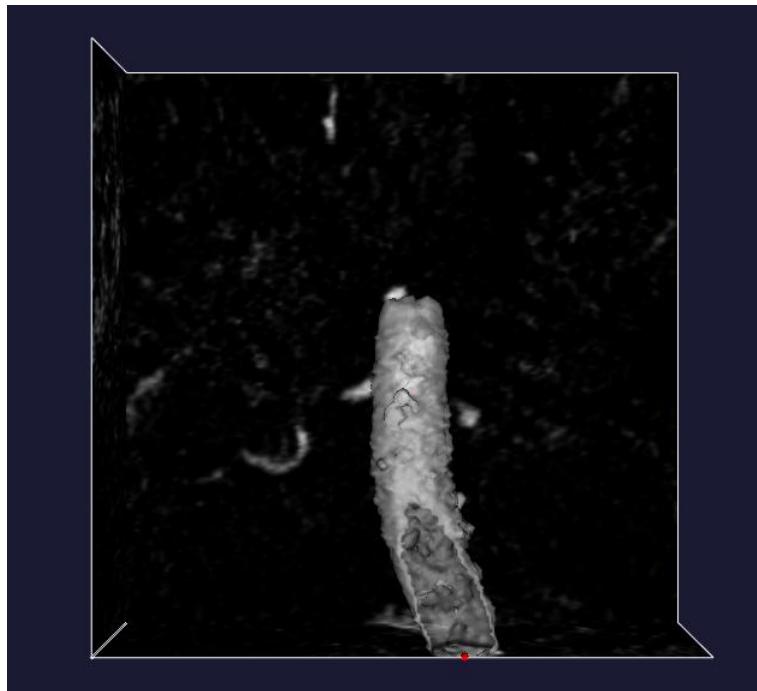


Fig. 2.16 Escala de Propagació 2

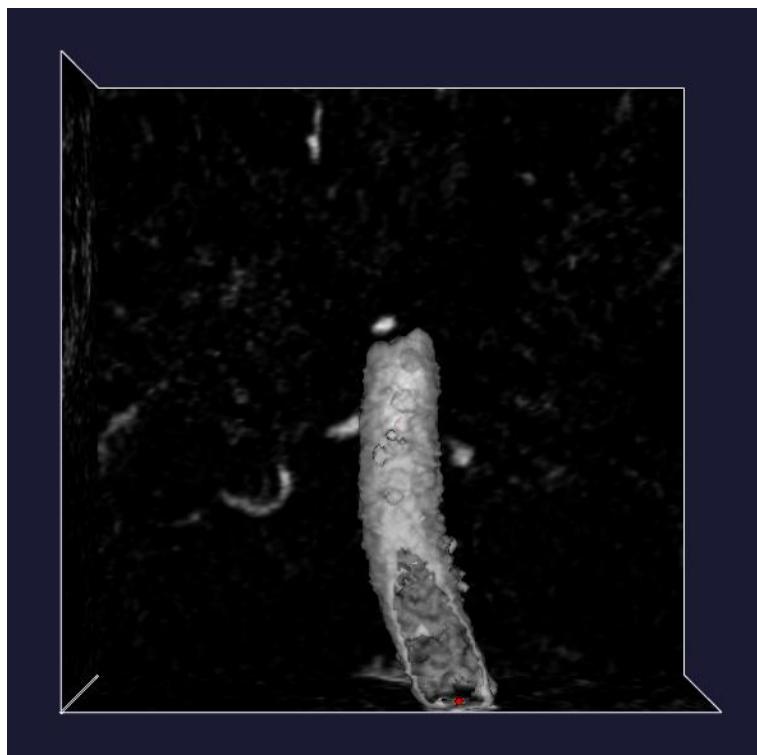


Fig. 2.17 Escala de Propagació 20

L'Escala d'Advecció regula la forma de les crestes. El número per defecte és l'1.

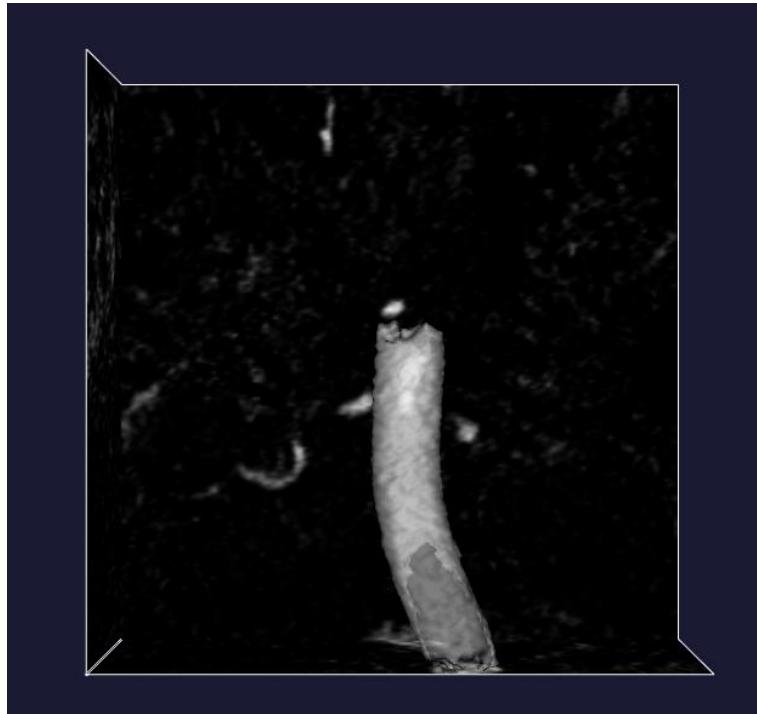


Fig. 2.18 Escala d'Advecció 10

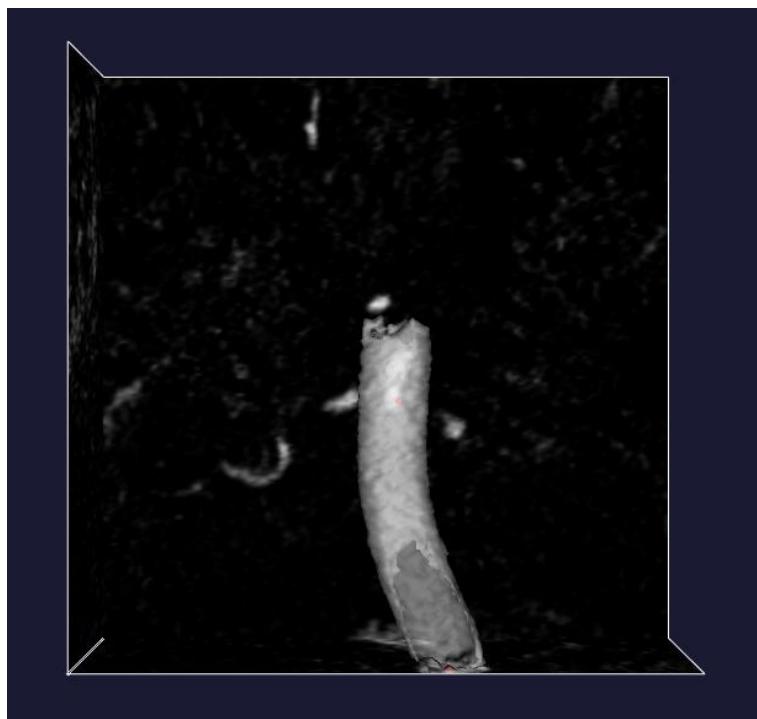


Fig. 2.19 Escala d'Advecció 20

La visualització sencera es mostra a l'Annex 4. Cada vegada que s'obre la finestra de visualització o es realitza alguna acció dins d'ella, s'ha de prémer la tecla **e** per finalitzar.

2.1.3.2. Mòdul *vmtkmarchingcubes*

En aquest mòdul es genera la superfície del volum escollit a partir de la imatge guardada anteriorment. Al crear-se una superfície, es passa al format *vtp*.

La visualització es mostra a l'Annex 5.

2.1.3.3. Mòdul *vmtkcenterlines*

Aquest mòdul s'utilitza per descriure la forma del volum escollit. Està determinat per un grup de trajectòries creades entre dos extrems. Les trajectòries passen pel centre d'unes esferes inscrites definides en el volum.

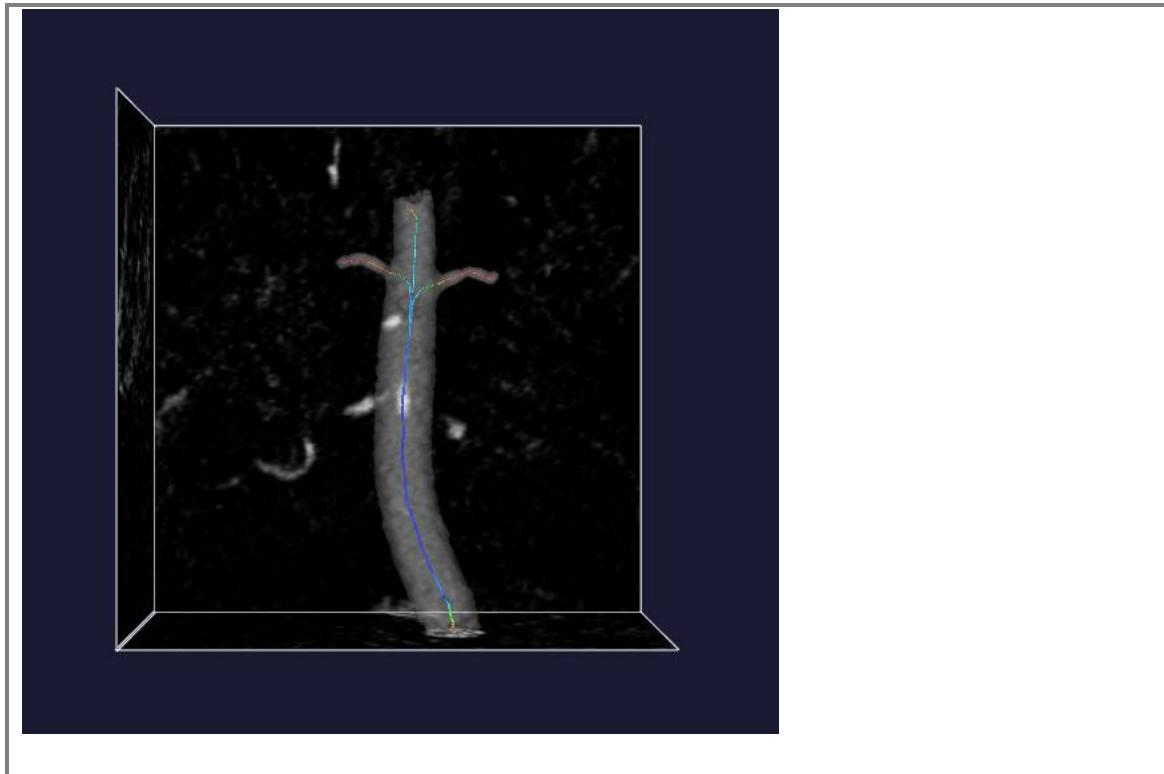
Les esferes són màximes quan no hi ha cap altra esfera que les contingui. Com més gran és el radi, el color característic és el blau; com més petit, el groc.

El conjunt d'esferes que es formen són la base del Diagrama de Voronoi, el qual s'ha d'indicar quan s'executa la visualització d'aquest mòdul. Si no s'indica res, només apareix la línia central de les esferes. Aquesta línia central pot aparèixer en blanc si no es s'escriu l'*array* que conté la informació.

La visualització es mostra a l'Annex 6.

En el cas següent, es fa ús del mòdul *vmtkrenderer* per poder visualitzar la imatge inicial junt amb la superfície escollida. Es crida l'*array* del mòdul *vmtksurfaceviewer*, per veure la línia central basada en el Diagrama de Voronoi.

```
vmtkimagereader -ifile voi.vti --pipe vmtkrenderer --pipe vmtkimageviewer  
--pipe vmtksurfacereader -ifile marching.vtp --pipe vmtksurfaceviewer -  
-opacity 0.25 --pipe vmtksurfacereader -ifile centerlinesauto.vtp --pipe  
vmtksurfaceviewer -array MaximumInscribedSphereRadius
```



2.1.4. Realització de millores en l'extracció

Un cop extret el volum del segment vascular es poden realitzar unes millores per donar-li un altre aspecte més acurat.

2.1.4.1. Mòdul *vmtksurfacesmoothing*

Aquest mòdul suavitza la superfície utilitzant l'algoritme de Taubin, el qual es basa en un filtre.

Hi ha dos paràmetres que controlen la suavitat, el Passabanda i les Iteracions. Com menor sigui el valor del passabanda, més suau serà la superfície. D'altra banda, amb un valor gran d'iteracions, més passos farà el mòdul i més definida es veurà la suavitat.

La visualització es mostra a l'Annex 7.

2.1.4.2. Mòdul *vmtksurfaceclipper*

Aquest mòdul permet obrir les parts tancades del volum ja que a partir de l'extracció el volum pot estar tancat.

Per escollir les parts a obrir, es clica la tecla *i* i apareix un cub. S'arrossega i es mou fins la posició adequada i es clica la tecla *espai* per tallar. Així tantes vegades com parts es vulguin obrir.

La visualització es mostra a l'Annex 8.

2.1.4.3. Mòdul *vmtksubdivision*

La superfície del volum del segment vascular es crea a partir de petits triangles que pràcticament són indetectables. No obstant, aquest mòdul permet dividir una altra vegada la superfície triangulada.

És una opció opcional, ja que moltes superfícies ja tenen el nombre correcte de divisions i no necessiten més; no obstant, quan la imatge inicial té poca resolució, el nombre final de triangles pot ser deficient i s'ha de subdividir.

El mètode que s'utilitza es diu Mètode Papallona (*Butterfly Method*), que correspon en afegir triangles sense perdre la continuïtat en la superfície.

La visualització es mostra a l'Annex 9.

2.1.4.4. Mòdul *vmtkflowextensions*

Aquest mòdul afegeix extensions cilíndriques en les parts obertes que són l'entrada i sortida del volum.

És important executar-lo després d'haver realitzat el mòdul *vmtkcenterlines*, ja que és on es designa una línia central que s'utilitza com a referència a les extensions.

Existeixen una paràmetres que donen forma a les extensions: *Adaptativelength*, *Extensionratio*, *Normalestimationratio* i *Interactive*. *Adaptativelength* correspon a la longitud de cada extensió, sempre proporcional al radi del perfil. *Extensoratio* és el factor de proporcionalitat. *Normalestimationratio* controla l'orientació de l'extensió dins el perill. *Interactive* significa que l'extensió sortirà per les parts obertes del volum automàticament. Per defecte té valor 0. Si es vol escollir manualment per on han de passar les extensions, ha de tenir el valor 1.

La visualització es mostra a l'Annex 10.

2.1.5. Divisió del segment vascular

El segment vascular junt amb les branques afegides en el mòdul vmtklevelsetsegmentation formen un conjunt que es pot dividir. El segment principal és el volum més gran que no té cap branca i els segments secundaris serien les branques adherides.

Dividir els segments fa possible un estudi més profund, ja que a la intersecció o bifurcació entre el volum principal i les branques és on hi ha canvis de direcció.

Per determinar cada bifurcació, s'utilitzen diferents números indicatius o punts de referència. El primer número es col·loca on el segment principal s'ajunta amb els secundaris, però com que a vegades és difícil saber quin és el principal, s'acaba posant en els llocs on hi ha canvi de segment. El segon número es col·loca on arriba la última esfera inscrita màxima de cada segment. Aquest valor és l'inici de la regió de bifurcació.

2.1.5.1. Mòdul vmtkbranchextractor

Aquest mòdul es basa en diferenciar les branques del segment vascular principal gràcies al treball realitzat amb el mòdul vmtkcenterlines, que permetia dividir el volum en esferes màximes inscrites.

Per poder realitzar la tasca, hi ha tres paràmetres associats: CenterlineId, que fa referència a l'esfera inscrita màxima última des de la qual començarà la branca; TractId, que identifica l'esfera màxima inscrita per sí sola; i, GroupId, que identifica el grup d'esferes inscrites màximes que pertanyen al mateix conjunt.

A més a més, hi ha el paràmetre Blanking que porta informació sobre si les esferes inscrites màximes formen part de la bifurcació o no. Si és 1, pertanyen a la bifurcació i si és 0, no.

Tots aquests paràmetres són emmagatzemats com una *cellarray*. Simplement amb el mòdul de visualització vmtkcenterlineviewer es poden veure els paràmetres.

La visualització es mostra a l'Annex 11.

2.1.5.2. Mòdul vmtkbranchclipper

Aquest mòdul s'utilitza seguit del mòdul anterior i permet dividir branques superficialment, un cop diferenciades.

Es pot obtenir la visualització del grup *GroupIds*.

La visualització es mostra a l'Annex 12.

De la mateixa manera es poden eliminar branques amb el paràmetre Insideout. En aquest cas, s'elimina el conjunt 1.

La visualització es mostra a l'Annex 13.

2.1.5.3. Mòdul *vmtkcenterlinelabeler*

Aquest mòdul es pot executar abans del mòdul vmtkbranchclipper per poder donar un nom específic a cada conjunt de *GroupIds*.

La visualització es mostra a l'Annex 14.

2.1.6. Anàlisi geomètric de l'extracció

L'anàlisi geomètric es basa en estudiar els angles i plans de les bifurcations i la curvatura i la torsió de les branques. Amb la majoria de resultats es necessita un altre programa de reconstrucció i visualització.

2.1.6.1. Mòdul *vmtkcenterlineattributes*

Aquest mòdul es basa en quantificar la posició angular d'un punt a la superfície del segment vascular i en el seu interior, a partir de les esferes inscrites màximes. Per tant, és un mòdul que necessita el mòdul vmtkcenterlines.

Els dos paràmetres atributius que s'utilitzen són: Abscissas i Normals.

L'atribut Abscissas mesura les distàncies en mm al llarg de la línia central de les esferes inscrites. Quan és 0 significa que és el primer punt des d'on es comença a mesurar.

L'atribut Normals mesura la posició angular i el canvi de direcció de la velocitat. Per fer això, s'estableix un sistema de referència, Frenet, en el qual, la normal va des de la línia de punts cap al centre del pla i el binormal és la normal del pla.

Per visualitzar els resultats, s'utilitza el mòdul vmtkcenterlineviewer i, el seu argument *pointarray*.

La visualització es mostra a l'Annex 15.

Un altres cas és quan s'uneixen els mòduls vmtkceterlineattributes i el vmtkbranchextractor. Així, les branques s'ajusten als valors d'abscisses resultants del primer mòdul.

La visualització es mostra a l'Annex 16.

2.1.6.2. Mòdul vmtkbifurcationreferencesystems

Aquest mòdul s'encarrega de crear un sistema de referència a la bifurcació escollida format per la normal i la binormal del pla on es troba.

El sistema de referència es defineix de la següent manera:

- L'origen de la bifurcació es defineix com el baricentre dels quatre punts de referència ponderat per la superfície de l'esfera inscrita màxima definida pels punts de referència inicials.
- La normal al pla de la bifurcació es defineix com la normal al polígon definit en les punts de referència de quatre computat en l'origen de la bifurcació.
- La binormal és el vector mitjana ponderada entre els vectors que apunta des del segon al primer punt de referència inicial de cada esfera inscrita màxima.

En aquest cas, el mòdul vmtkbranchextractor no té com argument de sortida *radiusarray*, no obstant, el mòdul vmtkbifurcationreferencesystems el necessita. Motiu pel que s'escriu @, que força l'argument per ser agafat d'entrada.

La visualització es mostra a l'Annex 17.

2.1.6.3. Mòdul vmtkcenterlineoffsetattributes

Aquest mòdul s'encarrega de desplaçar els atributs d'Abscisses y Normals del mòdul vmtkcenterlineattributes a la bifurcació de referència. El paràmetre referencegroupid, permet escollir el grup referent.

La visualització es mostra a l'Annex 18.

2.1.6.4. Mòdul vmtkbifurcationvectors

Aquest mòdul descriu com són de semblants, geomètricament, els diferents segments d'un volum vascular.

Cada segment, sigui principal o secundari, arriba a la regió de bifurcació amb una certa orientació. Aquesta orientació es divideix en dos components del pla de bifurcació. Per tant, existeixen dos angles que cada segment forma amb el sistema de referència de la bifurcació.

Si s'agafa com a condició inicial que la binormal té una orientació en el pla intern, cada segment es descriu per el seu angle intern amb la direcció binormal i el seu angle extern amb la normal.

L'arxiu creat sobre els vectors formen l'estructura d'un punt, per cada punt de referència de la bifurcació, més una sèrie d'altres dades:

- BifurcationVectors
- InPlaneBifurcationVectors
- OutOfPlaneBifurcationVectors
- InPlaneBifurcationVectorAngles
(l'angle entre el InPlaneBifurcationVectors i la binormal, en radians, de $-\pi$ a π ; 0 per un vector orientat com la binormal i positiu orientat a la dreta respecte la normal de la bifurcació)
- OutOfPlaneBifurcationVectorAngles
(l'angle entre el BifurcationVectors i el pla de bifurcació, en radians, positiu si el OutOfPlaneBifurcationVector té direcció de la normal de la bifurcació)
- BifurcationVectorsOrientation
(0 si el segment secundari no entra en la bifurcació i 1 si hi entra)
- GroupIds
- BifurcationGroupIds

Totes les dades anteriors són necessàries per extraure les característiques geomètriques en relació a la bifurcació.

La visualització es mostra a l'Annex 19.

2.1.6.5. Mòdul *vmtkcenterlinegeometry*

Aquest mòdul s'encarrega de computar la tangent, la normal i la binormal que constitueixen la curvatura i la torsió.

En aquest cas, no s'han de dividir les branques abans de poder executar el mòdul. No obstant, si que és necessari un filtre Laplaciana de suavitat, ja que el soroll que no s'aprecia sobre la línia superficial afecta en la realització del mòdul. La suavitat s'adjunta amb el paràmetre smoothing ja nomenat.

La suavitat es controla amb el paràmetre iterations i factor.

La visualització es mostra a l'Annex 20.

2.1.6.6. Mòdul vmtkbranchgeometry

Aquest mòdul proporciona les mateixes característiques geomètriques en totes les branques.

La visualització es mostra a l'Annex 21.

2.1.7. Anàlisi

Un cop realitzats tots aquests passos, s'arriba a un anàlisi. El punt inicial de l'anàlisi és aplanar i fixar la superfície, és a dir, fer que els triangles siguin aptes per formar una petita malla. Els mòduls que aplanen i fixen són: vmtkbranchmetrics, vmtkbranchmapping i vmtkbranchpatching. Per executar els mòduls es criden arrays i cellarrays múltiples.

No obstant, és necessari un programa més acurat de simulació de Dinàmica Computacional de Fluids (*Computational Fluid Dynamics- CFD*) per dur- a terme l'anàlisi.

En aquesta part de simulació, l'aplicació més comuna és el mapatge i l'ús de variables de dinàmica de fluids a la malla superficial del volum, com per exemple la tensió de cisallament (WWS) o l'índex de tall oscil·latori (OSI), un cop obtinguda la malla a partir d'un programa de simulació.

2.1.7.1. Mòdul vmtkbranchmetrics

Aquest mòdul crea dues matrius en cada branca. Les matrius són: AbscissaMetric, que es calcula a partir de l'abscissa longitudinal de les línies centrals; i, AngularMetric, que representa la coordenada circumferencial de la malla al voltant de les línies centrals.

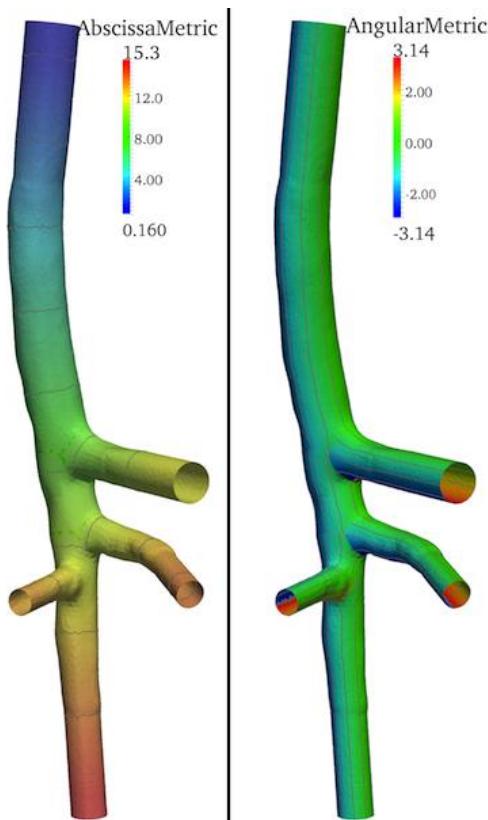


Fig. 2.20 Resultat del mòdul vmtkbranchmetrics

Per executar aquest mòdul es necessiten uns altres previs. A continuació es troba el codi inicial i, consecutivament, els que segueixen.

```
vmtk vmtkcenterlines -ifile marching.vtp --pipe vmtkcenterlineattributes --
pipe vmtkbranchextractor -ofile pas1.vtp

vmtk vmtkbifurcationreferencesystems -ifile pas1.vtp -radiusarray
MaximumInscribedSphereRadius -blankingarray Blanking -groupidsarray
GroupIds -ofile pas2.vtp

vmtk vmtkbranchclipper -ifile marching.vtp -centerlinesfile pas1.vtp -
groupidsarray GroupIds -radiusarray MaximumInscribedSphereRadius -
blankingarray Blanking -ofile pas_clipped.vtp

vmtk vmtkbranchmetrics -ifile pas_clipped.vtp -centerlinesfile pas1.vtp -
abscissasarray Abscissas -normalsarray ParallelTransportNormals -
groupidsarray Gropuids -centerlineidsarray Centerlinelids -tractidsarray
TractIds -blankingarray Blanking -radiusarray
MaximumInscribedSphereRadius -ofile pas_metrics.vtp
```

2.1.7.2. Mòdul vmtkbranchmapping

Aquest mòdul crea una funció harmònica que s'estén per la malla detectant les bifurcations.

Al final s'afegeix una array anomenada StretchedMapping.

```
vmtk vmtkbranchmapping -ifile pas_metrics.vtp -centerlinesfile pas1.vtp -referencesystemsfile pas2.vtp -normalsarray ParallelTransportNormals -abscissasarray Abscissas -groupidsarray GroupIds -centerlineidsarray CenterlineIds -tractidsarray TractIds -referencesystemsnormalarray Normal -radiusarray MaximumInscribedSphereRadius -blankingarray Blanking -angularmetricarray AngularMetric -abscissametricarray AbscissaMetric -ofile pas_mapping.vtp
```

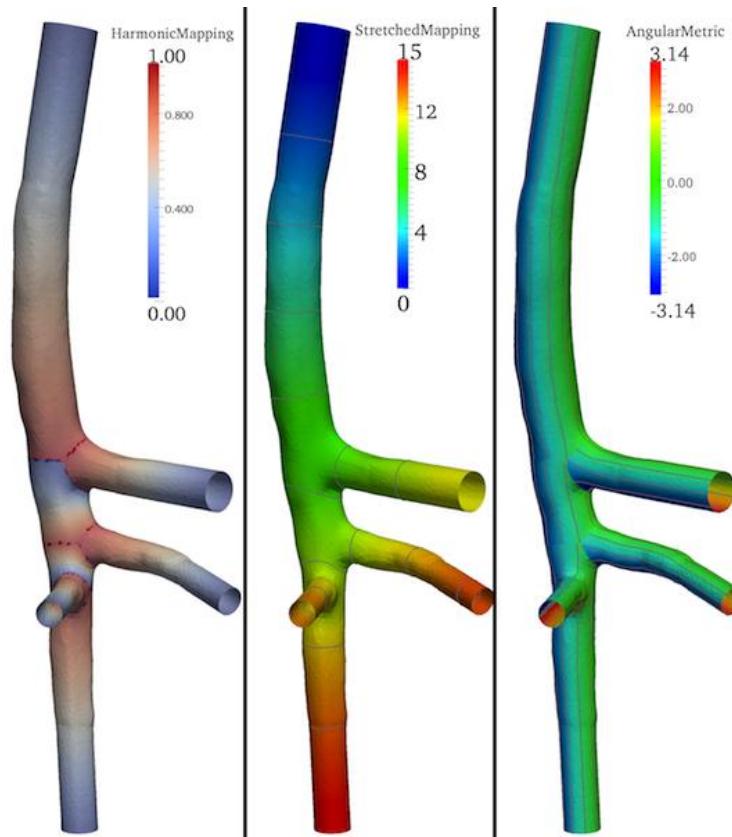


Fig. 2.21 Funció harmònica, StretchedMapping i AngularMetric

2.1.7.3. Mòdul vmtkbranchpatching

Aquest mòdul s'encarrega de tallar longitudinalment i de forma circular la superfície mallada seguint el contorn, informació del qual s'ha emmagatzemat en les array StretchedMapping i AngularMetric.

Amb els paràmetres longitudinalpatchsize i circularpatches es poden afegir les dimensions d'altura en mm per fixar la superfície en direcció longitudinal i el número de tall angular en l'interval de $[-\pi, \pi+]$.

```
vmtk vmtkbranchpatching -ifile pas_mapping.vtp -groupidsarray GroupIds  
-longitudinalmappingarray StretchedMapping -circularmappingarray  
AngularMetric -longitudinalpatchsize 0.5 -circularpatching 12 -ofile  
pas_paching.vtp
```

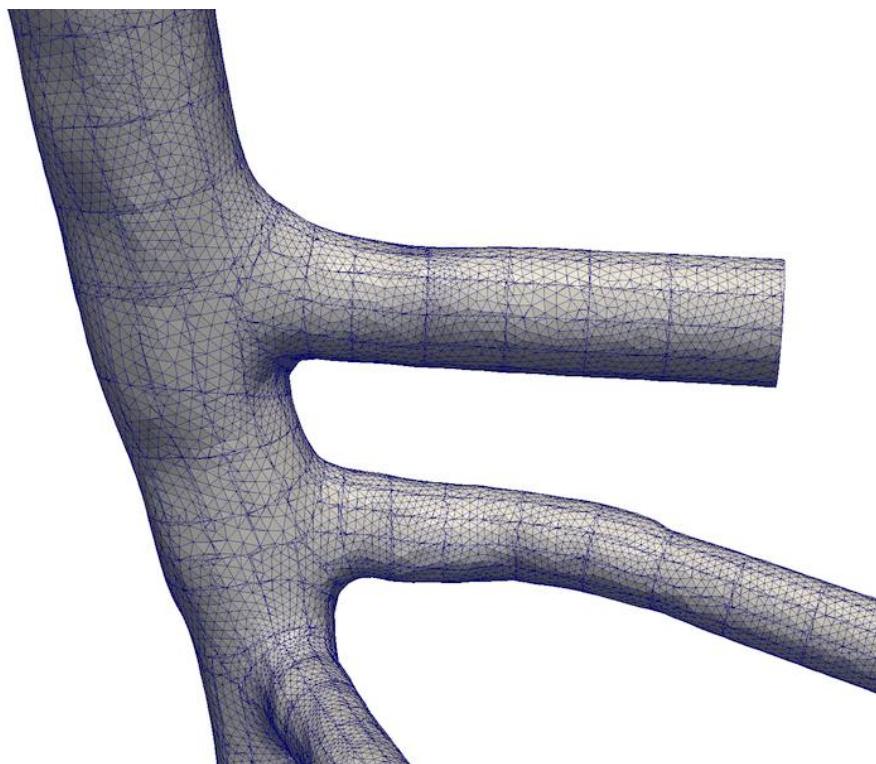


Fig. 2.22 Malla petita creada pel mòdul vmtkbranchpatching fixant la superfície

2.2. Introducció al mallatge

Tot i haver parlat de la malla petita que es pot crear aplanant i fixant els triangles de la superfície per fer un anàlisi de la superfície, existeixen dos mallatges diferents obtinguts de diverses maneres.

En primer lloc, s'utilitzen uns mòduls especialitzats en crear i retocar una malla; i, en segon lloc, es transforma la superfície a format stl, que correspon a un altre tipus de malla.

Amb aquest últim format també es poden fer estudis però fóra de la tècnica VMTK.

2.2.1. *Mallatge a partir de vmtk*

La creació d'una malla a partir de la tècnica VMTK es fa amb el mòdul generador de malles en format *vtu*. No hi ha relació amb la petita malla obtinguda anteriorment en format *vtp*, és a dir, com a superfície.

Al crear la malla, les dades són arguments de tipus *vtkUnstructuredGrid*.

2.2.1.1. *Mòdul vmtkmeshgenerator*

Aquest mòdul permet generar una malla a partir d'una superfície.



Fig. 2.23 Paràmetre edgelenght 3.5

La visualització es mostra a l'Annex 22.

2.2.1.2. *Mòdul vmtklineartoquadratic*

Aquest mòdul converteix els elements linears a quadràtics. La conversió es pot utilitzar per fer diferents estudis.

La visualització es mostra a l'Annex 23.

2.2.1.3. *Mòdul vmtkmeshscaling*

Aquest mòdul permet donar una escala en cm al mallatge.

La visualització es mostra a l'Annex 24.

2.2.1.4. *Mòdul vmtkboundaryinspector*

Aquest mòdul identifica els diferents límits del mallatge.

La visualització es mostra a l'Annex 25.

2.2.1.5. *Mòdul vmtkmeshtetrahedralize*

Aquest mòdul permet donar forma tetraèdrica a la malla per tal que els programes de simulació puguin detectar perfectament la malla.

En un sistema mallat és important la mida dels triangles (anomenats també facetes per alguns usuaris), ja que a menys triangles, més fàcil és analitzar el model ja que segueixen d'una forma més exacta la superfície 3D inicial.

La visualització es mostra a l'Annex 26.

2.2.1.6. *Mòdul vmtkmeshlinearize*

Aquest mòdul linealitza la malla preservant tots els elements.

La visualització es mostra a l'Annex 27.

2.2.2. *Mallatge a partir del format .stl*

Primerament, es passa del format *.vtp* al format *.stl* d'una forma senzilla amb la tècnica VMTK.

La visualització es mostra a l'Annex 28.

El format *.stl* (StereoLithography) és un arxiu estàndard dins l'àmbit de prototipat ràpid. Defineix la forma d'un objecte 3D gràcies a una mallatge format per triangles.

El format es delimita en aproximar-se a la superfície del cos de l'objecte i deixa de banda els punts, corbes i atributs. Únicament, es fixa en els vèrtexs i la direcció normal de cada triangle.

Quan es tracta d'una superfície plana, els triangles s'ajunten en quadrilàters, mentre que en superfícies corbes, s'afegeixen més triangles per minimitzar-los i poder resseguir la superfície sense perdre detall. En aquests casos, es parla de parts tetraèdriques o hexaèdriques si la simulació ho permet.

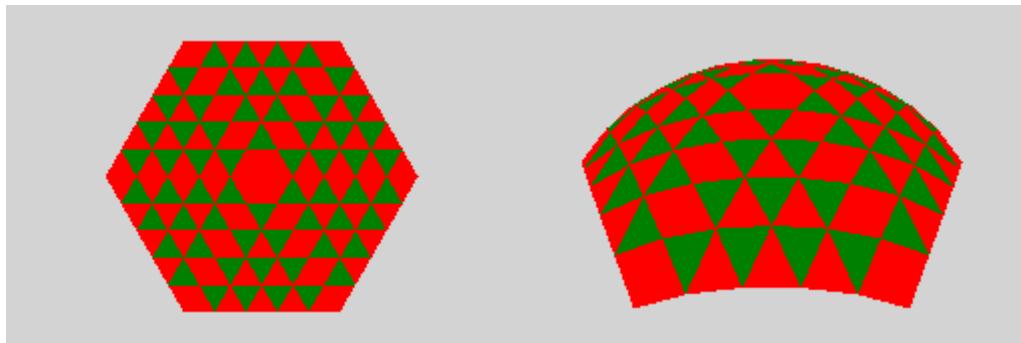


Fig. 2.24 Malla hexaèdrica simulant una corba

El format pot ser de dos tipus: binari i ASCII. El nombre de triangles i la qualitat del fitxer no varia entre els dos tipus, però per a un mateix número de triangles i de qualitat, el fitxer ASCII pot tenir deu o quinze vegades més pes que el binari.

Aquest format s'utilitza amb la majoria de programes de Disseny Assistit per Ordinador (CAD). En aquests programes es pot determinar el número de triangles que es necessiten i així fer un equilibri entre aquest número de triangles i el volum que pot ocupar un arxiu.

Per treballar amb els programes CAD, s'ha d'exportar un model de superfície en format *.stl*. Les conversions poden deixar errors estructurals i diferències

importants, així que els arxius han de ser comprovats per un software capaç de reparar els errors. Els més petits s'arreglen de forma automàtica, però els de més dimensió són feina de les persones que interactuen amb el software.

Un dels problemes és quan hi ha triangles adjacents que no comparteixen un vèrtex en comú. Això s'anomena la Regla Vèrtex a Vèrtex. Per a fer possible aquesta regla, se subdivideix el triangle inferior que es troba al costat dels adjacents, però pot causar la creació de forats en la malla.

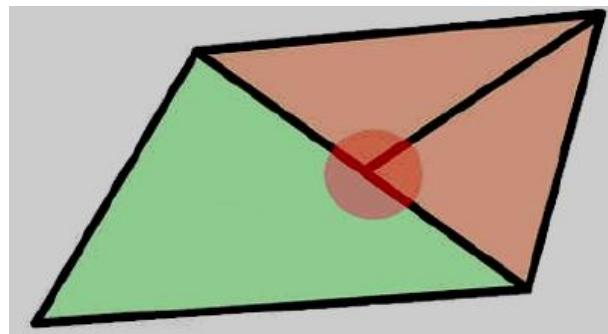


Fig. 2.25 Triangles adjacents amb vèrtex comú

Una possible solució a aquest fet és la reparació manual. El procediment es basa en esborrar les cares errònies i reconstruir-les utilitzant vèrtexs confrontants.

Un altre problema comú és trobar-se les normals invertides. Es detecten a partir de petits forats en la superfície. És un error degut a fer una incorrecta reparació dels triangles.

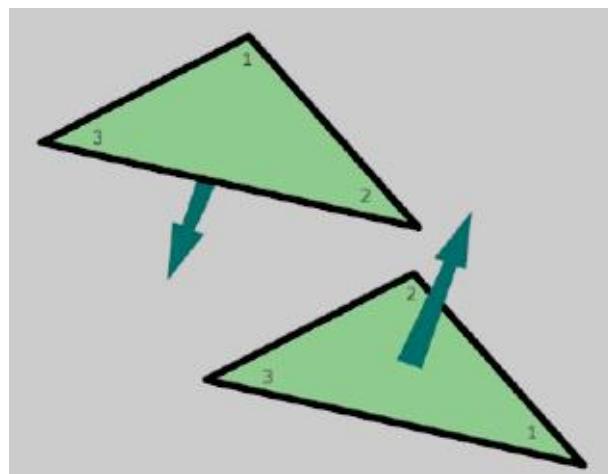


Fig. 2.26 Normals invertides

La solució es troba unificant les normals. Els programes tenen les eines necessàries per orientar-les perfectament.

Un tercer problema es genera quan es troben dobles cares i múltiples vores. En aquesta situació, si s'eliminen els duplicats i queden els originals ja està solucionat, però pot passar que siguin duplicats perquè el programa els reconeix com a duplicats però en realitat no pertanyen al mateix fragment de l'objecte. Llavors, abans d'eliminar els duplicats s'ha de fer un anàlisi per detectar el fragment d'origen per resoldar-los.

En relació als buits, forats, puntes i arestes obertes, tenen un tractament semblant. Els buits són menys visibles que els forats. Les puntes es poden comparar com una península que té les arestes obertes ens els extrems. Aquestes també són les vores d'un buit o forat.

Totes aquestes deformacions són degudes a una inapropiada reparació o una transformació al format inadequada o incomplerta.

Per solucionar els defectes s'utilitza una operació que els localitza, els tapa i els segella. No obstant, pot passar que les arestes es connectin de manera incorrecta i que apareixin cares aleatòries en els objectes, de manera que s'ha d'utilitzar amb molta precaució i mantenint la forma superficial de l'objecte.

A vegades es troben les cares no degenerades, cosa que no és crítica, però si afecta al volum de l'arxiu i pot causar confusió al programa d'anàlisi del mallatge. Els tipus de cares degenerades comporten que els vèrtexs siguin col-lineals o es transformin en col-lineals quan s'exporta l'arxiu.

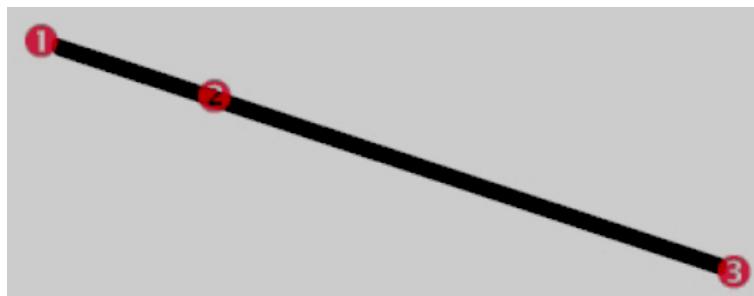


Fig. 2.27 Cares no degenerades

La única solució possible en aquest cas és diagnosticar i verificar que a la importació i l'exportació no es perd informació. Amb un canvi en la configuració per crear el mallatge i fixar els triangles hauria de ser suficient.

2.3. Interconnexió entre els mòduls

En aquest apartat es mostra com els diferents mòduls explicats durant tot el document es comuniquen entre ells o uns són de prèvia execució a d'altres.

En teoria, depenent de la funció per la qual es fa servir la tècnica VMTK i els possibles mallatges, s'ha de buscar el camí més idoni per arribar als millors resultats.

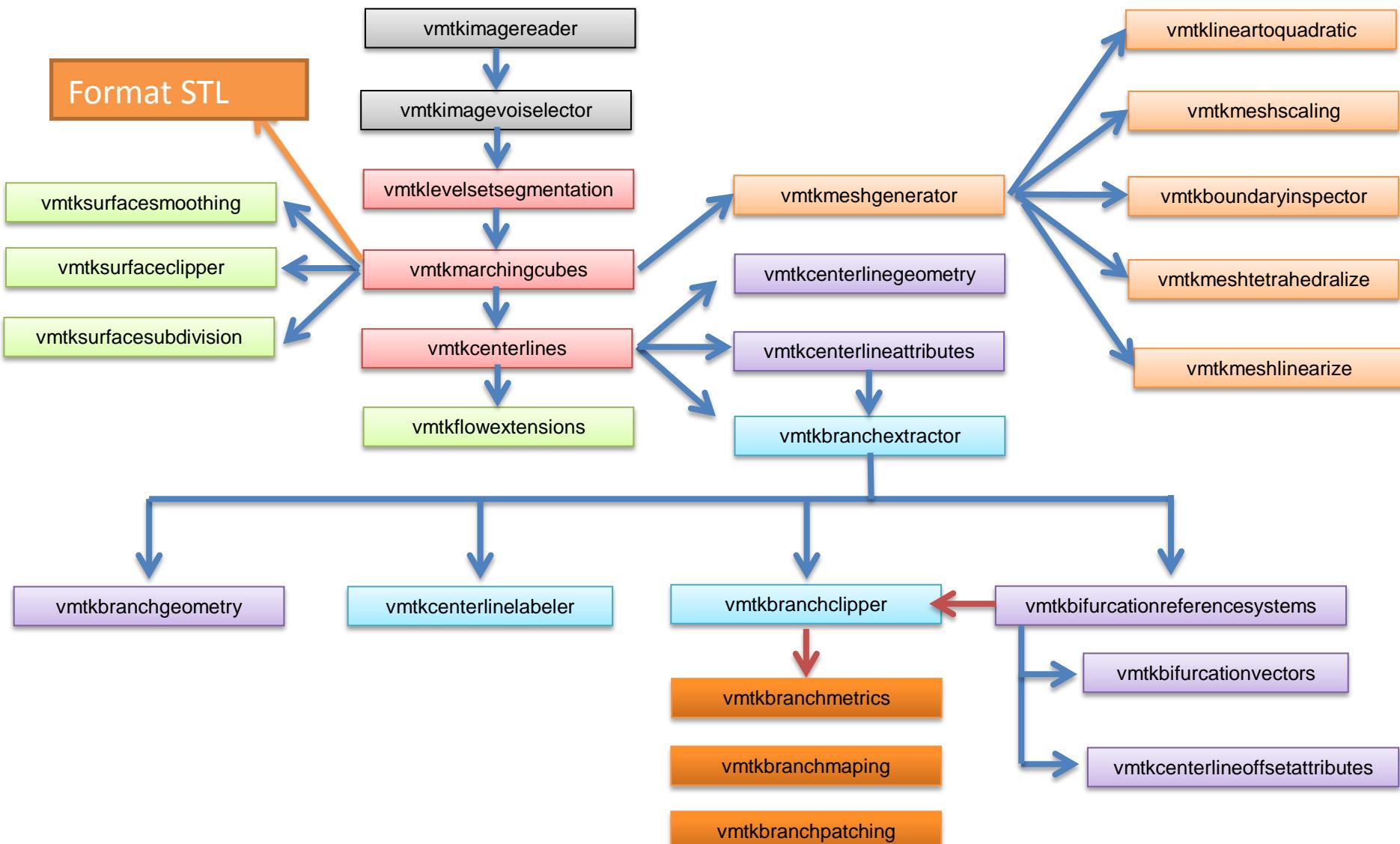
Les fletxes addicionals mostren direccions d'execució realitzades en el document que agafen mòduls dels diferents apartats, ja que les seves funcions són necessàries per dur a terme l'objectiu.

En relació als mòduls, en primer lloc, i de color blau, es troben els corresponents a l'apartat de reproducció d'imatges 3D. Pas previ a tot el procés.

Seguint les fletxes s'arriba als mòduls encarregats de la selecció i extracció d'un segment vascular amb branques, en color rosat. En aquest punt, es troben cinc camins de colors diferents.

Un permet realitzar millores en l'extracció, en color verd. Un altre permet passar directament al mallatge en format *stl*, en color taronja. Cap a la part dreta, el camí cap a la creació d'un altre mallatge de format *vtu*, amb els seus respectius mòduls, també en taronja. El següent porta als mòduls que divideixen el segment vascular, en blau clar. A partir d'aquest, es procedeix als mòduls que aplanen i fixen la superfície per fer una petita malla i un ànalisi estadístic, en color taronja fort.

Finalment, el cinquè camí, es basa en els mòduls d'ànalisi geomètric de l'extracció.

Tabla 2.1 Interconnexió entre mòduls

CAPÍTOL 3. ACTUALITAT

3.1. L'actualitat de la tècnica VMTK i el mallatge

En l'actualitat, les tècniques de simulació computacional de fluid han avançat àmpliament. Els diferents programes d'aquest àmbit intenten cada vegada més la creació de malles de tipus tetraèdriques precises i perfectes per poder analitzar-les amb tot detall. Gràcies a aquest tipus de mallatge, com s'ha indicat en el document, la complexitat dels segments vasculars disminueix i, aquest fet, garantitza un estudi i uns resultats òptims.

L'origen del procés es remunta a la tècnica VMTK que fa de pont a les noves metodologies de mallatge gràcies als arxius en format *stl*.

Una metodologia actual s'ha arribat a determinar més enllà de la forma tetraèdrica. Té l'objectiu de donar forma hexaèdrica al mallatge d'una superfície geomètrica triangular amb bifurcations. Aquest tipus de mallatge garanteix, encara més, una millora en l'aplicació de les tècniques de simulació [veure Annex 29].

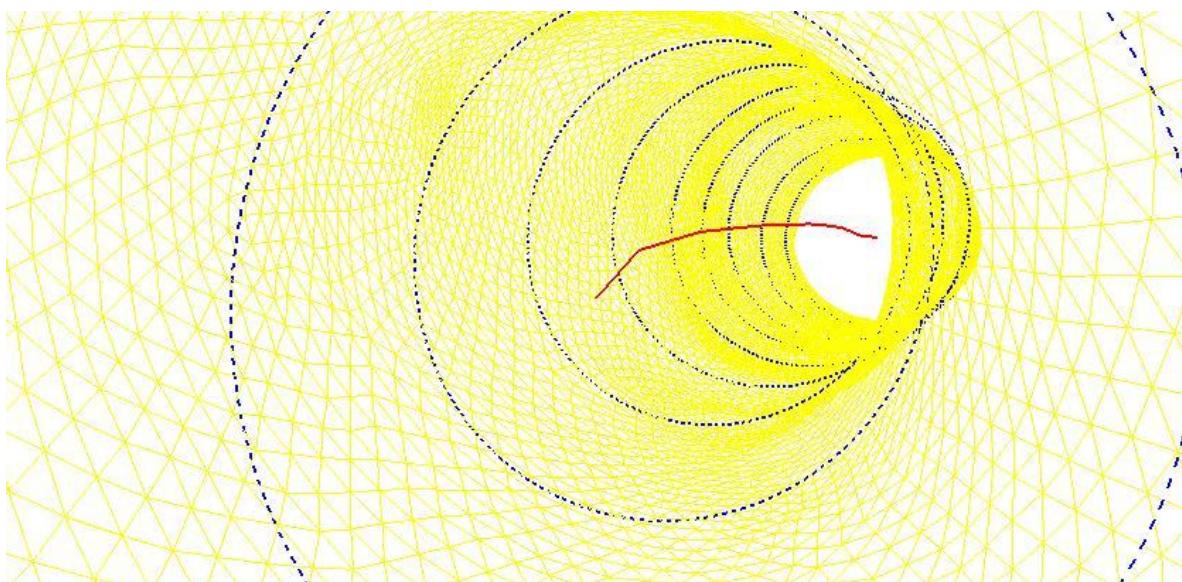


Fig. 3.1 Superfície triangulada d'un segment vascular

Un cop generada i reparada una malla en format *stl*, obtinguda a partir de la tècnica VMTK amb origen a les imatges d'angiografia, es crea una malla hexaèdrica generada utilitzant pyFormex. PyFormex és un software per crear, manipular i transformar models d'estructura geomètrica.

Primerament, s'orienta la superfície triangulada respecte el seu centre de gravetat per tal d'orientar el pla de la bifurcació perpendicularment a l'eix z. A partir d'aquí, l'usuari pot escollir uns punts en el pla x-y al llarg de cada costat de les branques. Aquests punts, junt amb un punt en el centre de la bifurcació formen la tipologia del segment vascular i s'utilitza per orientar i posicionar una sèrie de semiplans que es van apropiant al centre de la bifurcació.

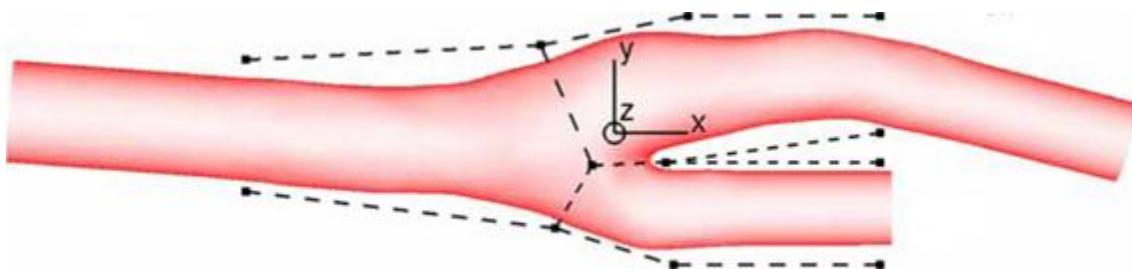


Fig. 3.2 Orientació del segament vascular en l'eix z

Un cop orientada la superfície, es fan una talls transversals en cada semibranca que prospera a partir de la bifurcació. Els talls en cada semibranca es divideixen en piles, una pila a la semibranca de dalt i l'altra, a la de baix. Les dues sobresurten de la bifurcació i s'ajunten amb la tercera branca a partir d'un tall de tres cares.

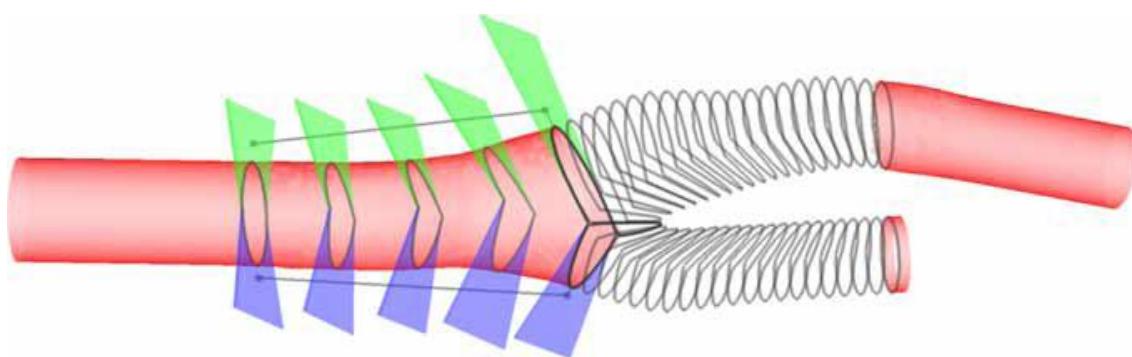


Fig. 3.3 Divisió de la superfície en talls transversals

Cada tall transversal es divideix en 13 punts que fan de marca per a la creació d'estries longitudinals que segueixen la corba del model. Aquestes estries corbes (Bezier) mantenen la continuïtat de la superfície i determinen la resolució longitudinal que depèn de la seva densitat.

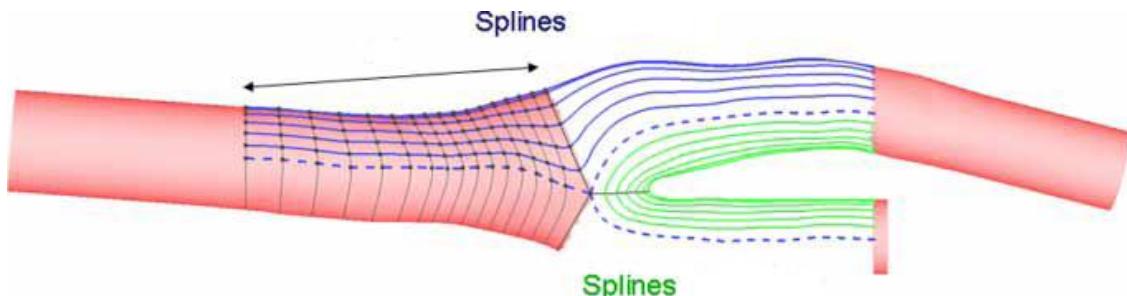


Fig. 3.4 Estries Bezier que determinen la resolució longitudinal. Les línies de punt es tallen en el centre de la bifurcació.

Els vèrtexs dels triangles que formen part d'un mateix tall transversal o longitudinal s'utilitzen per a controlar la transformació bicúbica, on la superfície original es plasma en una superfície cúbica.

La malla final s'obté mitjançant l'escombrada de les seccions apilades al llarg de les tres branques.

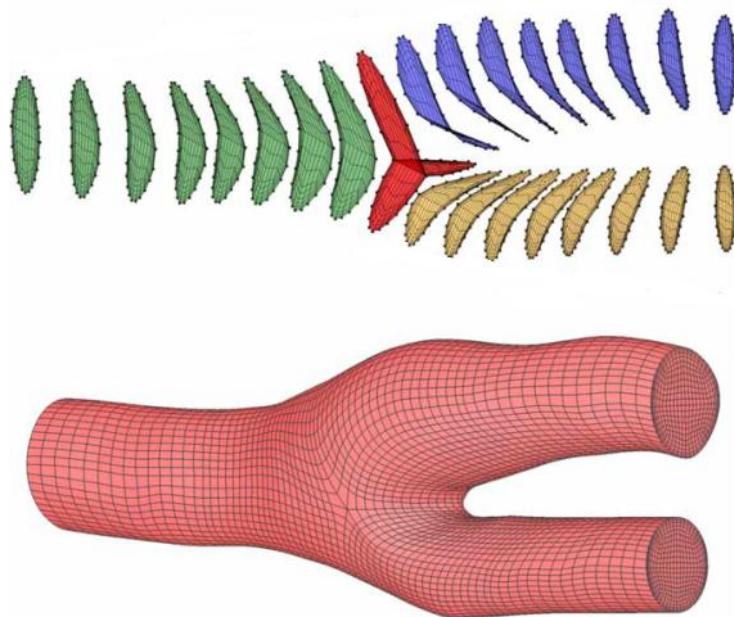


Fig. 3.5 Seccions transversals i mallatge.

En resum, els tall transversals fan de base per a les seccions a escombrar i, els tall longitudinals, fan de base per unir els vèrtexs dels triangles i mantenir la continuïtat del model.

La relativa diferència de volum i àrea del mallatge final respecte la superfície original es pren com a una mesura de precisió geomètrica. Es quantifica mesurant la distància entre les dues superfícies. Els valors positius corresponen als punts mesurats fóra del model *stl*.

El paràmetre *equiangle skew* (Qeas) correspon a la qualitat del mallatge, sent una mesura normalitzada d' asimetria des de 0.0, o perfecte equí angulació; fins 1, o màxima distorsió.

La metodologia explicada compta amb un resultat visible en forma de mallatge d'estructura hexaèdrica generat.

La nova malla generada a partir de la metodologia és més robusta i fiable que les seves anteriors, per tant, és una proposta amb molt de futur i expectatives.

L'actual desenvolupament innovador redueix el valor numèric d'errors i el temps computacional associat a fer els anàlisis. A més a més, s'amplia el nombre de segments vasculars que es poden analitzar ja que abans no optaven a tanta precisió.

CONCLUSIÓ

Com a conclusió d'aquest treball faig rellevant el fet que per realitzar tot el procés per obtenir una extracció i un estudi d'una imatge 3D es necessiten molts passos i s'arriba a un resultat gràcies a que tots s'utilitzen adequadament i no es prescindeix de cap: l'equip radiològic, la imatge, el software, la tècnica d'extracció i mallatge i el programa d'anàlisi. Per altra banda, no tot són procediments o passos en aquest document.

Mirant una mica més enllà es troben les persones.

Darrera de tot estudi, com l'iniciat en el document, les persones tenen un dels papers més importants: es necessita l'aportació d'informació i coneixement personal per realitzar tot el procés des de l'obtenció d'una imatge en 3D fins arribar al mallatge de l'extracció del seu volum i al seu estudi.

Tot el procés es desenvolupa gràcies a un hospital o centre mèdic a on al pacient li fan una radiografia, els metges la gestionen i els equips dels diferents fabricants, el quals es comuniquen en base a estàndards establerts per altres persones l'enven i l'emmagatzemen. Els investigadors tant de tècniques de programació com medicina i anàlisi obtenen resultats amb l'extracció dels segments vasculars del pacient gràcies a software especialitzat creat per altres experts en la matèria. I, finalment, se sap més sobre el pacient i es pot actuar amb detall sobre la seva situació única.

Que un pacient es faci una radiografia o un metge indiqui uns resultats i una solució, és un camí rutinari que succeeix cada dia. No obstant, la part central del procés millora i avança en relació als coneixements i l'experiència que els investigadors i treballadors demostren i els recursos i les aplicacions que tenen al seu abast; sense trencar el procés sencer.

Així, es justifica que la part més amagada del procés considerada com la investigació amb una imatge 3D és un bucle que no acaba mai però en el qual, els procediments han de ser cada vegada més exactes per arribar a uns resultats una mica més qualificats que els anteriors.

Tot això significa que, tot i que sigui lògic que darrera de tot hi ha persones, en aquest cas, les persones són una part importantíssima, la feina de les quals queda refugiada darrera les imatges i els estudis.

La dedicació i l'esforç per determinar nous paràmetres, ampliar coneixements, optar per recursos de fiabilitat i transparència i investigar amb la motivació d'aconseguir millores és un fet real i decisiu en l'avanc de la tècnica i, per tant, de la vida de tots els pacients que opten a una solució mèdica segura.

BIBLIOGRAFIA

Llibres:

Hofer M., *Manual Práctico de TC*, Editorial Medica Panamericana (2005).

Bueno G., Dorado J., *Gestión, procesado y análisis de imágenes biomédicas*, Ciencia y técnica 52 (2007).

Dougherty G., *Digital Image Processing for Medical Applications* (2009).

Pàgines web:

- Capítol 1:

http://books.google.es/books?id=tMVFkAEkfSQC&pg=PA119&lpg=PA119&dq=resoluci%C3%B3n+espacial+pelicula+radiografica&source=bl&ots=DLdQj50EWJ&sig=boxlvrKD8U-AvTTy-B3I8G72kWI&hl=es&ei=hi1uTayALcGFhQfA381N&sa=X&oi=book_result&ct=result&resnum=1&ved=0CBcQ6AEwAA#v=onepage&q&f=false

<http://www.nib.fmed.edu.uy/Corbo.pdf>

<http://www.galenusrevista.com/PET-CT-CARDIAC-una-nueva-dimension>

<http://dicom.nema.org/>

<http://www.hl7spain.org/>

- Capítol 2:

<http://www.vmtk.org/>

http://www.caddyspain.com/noticias/May06_4.htm

- Capítol 3:

<http://pyformex.berlios.de/>

ANNEXOS

TÍTOL DEL TFC: Obtaining geometries from CT scans of arteries to simulate the fluid flow of blood

TITULACIÓ: Enginyeria Tècnica Aeronàutica, especialitat Aeronavegació

AUTOR: Marta Lorente Muñoz

DIRECTOR: Gerber van der Graaf

DATA: 21 de novembre de 2011

VOLUM 2

ANNEXOS

ANNEX 1: DICOM.....	1
ANNEX 2: VMTKIMAGEREADER.....	25
ANNEX 3: VMTKIMAGEVOISELECTOR.....	29
ANNEX 4: VMTKLEVELSETSEGMENTATION.....	33
ANNEX 5: VMTKMARCHINGCUBES.....	41
ANNEX 6: VMTKCENTERLINES.....	44
ANNEX 7: VMTKSURFACEMOOHING.....	53
ANNEX 8: VMTKSUBDIVISION.....	56
ANNEX 9: VMTKSURFACLIPPER.....	59
ANNEX 10: VMTKFLOWEXTENSIONS.....	64
ANNEX 11: VMTKBRANCHEXTRACTOR.....	70
ANNEX 12: VMTKBRANCHCLIPPER.....	78
ANNEX 13: VMTKBRANCHCLIPPER (ELIMINAR BRANQUES).....	83
ANNEX 14: VMTKCENTERLINELABELER.....	88
ANNEX 15: VMTKCENTERLINEATTRIBUTES.....	94
ANNEX 16: VMTKCENTERLINEATTRIBUTES (VMTKBRANCHEXTRACTOR).....	99
ANNEX 17: VMTKBIFURCATIONREFERENCESYSTEMS.....	105
ANNEX 18: VMTKCENTERLINEOFFSETATTRIBUTES.....	109
ANNEX 19: VMTKBIFURCATIONVECTORS.....	115
ANNEX 20: VMTKCENTERLINEGEOMETRY.....	121
ANNEX 21: VMTKBRANCHGEOMETRY.....	125
ANNEX 22: VMTKMESHGENERATOR.....	129
ANNEX 23: VMTKLINEARTOQUADRATIC.....	133
ANNEX 24: VMTKMESHSCALING.....	136
ANNEX 25: VMTKBOUNDARYINSPECTOR.....	140
ANNEX 26: VMTKMESHTETRAHEDRALIZE.....	143
ANNEX 27: VMTKMESHLINEARIZE.....	147
ANNEX 28: MALLATGE A PARTIR DEL FORMAT STL.....	151
ANNEX 29: HEXAHEDRAL MESHES.....	154

ANNEX 1: DICOM

PS 3.1-2011

Digital Imaging and Communications in Medicine (DICOM)

Part 1: Introduction and Overview

Published by

National	Electrical	Manufacturers	Association
1300	N.	17th	Street
Rosslyn, Virginia 22209 USA			

© Copyright 2011 by the National Electrical Manufacturers Association. All rights including translation into other languages, reserved under the Universal Copyright Convention, the Berne Convention for the Protection of Literary and Artistic Works, and the International and Pan American Copyright Conventions.

NOTICE AND DISCLAIMER

The information in this publication was considered technically sound by the consensus of persons engaged in the development and approval of the document at the time it was developed. Consensus does not necessarily mean that there is unanimous agreement among every person participating in the development of this document.

NEMA standards and guideline publications, of which the document contained herein is one, are developed through a voluntary consensus standards development process. This process brings together volunteers and/or seeks out the views of persons who have an interest in the topic covered by this publication. While NEMA administers the process and establishes rules to promote fairness in the development of consensus, it does not write the document and it does not independently test, evaluate, or verify the accuracy or completeness of any information or the soundness of any judgments contained in its standards and guideline publications.

NEMA disclaims liability for any personal injury, property, or other damages of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, application, or reliance on this document. NEMA disclaims and makes no guaranty or warranty, expressed or implied, as to the accuracy or completeness of any information published herein, and disclaims and makes no warranty that the information in this document will fulfill any of your particular purposes or needs. NEMA does not undertake to guarantee the performance of any individual manufacturer or seller's products or services by virtue of this standard or guide.

In publishing and making this document available, NEMA is not undertaking to render professional or other services for or on behalf of any person or entity, nor is NEMA undertaking to perform any duty owed by any person or entity to someone else. Anyone using this document should rely on his or her own independent judgment or, as appropriate, seek the advice of a competent professional in determining the exercise of reasonable care in any given circumstances. Information and other standards on the topic covered by this publication may be available from other sources, which the user may wish to consult for additional views or information not covered by this publication.

NEMA has no power, nor does it undertake to police or enforce compliance with the contents of this document. NEMA does not certify, test, or inspect products, designs, or installations for safety or health purposes. Any certification or other statement of compliance with any health or safety-related information in this document shall not be attributable to NEMA and is solely the responsibility of the certifier or maker of the statement.

CONTENTS

NOTICE AND DISCLAIMER 3

CONTENTS 4

FOREWORD 6

INTRODUCTION 7

History.....	7
The DICOM Standard	7
Current direction	8
RETIREMENT	8

1 SCOPE AND FIELD OF APPLICATION 9

2 NORMATIVE REFERENCES 9

3 DEFINITIONS 9

4 SYMBOLS AND ABBREVIATIONS 10

5 GOALS OF THE DICOM STANDARD 11

6 OVERVIEW OF THE CONTENT OF THE DICOM STANDARD 13

6.1 Document Structure 13

6.2 PS 3.2: Conformance..... 13

6.3 PS 3.3: Information Object Definitions 16

6.4 PS 3.4: Service Class Specifications 17

6.5 PS 3.5: Data Structure and Semantics 17

6.6 PS 3.6: Data Dictionary 18

6.7 PS 3.7: Message Exchange..... 18

6.8 PS 3.8: Network Communication Support for Message Exchange 18

6.9 PS 3.9: Retired (Formerly Point-to-Point Communication Support for Message Exchange) 19

6.10 PS 3.10 Media Storage and File Format 19

6.11 PS 3.11: Media Storage Application Profiles.....	20
6.12 PS 3.12: Storage Functions and Media Formats for Data Interchange	21
6.13 PS 3.13: Retired (Formerly Print Management Point-to-point Communication Support)	21
6.14 PS 3.14: Grayscale Standard Display Function	21
6.15 PS 3.15: Security and System Management Profiles.....	22
6.16 PS 3.16: Content Mapping Resource	22
6.17 PS 3.17: Explanatory Information.....	22
6.18 PS 3.18: Web Access to DICOM Persistent Objects (WADO)	22
6.19 PS3.19: Application Hosting	22
6.20 PS 3.20: Transformation of DICOM to and from HL7 Standards	24

FOREWORD

This DICOM Standard was developed according to the Procedures of the DICOM Standards Committee.

The DICOM Standard is structured as a multi-part document using the guidelines established in the following document:

- ISO/IEC Directives, 1989 Part 3: Drafting and Presentation of International Standards.

INTRODUCTION

History

With the introduction of computed tomography (CT) followed by other digital diagnostic imaging modalities in the 1970's, and the increasing use of computers in clinical applications, the American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) recognized the emerging need for a standard method for transferring images and associated information between devices manufactured by various vendors. These devices produce a variety of digital image formats.

The American College of Radiology (ACR) and the National Electrical Manufacturers Association (NEMA) formed a joint committee in 1983 to develop a standard to:

- Promote communication of digital image information, regardless of device manufacturer
- Facilitate the development and expansion of picture archiving and communication systems (PACS) that can also interface with other systems of hospital information
- Allow the creation of diagnostic information data bases that can be interrogated by a wide variety of devices distributed geographically.

ACR-NEMA Standards Publication No. 300-1985, published in 1985 was designated version 1.0. The Standard was followed by two revisions: No. 1, dated October 1986 and No. 2, dated January 1988.

ACR-NEMA Standards Publication No. 300-1988, published in 1988 was designated version 2.0. It included version 1.0, the published revisions, and additional revisions. It also included new material to provide command support for display devices, to introduce a new hierarchy scheme to identify an image, and to add data elements for increased specificity when describing an image.

These Standards Publications specified a hardware interface, a minimum set of software commands, and a consistent set of data formats.

The DICOM Standard

This Standard, which is currently designated Digital Imaging and Communications in Medicine (DICOM), embodies a number of major enhancements to previous versions of the ACR-NEMA Standard:

- a. It is applicable to a networked environment. The ACR-NEMA Standard was applicable in a point-to-point environment only; for operation in a networked environment a Network Interface Unit (NIU) was required. DICOM supports operation in a networked environment using the industry standard networking protocol TCP/IP.
- b. It is applicable to an off-line media environment. The ACR-NEMA Standard did not specify a file format or choice of physical media or logical filesystem. DICOM supports operation in an off-line media environment using industry standard media such as CD-R and MOD and logical filesystems such as ISO 9660 and PC File System (FAT16).
- c. It specifies how devices claiming conformance to the Standard react to commands and data being exchanged. The ACR-NEMA Standard was confined to the transfer of data, but DICOM specifies, through the concept of Service Classes, the semantics of commands and associated data.
- d. It specifies levels of conformance. The ACR-NEMA Standard specified a minimum level of conformance. DICOM explicitly describes how an implementor must structure a Conformance Statement to select specific options.

- e. It is structured as a multi-part document. This facilitates evolution of the Standard in a rapidly evolving environment by simplifying the addition of new features. ISO directives which define how to structure multi-part documents have been followed in the construction of the DICOM Standard.
- f. It introduces explicit Information Objects not only for images and graphics but also for waveforms, reports, printing, etc.
- g. It specifies an established technique for uniquely identifying any Information Object. This facilitates unambiguous definitions of relationships between Information Objects as they are acted upon across the network.

Current direction

The DICOM Standard is an evolving standard and it is maintained in accordance with the Procedures of the DICOM Standards Committee. Proposals for enhancements are forthcoming from the DICOM Committee member organizations based on input from users of the Standard. These proposals are considered for inclusion in future editions of the Standard. A requirement in updating the Standard is to maintain effective compatibility with previous editions.

RETIREMENT

Part of the maintenance process involves retirement of sections of the Standard, including but not limited to, IODs, Attributes, Service Classes, SOP Classes, Transfer Syntaxes and Protocols.

Retirement does not imply that these features cannot be used. However, the DICOM Standards Committee will not maintain the documentation of retired features. The reader is referred to earlier editions of the Standard.

The use of the retired features is deprecated in new implementations, in favor of those alternatives remaining in the standard.

1 Scope and field of application

PS 3.1 provides an overview of the entire Digital Imaging and Communications in Medicine (DICOM) Standard. It describes the history, scope, goals, and structure of the Standard. In particular, it contains a brief description of the contents of each part of the Standard.

The DICOM Standard facilitates interoperability of medical imaging equipment by specifying:

- For network communications, a set of protocols to be followed by devices claiming conformance to the Standard.
- The syntax and semantics of Commands and associated information which can be exchanged using these protocols.
- For media communication, a set of media storage services to be followed by devices claiming conformance to the Standard, as well as a File Format and a medical directory structure to facilitate access to the images and related information stored on interchange media.
- Information that must be supplied with an implementation for which conformance to the Standard is claimed.

The DICOM Standard does not specify:

- The implementation details of any features of the Standard on a device claiming conformance.
- The overall set of features and functions to be expected from a system implemented by integrating a group of devices each claiming DICOM conformance.
- A testing/validation procedure to assess an implementation's conformance to the Standard.

The DICOM Standard pertains to the field of Medical Informatics. Within that field, it addresses the exchange of digital information between medical imaging equipment and other systems. Because such equipment may interoperate with other medical devices, the scope of this Standard needs to overlap with other areas of medical informatics. However, the DICOM Standard does not address the breadth of this field.

2 Normative references

ISO/IEC Directives, 1989 Part 3 - Drafting and presentation of International Standards.

ACR-NEMA 300-1988 Digital Imaging and Communications

ISO 8822, Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition.

ISO 8649, Information Processing Systems - Open Systems Interconnection - Service Definition for the Association Control Service Element.

3 Definitions

Attribute: A property of an Information Object. An Attribute has a name and a value which are independent of any encoding scheme.

Command: A request to operate on information across a network.

Command Element: An encoding of a parameter of a command which conveys this parameter's value.

Command Stream: The result of encoding a set of DICOM Command Elements using the DICOM encoding scheme.

Conformance Statement: A formal statement that describes a specific product implementation that uses the DICOM Standard. It specifies the Service Classes, Information Objects, and Communication Protocols supported by the implementation.

Data Dictionary: A registry of DICOM Data Elements which assigns a unique tag, a name, value characteristics, and semantics to each Data Element.

Data Element: A unit of information as defined by a single entry in the data dictionary.

Data Set: Exchanged information consisting of a structured set of Attributes. The value of each Attribute in a Data Set is expressed as a Data Element.

Data Stream: The result of encoding a Data Set using the DICOM encoding scheme (Data Element Numbers and representations as specified by the Data Dictionary).

Information Object: An abstraction of a real information entity (e.g., CT Image, Structured Report, etc.) which is acted upon by one or more DICOM Commands.

Note: This term is primarily used in PS 3.1, with a few references in PS 3.3. It is an informal term corresponding to a formal term that is introduced in PS 3.3. In all other parts of the DICOM Standard this formal term is known as an Information Object Definition.

Information Object Class: A formal description of an Information Object which includes a description of its purpose and the Attributes it possesses. It does not include values for these attributes.

Note: This term is only used in PS 3.1. It is an informal term corresponding to a formal term that is introduced in PS 3.4. This formal term is known as a Service-Object Pair Class or more commonly as a SOP Class.

Information Object Instance: A representation of an occurrence of a real-world entity, which includes values for the Attributes of the Information Object Class to which the entity belongs.

Note: This term is only used in PS 3.1. It is an informal term corresponding to a formal term that is introduced in PS 3.4. This formal term is known as a Service-Object Pair Instance or more commonly as a SOP Instance.

Message: A data unit of the Message Exchange Protocol exchanged between two cooperating DICOM Applications. A Message is composed of a Command Stream followed by an optional Data Stream.

Service Class: A structured description of a service which is supported by cooperating DICOM Applications using specific DICOM Commands acting on a specific class of Information Object.

4 Symbols and abbreviations

ACSE Association Control Service Element

CT Computed Tomography

DICOM Digital Imaging and Communications in Medicine

HIS Hospital Information System

JIRA Japan Industries Association of Radiological Systems

OSI Open Systems Interconnection

PACS Picture Archiving and Communication Systems

RIS Radiology Information System

TCP/IP Transmission Control Protocol/Internet Protocol

5 Goals of the DICOM standard

The DICOM Standard facilitates interoperability of devices claiming conformance. In particular, it:

- Addresses the semantics of Commands and associated data. For devices to interact, there must be standards on how devices are expected to react to Commands and associated data, not just the information which is to be moved between devices;
- Addresses the semantics of file services, file formats and information directories necessary for off-line communication;
- Is explicit in defining the conformance requirements of implementations of the Standard. In particular, a conformance statement must specify enough information to determine the functions for which interoperability can be expected with another device claiming conformance.
- Facilitates operation in a networked environment.
- Is structured to accommodate the introduction of new services, thus facilitating support for future medical imaging applications.
- Makes use of existing international standards wherever applicable, and itself conforms to established documentation guidelines for international standards.

Even though the DICOM Standard has the potential to facilitate implementations of PACS solutions, use of the Standard alone does not guarantee that all the goals of a PACS will be met. This Standard facilitates interoperability of systems claiming conformance in a multi-vendor environment, but does not, by itself, guarantee interoperability.

This Standard has been developed with an emphasis on diagnostic medical imaging as practiced in radiology, cardiology and related disciplines; however, it is also applicable to a wide range of image and non-image related information exchanged in clinical and other medical environments.

Figure 5-1 presents the general communication model of the Standard which spans both network (on-line) and media storage interchange (off-line) communication. Applications may rely on either one of the following boundaries:

- the Upper Layer Service, which provides independence from specific physical networking communication support and protocols such as TCP/IP.
- The Basic DICOM File Service, which provides access to Storage Media independently from specific media storage formats and file structures.

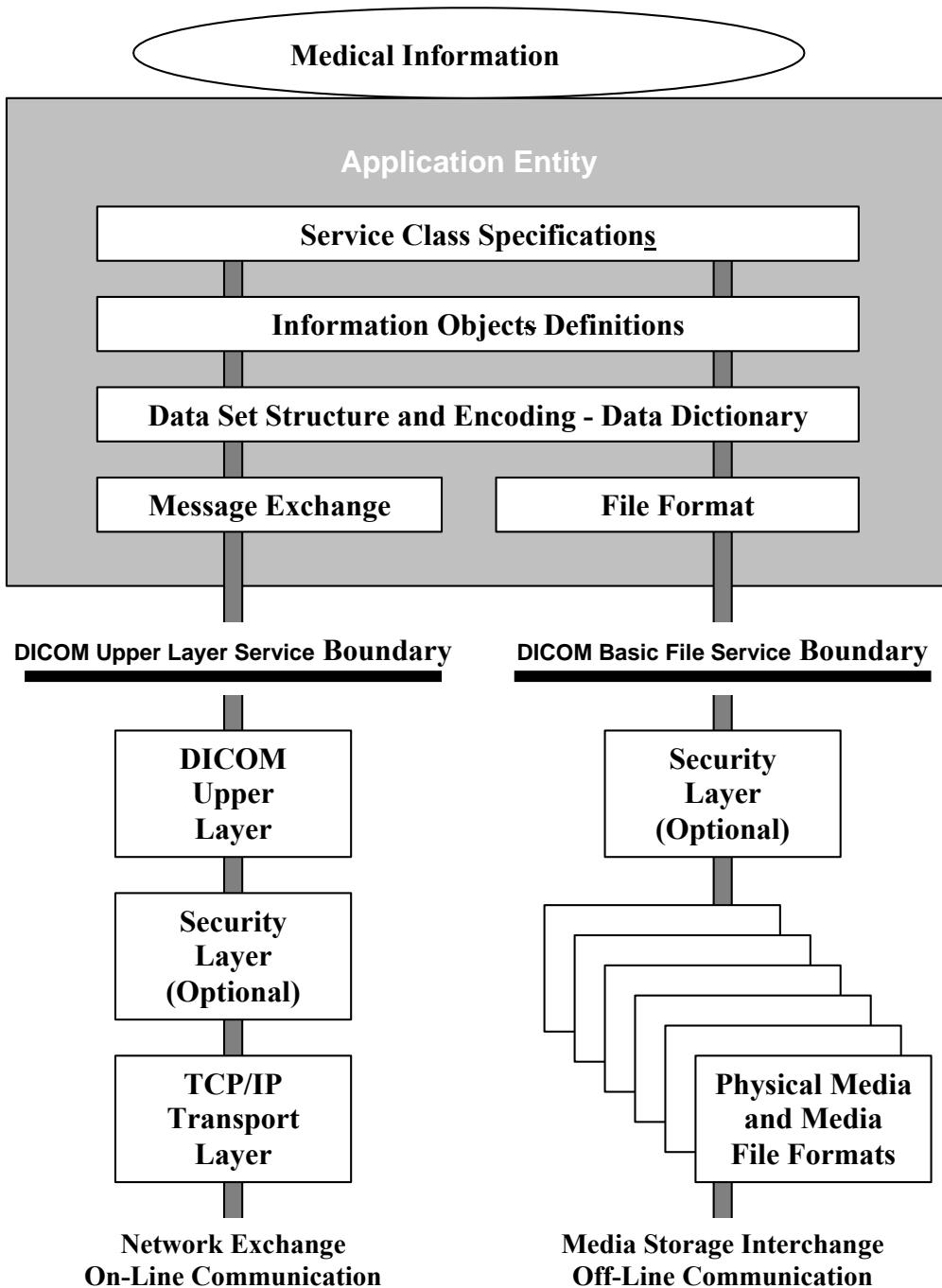


Figure 5-1 General Communication Model

6 Overview of the Content of the DICOM Standard

6.1 Document Structure

DICOM consists of the following parts:

PS 3.1:	Introduction and Overview (this document)
PS 3.2:	Conformance
PS 3.3:	Information Object Definitions
PS 3.4:	Service Class Specifications
PS 3.5:	Data Structure and Encoding
PS 3.6:	Data Dictionary
PS 3.7:	Message Exchange
PS 3.8:	Network Communication Support for Message Exchange
PS 3.9:	Retired
PS 3.10:	Media Storage and File Format for Data Interchange
PS 3.11:	Media Storage Application Profiles
PS 3.12:	Media Formats and Physical Media for Data Interchange
PS 3.13:	Retired
PS 3.14	Grayscale Standard Display Function
PS 3.15:	Security Profiles
PS 3.16:	Content Mapping Resource
PS 3.17:	Explanatory Information
PS 3.18:	Web Access to DICOM Persistent Objects (WADO)
PS 3.19:	Application Hosting
PS 3.20:	Transformation of DICOM to and from HL7 Standards

These parts of the Standard are related but independent documents. A brief description of each Part is provided in this section.

6.2 PS 3.2: Conformance

PS 3.2 of the DICOM Standard defines principles that implementations claiming conformance to the Standard shall follow:

- Conformance requirements. PS 3.2 specifies the general requirements which must be met by any implementation claiming conformance. It references the conformance sections of other parts of the Standard.
- Conformance Statement. PS 3.2 defines the structure of a Conformance Statement. It specifies the information which must be present in a Conformance Statement. It references the Conformance Statement sections of other parts of the Standard.

PS 3.2 does not specify a testing/validation procedure to assess an implementation's conformance to the Standard.

Figures 6.2-1 and 6.2-2 depict the construction process for a Conformance Statement for both network communication and media exchange. A Conformance Statement consists of the following parts:

- Set of Information Objects which is recognized by this implementation
- Set of Service Classes which this implementation supports
- Set of communications protocols or physical media which this implementation supports
- Set of security measures which this implementation supports.

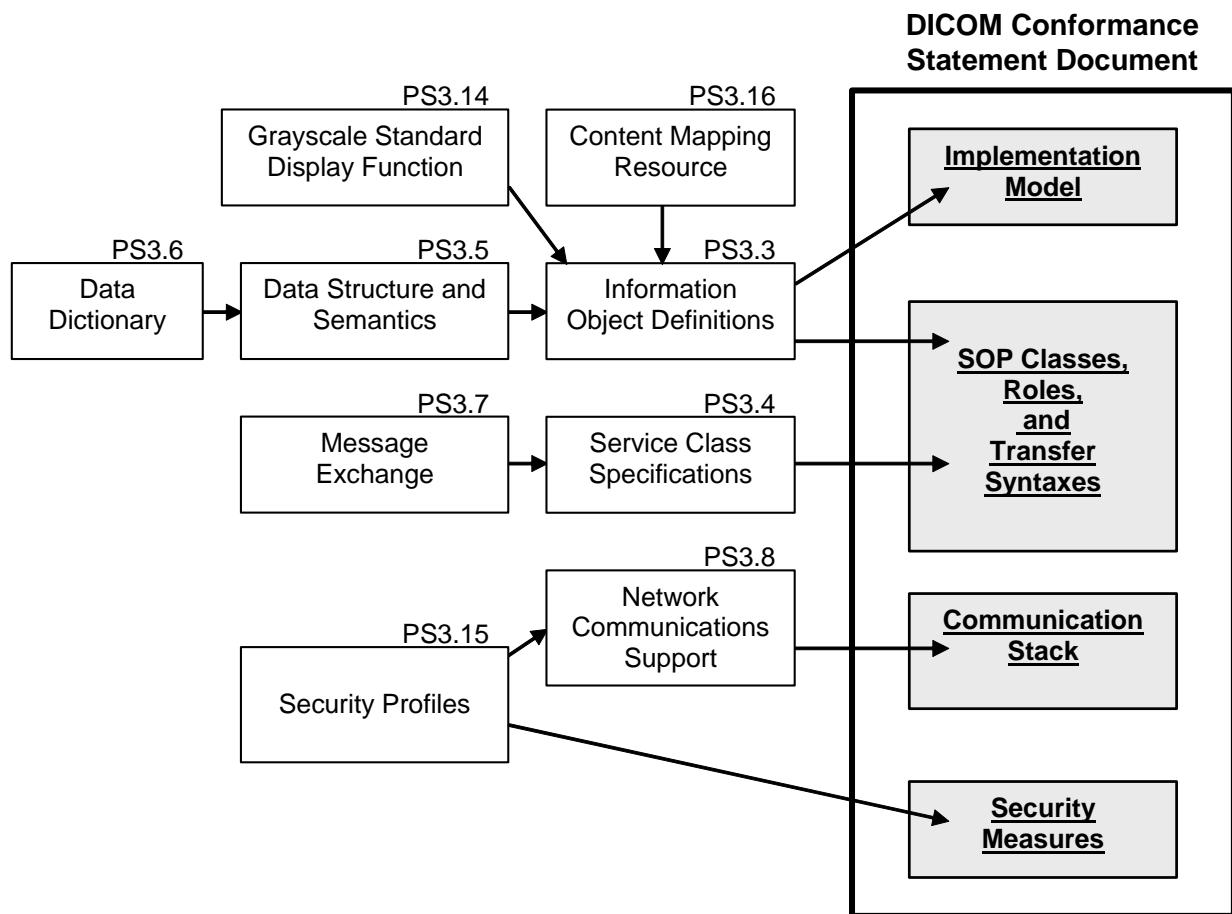
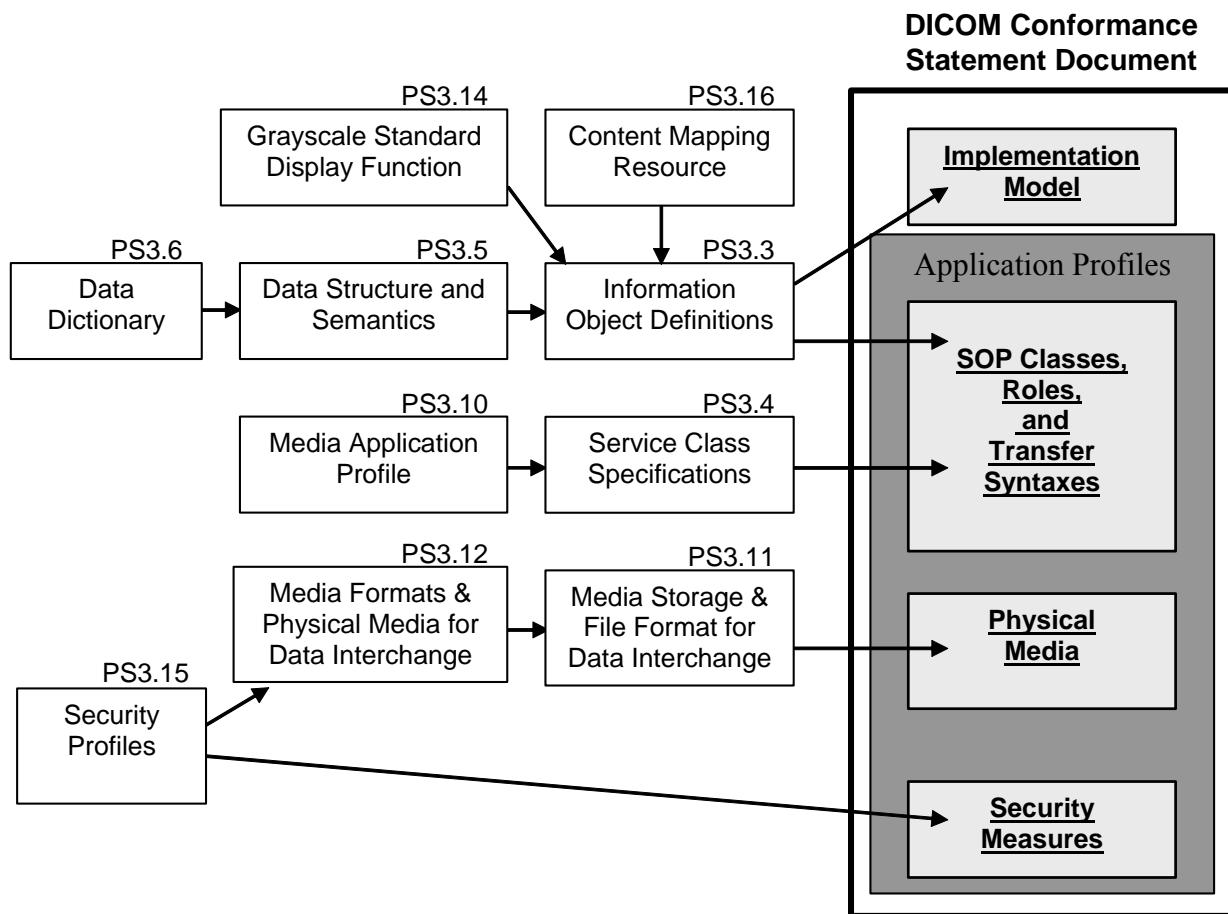


Figure 6.2-1
CONSTRUCTION PROCESS FOR A NETWORK CONFORMANCE CLAIM



**Figure 6.2-2
CONSTRUCTION PROCESS FOR A MEDIA CONFORMANCE CLAIM**

6.3 PS 3.3: Information Object Definitions

PS 3.3 of the DICOM Standard specifies a number of Information Object Classes which provide an abstract definition of real-world entities applicable to communication of digital medical images and related information (e.g., waveforms, structured reports, radiation therapy dose, etc.). Each Information Object Class definition consists of a description of its purpose and the Attributes which define it. An Information Object Class does not include the values for the Attributes which comprise its definition.

Two types of Information Object Classes are defined: normalized and composite.

Normalized Information Object Classes include only those Attributes inherent in the real-world entity represented. For example the study Information Object Class, which is defined as normalized, contains study date and study time Attributes because they are inherent in an actual study. Patient name, however, is not an Attribute of the study Information Object Class because it is inherent in the patient on which the study was performed and not the study itself.

Composite Information Object Classes may additionally include Attributes which are related to but not inherent in the real-world entity. For example, the Computed Tomography Image Information Object Class, which is defined as composite, contains both Attributes which are inherent in the image (e.g. image

date) and Attributes which are related to but not inherent in the image (e.g. patient name). Composite Information Object Classes provide a structured framework for expressing the communication requirements of images where image data and related data needs to be closely associated.

To simplify the Information Object Class definitions, the Attributes of each Information Object Class are partitioned with similar Attributes being grouped together. These groupings of Attributes are specified as independent modules and may be reused by other Composite Information Object Classes.

PS 3.3 defines a model of the Real World along with the corresponding Information Model that is reflected in the Information Object Definitions. Future editions of this Standard may extend this set of Information Objects to support new functionality.

To represent an occurrence of a real-world entity, an Information Object Instance is created, which includes values for the Attributes of the Information Object Class. The Attribute values of this Information Object Instance may change over time to accurately reflect the changing state of the entity which it represents. This is accomplished by performing different basic operations upon the Information Object Instance to render a specific set of services defined as a Service Class. These Service Classes are defined in PS 3.4 of the Standard.

6.4 PS 3.4: Service Class Specifications

PS 3.4 of the DICOM Standard defines a number of Service Classes. A Service Class associates one or more Information Objects with one or more Commands to be performed upon these objects. Service Class Specifications state requirements for Command Elements and how resulting Commands are applied to Information Objects. Service Class Specifications state requirements for both providers and users of communications services.

PS 3.4 of the DICOM Standard defines the characteristics shared by all Service Classes, and how a Conformance Statement to an individual Service Class is structured. It contains a number of normative annexes which describe individual Service Classes in detail.

Examples of Service Classes include the following:

- Storage Service Class
- Query/Retrieve Service Class
- basic Worklist Management Service Class
- Print Management Service Class.

PS 3.4 defines the operations performed upon the Information Objects defined in PS 3.3. PS 3.7 defines the Commands and protocols for using the Commands to accomplish the operations and notifications described in PS 3.4.

6.5 PS 3.5: Data Structure and Semantics

PS 3.5 of the DICOM Standard specifies how DICOM applications construct and encode the Data Set information resulting from the use of the Information Objects and Services Classes defined in PS 3.3 and PS 3.4 of the DICOM Standard. The support of a number of standard image compression techniques (e.g., JPEG lossless and lossy) is specified.

PS 3.5 addresses the encoding rules necessary to construct a Data Stream to be conveyed in a Message as specified in PS 3.7 of the DICOM Standard. This

Data Stream is produced from the collection of Data Elements making up the Data Set.

PS 3.5 also defines the semantics of a number of generic functions that are common to many Information Objects. PS 3.5 defines the encoding rules for international character sets used within DICOM.

6.6 PS 3.6: Data Dictionary

PS 3.6 of the DICOM Standard is the centralized registry which defines the collection of all DICOM Data Elements available to represent information, along with elements utilized for interchangeable media encoding and a list of uniquely identified items that are assigned by DICOM.

For each element, PS 3.6 specifies:

- its unique tag, which consists of a group and element number
- its name
- its value representation (character string, integer, etc)
- its value multiplicity (how many values per attribute)
- whether it is retired

For each uniquely identified item, PS 3.6 specifies:

- its unique value, which is numeric with multiple components separated by decimal points and limited to 64 characters
- its name
- its type, either Information Object Class, definition of encoding for data transfer, or certain well known Information Object Instances
- in which part of the DICOM Standard it is defined

6.7 PS 3.7: Message Exchange

PS 3.7 of the DICOM Standard specifies both the service and protocol used by an application in a medical imaging environment to exchange Messages over the communications support services defined in PS 3.8. A Message is composed of a Command Stream defined in PS 3.7 followed by an optional Data Stream as defined in PS 3.5.

PS 3.7 specifies:

- the operations and notifications (DIMSE Services) made available to Service Classes defined in PS 3.4,
- rules to establish and terminate associations provided by the communications support specified in PS 3.8, and the impact on outstanding transactions
- rules that govern the exchange of Command requests and responses
- encoding rules necessary to construct Command Streams and Messages.

6.8 PS 3.8: Network Communication Support for Message Exchange

PS 3.8 of the DICOM Standard specifies the communication services and the upper layer protocols necessary to support, in a networked environment, communication between DICOM applications as specified in PS 3.3, PS 3.4, PS 3.5, PS 3.6, and PS 3.7. These communication services and protocols ensure that communication between DICOM applications is performed in an efficient and coordinated manner across the network.

The communication services specified in PS 3.8 are a proper subset of the services offered by the OSI Presentation Service (ISO 8822) and of the OSI Association Control Service Element (ACSE) (ISO 8649). They are referred to as the Upper Layer Service, which allows peer applications to establish associations, transfer messages and terminate associations.

This definition of the Upper Layer Service specifies the use of the DICOM Upper Layer Protocol in conjunction with TCP/IP transport protocols.

The TCP/IP communication protocol specified by PS 3.8 is a general purpose communication protocol not specific to the DICOM Standard. Figure 5-1 shows this protocol stack.

6.9 PS 3.9: Retired (Formerly Point-to-Point Communication Support for Message Exchange)

PS 3.9 of the DICOM Standard previously specified the services and protocols used for point-to-point communications in a manner compatible with ACR-NEMA 2.0. It has been retired.

6.10 PS 3.10 Media Storage and File Format

PS 3.10 of the DICOM Standard specifies a general model for the storage of medical imaging information on removable media (see Figure 6.10-1). The purpose of this Part is to provide a framework allowing the interchange of various types of medical images and related information on a broad range of physical storage media.

Note: See Figure 5-1 for understanding how the media interchange model compares to the network model.

PS 3.10 Specifies:

- a layered model for the storage of medical images and related information on storage media. This model introduces the concept of media storage application profiles, which specify application specific subsets of the DICOM Standard to which a media storage implementation may claim conformance. Such a conformance applies only to the writing, reading and updating of the content of storage media.
- a DICOM file format supporting the encapsulation of any Information Object;
- a secure DICOM file format supporting the encapsulation of a DICOM file format in a cryptographic envelope;
- a DICOM file service providing independence from the underlying media format and physical media.

PS 3.10 defines various media storage concepts:

- a) the method to identify a set of files on a single medium
- b) the method for naming a DICOM file within a specific file system

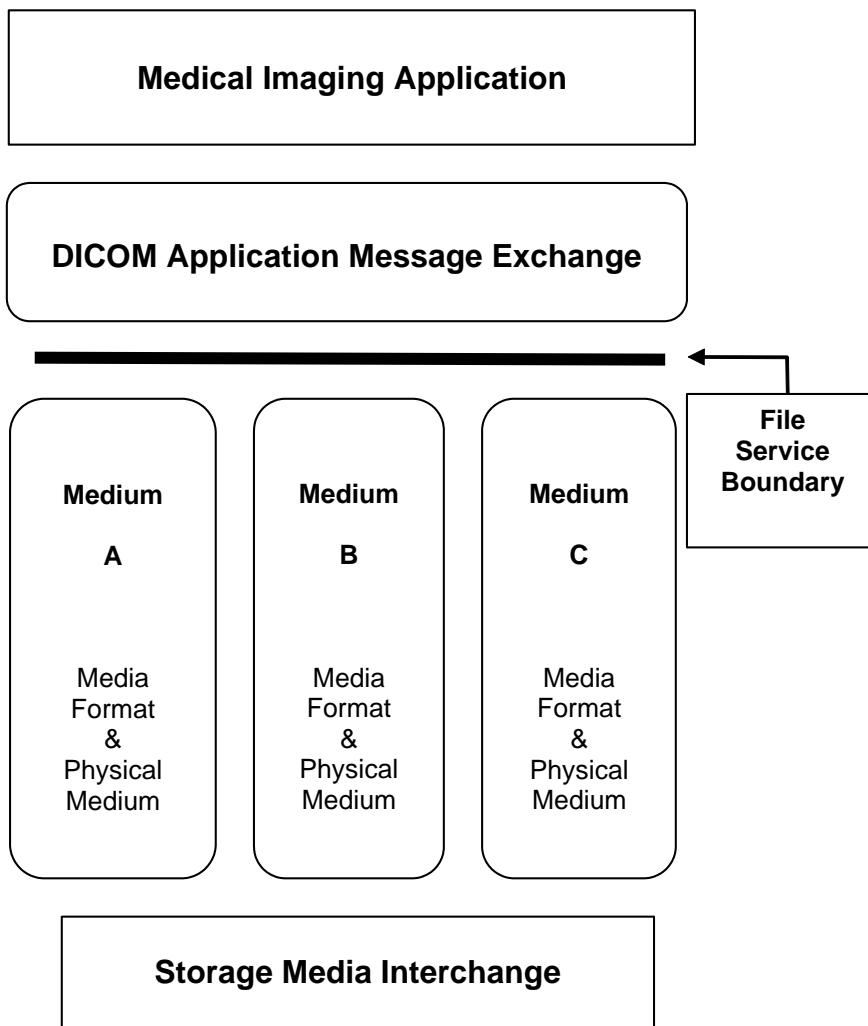


Figure 6.10-1
DICOM Media Communication Model

6.11 PS 3.11: Media Storage Application Profiles

PS 3.11 of the DICOM Standard specifies application specific subsets of the DICOM Standard to which an implementation may claim conformance. These application specific subsets will be referred to as Application Profiles in this section. Such a conformance statement applies to the interoperable interchange of medical images and related information on storage media for specific clinical uses. It follows the framework, defined in PS 3.10, for the interchange of various types of information on storage media.

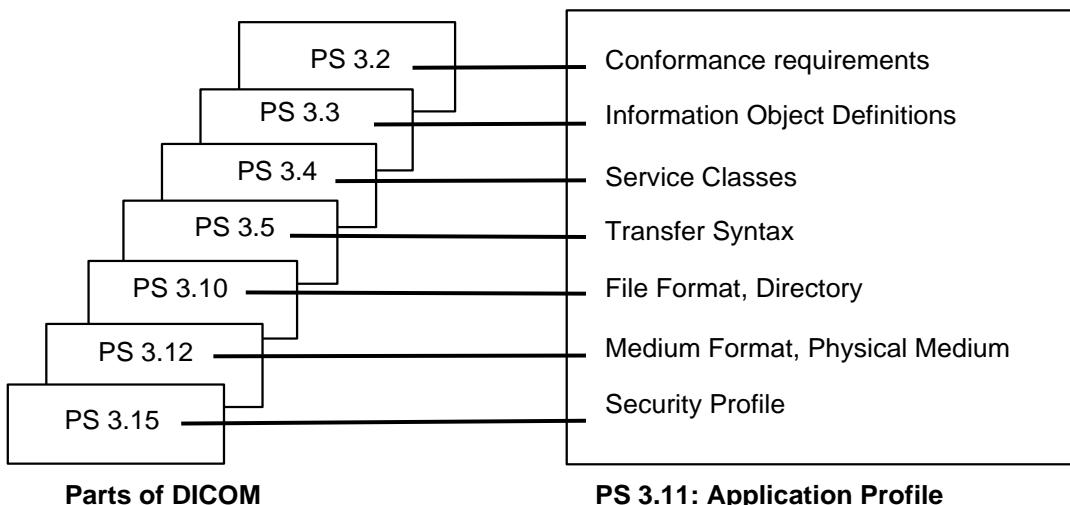
An Application Profile annex is organized into the following major parts:

- a) The name of the Application Profile, or the list of Application Profiles grouped in a related class
- b) A description of the clinical context of the Application Profile
- c) The definition of the media storage Service Class with the device roles for the Application Profile and associated options
- d) Informative section describing the operational requirements of the Application Profile
- e) Specification of the Information Object Classes and associated Information Objects supported and the encoding to be used for the data transfer

- f) The selection of media formats and physical media to be used
- g) Other parameters which need to be specified to ensure interoperable media interchange
- h) Security parameters which select the cryptographic techniques to be used with secure media storage Application Profiles

The structure of DICOM and the design of the Application Profile mechanism is such that extension to additional Information Object Classes and the new exchange media is straightforward.

Note: Figure 6.11-1 shows how individual aspects of an Application profile map to the various parts of the DICOM Standard.



**Figure 6.11-1
Relationship Between Application Profile and Parts of DICOM**

6.12 PS 3.12: Storage Functions and Media Formats for Data Interchange

This part of the DICOM Standard facilitates the interchange of information between applications in medical environments by specifying:

- a) A structure for describing the relationship between the media storage model and a specific physical media and media format.
- b) Specific physical media characteristics and associated media formats.

6.13 PS 3.13: Retired (Formerly Print Management Point-to-point Communication Support)

PS 3.13 previously specified the services and protocols used for point-to-point communication of print management services. It has been retired.

6.14 PS 3.14: Grayscale Standard Display Function

PS 3.14 specifies a standardized display function for consistent display of grayscale images. This function provides methods for calibrating a particular display system for the purpose of presenting images consistently on different display media (e.g. monitors and printers).

The chosen display function is based on human visual perception. Human eye contrast sensitivity is distinctly non-linear within the luminance range of display devices. This standard uses Barten's model of the human visual system.

6.15 PS 3.15: Security and System Management Profiles

PS 3.15 of the DICOM Standard specifies security and system management profiles to which implementations may claim conformance. Security and system management profiles are defined by referencing externally developed standard protocols, such as DHCP, LDAP, TLS and ISCL. Security protocols may use security techniques like public keys and “smart cards”. Data encryption can use various standardized data encryption schemes.

This part does not address issues of security policies. The standard only provides mechanisms that can be used to implement security policies with regard to the interchange of DICOM objects. It is the local administrator’s responsibility to establish appropriate security policies.

6.16 PS 3.16: Content Mapping Resource

PS 3.16 of the DICOM Standard specifies:

- templates for structuring documents as DICOM Information Objects
- sets of coded terms for use in Information Objects
- a lexicon of terms defined and maintained by DICOM
- country specific translations of coded terms

6.17 PS 3.17: Explanatory Information

PS 3.17 of the DICOM Standard specifies:

- informative and normative annexes containing explanatory information

6.18 PS 3.18: Web Access to DICOM Persistent Objects (WADO)

PS 3.18 of the DICOM Standard specifies the means whereby a request for access to a DICOM persistent object can be expressed as an HTTP URL/URI request that includes a pointer to a specific DICOM persistent object in the form of its Instance UID.

The request also specifies the format of the result to be returned in response to the request.

Examples include:

1. (MIME) Content-type, e.g., application/dicom or image/jpeg for images, application/dicom or application/rtf or xml for reports
2. Content-Encodings
3. reports as HL7/CDA Level 1

The parameters of the query URL as defined within this standard are sufficient for the HTTP server to act as a DICOM SCU (Service Class User) to retrieve the requested object from an appropriate DICOM SCP (Service Class Provider) using baseline DICOM functionality as defined in PS 3.4 and PS 3.7.

6.19 PS3.19: Application Hosting

PS3.19 of the DICOM Standard specifies an Application Programming Interface (API) to a DICOM-based medical computing system into which programs written to that standardized interface can ‘plug-in’ (see Figure 6.19-1). A Hosting System implementer only needs to create the standardized API once to support a wide variety of add-on Hosted Applications.

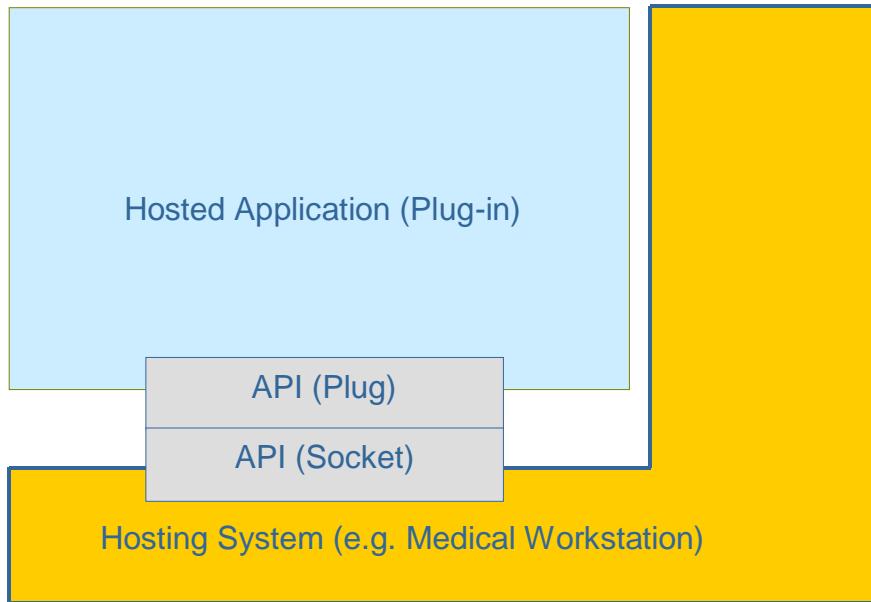


Figure 6.19-1. Interface between a Hosted Application and a Hosting System

In the traditional ‘plug-in’ model, the ‘plug-in’ is dedicated to a particular host system (e.g. a web browsing program), and might not run under other host systems (e.g. other web browsing programs). PS3.19 defines an API that may be implemented by any Hosting System. A ‘plug-in’ Hosted Application written to the API would be able run in any environment provided by a Hosting System that implements that API (see Figure 6.19-2).

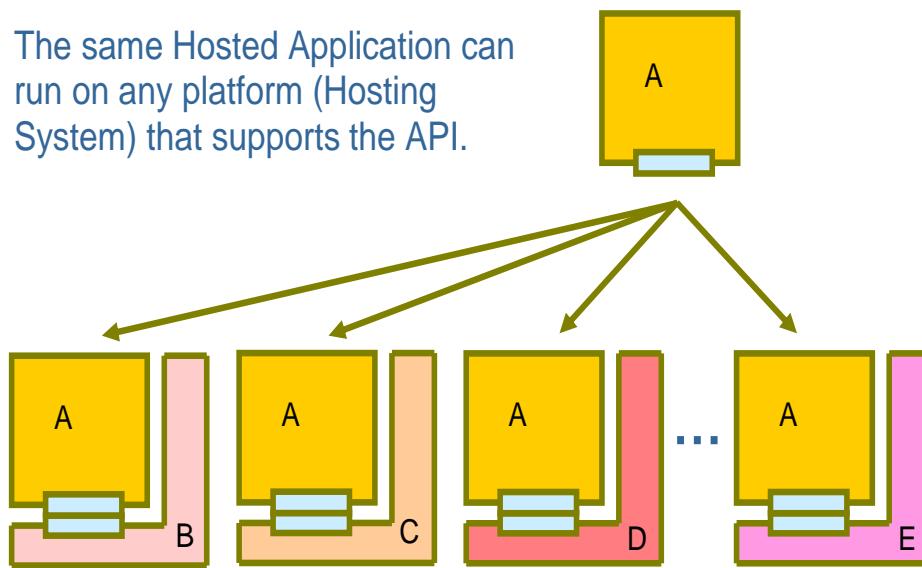


Figure 6.19-2. Illustration of platform independence via the Hosted Application architecture.

PS3.19 specifies both the interactions and the Application Programming Interfaces (API) between Hosting Systems and Hosted Applications. PS3.19 also defines the data models that are used by the API.

6.20 PS 3.20: Transformation of DICOM to and from HL7 Standards

PS 3.20 of the DICOM Standard specifies:

- Transformations of DICOM data to and from HL7 standards

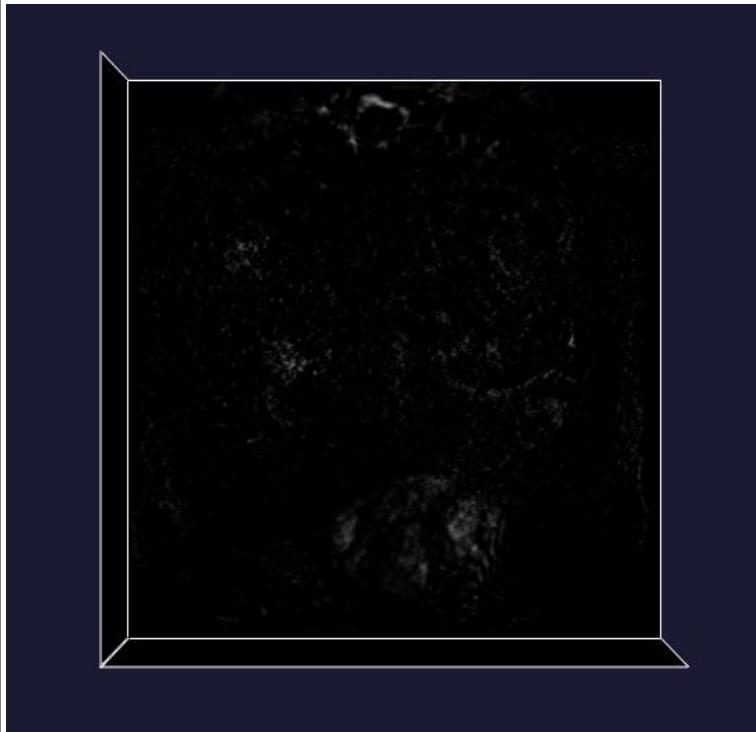
ANNEX 2: VMTKIMAGEREADER

```
vmtk vmtkimagereader -ifile 3/IM-0001-0001.dcm -ofile dicomimages.vti --pipe vmtkimageviewer
Executing vmtkimagereader -ifile 3/IM-0001-0001.dcm -ofile dicomimages.vti --pipe vmtkimageviewer

Creating vmtkImageReader instance.
Automatic piping vmtkimagereader
Parsing options vmtkimagereader
  InputFileName = 3/IM-0001-0001.dcm
  ImageOutputFileName = dicomimages.vti
Explicit piping vmtkimagereader
Input vmtkimagereader members:
  Id = 0
  Disabled = 0
  Format =
  GuessFormat = 1
  UseITKIO = 1
  Image = 0
  InputFileName = 3/IM-0001-0001.dcm
  InputFilePrefix =
  InputFilePattern =
  InputDirectoryName =
  DataExtent = [-1, -1, -1, -1, -1, -1]
  HeaderSize = 0
  DataSpacing = [1.0, 1.0, 1.0]
  DataOrigin = [0.0, 0.0, 0.0]
  DataByteOrder = little endian
  DataScalarType = float
  FileDimensionality = 3
  Flip = [0, 0, 0]
  AutoOrientDICOMImage = 1
  ImageOutputFileName = dicomimages.vti
Executing vmtkimagereader ...
Spacing (1.0026041666667, 1.0026041666667, 1.4999980926513672)
Origin (182.05859375, 32.531246185302734, 192.5)
Dimensions (384, 384, 72)
Done executing vmtkimagereader.
Writing VTK XML image file.
Output vmtkimagereader members:
  Id = 0
  Image = vtkImageData
  RasToljkMatrixCoefficients = [-0.9974025974025642, -0.0, 0.0,
  181.58571428570826, -0.0, 0.0, -0.9974025974025642, 191.99999999999936,
  0.0, -0.666667514378248, 0.0, 21.687525033982634, 0.0, 0.0, -0.0, 1.0]
  XyzToRasMatrixCoefficients = [-1.0, 0.0, 0.0, 364.1171875, 0.0, 0.0, -1.0,
  225.03124618530273, 0.0, -1.0, 0.0, 225.03124618530273, 0.0, 0.0, 0.0, 1.0]

Creating vmtkImageViewer instance.
Automatic piping vmtkimageviewer
Parsing options vmtkimageviewer
```

```
ImageInputFileName = dicomimages.vti
Explicit piping vmtkimageviewer
Input vmtkimageviewer members:
Id = 0
Disabled = 0
Image = None
ImageInputFileName = dicomimages.vti
ArrayName =
vmtkRenderer = None
WindowLevel = [0.0, 0.0]
Display = 1
Margins = 0
TextureInterpolation = 1
ContinuousCursor = 0
ImageOutputFileName =
Reading VTK XML image file.
Spacing (1.0026041667, 1.0026041667, 1.4999980927)
Origin (182.05859375, 32.531246185, 192.5)
Dimensions (384, 384, 72)
Executing vmtkimageviewer ...
```



Done executing vmtkimageviewer.

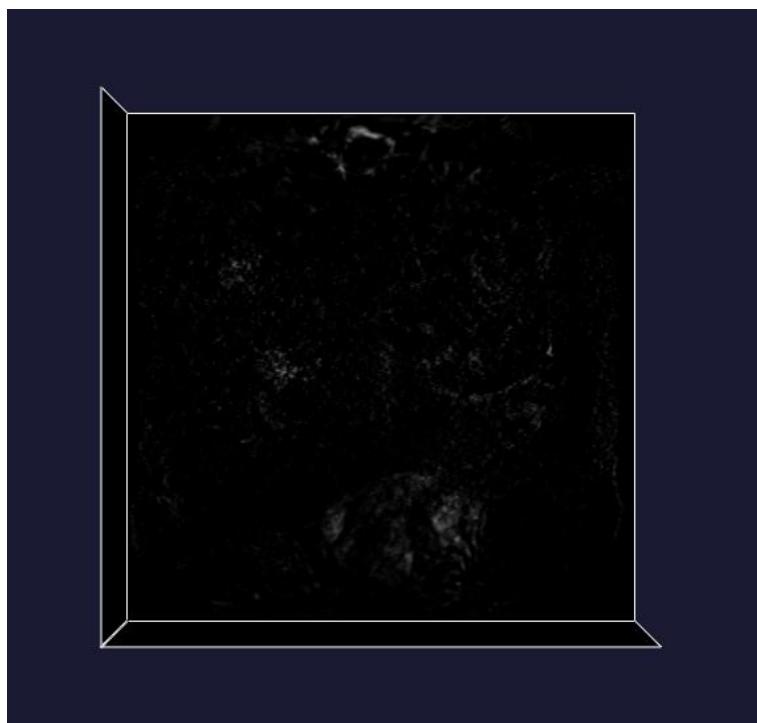
Output vmtkimageviewer members:

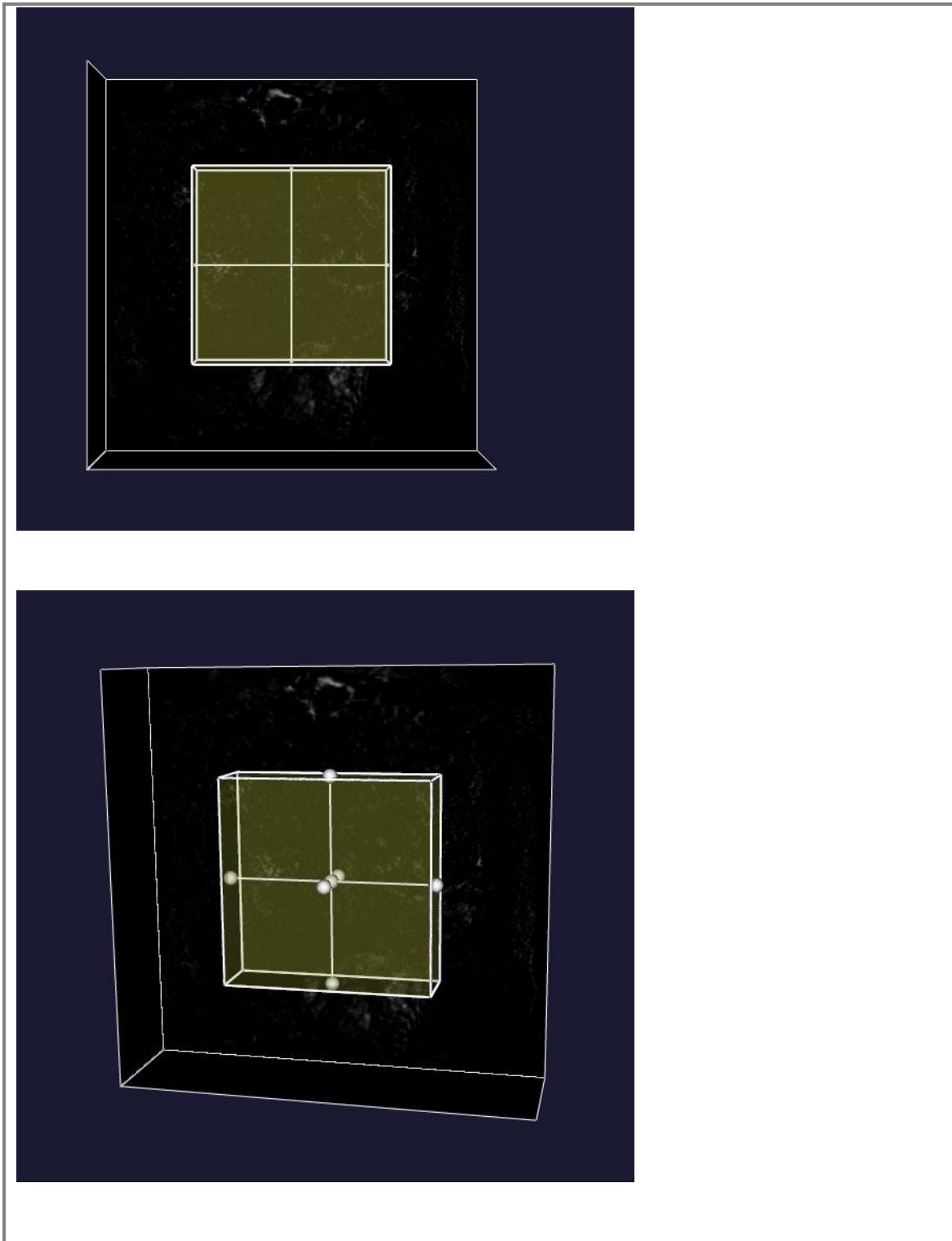
```
Id = 0
Image = vtkImageData
PlaneWidgetX = vtkImagePlaneWidget
PlaneWidgetY = vtkImagePlaneWidget
PlaneWidgetZ = vtkImagePlaneWidget
```

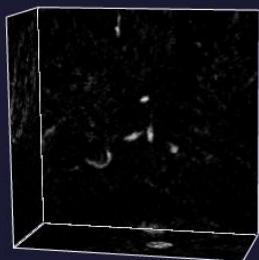


ANNEX 3: VMTKIMAGEVOISELECTOR

```
vmtk vmtkimagevoiselector -ifile dicomimages.vti -ofile voi.vti
Executing vmtkimagevoiselector -ifile dicomimages.vti -ofile voi.vti
Creating vmtkImageVOISelector instance.
Automatic piping vmtkimagevoiselector
Parsing options vmtkimagevoiselector
  ImageInputFileName = dicomimages.vti
  ImageOutputFileName = voi.vti
Explicit piping vmtkimagevoiselector
Input vmtkimagevoiselector members:
  Id = 0
  Disabled = 0
  Image = None
  ImageInputFileName = dicomimages.vti
  Interactive = 1
  BoxBounds = [0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
  vmtkRenderer = None
  ImageOutputFileName = voi.vti
Reading VTK XML image file.
Spacing (1.0026041667, 1.0026041667, 1.4999980927)
Origin (182.05859375, 32.531246185, 192.5)
Dimensions (384, 384, 72)
Executing vmtkimagevoiselector ...
```







Done executing vmtkimagevoiselector.
Writing VTK XML image file.
Output vmtkimagevoiselector members:
Id = 0
Image = vtkImageData

ANNEX 4: VMTKLEVELSETSEGMENTATION

```
vmtk vmtklevelsetsegmentation -ifile voi.vti -ofile levelset.vti
```

Executing vmtklevelsetsegmentation -ifile voi.vti -ofile levelset.vti

Creating vmtkLevelSetSegmentation instance.

Automatic piping vmtklevelsetsegmentation

Parsing options vmtklevelsetsegmentation

ImageInputFileName = voi.vti

LevelSetsOutputFileName = levelset.vti

Explicit piping vmtklevelsetsegmentation

Input vmtklevelsetsegmentation members:

Id = 0

Disabled = 0

Image = None

ImageInputFileName = voi.vti

FeatureImage = None

FeatureImageInputFileName =

InitializationImage = None

InitializationImageInputFileName =

InitialLevelSets = None

InitialLevelSetsInputFileName =

LevelSets = None

LevelSetsInputFileName =

LevelSetsType = geodesic

FeatureImageType = gradient

SigmoidRemapping = 0

IsoSurfaceValue = 0.0

DerivativeSigma = 0.0

FeatureDerivativeSigma = 0.0

UpwindFactor = 1.0

FWHMRadius = [1.0, 1.0, 1.0]

FWHMBackgroundValue = 0.0

NumberOflterations = 0

PropagationScaling = 0.0

CurvatureScaling = 0.0

AdvectionScaling = 1.0

EdgeWeight = 0.0

SmoothingIterations = 5

SmoothingTimeStep = 0.1

SmoothingConductance = 0.8

vmtkRenderer = None

LevelSetsOutputFileName = levelset.vti

FeatureImageOutputFileName =

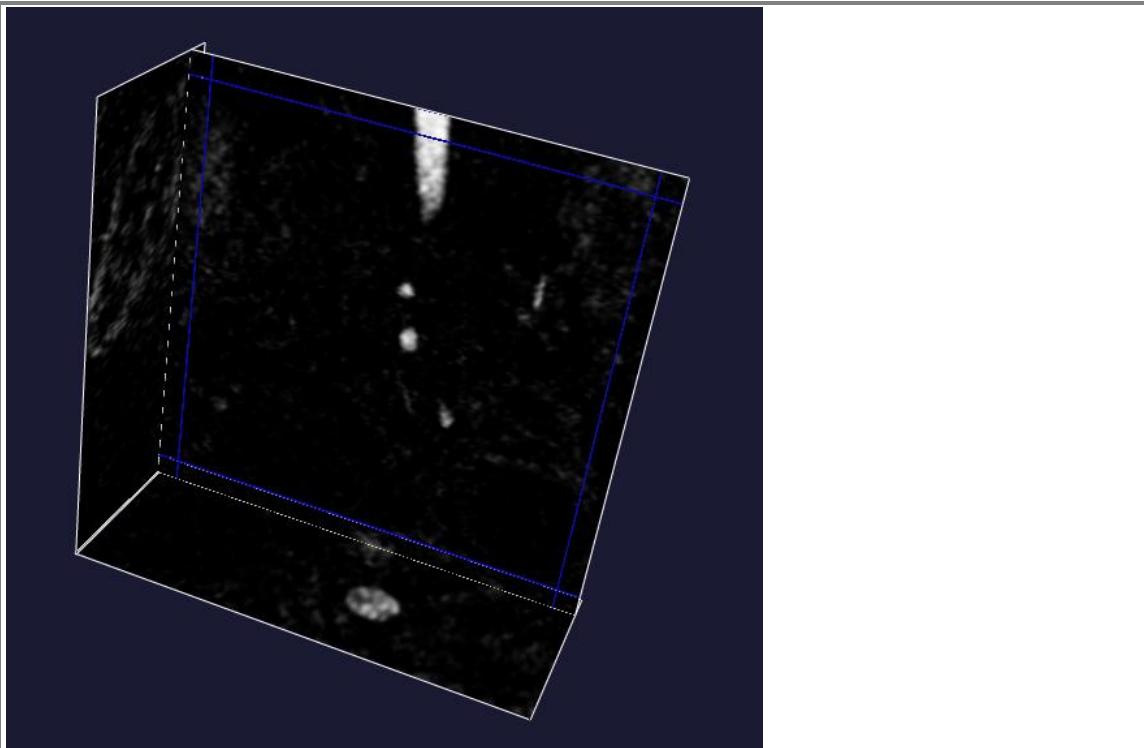
Reading VTK XML image file.

Spacing (1.0026041667, 1.0026041667, 1.4999980927)

Origin (182.05859375, 32.531246185, 192.5)

Dimensions (192, 192, 54)

Executing vmtklevelsetsegmentation ...



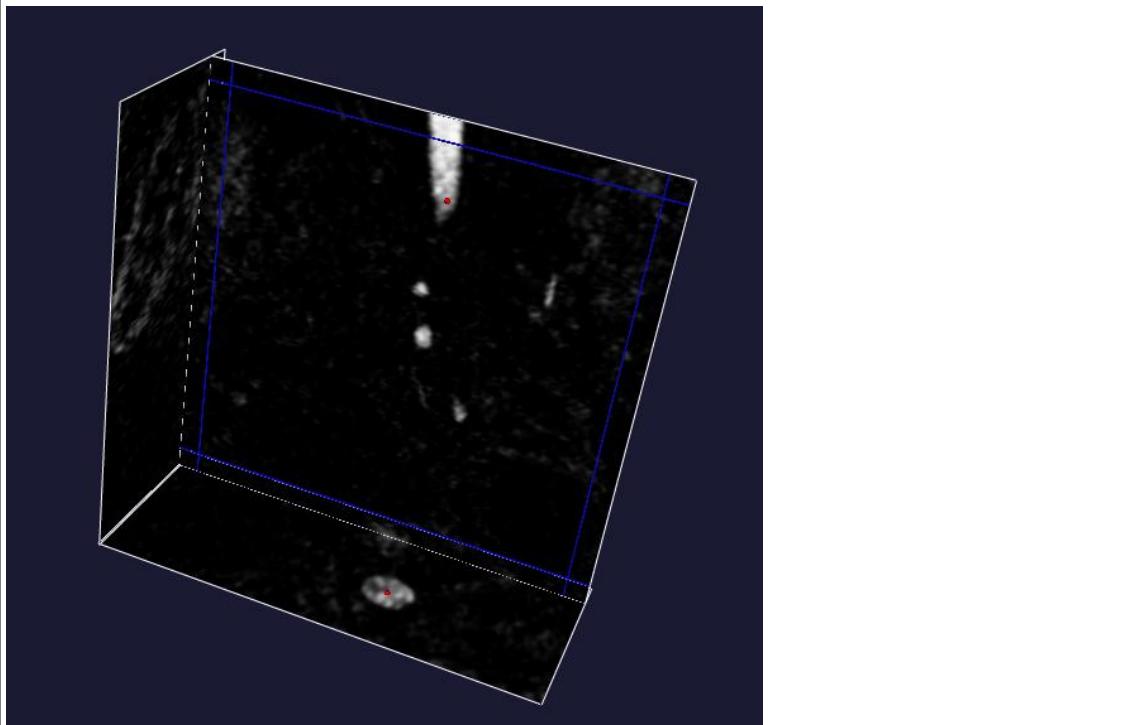
Please choose initialization type: (0: colliding fronts; 1: fast marching; 2: threshold; 3: isosurface, 4: seed): 0

Colliding fronts initialization.

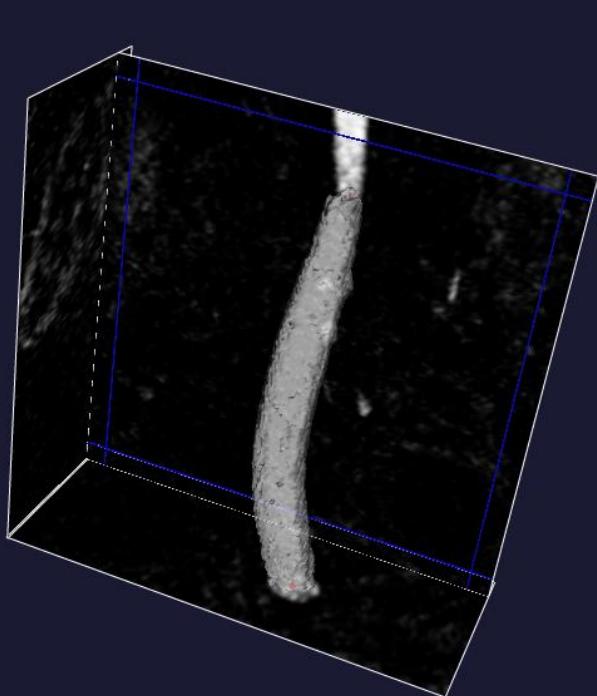
Please input lower threshold ('i' to activate image, 'n' for none): n

Please input upper threshold ('i' to activate image, 'n' for none): n

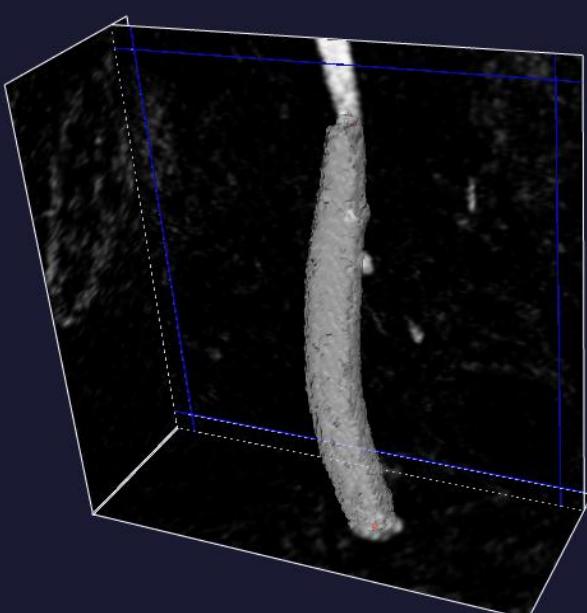
Please place two seeds (click on the image while pressing Ctrl).



Displaying.



Accept initialization? (y/n): y
Displaying.



Initialize another branch? (y/n): y

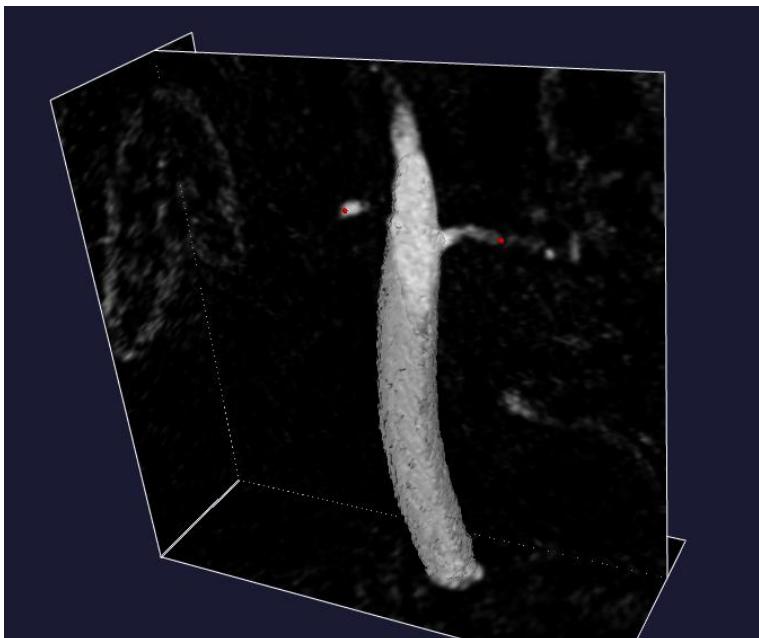
Please choose initialization type: (0: colliding fronts; 1: fast marching; 2: threshold; 3: isosurface, 4: seed): 0

Colliding fronts initialization.

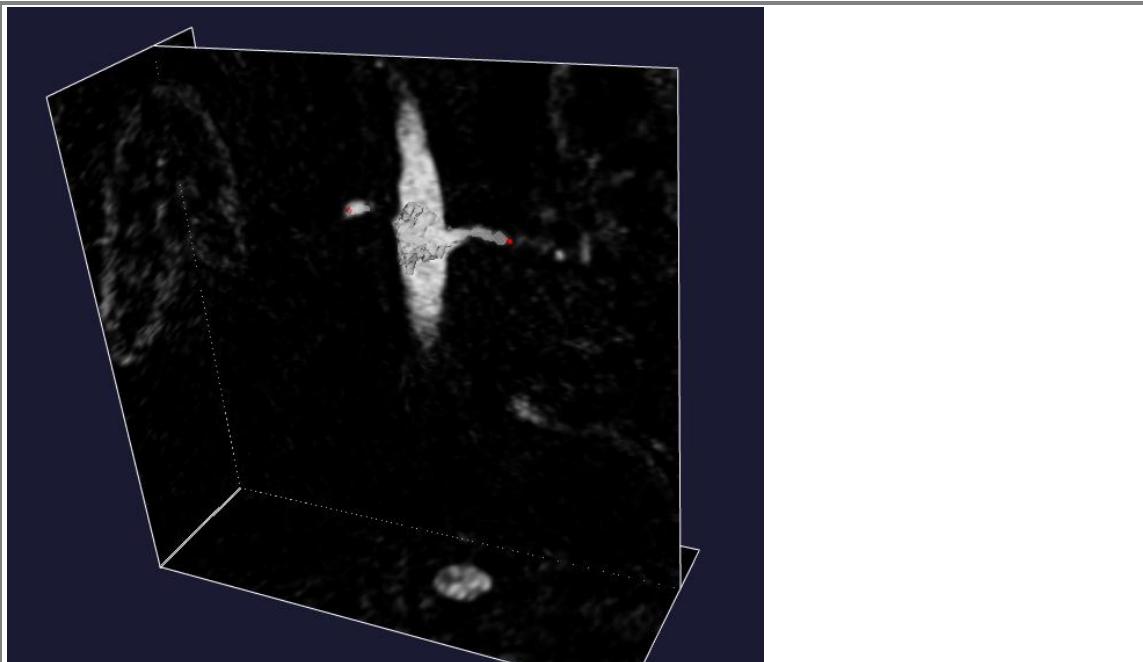
Please input lower threshold ('i' to activate image, 'n' for none): n

Please input upper threshold ('i' to activate image, 'n' for none): n

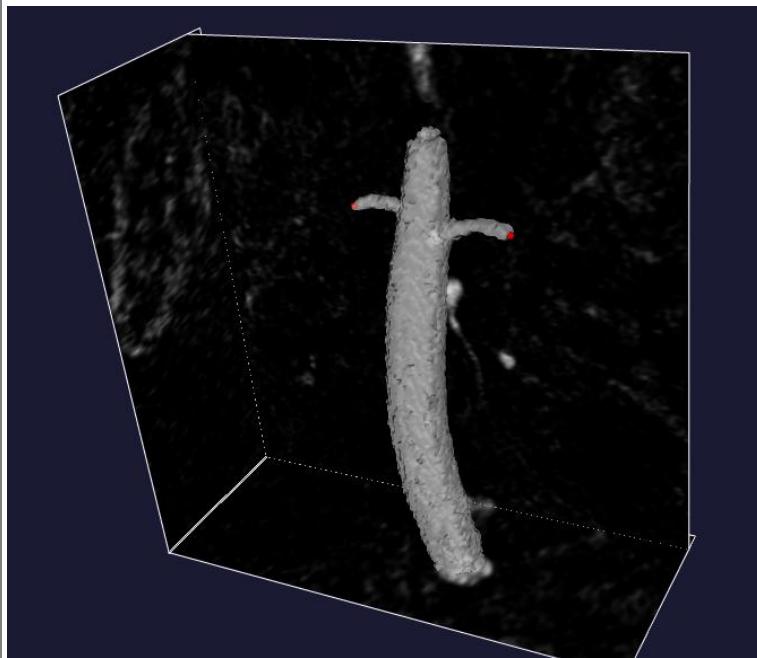
Please place two seeds (click on the image while pressing Ctrl).



Displaying.



Accept initialization? (y/n): y
Displaying.



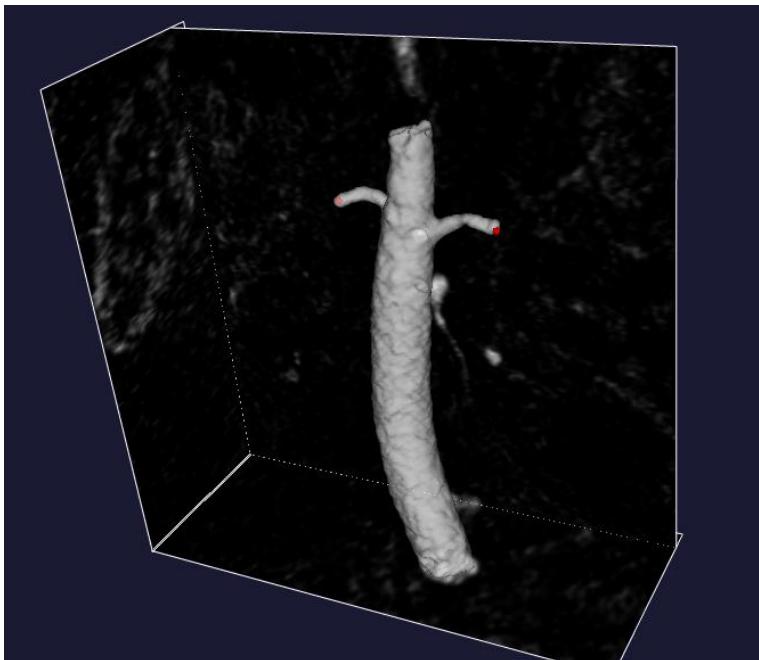
Initialize another branch? (y/n): n
Please input parameters (type return to accept current values, 'e' to end, 'q' to quit):

NumberOfIterations(0) [PropagationScaling(0.0) CurvatureScaling(0.0)
AdvectionScaling(1.0)]: 300 0 0 1

Progress: 100%
Displaying.



Accept result? (y/n): y
Merge branch? (y/n): y



Displaying.
Segment another branch? (y/n): n

```
Done executing vmtklevelsetsegmentation.  
Writing VTK XML image file.  
Output vmtklevelsetsegmentation members:  
  Id = 0  
  LevelSets = vtkImageData  
  FeatureImage = vtkImageData
```

ANNEX 5: VMTKMARCHINGCUBES

```
vmtk vmtkmarchingcubes -ifile levelset.vti -ofile marching.vtp --pipe
vmtksurfaceviewer
Executing vmtkmarchingcubes -ifile levelset.vti -ofile marching.vtp --pipe
vmtksurfaceviewer

Creating vmtkMarchingCubes instance.
Automatic piping vmtkmarchingcubes
Parsing options vmtkmarchingcubes
  ImageInputFileName = levelset.vti
  SurfaceOutputFileName = marching.vtp
Explicit piping vmtkmarchingcubes
Input vmtkmarchingcubes members:
  Id = 0
  Disabled = 0
  Image = None
  ImageInputFileName = levelset.vti
  ArrayName =
  Level = 0.0
  Connectivity = 0
  SurfaceOutputFileName = marching.vtp
Reading VTK XML image file.
Spacing (1.0026041667, 1.0026041667, 1.4999980927)
Origin (182.05859375, 32.531246185, 192.5)
Dimensions (192, 192, 54)
Executing vmtkmarchingcubes ...
Done executing vmtkmarchingcubes.
Writing VTK XML surface file.
Output vmtkmarchingcubes members:
  Id = 0
  Surface = vtkPolyData

Creating vmtkSurfaceViewer instance.
Automatic piping vmtksurfaceviewer
Parsing options vmtksurfaceviewer
  SurfaceInputFileName = marching.vtp
Explicit piping vmtksurfaceviewer
Input vmtksurfaceviewer members:
  Id = 0
  Disabled = 0
  Surface = None
  SurfaceInputFileName = marching.vtp
  vmtkRenderer = None
  Display = 1
  Opacity = 1.0
  ArrayName =
  ScalarRange = [0.0, 0.0]
  Legend = 0
  Grayscale = 0
  FlatInterpolation = 0
  DisplayCellData = 0
```

```
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName =
Reading VTK XML surface file.
Executing vmtksurfaceviewer ...
```

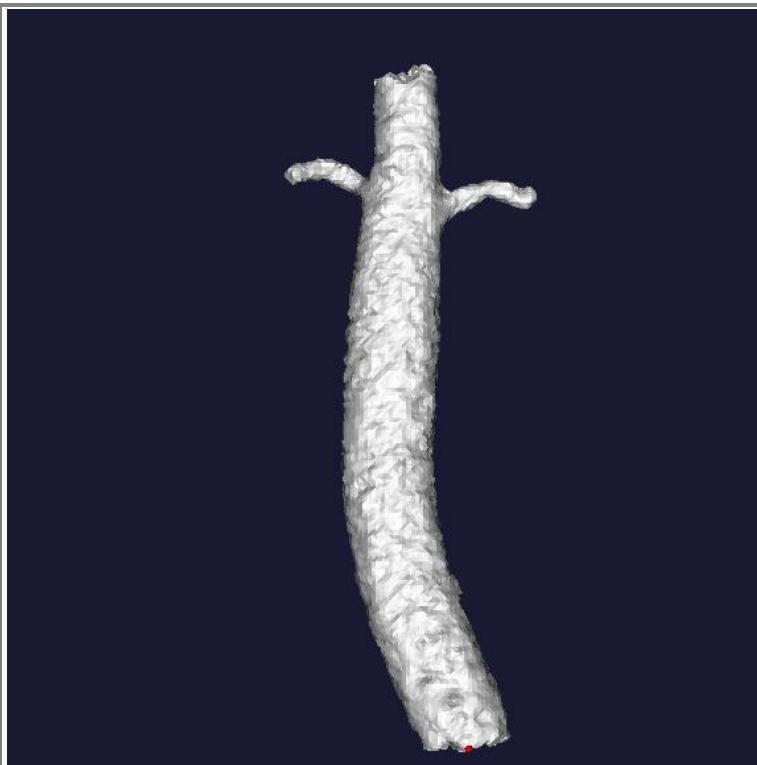


```
Done executing vmtksurfaceviewer.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor
```

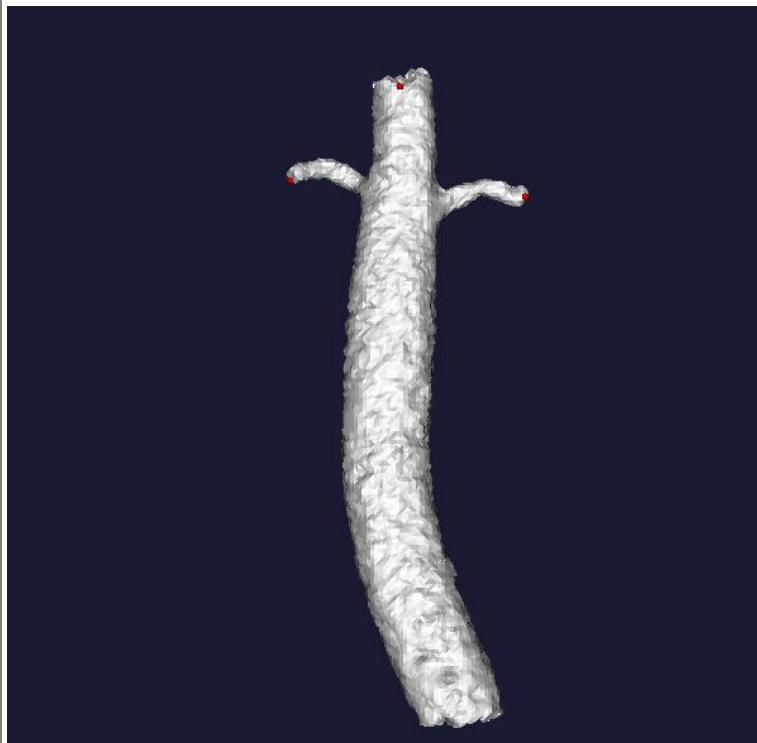
ANNEX 6: VMTKCENTERLINES

```
vmtk vmtkcenterlines -ifile marching.vtp -ofile centerlinesauto.vtp --pipe
vmtksurfaceviewer
Executing vmtkcenterlines -ifile marching.vtp -ofile centerlinesauto.vtp --pipe
vmtksurfaceviewer

Creating vmtkCenterlines instance.
Automatic piping vmtkcenterlines
Parsing options vmtkcenterlines
    SurfaceInputFileName = marching.vtp
    CenterlinesOutputFileName = centerlinesauto.vtp
Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
    Id = 0
    Disabled = 0
    Surface = None
    SurfaceInputFileName = marching.vtp
    SeedSelectorName = pickpoint
    SourceIds = []
    TargetIds = []
    SourcePoints = []
    TargetPoints = []
    AppendEndPoints = 0
    CheckNonManifold = 0
    FlipNormals = 0
    CapDisplacement = 0.0
    RadiusArrayName = MaximumInscribedSphereRadius
    AppendEndPoints = 0
    Resampling = 0
    ResamplingStepLength = 1.0
    DelaunayTessellation = None
    SimplifyVoronoi = 0
    UseTetGen = 0
    TetGenDetectInter = 1
    CostFunction = 1/R
    vmtkRenderer = None
    CenterlinesOutputFileName = centerlinesauto.vtp
    DelaunayTessellationOutputFileName =
    VoronoiDiagramOutputFileName =
Reading VTK XML surface file.
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
```



Please position the mouse and press space to add target points, 'u' to undo



Computing centerlines.
Done executing vmtkcenterlines.

Writing VTK XML surface file.

Output vmtkcenterlines members:

```
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList
```

Creating vmtkSurfaceViewer instance.

Automatic piping vmtksurfaceviewer

Parsing options vmtksurfaceviewer

```
SurfaceInputFileName = centerlinesauto.vtp
```

Explicit piping vmtksurfaceviewer

Input vmtksurfaceviewer members:

```
Id = 0
Disabled = 0
Surface = None
SurfaceInputFileName = centerlinesauto.vtp
vmtkRenderer = None
Display = 1
Opacity = 1.0
ArrayName =
ScalarRange = [0.0, 0.0]
Legend = 0
Grayscale = 0
FlatInterpolation = 0
DisplayCellData = 0
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName =
Reading VTK XML surface file.
Executing vmtksurfaceviewer ...
```



Done executing vmtksurfaceviewer.
 Output vmtksurfaceviewer members:
 Id = 0
 Surface = vtkPolyData
 Actor = vtkActor

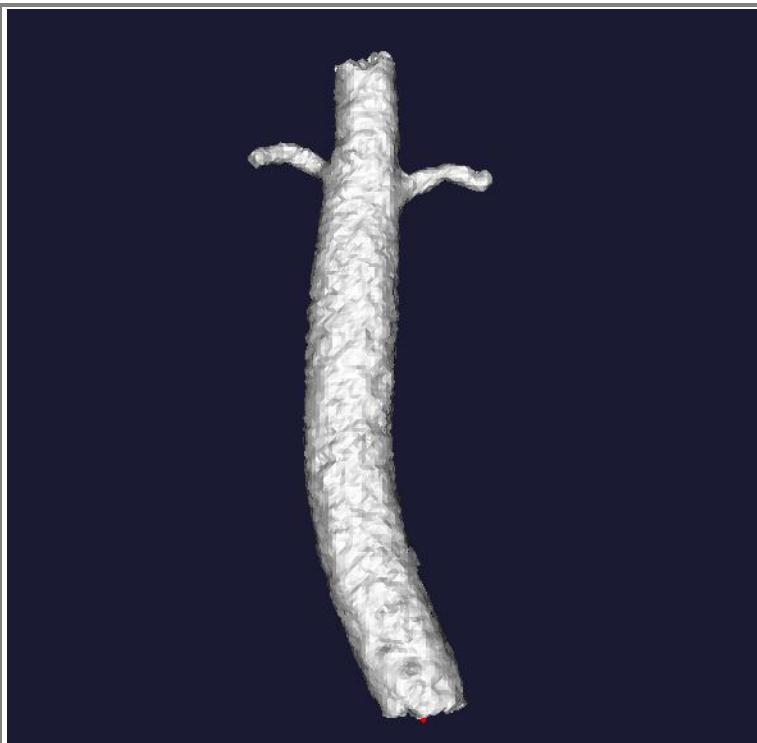
En el següent cas, es mostren les esferes inscrites màximes a partir del Diagrama de Voronoi.

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtksurfaceviewer -i @vmtkcenterlines.voronoidiagram -array
MaximumInscribedSphereRadius
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtksurfaceviewer -i @vmtkcenterlines.voronoidiagram -array
MaximumInscribedSphereRadius

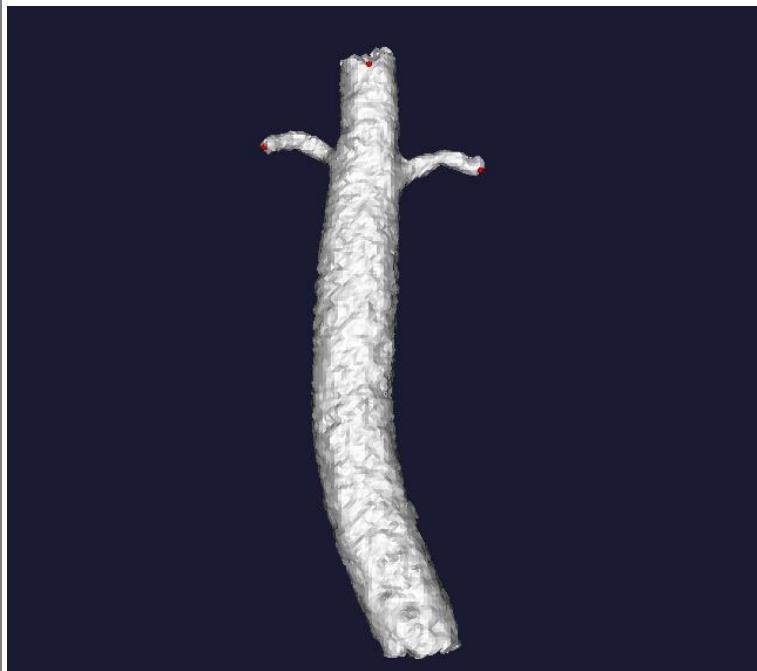
Creating vmtkSurfaceReader instance.
Automatic piping vmtksurfacereader
Parsing options vmtksurfacereader
  InputFileName = marching.vtp
Explicit piping vmtksurfacereader
Input vmtksurfacereader members:
  Id = 0
  Disabled = 0
  Format =
  GuessFormat = 1
```

```
Surface = 0
InputFileName = marching.vtp
SurfaceOutputFileName =
Executing vmtksurfacereader ...
Reading VTK XML surface file.
Done executing vmtksurfacereader.
Output vmtksurfacereader members:
Id = 0
Surface = vtkPolyData

Creating vmtkCenterlines instance.
Automatic piping vmtkcenterlines
    Surface = vmtksurfacereader-0.Surface
Parsing options vmtkcenterlines
Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
Id = 0
Disabled = 0
Surface = vtkPolyData
SurfaceInputFileName =
SeedSelectorName = pickpoint
SourceIds = []
TargetIds = []
SourcePoints = []
TargetPoints = []
AppendEndPoints = 0
CheckNonManifold = 0
FlipNormals = 0
CapDisplacement = 0.0
RadiusArrayName = MaximumInscribedSphereRadius
AppendEndPoints = 0
Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
```



Please position the mouse and press space to add target points, 'u' to undo



Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData

```
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList
```

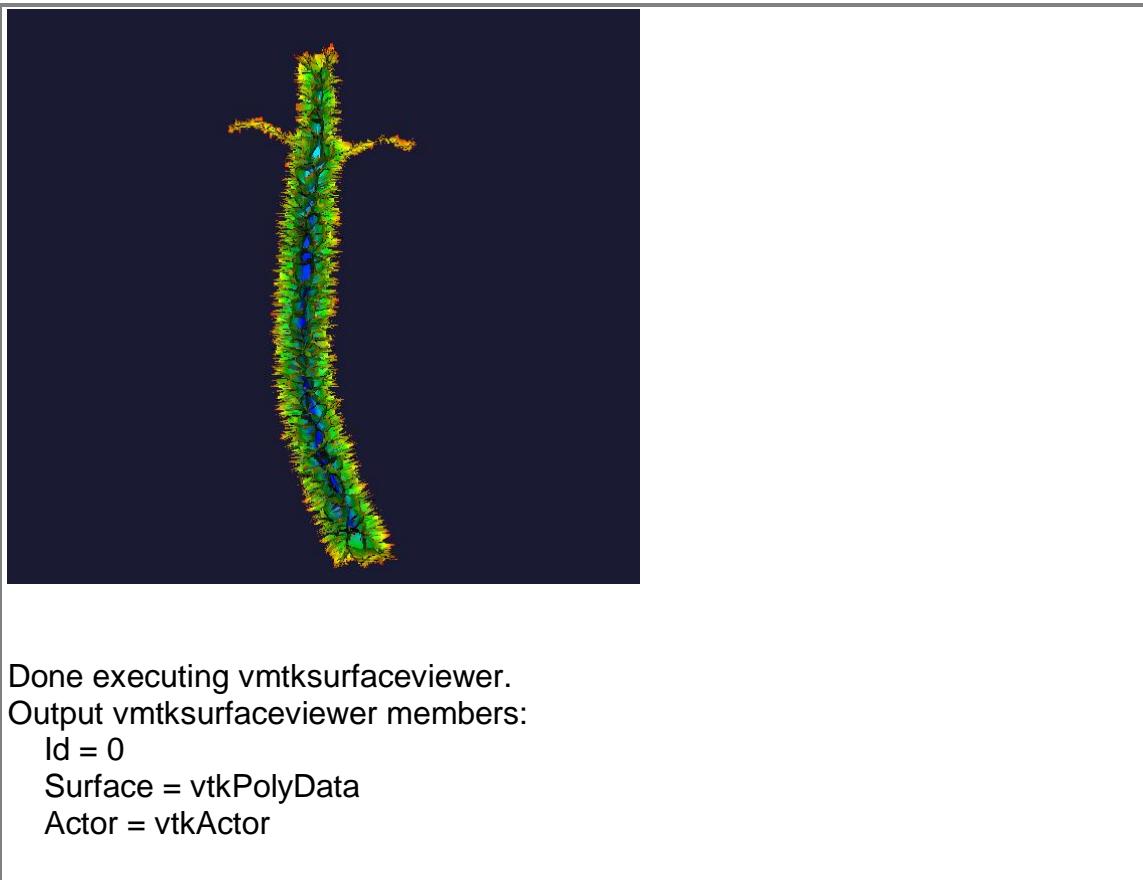
Creating vmtkSurfaceViewer instance.

```
Automatic piping vmtksurfaceviewer
  Surface = vmtksurfacereader-0.Surface
Parsing options vmtksurfaceviewer
  Surface = @vmtkcenterlines.voronoidiagram
  ArrayName = MaximumInscribedSphereRadius
```

```
Explicit piping vmtksurfaceviewer
  Surface = vmtkcenterlines-0.VoronoiDiagram
```

Input vmtksurfaceviewer members:

```
  Id = 0
  Disabled = 0
  Surface = vtkPolyData
  SurfaceInputFileName =
  vmtkRenderer = None
  Display = 1
  Opacity = 1.0
  ArrayName = MaximumInscribedSphereRadius
  ScalarRange = [0.0, 0.0]
  Legend = 0
  Grayscale = 0
  FlatInterpolation = 0
  DisplayCellData = 0
  Color = [-1.0, -1.0, -1.0]
  LineWidth = 1
  LegendTitle =
  SurfaceOutputFileName =
Executing vmtksurfaceviewer ...
```



Done executing vmtksurfaceviewer.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor

ANNEX 7: VMTKSURFACESMOOTHING

```
vmtk vmtksurfacesmoothing -ifile marching.vtp -passband 0.1 -iterations  
30 --pipe vmtksurfaceviewer -ofile smoothing.vtp  
Executing vmtksurfacesmoothing -ifile marching.vtp -passband 0.1 -iterations  
30 --pipe vmtksurfaceviewer -ofile smoothing.vtp
```

Creating vmtkSurfaceSmoothing instance.

Automatic piping vmtksurfacesmoothing

Parsing options vmtksurfacesmoothing

 SurfaceInputFileName = marching.vtp

 NumberOfIterations = 30

 PassBand = 0.1

Explicit piping vmtksurfacesmoothing

Input vmtksurfacesmoothing members:

 Id = 0

 Disabled = 0

 Surface = None

 SurfaceInputFileName = marching.vtp

 NumberOfIterations = 30

 Method = taubin

 PassBand = 0.1

 RelaxationFactor = 0.01

 BoundarySmoothing = 1

 SurfaceOutputFileName =

Reading VTK XML surface file.

Executing vmtksurfacesmoothing ...

Done executing vmtksurfacesmoothing.

Output vmtksurfacesmoothing members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkSurfaceViewer instance.

Automatic piping vmtksurfaceviewer

 Surface = vmtksurfacesmoothing-0.Surface

Parsing options vmtksurfaceviewer

 SurfaceOutputFileName = smoothing.vtp

Explicit piping vmtksurfaceviewer

Input vmtksurfaceviewer members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 vmtkRenderer = None

 Display = 1

 Opacity = 1.0

 ArrayName =

 ScalarRange = [0.0, 0.0]

 Legend = 0

 Grayscale = 0

 FlatInterpolation = 0

```
DisplayCellData = 0
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName = smoothing.vtp
Executing vmtksurfaceviewer ...
```



```
Done executing vmtksurfaceviewer.
Writing VTK XML surface file.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor
```

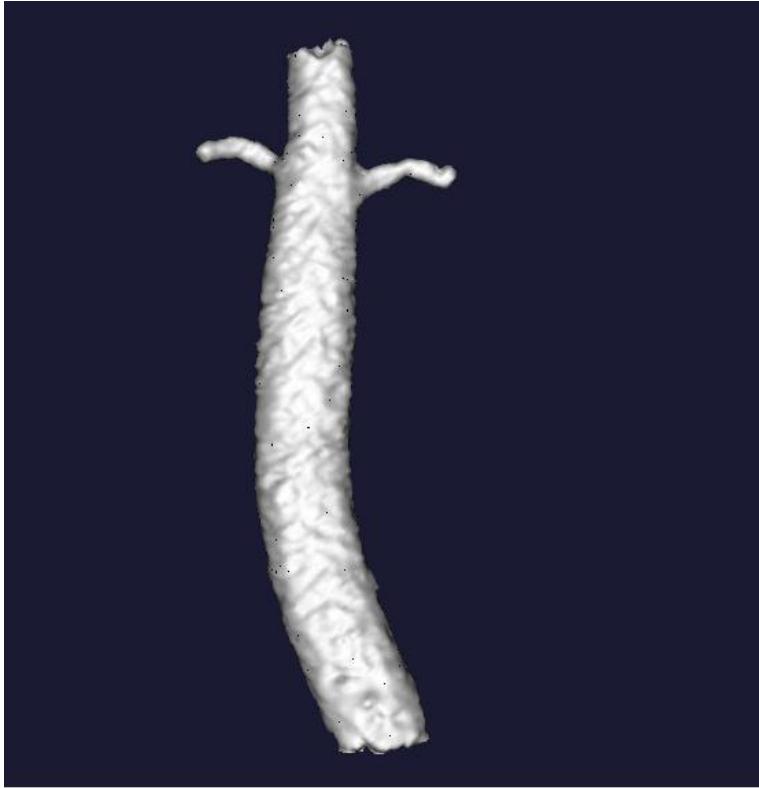
ANNEX 8: VMTKSUBDIVISION

```
vmtk vmtksurfacesubdivision -ifile marching.vtp -method butterfly --pipe
vmtksurfaceviewer -ofile sdivision.vtp
Executing vmtksurfacesubdivision -ifile marching.vtp -method butterfly --pipe
vmtksurfaceviewer -ofile sdivision.vtp

Creating vmtkSurfaceSubdivision instance.
Automatic piping vmtksurfacesubdivision
Parsing options vmtksurfacesubdivision
    SurfaceInputFileName = marching.vtp
    Method = butterfly
Explicit piping vmtksurfacesubdivision
Input vmtksurfacesubdivision members:
    Id = 0
    Disabled = 0
    Surface = None
    SurfaceInputFileName = marching.vtp
    NumberOfSubdivisions = 1
    Method = butterfly
    SurfaceOutputFileName =
Reading VTK XML surface file.
Executing vmtksurfacesubdivision ...
Done executing vmtksurfacesubdivision.
Output vmtksurfacesubdivision members:
    Id = 0
    Surface = vtkPolyData

Creating vmtkSurfaceViewer instance.
Automatic piping vmtksurfaceviewer
    Surface = vmtksurfacesubdivision-0.Surface
Parsing options vmtksurfaceviewer
    SurfaceOutputFileName = sdivision.vtp
Explicit piping vmtksurfaceviewer
Input vmtksurfaceviewer members:
    Id = 0
    Disabled = 0
    Surface = vtkPolyData
    SurfaceInputFileName =
    vmtkRenderer = None
    Display = 1
    Opacity = 1.0
    ArrayName =
    ScalarRange = [0.0, 0.0]
    Legend = 0
    Grayscale = 0
    FlatInterpolation = 0
    DisplayCellData = 0
    Color = [-1.0, -1.0, -1.0]
    LineWidth = 1
    LegendTitle =
    SurfaceOutputFileName = sdivision.vtp
```

Executing vmtksurfaceviewer ...



Done executing vmtksurfaceviewer.

Writing VTK XML surface file.

Output vmtksurfaceviewer members:

Id = 0

Surface = vtkPolyData

Actor = vtkActor

ANNEX 9: VMTKSURFACLIPPER

```
vmtk vmtksurfaceclipper -ifile marching.vtp --pipe vmtksurfaceviewer -  
ofile clipper.vtp
```

Executing vmtksurfaceclipper -ifile marching.vtp --pipe vmtksurfaceviewer -ofile
clipper.vtp

Creating vmtkSurfaceClipper instance.

Automatic piping vmtksurfaceclipper

Parsing options vmtksurfaceclipper

 SurfaceInputFileName = marching.vtp

Explicit piping vmtksurfaceclipper

Input vmtksurfaceclipper members:

 Id = 0

 Disabled = 0

 Surface = None

 SurfaceInputFileName = marching.vtp

 WidgetType = box

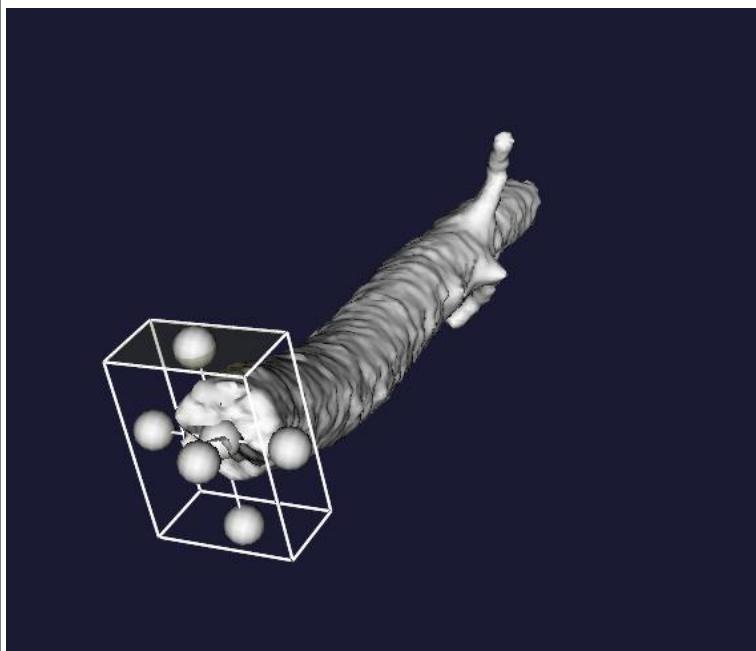
 CleanOutput = 1

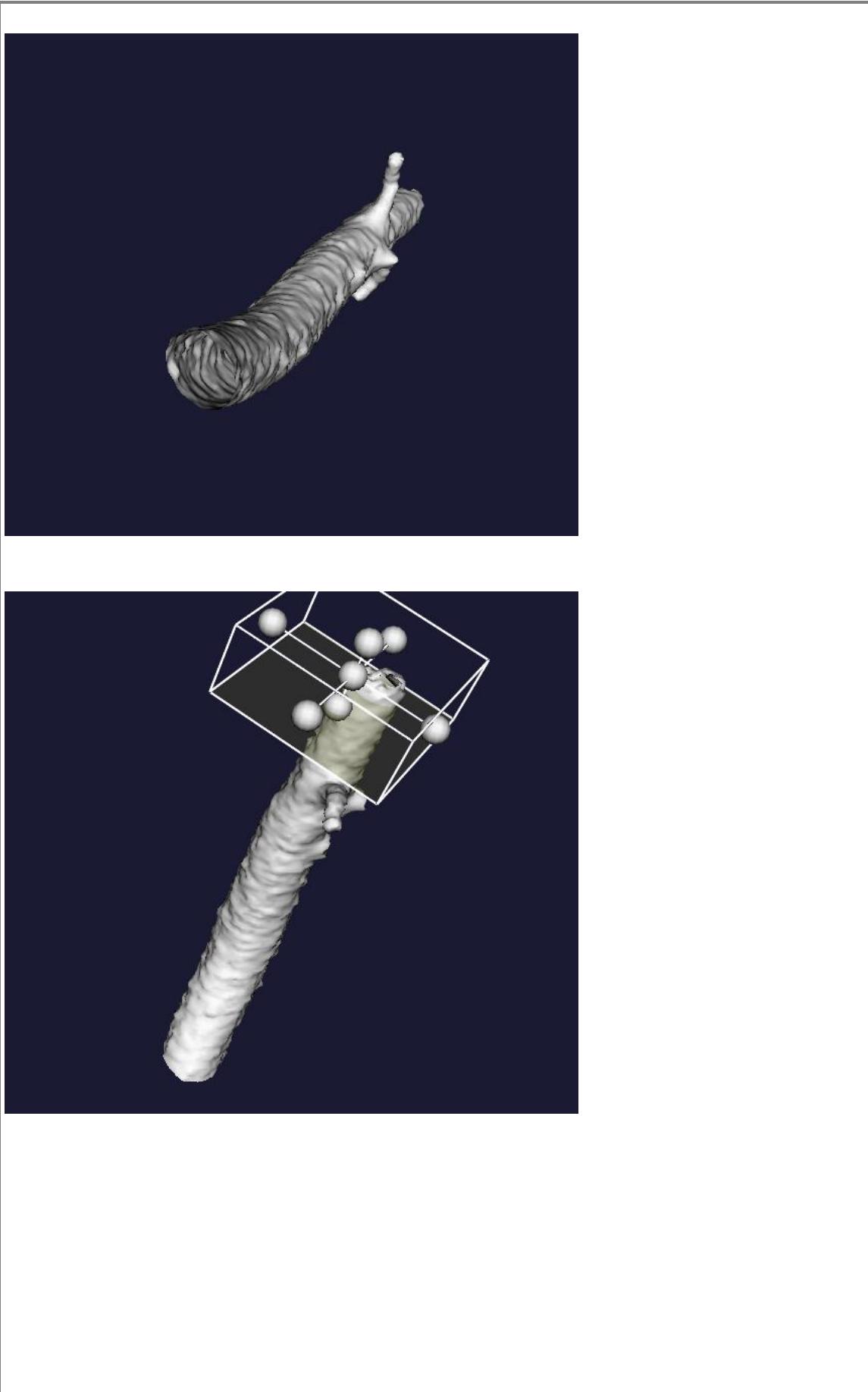
 vmtkRenderer = None

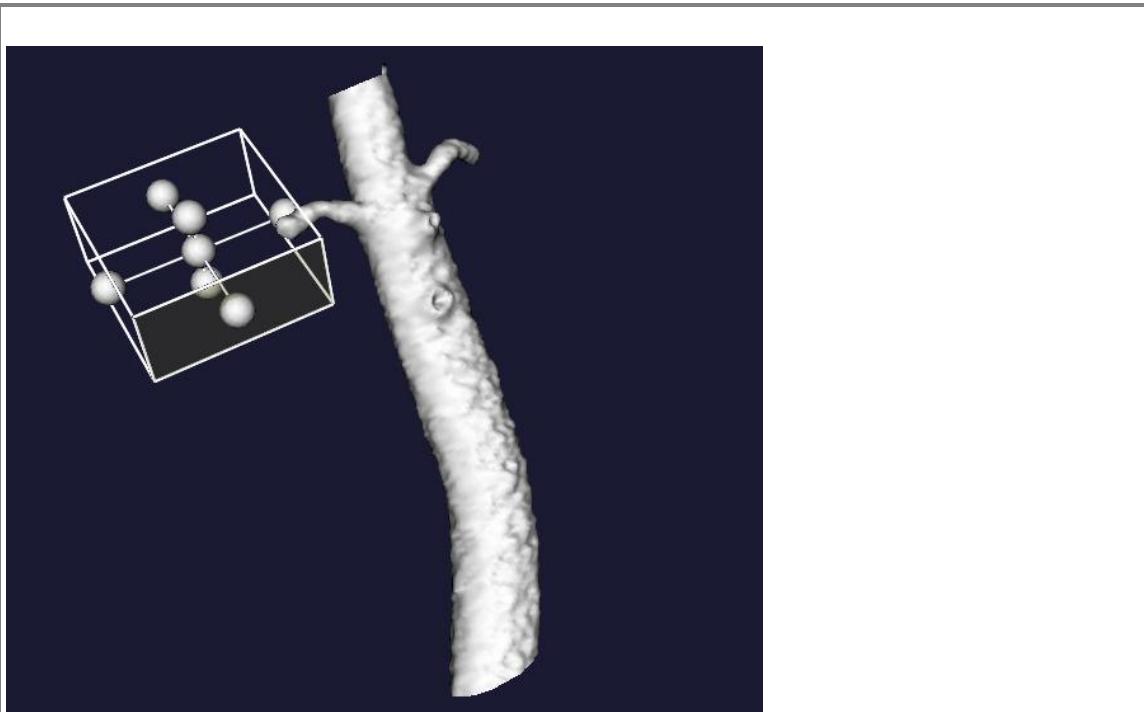
 SurfaceOutputFileName =

Reading VTK XML surface file.

Executing vmtksurfaceclipper ...







Done executing vmtksurfaceclipper.

Output vmtksurfaceclipper members:

Id = 0

Surface = vtkPolyData

Creating vmtkSurfaceViewer instance.

Automatic piping vmtksurfaceviewer

Surface = vmtksurfaceclipper-0.Surface

Parsing options vmtksurfaceviewer

SurfaceOutputFileName = sclipper.vtp

Explicit piping vmtksurfaceviewer

Input vmtksurfaceviewer members:

Id = 0

Disabled = 0

Surface = vtkPolyData

SurfaceInputFileName =

vmtkRenderer = None

Display = 1

Opacity = 1.0

ArrayName =

ScalarRange = [0.0, 0.0]

Legend = 0

Grayscale = 0

FlatInterpolation = 0

DisplayCellData = 0

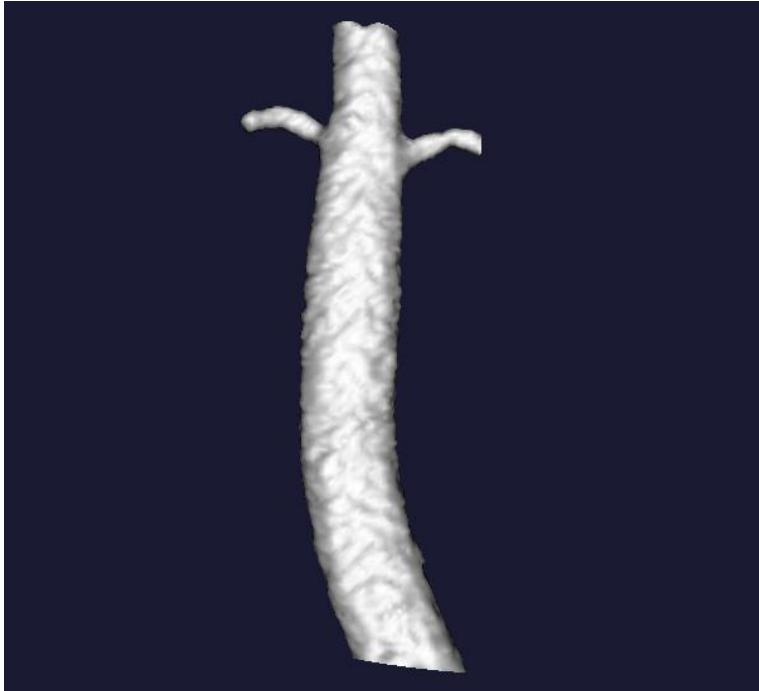
Color = [-1.0, -1.0, -1.0]

LineWidth = 1

LegendTitle =

SurfaceOutputFileName = sclipper.vtp

Executing vmtksurfaceviewer ...



Done executing vmtksurfaceviewer.

Writing VTK XML surface file.

Output vmtksurfaceviewer members:

Id = 0

Surface = vtkPolyData

Actor = vtkActor

ANNEX 10: VMTKFLOWEXTENSIONS

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtksurfaceclipper --  
pipe vmtkcenterlines --pipe vmtkflowextensions -adaptivelength 1 -  
extensionratio 10 -normalestimationratio 1 -interactive 0 --pipe  
vmtksurfaceviewer -ofile flowex.vtp
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtksurfaceclipper --pipe
vmtkcenterlines --pipe vmtkflowextensions -adaptivelength 1 -extensionratio 10
-normalestimationratio 1 -interactive 0 --pipe vmtksurfaceviewer -ofile flowex.vtp

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkSurfaceClipper instance.

Automatic piping vmtksurfaceclipper

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtksurfaceclipper

Explicit piping vmtksurfaceclipper

Input vmtksurfaceclipper members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 WidgetType = box

 CleanOutput = 1

 vmtkRenderer = None

 SurfaceOutputFileName =

Executing vmtksurfaceclipper ...

Done executing vmtksurfaceclipper.

Output vmtksurfaceclipper members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

```
Surface = vmtksurfaceclipper-0.Surface
Parsing options vmtkcenterlines
Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
Id = 0
Disabled = 0
Surface = vtkPolyData
SurfaceInputFileName =
SeedSelectorName = pickpoint
SourceIds = []
TargetIds = []
SourcePoints = []
TargetPoints = []
AppendEndPoints = 0
CheckNonManifold = 0
FlipNormals = 0
CapDisplacement = 0.0
RadiusArrayName = MaximumInscribedSphereRadius
AppendEndPoints = 0
Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList
```

```
Creating vmtkFlowExtensions instance.  
Automatic piping vmtkflowextensions  
    Surface = vmtksurfaceclipper-0.Surface  
    Centerlines = vmtkcenterlines-0.Centerlines  
Parsing options vmtkflowextensions  
    AdaptiveExtensionLength = 1  
    ExtensionRatio = 10.0  
    Interactive = 0  
    CenterlineNormalEstimationDistanceRatio = 1.0  
Explicit piping vmtkflowextensions  
Input vmtkflowextensions members:  
    Id = 0  
    Disabled = 0  
    Surface = vtkPolyData  
    SurfaceInputFileName =  
    Centerlines = vtkPolyData  
    CenterlinesInputFileName =  
    ExtensionMode = centerlinedirection  
    InterpolationMode = thinplatespline  
    Sigma = 1.0  
    AdaptiveExtensionLength = 1  
    AdaptiveExtensionRadius = 1  
    AdaptiveNumberOfBoundaryPoints = 0  
    ExtensionLength = 1.0  
    ExtensionRatio = 10.0  
    ExtensionRadius = 1.0  
    TransitionRatio = 0.25  
    TargetNumberOfBoundaryPoints = 50  
    Interactive = 0  
    vmtkRenderer = None  
    CenterlineNormalEstimationDistanceRatio = 1.0  
    SurfaceOutputFileName =  
Executing vmtkflowextensions ...  
Done executing vmtkflowextensions.  
Output vmtkflowextensions members:  
    Id = 0  
    Surface = vtkPolyData  
    Centerlines = vtkPolyData  
  
Creating vmtkSurfaceViewer instance.  
Automatic piping vmtksurfaceviewer  
    Surface = vmtkflowextensions-0.Surface  
Parsing options vmtksurfaceviewer  
    SurfaceOutputFileName = flowex.vtp  
Explicit piping vmtksurfaceviewer  
Input vmtksurfaceviewer members:  
    Id = 0  
    Disabled = 0  
    Surface = vtkPolyData  
    SurfaceInputFileName =
```

```
vmtkRenderer = None
Display = 1
Opacity = 1.0
ArrayName =
ScalarRange = [0.0, 0.0]
Legend = 0
Grayscale = 0
FlatInterpolation = 0
DisplayCellData = 0
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName = flowex.vtp
Executing vmtksurfaceviewer ...
```





Done executing vmtksurfaceviewer.
Writing VTK XML surface file.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor

ANNEX 11: VMTKBRANCHEXTRACTOR

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor --pipe vmtkcenterlineviewer -cellarray CenterlineIds -
-pipe vmtkcenterlineviewer -cellarray TractIds --pipe vmtkcenterlineviewer
-cellarray GroupIds --pipe vmtkcenterlineviewer -cellarray Blanking
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor --pipe vmtkcenterlineviewer -cellarray CenterlineIds --pipe
vmtkcenterlineviewer -cellarray TractIds --pipe vmtkcenterlineviewer -cellarray
GroupIds --pipe vmtkcenterlineviewer -cellarray Blanking -ofile bextractor.vtp

Creating vmtkSurfaceReader instance.
Automatic piping vmtksurfacereader
Parsing options vmtksurfacereader
    InputFileName = marching.vtp
Explicit piping vmtksurfacereader
Input vmtksurfacereader members:
    Id = 0
    Disabled = 0
    Format =
    GuessFormat = 1
    Surface = 0
    InputFileName = marching.vtp
    SurfaceOutputFileName =
Executing vmtksurfacereader ...
Reading VTK XML surface file.
Done executing vmtksurfacereader.
Output vmtksurfacereader members:
    Id = 0
    Surface = vtkPolyData

Creating vmtkCenterlines instance.
Automatic piping vmtkcenterlines
    Surface = vmtksurfacereader-0.Surface
Parsing options vmtkcenterlines
Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
    Id = 0
    Disabled = 0
    Surface = vtkPolyData
    SurfaceInputFileName =
    SeedSelectorName = pickpoint
    SourceIds = []
    TargetIds = []
    SourcePoints = []
    TargetPoints = []
    AppendEndPoints = 0
    CheckNonManifold = 0
    FlipNormals = 0
    CapDisplacement = 0.0
    RadiusArrayName = MaximumInscribedSphereRadius
    AppendEndPoints = 0
```

```

Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
  Centerlines = vmtkcenterlines-0.Centerlines
  RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.

```

Output vmtkbranchextractor members:

Id = 0
Centerlines = vtkPolyData
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking

Creating vmtkCenterlineViewer instance.

Automatic piping vmtkcenterlineviewer
Centerlines = vmtkbranchextractor-0.Centerlines

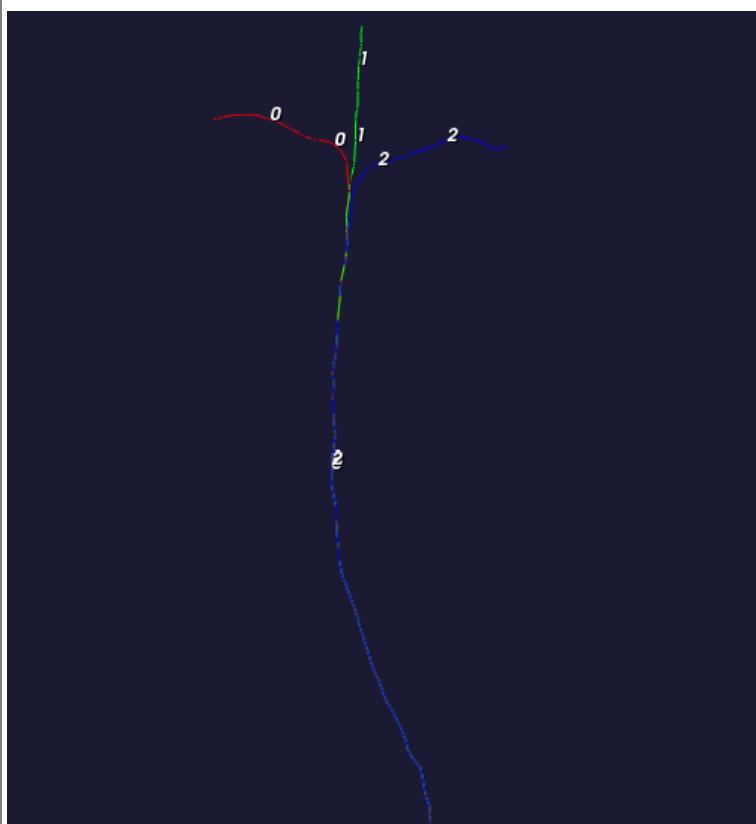
Parsing options vmtkcenterlineviewer

CellDataArrayName = CenterlineIds

Explicit piping vmtkcenterlineviewer

Input vmtkcenterlineviewer members:

Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
PointDataArrayName =
CellDataArrayName = CenterlineIds
Legend = 1
vmtkRenderer = None
CenterlinesOutputFileName =
Executing vmtkcenterlineviewer ...

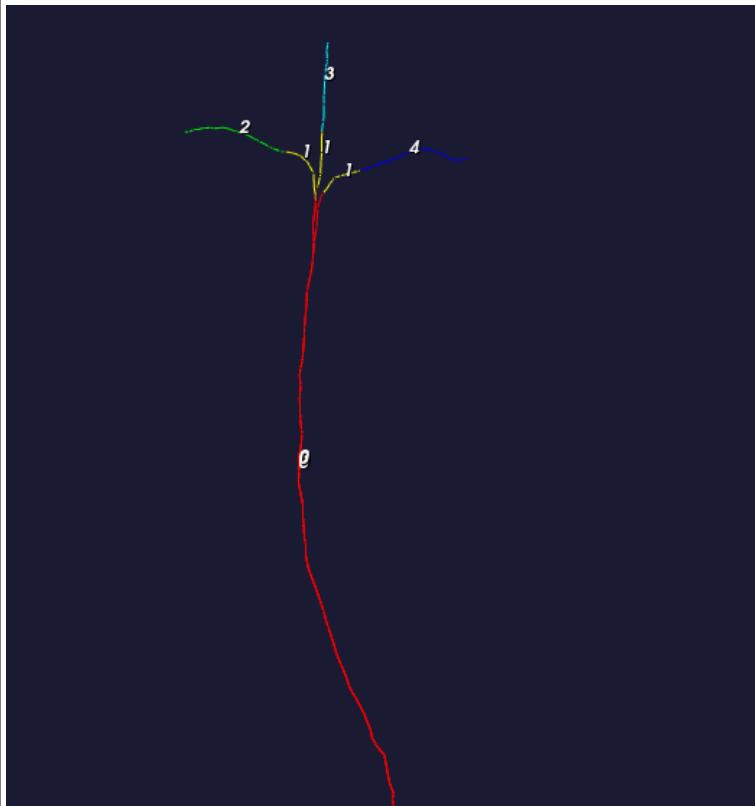


```
Done executing vmtkcenterlineviewer.  
Output vmtkcenterlineviewer members:  
  Id = 0  
  Centerlines = vtkPolyData  
  
Creating vmtkCenterlineViewer instance.  
Automatic piping vmtkcenterlineviewer  
  Centerlines = vmtkcenterlineviewer-0.Centerlines  
Parsing options vmtkcenterlineviewer  
  CellDataArrayName = TractIds  
Explicit piping vmtkcenterlineviewer  
Input vmtkcenterlineviewer members:  
  Id = 0  
  Disabled = 0  
  Centerlines = vtkPolyData  
  CenterlinesInputFileName =  
  PointDataArrayName =  
  CellDataArrayName = TractIds  
  Legend = 1  
  vmtkRenderer = None  
  CenterlinesOutputFileName =  
Executing vmtkcenterlineviewer ...
```



```
Done executing vmtkcenterlineviewer.
```

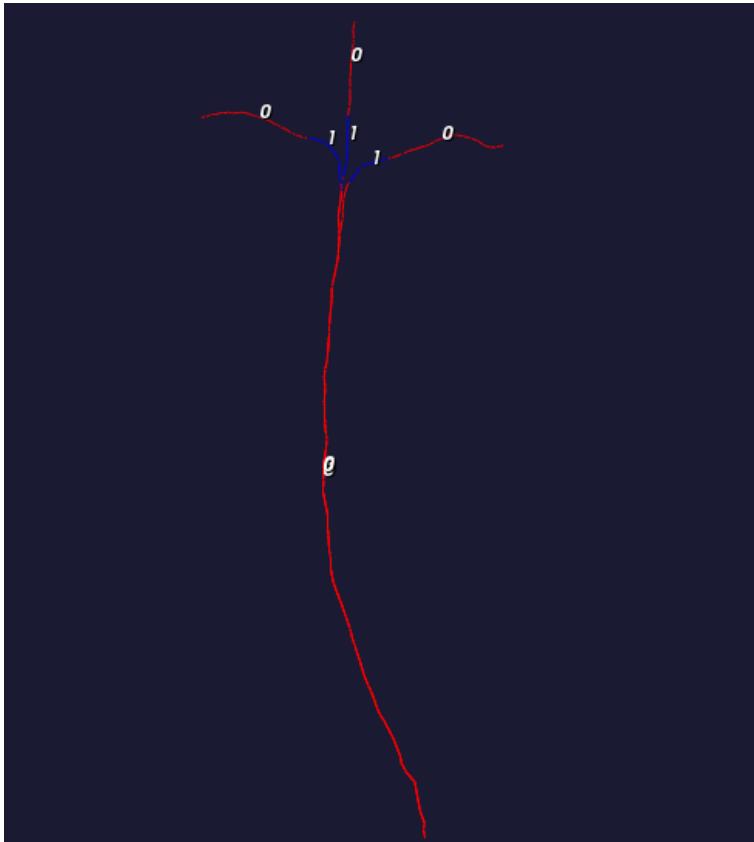
```
Output vmtkcenterlineviewer members:  
Id = 0  
Centerlines = vtkPolyData  
  
Creating vmtkCenterlineViewer instance.  
Automatic piping vmtkcenterlineviewer  
Centerlines = vmtkcenterlineviewer-0.Centerlines  
Parsing options vmtkcenterlineviewer  
CellDataArrayName = GroupIds  
Explicit piping vmtkcenterlineviewer  
Input vmtkcenterlineviewer members:  
Id = 0  
Disabled = 0  
Centerlines = vtkPolyData  
CenterlinesInputFileName =  
PointDataArrayName =  
CellDataArrayName = GroupIds  
Legend = 1  
vmtkRenderer = None  
CenterlinesOutputFileName =  
Executing vmtkcenterlineviewer ...
```



```
Done executing vmtkcenterlineviewer.  
Output vmtkcenterlineviewer members:  
Id = 0
```

```
Centerlines = vtkPolyData

Creating vmtkCenterlineViewer instance.
Automatic piping vmtkcenterlineviewer
    Centerlines = vmtkcenterlineviewer-0.Centerlines
Parsing options vmtkcenterlineviewer
    CellDataArrayName = Blanking
Explicit piping vmtkcenterlineviewer
Input vmtkcenterlineviewer members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    PointDataArrayName =
    CellDataArrayName = Blanking
    Legend = 1
    vmtkRenderer = None
    CenterlinesOutputFileName =
Executing vmtkcenterlineviewer ...
```



```
Done executing vmtkcenterlineviewer.
Output vmtkcenterlineviewer members:
    Id = 0
    Centerlines = vtkPolyData
```

```
Creating vmtkCenterlineViewer instance.  
Automatic piping vmtkcenterlineviewer  
    Centerlines = vmtkcenterlineviewer-0.Centerlines  
Parsing options vmtkcenterlineviewer  
    CellDataArrayName = Blanking  
    CenterlinesOutputFileName = bextractor.vtp  
Explicit piping vmtkcenterlineviewer  
Input vmtkcenterlineviewer members:  
    Id = 0  
    Disabled = 0  
    Centerlines = vtkPolyData  
    CenterlinesInputFileName =  
    PointDataArrayName =  
    CellDataArrayName = Blanking  
    Legend = 1  
    vmtkRenderer = None  
    CenterlinesOutputFileName = bextractor.vtp  
Executing vmtkcenterlineviewer ...  
Done executing vmtkcenterlineviewer.  
Writing VTK XML surface file.  
Output vmtkcenterlineviewer members:  
    Id = 0  
    Centerlines = vtkPolyData
```

ANNEX 12: VMTKBRANCHCLIPPER

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor --pipe vmtkbranchclipper --pipe vmtksurfaceviewer -
array GroupIds -ofile bclipper.vtp
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe vmtkbranchextractor --pipe vmtkbranchclipper --pipe vmtksurfaceviewer -array GroupIds -ofile bclipper.vtp

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

 RadiusArrayName = MaximumInscribedSphereRadius

 AppendEndPoints = 0

 Resampling = 0

 ResamplingStepLength = 1.0

```
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlines-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
Output vmtkbranchextractor members:
Id = 0
```

```
Centerlines = vtkPolyData
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
```

Creating vmtkBranchClipper instance.

Automatic piping vmtkbranchclipper

```
Surface = vmtksurfacereader-0.Surface
Centerlines = vmtkbranchextractor-0.Centerlines
GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
BlankingArrayName = vmtkbranchextractor-0.BlankingArrayName
```

Parsing options vmtkbranchclipper

Explicit piping vmtkbranchclipper

Input vmtkbranchclipper members:

```
Id = 0
Disabled = 0
Surface = vtkPolyData
SurfaceInputFileName =
Centerlines = vtkPolyData
CenterlinesInputFileName =
GroupIdsArrayName = GroupIds
GroupIds = []
InsideOut = 0
UseRadiusInformation = 1
RadiusArrayName = MaximumInscribedSphereRadius
BlankingArrayName = Blanking
CutoffRadiusFactor = 1e+16
ClipValue = 0.0
Interactive = 0
```

```
vmtkRenderer = None
SurfaceOutputFileName =
CenterlinesOutputFileName =
Executing vmtkbranchclipper ...
Done executing vmtkbranchclipper.
```

Output vmtkbranchclipper members:

```
Id = 0
Surface = vtkPolyData
Centerlines = vtkPolyData
```

Creating vmtkSurfaceViewer instance.

Automatic piping vmtksurfaceviewer

```
Surface = vmtkbranchclipper-0.Surface
```

Parsing options vmtksurfaceviewer

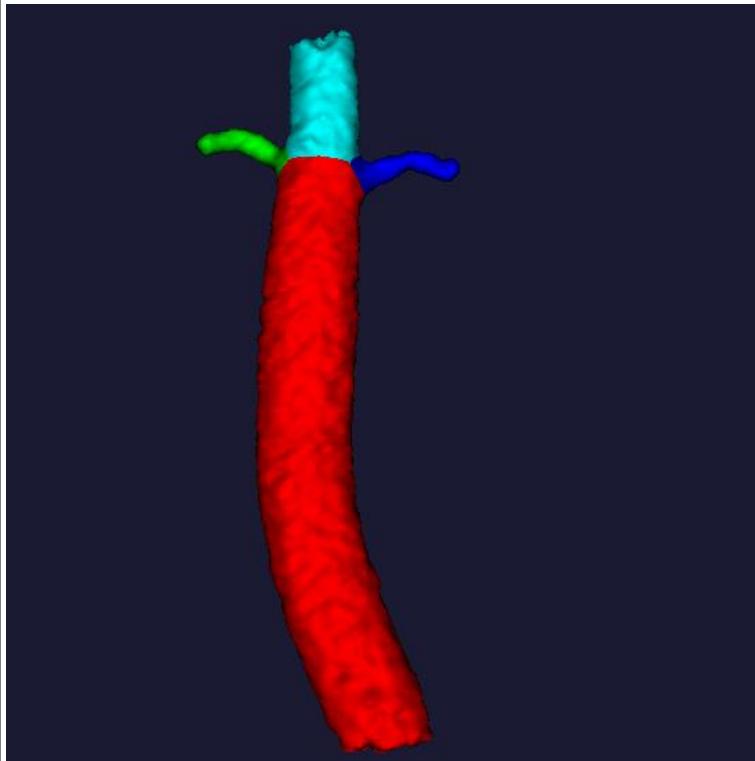
```
ArrayName = GroupIds
SurfaceOutputFileName = bclipper.vtp
```

Explicit piping vmtksurfaceviewer

Input vmtksurfaceviewer members:

```
Id = 0
```

```
Disabled = 0
Surface = vtkPolyData
SurfaceInputFileName =
vmtkRenderer = None
Display = 1
Opacity = 1.0
ArrayName = GroupIds
ScalarRange = [0.0, 0.0]
Legend = 0
Grayscale = 0
FlatInterpolation = 0
DisplayCellData = 0
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName = bclipper.vtp
Executing vmtksurfaceviewer ...
```



```
Done executing vmtksurfaceviewer.
Writing VTK XML surface file.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor
```

**ANNEX 13: VMTKBRANCHCLIPPER (ELIMINAR
BRANQUES)**

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe  
vmtkbranchextractor --pipe vmtkbranchclipper -groupids 3 -insideout 1 --  
pipe vmtksurfaceviewer
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor --pipe vmtkbranchclipper -groupids 3 -insideout 1 --pipe
vmtksurfaceviewer

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

 RadiusArrayName = MaximumInscribedSphereRadius

 AppendEndPoints = 0

 Resampling = 0

 ResamplingStepLength = 1.0

```
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlines-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
Output vmtkbranchextractor members:
Id = 0
```

```
Centerlines = vtkPolyData
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking

Creating vmtkBranchClipper instance.
Automatic piping vmtkbranchclipper
    Surface = vmtksurfacereader-0.Surface
    Centerlines = vmtkbranchextractor-0.Centerlines
    GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
    BlankingArrayName = vmtkbranchextractor-0.BlinkingArrayName
Parsing options vmtkbranchclipper
    GroupIds = [3]
    InsideOut = 1
Explicit piping vmtkbranchclipper
Input vmtkbranchclipper members:
    Id = 0
    Disabled = 0
    Surface = vtkPolyData
    SurfaceInputFileName =
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    GroupIdsArrayName = GroupIds
    GroupIds = [3]
    InsideOut = 1
    UseRadiusInformation = 1
    RadiusArrayName = MaximumInscribedSphereRadius
    BlankingArrayName = Blanking
    CutoffRadiusFactor = 1e+16
    ClipValue = 0.0
    Interactive = 0
    vmtkRenderer = None
    SurfaceOutputFileName =
    CenterlinesOutputFileName =
Executing vmtkbranchclipper ...
Done executing vmtkbranchclipper.
Output vmtkbranchclipper members:
    Id = 0
    Surface = vtkPolyData
    Centerlines = vtkPolyData

Creating vmtkSurfaceViewer instance.
Automatic piping vmtksurfaceviewer
    Surface = vmtkbranchclipper-0.Surface
Parsing options vmtksurfaceviewer
Explicit piping vmtksurfaceviewer
Input vmtksurfaceviewer members:
    Id = 0
```

```
Disabled = 0
Surface = vtkPolyData
SurfaceInputFileName =
vmtkRenderer = None
Display = 1
Opacity = 1.0
ArrayName =
ScalarRange = [0.0, 0.0]
Legend = 0
Grayscale = 0
FlatInterpolation = 0
DisplayCellData = 0
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName =
Executing vmtksurfaceviewer ...
```



```
Done executing vmtksurfaceviewer.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor
```

ANNEX 14: VMTKCENTERLINELABELER

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe  
vmtkbranchextractor --pipe vmtkcenterlinelabeler -labelidsarray GroupIds  
--pipe vmtkcenterlineviewer -cellarray GroupIds
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe vmtkbranchextractor --pipe vmtkcenterlinelabeler -labelidsarray GroupIds --pipe vmtkbranchclipper --pipe vmtkcenterlineviewer -cellarray GroupIds

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

 RadiusArrayName = MaximumInscribedSphereRadius

 AppendEndPoints = 0

 Resampling = 0

 ResamplingStepLength = 1.0

```
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlines-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
Output vmtkbranchextractor members:
Id = 0
```

```
Centerlines = vtkPolyData
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
```

Creating vmtkCenterlineLabeler instance.

Automatic piping vmtkcenterlinelabeler

```
Centerlines = vmtkbranchextractor-0.Centerlines
GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
```

Parsing options vmtkcenterlinelabeler

```
LabelIdsArrayName = GroupIds
```

Explicit piping vmtkcenterlinelabeler

Input vmtkcenterlinelabeler members:

```
Id = 0
```

```
Disabled = 0
```

```
Centerlines = vtkPolyData
```

```
CenterlinesInputFileName =
```

```
GroupIdsArrayName = GroupIds
```

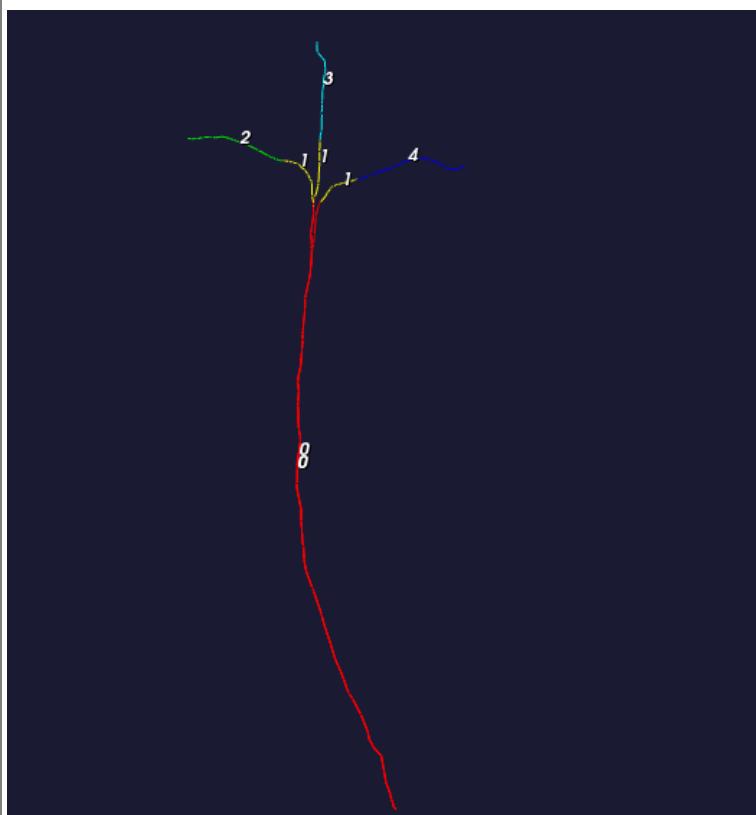
```
LabelIdsArrayName = GroupIds
```

```
Labeling = []
```

```
vmtkRenderer = None
```

```
CenterlinesOutputFileName =
```

Executing vmtkcenterlinelabeler ...



Please input labels for the following groupIds:

0 1 2 3 4

5 6 7 8 9

Done executing vmtkcenterlinelabeler.

Output vmtkcenterlinelabeler members:

Id = 0

Centerlines = vtkPolyData

LabelIdsArrayName = GroupIds

Creating vmtkCenterlineViewer instance.

Automatic piping vmtkcenterlineviewer

Centerlines = vmtkbranchclipper-0.Centerlines

Parsing options vmtkcenterlineviewer

CellDataArrayName = GroupIds

Explicit piping vmtkcenterlineviewer

Input vmtkcenterlineviewer members:

Id = 0

Disabled = 0

Centerlines = vtkPolyData

CenterlinesInputFileName =

PointDataArrayName =

CellDataArrayName = GroupIds

Legend = 1

vmtkRenderer = None

CenterlinesOutputFileName =

Executing vmtkcenterlineviewer ...



Done executing vmtkcenterlineviewer.
Output vmtkcenterlineviewer members:
Id = 0
Centerlines = vtkPolyData

ANNEX 15: VMTKCENTERLINEATTRIBUTES

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkcenterlineviewer -pointarray
Abscissas --pipe vmtkcenterlineviewer -pointarray
ParallelTransportNormals -ofile catributes.vtp
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkcenterlineviewer -pointarray Abscissas --
pipe vmtkcenterlineviewer -pointarray ParallelTransportNormals -ofile
catributes.vtp

Creating vmtkSurfaceReader instance.
Automatic piping vmtksurfacereader
Parsing options vmtksurfacereader
  InputFileName = marching.vtp
Explicit piping vmtksurfacereader
Input vmtksurfacereader members:
  Id = 0
  Disabled = 0
  Format =
  GuessFormat = 1
  Surface = 0
  InputFileName = marching.vtp
  SurfaceOutputFileName =
Executing vmtksurfacereader ...
Reading VTK XML surface file.
Done executing vmtksurfacereader.
Output vmtksurfacereader members:
  Id = 0
  Surface = vtkPolyData

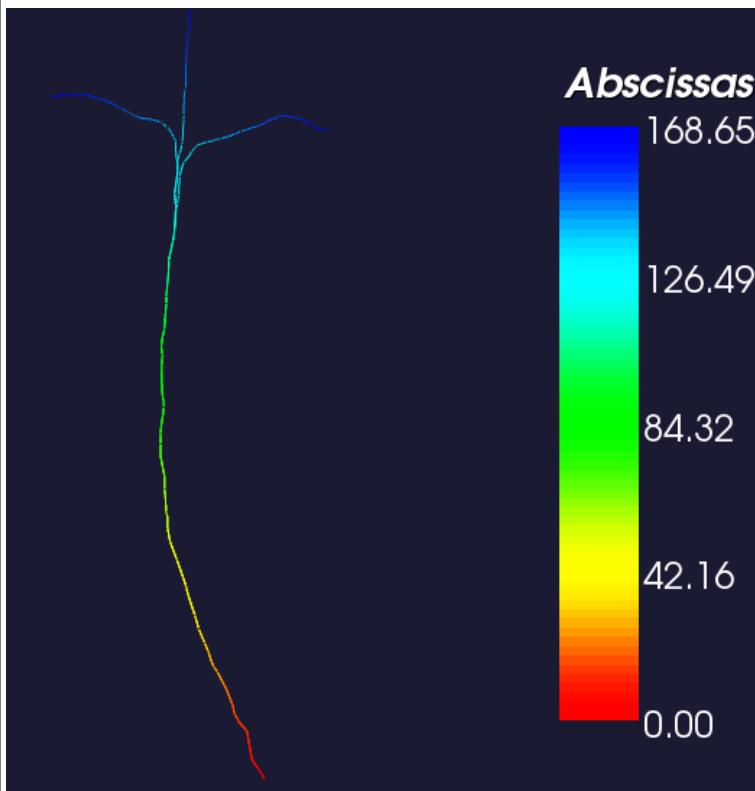
Creating vmtkCenterlines instance.
Automatic piping vmtkcenterlines
  Surface = vmtksurfacereader-0.Surface
Parsing options vmtkcenterlines
Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
  Id = 0
  Disabled = 0
  Surface = vtkPolyData
  SurfaceInputFileName =
  SeedSelectorName = pickpoint
  SourceIds = []
  TargetIds = []
  SourcePoints = []
  TargetPoints = []
  AppendEndPoints = 0
  CheckNonManifold = 0
  FlipNormals = 0
  CapDisplacement = 0.0
  RadiusArrayName = MaximumInscribedSphereRadius
  AppendEndPoints = 0
```

```
Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkCenterlineAttributes instance.
Automatic piping vmtkcenterlineattributes
    Centerlines = vmtkcenterlines-0.Centerlines
Parsing options vmtkcenterlineattributes
Explicit piping vmtkcenterlineattributes
Input vmtkcenterlineattributes members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
AbscissasArrayName = Abscissas
NormalsArrayName = ParallelTransportNormals
CenterlinesOutputFileName =
Executing vmtkcenterlineattributes ...
Done executing vmtkcenterlineattributes.
Output vmtkcenterlineattributes members:
Id = 0
Centerlines = vtkPolyData
AbscissasArrayName = Abscissas
```

```
NormalsArrayName = ParallelTransportNormals

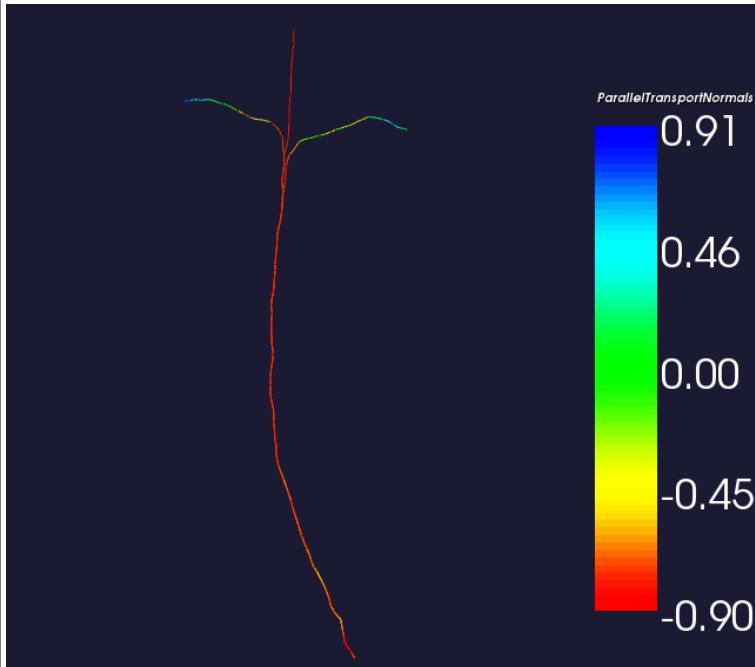
Creating vmtkCenterlineViewer instance.
Automatic piping vmtkcenterlineviewer
  Centerlines = vmtkcenterlineattributes-0.Centerlines
Parsing options vmtkcenterlineviewer
  PointDataArrayName = Abscissas
Explicit piping vmtkcenterlineviewer
Input vmtkcenterlineviewer members:
  Id = 0
  Disabled = 0
  Centerlines = vtkPolyData
  CenterlinesInputFileName =
  PointDataArrayName = Abscissas
  CellDataArrayName =
  Legend = 1
  vmtkRenderer = None
  CenterlinesOutputFileName =
Executing vmtkcenterlineviewer ...
```



```
Done executing vmtkcenterlineviewer.
Output vmtkcenterlineviewer members:
  Id = 0
  Centerlines = vtkPolyData
```

```
Creating vmtkCenterlineViewer instance.
```

```
Automatic piping vmtkcenterlineviewer
  Centerlines = vmtkcenterlineviewer-0.Centerlines
Parsing options vmtkcenterlineviewer
  PointDataArrayName = ParallelTransportNormals
  CenterlinesOutputFileName = cattributes.vtp
Explicit piping vmtkcenterlineviewer
Input vmtkcenterlineviewer members:
  Id = 0
  Disabled = 0
  Centerlines = vtkPolyData
  CenterlinesInputFileName =
  PointDataArrayName = ParallelTransportNormals
  CellDataArrayName =
  Legend = 1
  vmtkRenderer = None
  CenterlinesOutputFileName = cattributes.vtp
Executing vmtkcenterlineviewer ...
```



```
Done executing vmtkcenterlineviewer.
Writing VTK XML surface file.
Output vmtkcenterlineviewer members:
  Id = 0
  Centerlines = vtkPolyData
```

ANNEX 16:
VMTKCENTERLINEATTRIBUTES
(VMTKBRANCHEXTRACTOR)

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkbranchextractor --pipe
vmtkcenterlineviewer -pointarray Abscissas --pipe vmtkcenterlineviewer -
-pointarray ParallelTransportNormals -ofile catributes.vtp
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe vmtkcenterlineattributes --pipe vmtkbranchextractor --pipe vmtkcenterlineviewer -pointarray Abscissas --pipe vmtkcenterlineviewer -pointarray ParallelTransportNormals -ofile catributes.vtp

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

 RadiusArrayName = MaximumInscribedSphereRadius

 AppendEndPoints = 0

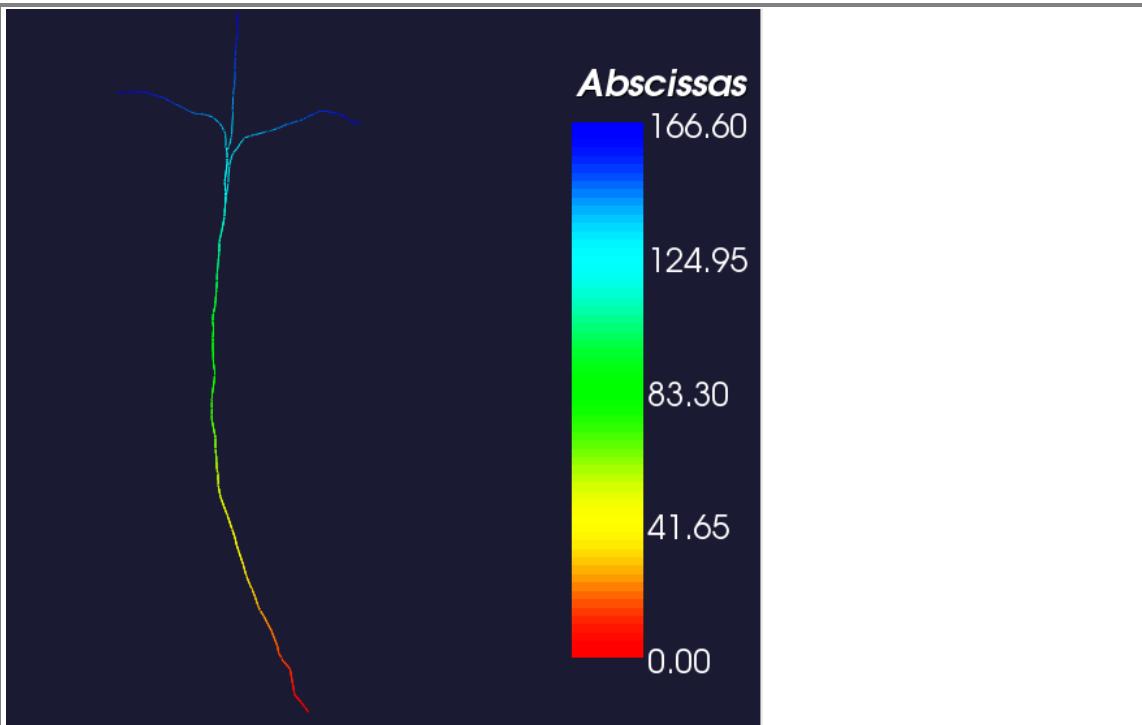
```
Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkCenterlineAttributes instance.
Automatic piping vmtkcenterlineattributes
    Centerlines = vmtkcenterlines-0.Centerlines
Parsing options vmtkcenterlineattributes
Explicit piping vmtkcenterlineattributes
Input vmtkcenterlineattributes members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
AbscissasArrayName = Abscissas
NormalsArrayName = ParallelTransportNormals
CenterlinesOutputFileName =
Executing vmtkcenterlineattributes ...
Done executing vmtkcenterlineattributes.
Output vmtkcenterlineattributes members:
Id = 0
Centerlines = vtkPolyData
AbscissasArrayName = Abscissas
```

```
NormalsArrayName = ParallelTransportNormals

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlineattributes-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    RadiusArrayName = MaximumInscribedSphereRadius
    GroupIdsArrayName = GroupIds
    CenterlineIdsArrayName = CenterlineIds
    TractIdsArrayName = TractIds
    BlankingArrayName = Blanking
    CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
Output vmtkbranchextractor members:
    Id = 0
    Centerlines = vtkPolyData
    GroupIdsArrayName = GroupIds
    CenterlineIdsArrayName = CenterlineIds
    TractIdsArrayName = TractIds
    BlankingArrayName = Blanking

Creating vmtkCenterlineViewer instance.
Automatic piping vmtkcenterlineviewer
    Centerlines = vmtkbranchextractor-0.Centerlines
Parsing options vmtkcenterlineviewer
    PointDataArrayName = Abscissas
Explicit piping vmtkcenterlineviewer
Input vmtkcenterlineviewer members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    PointDataArrayName = Abscissas
    CellDataArrayName =
    Legend = 1
    vmtkRenderer = None
    CenterlinesOutputFileName =
Executing vmtkcenterlineviewer ...
```



Done executing vmtkcenterlineviewer.

Output vmtkcenterlineviewer members:

Id = 0

Centerlines = vtkPolyData

Creating vmtkCenterlineViewer instance.

Automatic piping vmtkcenterlineviewer

 Centerlines = vmtkcenterlineviewer-0.Centerlines

Parsing options vmtkcenterlineviewer

 PointDataArrayName = ParallelTransportNormals

 CenterlinesOutputFileName = cattributes.vtp

Explicit piping vmtkcenterlineviewer

Input vmtkcenterlineviewer members:

 Id = 0

 Disabled = 0

 Centerlines = vtkPolyData

 CenterlinesInputFileName =

 PointDataArrayName = ParallelTransportNormals

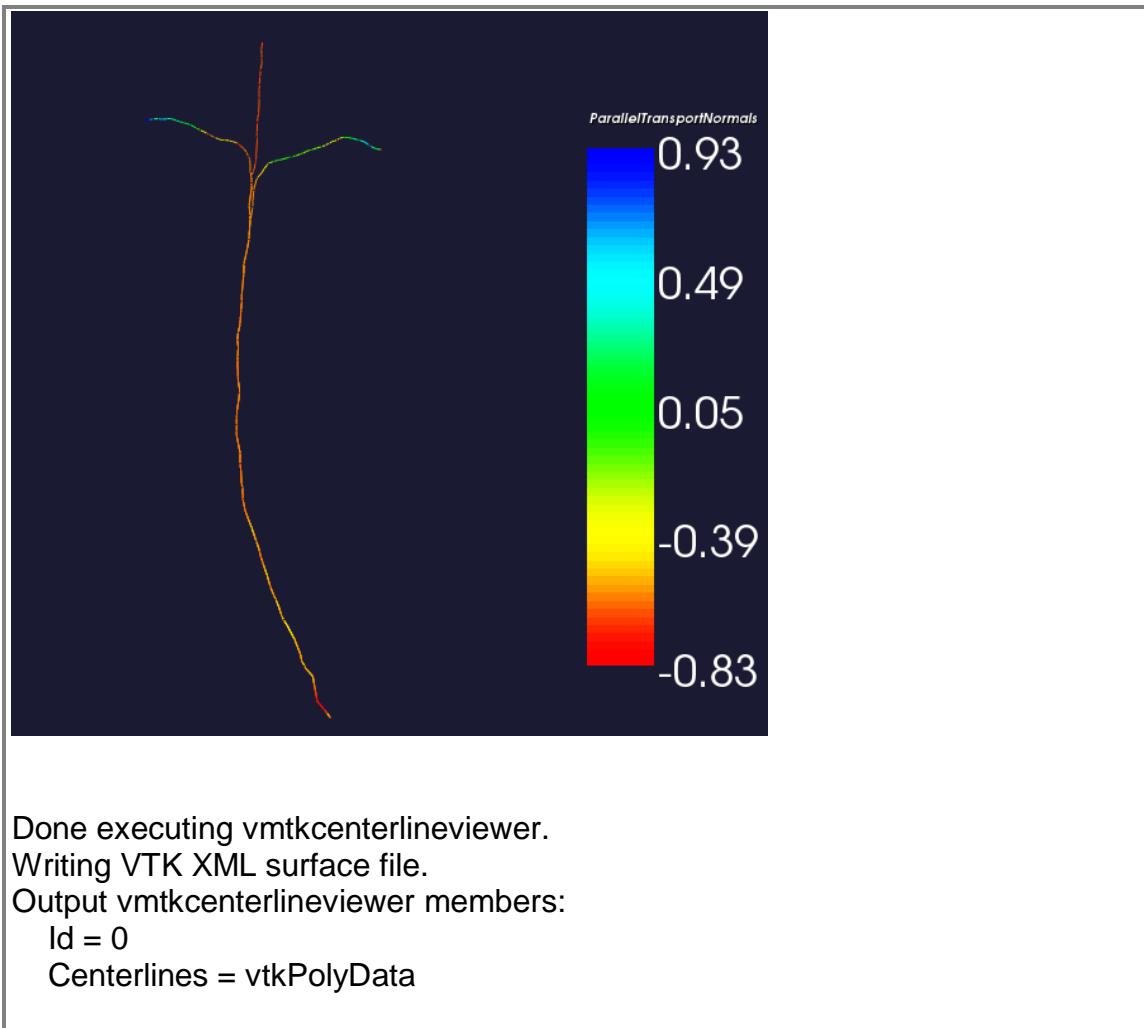
 CellDataArrayName =

 Legend = 1

 vmtkRenderer = None

 CenterlinesOutputFileName = cattributes.vtp

Executing vmtkcenterlineviewer ...



ANNEX 17: VMTKBIFURCATIONREFERENCESYSTEMS

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor -radiusarray@ MaximumInscribedSphereRadius --
pipe vmtkbifurcationreferencesystems -ofile biresys.vtp
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe vmtkbranchextractor -radiusarray@ MaximumInscribedSphereRadius --pipe vmtkbifurcationreferencesystems --pipe vmtkcenterlineviewer -cellarray GroupIds -ofile biresys.vtp

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

 RadiusArrayName = MaximumInscribedSphereRadius

 AppendEndPoints = 0

 Resampling = 0

```
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlines-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
    RadiusArrayName = MaximumInscribedSphereRadius
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
```

```
Output vmtkbranchextractor members:
```

```
Id = 0
Centerlines = vtkPolyData
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
```

```
Creating vmtkBifurcationReferenceSystems instance.
```

```
Automatic piping vmtkbifurcationreferencesystems
```

```
Centerlines = vmtkbranchextractor-0.Centerlines
RadiusArrayName = vmtkbranchextractor-0.RadiusArrayName
BlankingArrayName = vmtkbranchextractor-0.BlankingArrayName
GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
```

```
Parsing options vmtkbifurcationreferencesystems
```

```
Explicit piping vmtkbifurcationreferencesystems
```

```
Input vmtkbifurcationreferencesystems members:
```

```
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
BlankingArrayName = Blanking
GroupIdsArrayName = GroupIds
ReferenceSystemsNormalArrayName = Normal
ReferenceSystemsUpNormalArrayName = UpNormal
ReferenceSystemsOutputFileName =
```

```
Executing vmtkbifurcationreferencesystems ...
```

```
Done executing vmtkbifurcationreferencesystems.
```

```
Output vmtkbifurcationreferencesystems members:
```

```
Id = 0
ReferenceSystems = vtkPolyData
ReferenceSystemsNormalArrayName = Normal
ReferenceSystemsUpNormalArrayName = UpNormal
```

ANNEX 18: VMTKCENTERLINEOFFSETATTRIBUTES

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkbranchextractor -radiusarray@
MaximumInscribedSphereRadius --pipe vmtkbifurcationreferencesystems
--pipe vmtkcenterlineoffsetattributes -referencegroupid 1 -ofile offset.vtp
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkbranchextractor -radiusarray@
MaximumInscribedSphereRadius --pipe vmtkbifurcationreferencesystems --pipe
vmtkcenterlineoffsetattributes -referencegroupid 1 -ofile offset.vtp
```

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

 RadiusArrayName = MaximumInscribedSphereRadius

 AppendEndPoints = 0

```
Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkCenterlineAttributes instance.
Automatic piping vmtkcenterlineattributes
    Centerlines = vmtkcenterlines-0.Centerlines
Parsing options vmtkcenterlineattributes
Explicit piping vmtkcenterlineattributes
Input vmtkcenterlineattributes members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
AbscissasArrayName = Abscissas
NormalsArrayName = ParallelTransportNormals
CenterlinesOutputFileName =
Executing vmtkcenterlineattributes ...
Done executing vmtkcenterlineattributes.
Output vmtkcenterlineattributes members:
Id = 0
Centerlines = vtkPolyData
AbscissasArrayName = Abscissas
```

```

NormalsArrayName = ParallelTransportNormals

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
Centerlines = vmtkcenterlineattributes-0.Centerlines
RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
RadiusArrayName = MaximumInscribedSphereRadius
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
Output vmtkbranchextractor members:
Id = 0
Centerlines = vtkPolyData
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking

Creating vmtkBifurcationReferenceSystems instance.
Automatic piping vmtkbifurcationreferencesystems
Centerlines = vmtkbranchextractor-0.Centerlines
RadiusArrayName = vmtkbranchextractor-0.RadiusArrayName
BlankingArrayName = vmtkbranchextractor-0.BlinkingArrayName
GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
Parsing options vmtkbifurcationreferencesystems
Explicit piping vmtkbifurcationreferencesystems
Input vmtkbifurcationreferencesystems members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
BlankingArrayName = Blanking
GroupIdsArrayName = GroupIds
ReferenceSystemsNormalArrayName = Normal
ReferenceSystemsUpNormalArrayName = UpNormal
ReferenceSystemsOutputFileName =
Executing vmtkbifurcationreferencesystems ...

```

```
Done executing vmtkbifurcationreferencesystems.
Output vmtkbifurcationreferencesystems members:
Id = 0
ReferenceSystems = vtkPolyData
ReferenceSystemsNormalArrayName = Normal
ReferenceSystemsUpNormalArrayName = UpNormal

Creating vmtkCenterlineOffsetAttributes instance.
Automatic piping vmtkcenterlineoffsetattributes
    Centerlines = vmtkbranchextractor-0.Centerlines
    ReferenceSystems = vmtkbifurcationreferencesystems-0.ReferenceSystems
    AbscissasArrayName = vmtkcenterlineattributes-0.AbscissasArrayName
    NormalsArrayName = vmtkcenterlineattributes-0.NormalsArrayName
    GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
    CenterlineIdsArrayName = vmtkbranchextractor-0.CenterlineIdsArrayName
    ReferenceSystemsNormalArrayName = vmtkbifurcationreferencesystems-
0.ReferenceSystemsNormalArrayName
Parsing options vmtkcenterlineoffsetattributes
    ReferenceGroupId = 1
    CenterlinesOutputFileName = offset.vtp
Explicit piping vmtkcenterlineoffsetattributes
Input vmtkcenterlineoffsetattributes members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
ReferenceSystems = vtkPolyData
ReferenceSystemsInputFileName =
ReferenceGroupId = 1
ReplaceAttributes = 1
AbscissasArrayName = Abscissas
NormalsArrayName = ParallelTransportNormals
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
ReferenceSystemsNormalArrayName = Normal
OffsetAbscissasArrayName = OffsetAbscissas
OffsetNormalsArrayName = OffsetNormals
Interactive = 0
vmtkRenderer = None
CenterlinesOutputFileName = offset.vtp
Executing vmtkcenterlineoffsetattributes ...
Done executing vmtkcenterlineoffsetattributes.
Writing VTK XML surface file.
Output vmtkcenterlineoffsetattributes members:
Id = 0
Centerlines = vtkPolyData
ReferenceGroupId = 1
OffsetAbscissasArrayName = OffsetAbscissas
OffsetNormalsArrayName = OffsetNormals
AbscissasArrayName = Abscissas
```

NormalsArrayName = ParallelTransportNormals

ANNEX 19: VMTKBIFURCATIONVECTORS

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkbranchextractor -radiusarray@
MaximumInscribedSphereRadius --pipe vmtkbifurcationreferencesystems
--pipe vmtkbifurcationvectors
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlineattributes --pipe vmtkbranchextractor -radiusarray@
MaximumInscribedSphereRadius --pipe vmtkbifurcationreferencesystems --pipe
vmtkbifurcationvectors

Creating vmtkSurfaceReader instance.
Automatic piping vmtksurfacereader
Parsing options vmtksurfacereader
    InputFileName = marching.vtp
Explicit piping vmtksurfacereader
Input vmtksurfacereader members:
    Id = 0
    Disabled = 0
    Format =
    GuessFormat = 1
    Surface = 0
    InputFileName = marching.vtp
    SurfaceOutputFileName =
Executing vmtksurfacereader ...
Reading VTK XML surface file.
Done executing vmtksurfacereader.
Output vmtksurfacereader members:
    Id = 0
    Surface = vtkPolyData

Creating vmtkCenterlines instance.
Automatic piping vmtkcenterlines
    Surface = vmtksurfacereader-0.Surface
Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
    Id = 0
    Disabled = 0
    Surface = vtkPolyData
    SurfaceInputFileName =
    SeedSelectorName = pickpoint
    SourceIds = []
    TargetIds = []
    SourcePoints = []
    TargetPoints = []
    AppendEndPoints = 0
    CheckNonManifold = 0
    FlipNormals = 0
    CapDisplacement = 0.0
    RadiusArrayName = MaximumInscribedSphereRadius
```

```
AppendEndPoints = 0
Resampling = 0
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
    UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
    Id = 0
    Centerlines = vtkPolyData
    RadiusArrayName = MaximumInscribedSphereRadius

EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkCenterlineAttributes instance.
Automatic piping vmtkcenterlineattributes
    Centerlines = vmtkcenterlines-0.Centerlines
Parsing options vmtkcenterlineattributes
Explicit piping vmtkcenterlineattributes
Input vmtkcenterlineattributes members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    AbscissasArrayName = Abscissas
    NormalsArrayName = ParallelTransportNormals
    CenterlinesOutputFileName =
Executing vmtkcenterlineattributes ...
Done executing vmtkcenterlineattributes.
Output vmtkcenterlineattributes members:
    Id = 0
```

```
Centerlines = vtkPolyData
AbscissasArrayName = Abscissas
NormalsArrayName = ParallelTransportNormals

Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlineattributes-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
    RadiusArrayName = MaximumInscribedSphereRadius
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    RadiusArrayName = MaximumInscribedSphereRadius
    GroupIdsArrayName = GroupIds
    CenterlineIdsArrayName = CenterlineIds
    TractIdsArrayName = TractIds
    BlankingArrayName = Blanking
    CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.

Output vmtkbranchextractor members:
    Id = 0
    Centerlines = vtkPolyData
    GroupIdsArrayName = GroupIds
    CenterlineIdsArrayName = CenterlineIds
    TractIdsArrayName = TractIds
    BlankingArrayName = Blanking

Creating vmtkBifurcationReferenceSystems instance.
Automatic piping vmtkbifurcationreferencesystems
    Centerlines = vmtkbranchextractor-0.Centerlines
    RadiusArrayName = vmtkbranchextractor-0.RadiusArrayName
    BlankingArrayName = vmtkbranchextractor-0.BlinkingArrayName
    GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
Parsing options vmtkbifurcationreferencesystems
Explicit piping vmtkbifurcationreferencesystems
Input vmtkbifurcationreferencesystems members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    RadiusArrayName = MaximumInscribedSphereRadius
    BlankingArrayName = Blanking
    GroupIdsArrayName = GroupIds
    ReferenceSystemsNormalArrayName = Normal
```

```
ReferenceSystemsUpNormalArrayName = UpNormal
ReferenceSystemsOutputFileName =
Executing vmtkbifurcationreferencesystems ...
Done executing vmtkbifurcationreferencesystems.
Output vmtkbifurcationreferencesystems members:
Id = 0
ReferenceSystems = vtkPolyData
ReferenceSystemsNormalArrayName = Normal
ReferenceSystemsUpNormalArrayName = UpNormal

Creating vmtkBifurcationVectors instance.
Automatic piping vmtkbifurcationvectors
Centerlines = vmtkbranchextractor-0.Centerlines
ReferenceSystems = vmtkbifurcationreferencesystems-0.ReferenceSystems
RadiusArrayName = vmtkbranchextractor-0.RadiusArrayName
GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
CenterlineIdsArrayName = vmtkbranchextractor-0.CenterlineIdsArrayName
TractIdsArrayName = vmtkbranchextractor-0.TtractIdsArrayName
BlankingArrayName = vmtkbranchextractor-0.BlanckingArrayName
ReferenceSystemsNormalArrayName = vmtkbifurcationreferencesystems-
0.ReferenceSystemsNormalArrayName
ReferenceSystemsUpNormalArrayName = vmtkbifurcationreferencesystems-
0.ReferenceSystemsUpNormalArrayName
Parsing options vmtkbifurcationvectors

Explicit piping vmtkbifurcationvectors
Input vmtkbifurcationvectors members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
ReferenceSystems = vtkPolyData
ReferenceSystemsInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
ReferenceSystemsNormalArrayName = Normal
ReferenceSystemsUpNormalArrayName = UpNormal
BifurcationVectorsArrayName = BifurcationVectors
InPlaneBifurcationVectorsArrayName = InPlaneBifurcationVectors
OutOfPlaneBifurcationVectorsArrayName = OutOfPlaneBifurcationVectors
InPlaneBifurcationVectorAnglesArrayName= InPlaneBifurcationVectorAngles
OutOfPlaneBifurcationVectorAnglesArrayName=
OutOfPlaneBifurcationVectorAngles
BifurcationVectorsOrientationArrayName = BifurcationVectorsOrientation
BifurcationGroupIdsArrayName = BifurcationGroupIds
NormalizeBifurcationVectors = 0
BifurcationVectorsOutputFileName =
```

```
Executing vmtkbifurcationvectors ...
Done executing vmtkbifurcationvectors.
Output vmtkbifurcationvectors members:
Id = 0
BifurcationVectors = vtkPolyData
BifurcationVectorsArrayName = BifurcationVectors
InPlaneBifurcationVectorsArrayName = InPlaneBifurcationVectors
OutOfPlaneBifurcationVectorsArrayName = OutOfPlaneBifurcationVectors
InPlaneBifurcationVectorAnglesArrayName= InPlaneBifurcationVectorAngles
OutOfPlaneBifurcationVectorAnglesArrayName=
OutOfPlaneBifurcationVectorAngles
BifurcationVectorsOrientationArrayName = BifurcationVectorsOrientation
BifurcationGroupIdsArrayName = BifurcationGroupIds
```

ANNEX 20: VMTKCENTERLINEGEOMETRY

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe  
vmtkcenterlinegeometry -smoothing 1 -iterations 100 -factor 0.1  
-outputssmoothed 1
```

Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkcenterlinegeometry -smoothing 1 -iterations 100 -factor 0.1 -
outputssmoothed 1

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkCenterlines instance.

Automatic piping vmtkcenterlines

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkcenterlines

Explicit piping vmtkcenterlines

Input vmtkcenterlines members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 SeedSelectorName = pickpoint

 SourceIds = []

 TargetIds = []

 SourcePoints = []

 TargetPoints = []

 AppendEndPoints = 0

 CheckNonManifold = 0

 FlipNormals = 0

 CapDisplacement = 0.0

RadiusArrayName = MaximumInscribedSphereRadius

AppendEndPoints = 0

Resampling = 0

```
ResamplingStepLength = 1.0
DelaunayTessellation = None
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData
RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList

Creating vmtkCenterlineGeometry instance.
Automatic piping vmtkcenterlinegeometry
    Centerlines = vmtkcenterlines-0.Centerlines
Parsing options vmtkcenterlinegeometry
LineSmoothing = 1
OutputSmoothedLines = 1
NumberOfSmoothingIterations = 100
SmoothingFactor = 0.1
Explicit piping vmtkcenterlinegeometry
Input vmtkcenterlinegeometry members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
CurvatureArrayName = Curvature
TorsionArrayName = Torsion
FrenetTangentArrayName = FrenetTangent

FrenetNormalArrayName = FrenetNormal
FrenetBinormalArrayName = FrenetBinormal
LineSmoothing = 1
```

```
OutputSmoothedLines = 1
NumberOfSmoothingIterations = 100
SmoothingFactor = 0.1
CenterlinesOutputFileName =
Executing vmtkcenterlinegeometry ...
ERROR:           In          /build/buildd-vmtk_0.9.0-5-amd64-KTx19t/vmtk-
.9.0/vtkVmtk/ComputationalGeometry/vtkvmtkCenterlineGeometry.cxx, line 118
vtkvmtkCenterlineGeometry (0x35a35f0): LengthArrayName not specified

Done executing vmtkcenterlinegeometry.
Output vmtkcenterlinegeometry members:
Id = 0
Centerlines = vtkPolyData
CurvatureArrayName = Curvature
TorsionArrayName = Torsion
```

ANNEX 21: VMTKBRANCHGEOMETRY

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor --pipe vmtkbranchgeometry
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkcenterlines --pipe
vmtkbranchextractor --pipe vmtkbranchgeometry

Creating vmtkSurfaceReader instance.
Automatic piping vmtksurfacereader
Parsing options vmtksurfacereader
  InputFileName = marching.vtp
Explicit piping vmtksurfacereader
Input vmtksurfacereader members:
  Id = 0
  Disabled = 0
  Format =
  GuessFormat = 1
  Surface = 0
  InputFileName = marching.vtp
  SurfaceOutputFileName =
Executing vmtksurfacereader ...
Reading VTK XML surface file.
Done executing vmtksurfacereader.
Output vmtksurfacereader members:
  Id = 0

  Surface = vtkPolyData

Creating vmtkCenterlines instance.
Automatic piping vmtkcenterlines
  Surface = vmtksurfacereader-0.Surface
Parsing options vmtkcenterlines
Explicit piping vmtkcenterlines
Input vmtkcenterlines members:
  Id = 0
  Disabled = 0
  Surface = vtkPolyData
  SurfaceInputFileName =
  SeedSelectorName = pickpoint
  SourceIds = []
  TargetIds = []
  SourcePoints = []
  TargetPoints = []
  AppendEndPoints = 0
  CheckNonManifold = 0
  FlipNormals = 0
  CapDisplacement = 0.0
  RadiusArrayName = MaximumInscribedSphereRadius
  AppendEndPoints = 0
  Resampling =
  ResamplingStepLength = 1.0
  DelaunayTessellation = None
```

```
SimplifyVoronoi = 0
UseTetGen = 0
TetGenDetectInter = 1
CostFunction = 1/R
vmtkRenderer = None
CenterlinesOutputFileName =
DelaunayTessellationOutputFileName =
VoronoiDiagramOutputFileName =
Executing vmtkcenterlines ...
Cleaning surface.
Triangulating surface.
Capping surface.
Please position the mouse and press space to add source points, 'u' to undo
Please position the mouse and press space to add target points, 'u' to undo
Computing centerlines.
Done executing vmtkcenterlines.
Output vmtkcenterlines members:
Id = 0
Centerlines = vtkPolyData

RadiusArrayName = MaximumInscribedSphereRadius
EikonalSolutionArrayName = EikonalSolutionArray
EdgeArrayName = EdgeArray
EdgePCoordArrayName = EdgePCoordArray
CostFunctionArrayName = CostFunctionArray
DelaunayTessellation = vtkUnstructuredGrid
VoronoiDiagram = vtkPolyData
PoleIds = vtkIdList
Creating vmtkBranchExtractor instance.
Automatic piping vmtkbranchextractor
    Centerlines = vmtkcenterlines-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
Parsing options vmtkbranchextractor
Explicit piping vmtkbranchextractor
Input vmtkbranchextractor members:
Id = 0
Disabled = 0
Centerlines = vtkPolyData
CenterlinesInputFileName =
RadiusArrayName = MaximumInscribedSphereRadius
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking
CenterlinesOutputFileName =
Executing vmtkbranchextractor ...
Done executing vmtkbranchextractor.
Output vmtkbranchextractor members:
Id = 0
Centerlines = vtkPolyData
```

```
GroupIdsArrayName = GroupIds
CenterlineIdsArrayName = CenterlineIds
TractIdsArrayName = TractIds
BlankingArrayName = Blanking

Creating vmtkBranchGeometry instance.
Automatic piping vmtkbranchgeometry
    Centerlines = vmtkbranchextractor-0.Centerlines
    RadiusArrayName = vmtkcenterlines-0.RadiusArrayName
    GroupIdsArrayName = vmtkbranchextractor-0.GroupIdsArrayName
    BlankingArrayName = vmtkbranchextractor-0.BlankingArrayName
Parsing options vmtkbranchgeometry
Explicit piping vmtkbranchgeometry
Input vmtkbranchgeometry members:
    Id = 0
    Disabled = 0
    Centerlines = vtkPolyData
    CenterlinesInputFileName =
    RadiusArrayName = MaximumInscribedSphereRadius

    GroupIdsArrayName = GroupIds
    BlankingArrayName = Blanking
    LengthArrayName = Length
    CurvatureArrayName = Curvature
    TorsionArrayName = Torsion
    TortuosityArrayName = Tortuosity
    LineSmoothing = 0
    NumberOfSmoothingIterations = 100
    SmoothingFactor = 0.1
    GeometryDataOutputFileName =
Executing vmtkbranchgeometry ...
Done executing vmtkbranchgeometry.
Output vmtkbranchgeometry members:
    Id = 0
    GeometryData = vtkPolyData
    LengthArrayName = Length
    CurvatureArrayName = Curvature
    TorsionArrayName = Torsion
    TortuosityArrayName = Tortuosity
```

ANNEX 22: VMTKMESHGENERATOR

```
vmtk vmtksurfacereader -ifile marching.vtp --pipe vmtkmeshgenerator -  
edgelength 0.5 --pipe vmtkmeshviewer -ofile mesh.vtu
```

```
Executing vmtksurfacereader -ifile marching.vtp --pipe vmtkmeshgenerator -  
edgelength 0.5 --pipe vmtkmeshviewer -ofile mesh.vtu
```

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName =

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkMeshGenerator instance.

Automatic piping vmtkmeshgenerator

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtkmeshgenerator

 TargetEdgeLength = 0.5

Explicit piping vmtkmeshgenerator

Input vmtkmeshgenerator members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 TargetEdgeLength = 0.5

 TargetEdgeLengthArrayName =

 TargetEdgeLengthFactor = 1.0

 MaxEdgeLength = 1e+16

 MinEdgeLength = 0.0

 CellEntityIdsArrayName = CellEntityIds

 ElementSizeMode = edgelength

 VolumeElementScaleFactor = 0.8

 BoundaryLayer = 0

 NumberOfSubLayers = 2

 BoundaryLayerThicknessFactor = 0.25

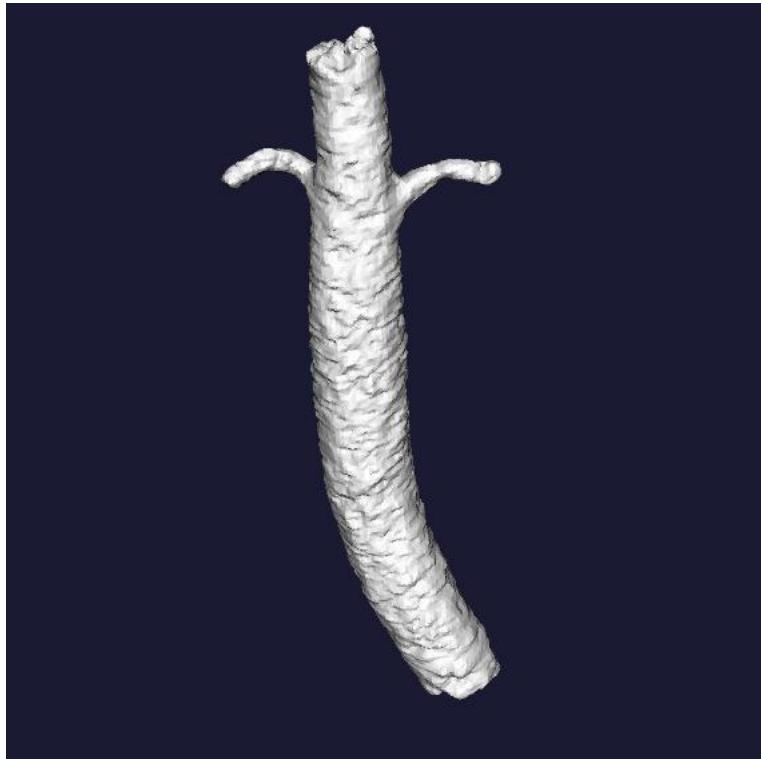
```
Tetrahedralize = 0
MeshOutputFileName =
Executing vmtkmeshgenerator ...
Capping surface
Remeshing surface
Iteration 1/10
Iteration 2/10

Iteration 3/10
Iteration 4/10
Iteration 5/10
Iteration 6/10
Iteration 7/10
Iteration 8/10
Iteration 9/10
Iteration 10/10
Final mesh improvement
Computing sizing function
Converting surface to mesh
Generating volume mesh
TetGen           command          line      options:
pq1.414000q10.000000q165.000000YsT1.000000e-08zQm
Done executing vmtkmeshgenerator.
Output vmtkmeshgenerator members:
  Id = 0
  Mesh = vtkUnstructuredGrid

CellEntityIdsArrayName = CellEntityIds

Creating vmtkMeshViewer instance.
Automatic piping vmtkmeshviewer
  Mesh = vmtkmeshgenerator-0.Mesh
Parsing options vmtkmeshviewer
  MeshOutputFileName = mesh.vtu
Explicit piping vmtkmeshviewer
Input vmtkmeshviewer members:
  Id = 0
  Disabled = 0
  Mesh = vtkUnstructuredGrid
  MeshInputFileName =
  vmtkRenderer = None
  Display = 1
  Opacity = 1.0
  ArrayName =
  ScalarRange = [0.0, 0.0]
  Legend = 0
  Grayscale = 0
  FlatInterpolation = 0
  MeshOutputFileName = mesh.vtu
```

Executing vmtkmeshviewer ...



Done executing vmtkmeshviewer.
Writing VTK XML mesh file.
Output vmtkmeshviewer members:

Id = 0
Mesh = vtkUnstructuredGrid

ANNEX 23: VMTKLINEARTOQUADRATIC

```
vmtk vmtklineartoquadratic -ifile mesh.vtu -entityidsarray CellEntityIds --
pipe vmtkmeshviewer -ofile meshlineartoqua.vtu
Executing vmtklineartoquadratic -ifile mesh.vtu -entityidsarray CellEntityIds --
pipe vmtkmeshviewer -ofile meshlineartoqua.vtu

Creating vmtkLinearToQuadratic instance.
Automatic piping vmtklineartoquadratic
Parsing options vmtklineartoquadratic
    MeshInputFileName = mesh.vtu
    CellEntityIdsArrayName = CellEntityIds
Explicit piping vmtklineartoquadratic
Input vmtklineartoquadratic members:
    Id = 0
    Disabled = 0
    Mesh = None
    MeshInputFileName = mesh.vtu
    Surface = None
    SurfaceInputFileName =
    Mode = volume
    UseBiquadraticWedge = True
    CapSurface = False
    CellEntityIdsArrayName = CellEntityIds
    ProjectedCellEntityId = 1
    QuadratureOrder = 10
    NegativeJacobianTolerance = 0.0
    SubdivisionMethod = linear
    MeshOutputFileName =
Reading VTK XML mesh file.
Executing vmtklineartoquadratic ...
Done executing vmtklineartoquadratic.
Output vmtklineartoquadratic members:
    Id = 0
    Mesh = vtkUnstructuredGrid

Creating vmtkMeshViewer instance.
Automatic piping vmtkmeshviewer
    Mesh = vmtklineartoquadratic-0.Mesh
Parsing options vmtkmeshviewer
    MeshOutputFileName = meshlineartoqua.vtu

Explicit piping vmtkmeshviewer
Input vmtkmeshviewer members:
    Id = 0
    Disabled = 0
    Mesh = vtkUnstructuredGrid
    MeshInputFileName =
    vmtkRenderer = None
    Display = 1
    Opacity = 1.0
```

```
ArrayName =
ScalarRange = [0.0, 0.0]
Legend = 0
Grayscale = 0
FlatInterpolation = 0
MeshOutputFileName = meshlineartoqua.vtu
Executing vmtkmeshviewer ...
```



```
Done executing vmtkmeshviewer.
Writing VTK XML mesh file.
Output vmtkmeshviewer members:
Id = 0
Mesh = vtkUnstructuredGrid
```

ANNEX 24: VMTKMESHSCALING

```
vmtk vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshscaling -scale 0.1 --
pipe vmtkmeshviewer -ofile meshsacl.vtu
```

Executing vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshscaling -scale 0.1 --
pipe vmtkmeshviewer -ofile meshsacl.vtu

Creating vmtkMeshReader instance.

Automatic piping vmtkmeshreader

Parsing options vmtkmeshreader

 InputFileName = mesh.vtu

Explicit piping vmtkmeshreader

Input vmtkmeshreader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Mesh = 0

 InputFileName = mesh.vtu

 GhostNodes = 1

 VolumeElementsOnly = 0

 CellEntityIdsArrayName = CellEntityIds

 MeshOutputFileName =

Executing vmtkmeshreader ...

Reading VTK XML mesh file.

Done executing vmtkmeshreader.

Output vmtkmeshreader members:

 Id = 0

 Mesh = vtkUnstructuredGrid

 CellEntityIdsArrayName = CellEntityIds

Creating vmtkMeshScaling instance.

Automatic piping vmtkmeshscaling

 Mesh = vmtkmeshreader-0.Mesh

Parsing options vmtkmeshscaling

 ScaleFactor = 0.1

Explicit piping vmtkmeshscaling

Input vmtkmeshscaling members:

 Id = 0

 Disabled = 0

 Mesh = vtkUnstructuredGrid

 MeshInputFileName =

 ScaleFactor = 0.1

 MeshOutputFileName =

Executing vmtkmeshscaling ...

Done executing vmtkmeshscaling.

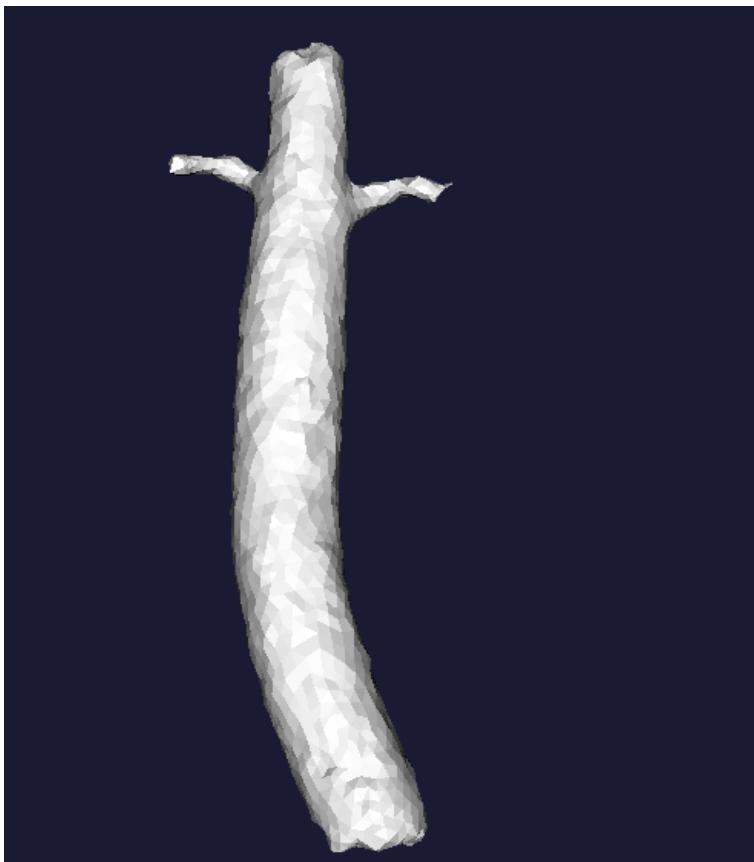
Output vmtkmeshscaling members:

 Id = 0

 Mesh = vtkUnstructuredGrid

Creating vmtkMeshViewer instance.

```
Automatic piping vmtkmeshviewer
  Mesh = vmtkmeshscaling-0.Mesh
Parsing options vmtkmeshviewer
  MeshOutputFileName = meshsacl.vtu
Explicit piping vmtkmeshviewer
Input vmtkmeshviewer members:
  Id = 0
  Disabled = 0
  Mesh = vtkUnstructuredGrid
  MeshInputFileName =
  vmtkRenderer = None
  Display = 1
  Opacity = 1.0
  ArrayName =
  ScalarRange = [0.0, 0.0]
  Legend = 0
  Grayscale = 0
  FlatInterpolation = 0
  MeshOutputFileName = meshsacl.vtu
Executing vmtkmeshviewer ...
```



```
Done executing vmtkmeshviewer.
Writing VTK XML mesh file.
Output vmtkmeshviewer members:
```

Id = 0
Mesh = vtkUnstructuredGrid

ANNEX 25: VMTKBOUNDARYINSPECTOR

```
vmtk vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshboundaryinspector  
-entityidsarray CellEntityIds -ofile meshbinspec.vtu  
Executing vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshboundaryinspector -  
entityidsarray CellEntityIds -ofile meshbinspec.vtu

Creating vmtkMeshReader instance.  
Automatic piping vmtkmeshreader  
Parsing options vmtkmeshreader  
    InputFileName = mesh.vtu  
Explicit piping vmtkmeshreader  
Input vmtkmeshreader members:  
    Id = 0  
    Disabled = 0  
    Format =  
    GuessFormat = 1  
    Mesh = 0  
    InputFileName = mesh.vtu  
    GhostNodes = 1  
    VolumeElementsOnly = 0  
    CellEntityIdsArrayName = CellEntityIds  
    MeshOutputFileName =  
Executing vmtkmeshreader ...  
Reading VTK XML mesh file.  
Done executing vmtkmeshreader.  
Output vmtkmeshreader members:  
    Id = 0  
    Mesh = vtkUnstructuredGrid  
    CellEntityIdsArrayName = CellEntityIds

Creating vmtkMeshBoundaryInspector instance.  
Automatic piping vmtkmeshboundaryinspector  
    Mesh = vmtkmeshreader-0.Mesh  
    CellEntityIdsArrayName = vmtkmeshreader-0.CellEntityIdsArrayName  
Parsing options vmtkmeshboundaryinspector  
    CellEntityIdsArrayName = CellEntityIds  
Explicit piping vmtkmeshboundaryinspector  
Input vmtkmeshboundaryinspector members:  
    Id = 0  
    Disabled = 0  
    Mesh = vtkUnstructuredGrid  
    MeshInputFileName =  
    CellEntityIdsArrayName = CellEntityIds  
    VolumeCellEntityId = 0  
    WallCellEntityId = 1  
    vmtkRenderer = None  
    ReferenceSystemsOutputFileName =  
Executing vmtkmeshboundaryinspector ...

CellEntityId: 1
```

```
Origin: 393.181519, 128.781250, 264.624054
Normal: 0.000000, -1.000000, 0.000000
Radius: 4.220500
```

```
CellEntityId: 1
Origin: 386.740112, 128.781250, 270.662048
Normal: 0.000000, -1.000000, 0.000000
Radius: 1.824768
```

```
CellEntityId: 1
Origin: 400.546356, 128.781250, 267.497162
Normal: 0.000000, -1.000000, 0.000000
Radius: 0.264853
```



```
Done executing vmtkmeshboundaryinspector.
```

```
Output vmtkmeshboundaryinspector members:
```

```
    Id = 0
```

```
    ReferenceSystems = vtkPolyData
```

```
Writing VTK XML mesh file.
```

```
Output vmtkmeshviewer members:
```

```
    Id = 0
```

```
    Mesh = vtkUnstructuredGrid
```

ANNEX 26: VMTKMESHTETRAHEDRALIZE

```
vmtk vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshtetrahedralize --pipe  
vmtkmeshviewer -ofile meshtetrah.vtu
```

Executing vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshtetrahedralize --pipe
vmtkmeshviewer -ofile meshtetrah.vtu

Creating vmtkMeshReader instance.

Automatic piping vmtkmeshreader

Parsing options vmtkmeshreader

 InputFileName = mesh.vtu

Explicit piping vmtkmeshreader

Input vmtkmeshreader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Mesh = 0

 InputFileName = mesh.vtu

 GhostNodes = 1

 VolumeElementsOnly = 0

 CellEntityIdsArrayName = CellEntityIds

 MeshOutputFileName =

Executing vmtkmeshreader ...

Reading VTK XML mesh file.

Done executing vmtkmeshreader.

Output vmtkmeshreader members:

 Id = 0

 Mesh = vtkUnstructuredGrid

 CellEntityIdsArrayName = CellEntityIds

Creating vmtkMeshTetrahedralize instance.

Automatic piping vmtkmeshtetrahedralize

 Mesh = vmtkmeshreader-0.Mesh

Parsing options vmtkmeshtetrahedralize

Explicit piping vmtkmeshtetrahedralize

Input vmtkmeshtetrahedralize members:

 Id = 0

 Disabled = 0

 Mesh = vtkUnstructuredGrid

 MeshInputFileName =

 TetrahedraOnly = 0

 MeshOutputFileName =

Executing vmtkmeshtetrahedralize ...

Done executing vmtkmeshtetrahedralize.

Output vmtkmeshtetrahedralize members:

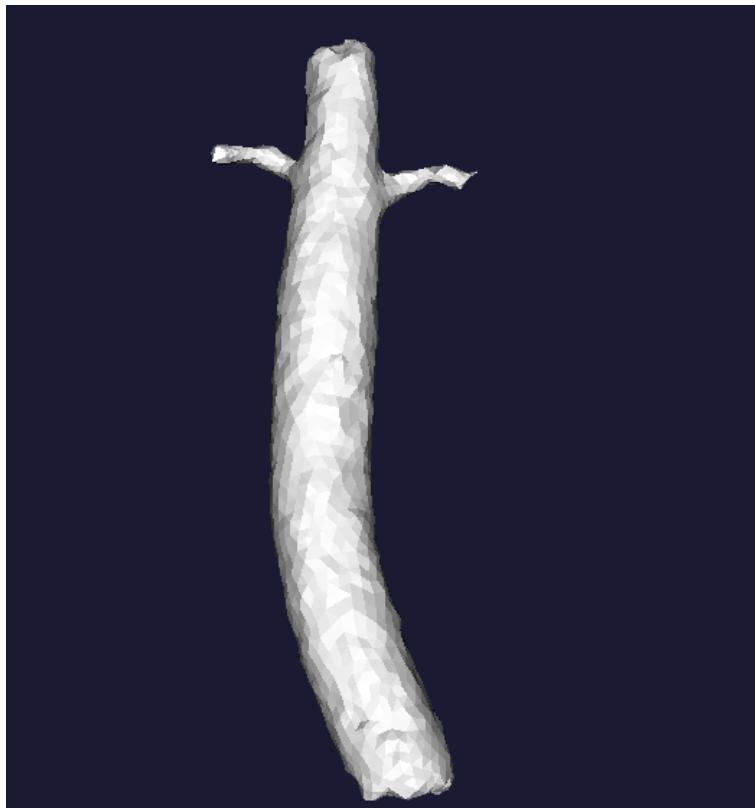
 Id = 0

 Mesh = vtkUnstructuredGrid

Creating vmtkMeshViewer instance.

Automatic piping vmtkmeshviewer

```
Mesh = vmtkmeshtetrahedralize-0.Mesh
Parsing options vmtkmeshviewer
  MeshOutputFileName = meshtetrah.vtu
Explicit piping vmtkmeshviewer
Input vmtkmeshviewer members:
  Id = 0
  Disabled = 0
  Mesh = vtkUnstructuredGrid
  MeshInputFileName =
  vmtkRenderer = None
  Display = 1
  Opacity = 1.0
  ArrayName =
  ScalarRange = [0.0, 0.0]
  Legend = 0
  Grayscale = 0
  FlatInterpolation = 0
  MeshOutputFileName = meshtetrah.vtu
Executing vmtkmeshviewer ...
```



```
Done executing vmtkmeshviewer.
Writing VTK XML mesh file.
Output vmtkmeshviewer members:
  Id = 0
```

Mesh = vtkUnstructuredGrid

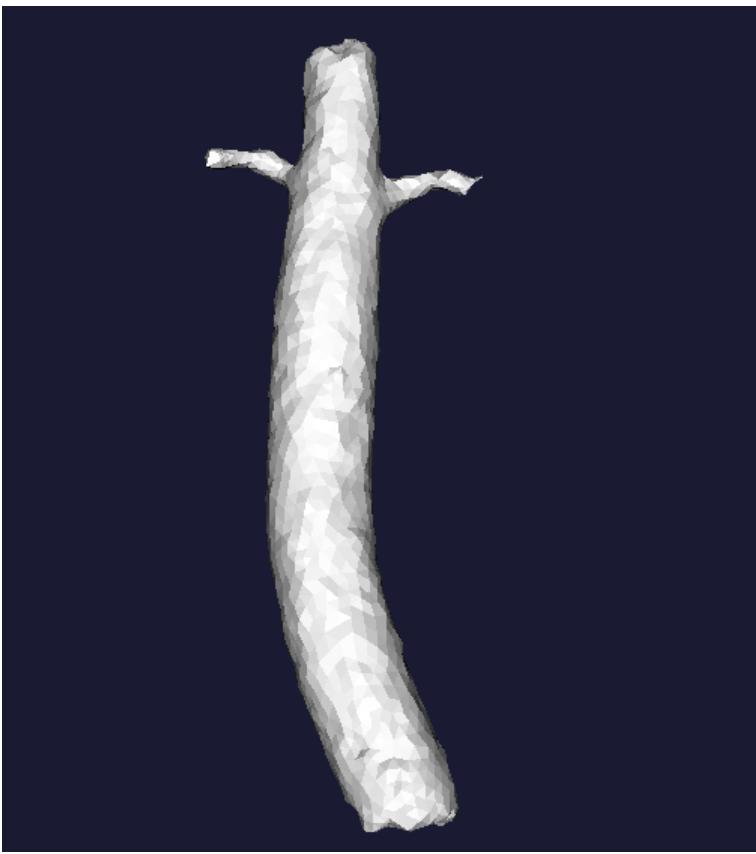
ANNEX 27: VMTKMESHLINEARIZE

```
vmtk vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshlinearize --pipe
vmtkmeshviewer -ofile meshlinea.vtu
Executing vmtkmeshreader -ifile mesh.vtu --pipe vmtkmeshlinearize --pipe
vmtkmeshviewer -ofile meshlinea.vtu

Creating vmtkMeshReader instance.
Automatic piping vmtkmeshreader
Parsing options vmtkmeshreader
  InputFileName = mesh.vtu
Explicit piping vmtkmeshreader
Input vmtkmeshreader members:
  Id = 0
  Disabled = 0
  Format =
  GuessFormat = 1
  Mesh = 0
  InputFileName = mesh.vtu
  GhostNodes = 1
  VolumeElementsOnly = 0
  CellEntityIdsArrayName = CellEntityIds
  MeshOutputFileName =
Executing vmtkmeshreader ...
Reading VTK XML mesh file.
Done executing vmtkmeshreader.
Output vmtkmeshreader members:
  Id = 0
  Mesh = vtkUnstructuredGrid
  CellEntityIdsArrayName = CellEntityIds

Creating vmtkMeshLinearize instance.
Automatic piping vmtkmeshlinearize
  Mesh = vmtkmeshreader-0.Mesh
Parsing options vmtkmeshlinearize
Explicit piping vmtkmeshlinearize
Input vmtkmeshlinearize members:
  Id = 0
  Disabled = 0
  Mesh = vtkUnstructuredGrid
  MeshInputFileName =
  CleanOutput = 1
  MeshOutputFileName =
Executing vmtkmeshlinearize ...
Done executing vmtkmeshlinearize.
Output vmtkmeshlinearize members:
  Id = 0
  Mesh = vtkUnstructuredGrid
```

```
Creating vmtkMeshViewer instance.  
Automatic piping vmtkmeshviewer  
    Mesh = vmtkmeshlinearize-0.Mesh  
Parsing options vmtkmeshviewer  
    MeshOutputFileName = meshlinea.vtu  
Explicit piping vmtkmeshviewer  
Input vmtkmeshviewer members:  
    Id = 0  
    Disabled = 0  
    Mesh = vtkUnstructuredGrid  
    MeshInputFileName =  
    vmtkRenderer = None  
    Display = 1  
    Opacity = 1.0  
    ArrayName =  
    ScalarRange = [0.0, 0.0]  
    Legend = 0  
    Grayscale = 0  
    FlatInterpolation = 0  
    MeshOutputFileName = meshlinea.vtu  
Executing vmtkmeshviewer ...
```



Done executing vmtkmeshviewer.
Writing VTK XML mesh file.
Output vmtkmeshviewer members:
Id = 0
Mesh = vtkUnstructuredGrid

ANNEX 28: MALLATGE A PARTIR DEL FORMAT STL

```
vmtk vmtksurfacereader -ifile marching.vtp -ofile meshformat.stl --pipe  
vmtksurfaceviewer
```

```
Executing vmtksurfacereader -ifile marching.vtp -ofile meshformat.stl --pipe  
vmtksurfaceviewer
```

Creating vmtkSurfaceReader instance.

Automatic piping vmtksurfacereader

Parsing options vmtksurfacereader

 InputFileName = marching.vtp

 SurfaceOutputFileName = meshformat.stl

Explicit piping vmtksurfacereader

Input vmtksurfacereader members:

 Id = 0

 Disabled = 0

 Format =

 GuessFormat = 1

 Surface = 0

 InputFileName = marching.vtp

 SurfaceOutputFileName = meshformat.stl

Executing vmtksurfacereader ...

Reading VTK XML surface file.

Done executing vmtksurfacereader.

Writing STL surface file.

Output vmtksurfacereader members:

 Id = 0

 Surface = vtkPolyData

Creating vmtkSurfaceViewer instance.

Automatic piping vmtksurfaceviewer

 Surface = vmtksurfacereader-0.Surface

Parsing options vmtksurfaceviewer

Explicit piping vmtksurfaceviewer

Input vmtksurfaceviewer members:

 Id = 0

 Disabled = 0

 Surface = vtkPolyData

 SurfaceInputFileName =

 vmtkRenderer = None

 Display = 1

 Opacity = 1.0

 ArrayName =

 ScalarRange = [0.0, 0.0]

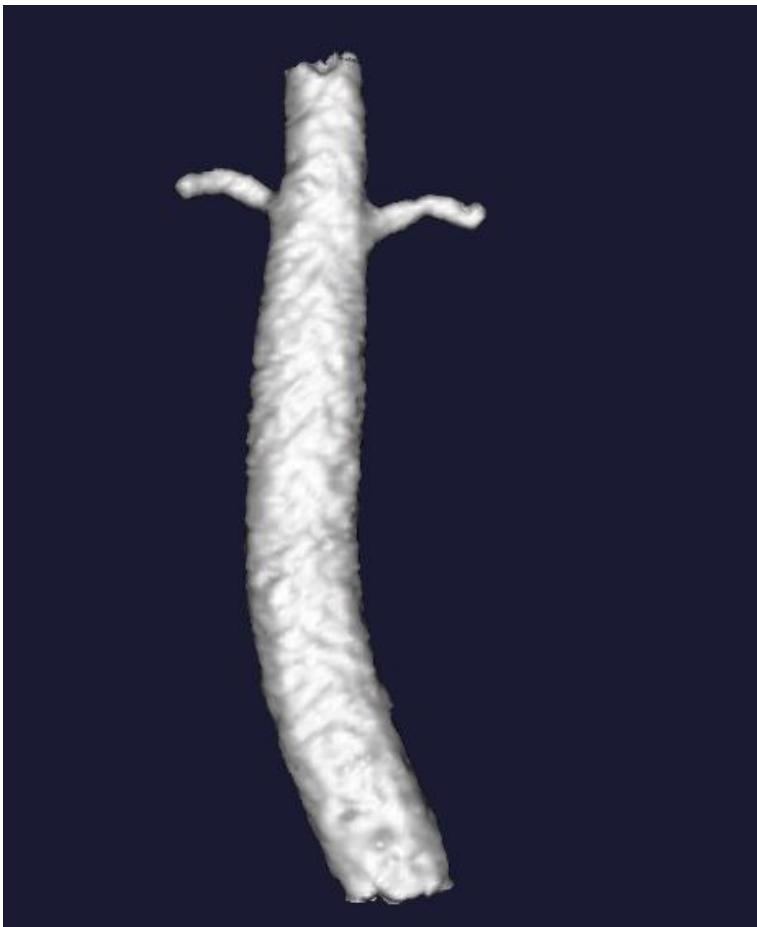
 Legend = 0

 Grayscale = 0

 FlatInterpolation = 0

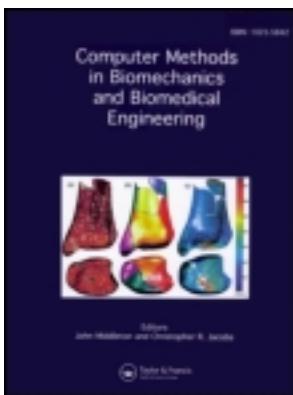
 DisplayCellData = 0

```
Color = [-1.0, -1.0, -1.0]
LineWidth = 1
LegendTitle =
SurfaceOutputFileName =
Executing vmtksurfaceviewer ...
```



```
Done executing vmtksurfaceviewer.
Output vmtksurfaceviewer members:
Id = 0
Surface = vtkPolyData
Actor = vtkActor
```

ANNEX 29: HEXAHEDRAL MESHES



Computer Methods in Biomechanics and Biomedical Engineering

Publication details, including instructions for authors and subscription information:

<http://www.tandfonline.com/loi/gcmb20>

Patient-specific computational haemodynamics: generation of structured and conformal hexahedral meshes from triangulated surfaces of vascular bifurcations

G. De Santis ^a, M. De Beule ^a, P. Segers ^a, P. Verdonck ^a & B. Verhegghe ^a

^a bioMMeda, Institute Biomedical Technology (IBiTech), Ghent University, De Pintelaan 185, Block B, BE-9000, Gent, Belgium

Available online: 24 May 2011

To cite this article: G. De Santis, M. De Beule, P. Segers, P. Verdonck & B. Verhegghe (2011): Patient-specific computational haemodynamics: generation of structured and conformal hexahedral meshes from triangulated surfaces of vascular bifurcations, *Computer Methods in Biomechanics and Biomedical Engineering*, 14:9, 797-802

To link to this article: <http://dx.doi.org/10.1080/10255842.2010.495066>

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: <http://www.tandfonline.com/page/terms-and-conditions>

This article may be used for research, teaching and private study purposes. Any substantial or systematic reproduction, re-distribution, re-selling, loan, sub-licensing, systematic supply or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

Patient-specific computational haemodynamics: generation of structured and conformal hexahedral meshes from triangulated surfaces of vascular bifurcations

G. De Santis*, M. De Beule, P. Segers, P. Verdonck and B. Verhegge

bioMMeda, Institute Biomedical Technology (IBiTech), Ghent University, De Pintelaan 185, Block B, BE-9000 Gent, Belgium

(Received 24 February 2010; final version received 18 May 2010)

Measuring the blood flow is still limited by current imaging technologies and is generally overcome using computational fluid dynamics (CFD) which, because of the complex geometry of blood vessels, has widely relied on tetrahedral meshes. Hexahedral meshes offer more accurate results with lower-density meshes and faster computation as compared to tetrahedral meshes, but their use is limited by the far more complex mesh generation. We present a robust methodology for conformal and structured hexahedral mesh generation – applicable to complex arterial geometries as bifurcating vessels – starting from triangulated surfaces. Cutting planes are used to slice the lumen surface and to construct longitudinal Bezier splines. Afterwards, an isoparametric transformation is used to map a parametrically defined quadrilateral surface mesh into the vessel volume, resulting in stacks of sections which can then be used for sweeping. Being robust and open source based, this methodology may improve the current standard in patient-specific mesh generation and enhance the reliability of CFD to patient-specific haemodynamics.

Keywords: computational fluid dynamics; pyFormex; patient specific; hexahedral mesh; STL; bifurcations

Introduction

Endothelial cells of blood vessels are known to be mechanosensitive and to be involved in controlling arterial tone (and lumen cross-section) and vascular remodelling (Malek et al. 1999; Carlier et al. 2003). Facing the blood stream, they resist the frictional forces exerted by blood (i.e. the wall shear stress, (WSS)), which is a viscous fluid. It is known that high WSS (~ 1.5 Pa or 15 dyne/cm 2) promotes atheroprotective endothelial phenotype whereas low (~ 0.4 Pa or 4 dyne/cm 2) and oscillating WSS promotes atherogenic commitment, which may result in atherosclerotic manifestations (Malek et al. 1999). Because direct measurements of WSS (it is a differential quantity) *in vivo* are difficult due to limited spatial and temporal resolution offered by current imaging technologies (mainly Doppler ultrasound and phase contrast MRI) (Katritsis et al. 2007; Ponzini et al. 2008; Morbiducci et al. 2009; Swillens et al. 2009), an indirect approach is often taken, integrating medical imaging techniques (biplane angiography, CT, MRI) with computational fluid dynamics (CFD) for patient-specific WSS profiling (Antiga et al. 2008).

In order to access the patient-specific haemodynamics, a number of steps need to be performed starting from the rough medical images. Software packages such as the vascular modelling toolkit (vmtk, www.vmtk.org) and Mimics (Materialise, Leuven, Belgium, www.materialise.com/mimics) offer advanced tools to generate the lumen surface, usually a triangulated surface (STL), from medical

images (3D angiography, CT, MRI). From this surface, the lumen volume needs to be discretised in order to produce a computational mesh for the numerical resolution of the Navier–Stokes equations. Volume discretisation (i.e. mesh generation) is a critical step in this procedure because the accuracy of the solution (e.g. velocity or WSS) is affected by (1) the mesh resolution, (2) the mesh element type (tetrahedral, prismatic or hexahedral) and (3) the mesh topology (structured or unstructured mesh). Because of the geometrical complexity of the blood vessels, which may include bifurcations and stenoses, almost all the current patient-specific CFD investigations rely on (semi-) automatically generated unstructured tetrahedral meshes, possibly with a prismatic boundary layer. In a previous study, we have investigated the performance of such meshes with respect to structured hexahedral meshes in calculating WSS and we have found that tetrahedral meshes required six times more cells to reach a mesh-independent solution (within 5% error) and 14 times longer computation times. Further refinement of these tetrahedral meshes did not reduce the mesh-dependent error, whereas with structured meshes, this error was progressively reduced to 0.6% (De Santis et al. 2010). In this paper, we propose a new semi-automatic tool to generate structured hexahedral meshes from the STL surface of a bifurcating vessel by combining a spline reconstruction of the luminal surface with cubic isoparametric mapping. Such meshes are required for reliable

CFD calculations, especially when aiming to investigate the local haemodynamics.

Methods

Angiographic images of a healthy carotid bifurcation were segmented using vmtk in order to generate a triangulated surface mesh representing the lumen profile without secondary branches (Figure 1). From this surface, a structured hexahedral mesh was generated using pyFormex, which is an open source software for generating, manipulating and transforming large geometrical models of 3D structures by sequences of mathematical transformations, currently under development at Ghent University (www.pyFormex.org).

Vessel surface reconstruction by longitudinal Bezier splines and longitudinal mesh resolution

The principal axes of inertia and the centre of gravity of the surface are calculated in order to orient the bifurcation plane perpendicularly to the z-axis and allow the user to select a set of points on the x-y plane (at least 2) along each side of the branches. These landmarks, together with an extra point which identifies the bifurcation centre, define the bifurcation branching topology (Figure 1) and are used to automatically orient and position a series of slicing semi-planes which gradually approach the bifurcation centre. Cutting the triangulated surface with each semi-plane produces an intersecting boundary line which reproduces half a cross-section of a branch (semi-slice). Repeating this slicing operation over an entire branch produces two stacks of semi-planar slices, which protrude inside the bifurcation towards a triple-sided slice, required to join the three branches together (Figure 2). Each semi-slice is split into 12 equally spaced segments which are used to define 13 longitudinal Bezier splines along each semi-branch (a different number of splines per semi-branch may be chosen depending on the accuracy of the original STL surface). Splines of contiguous

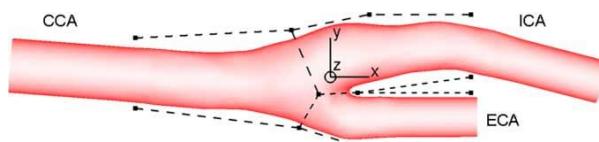


Figure 1. A triangulated surface of a carotid bifurcation of a healthy subject is oriented according to the principal axes of inertia (solid lines indicate the coordinate system). Using the new coordinate system, the user can indicate some lines (dotted lines) and a central point in order to define the topological branching: common carotid artery (CCA), internal carotid artery (ICA) and external carotid artery (ECA).

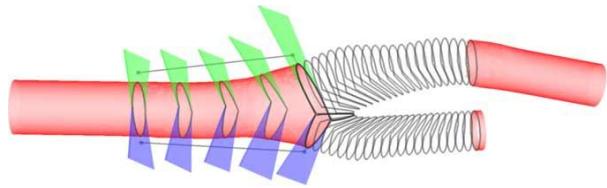


Figure 2. The STL is sliced using cutting semi-planes according to the points inputted by the user (only four black points on the CCA are shown). The use of semi-planes allows the acquisition of semi-slices within the bifurcating region up to a three-sided section (dark lines at the centre). The number of cutting semi-planes per branch can be changed because a high (low) number of semi-slices may be beneficial if the STL has a high (low) resolution.

semi-branches are connected together in order to keep G1 continuity (each spline is a path with a continuous tangent between two flow boundaries, i.e., inlet to first outlet, first to second outlet, inlet to second outlet), except for the first and the last splines which end at the top and at the bottom point of the bifurcation axis. As a result, 24 splines around the centreline describe a single branch, which can be partitioned in quarters (each quarter is delimited by 8 splines, 7 on the boundary and 1 on the vessel centreline). The splines on each branch are divided into longitudinal segments, the density of which determines the longitudinal resolution of the final mesh on each branch. This longitudinal division is performed using curved profiles based on the cutting planes, in order to avoid the creation of sharp edges where the quarters of each branch connect together (Figure 3).

Isoparametric transformation and cross-sectional mesh resolution

The vertices used for the spline division are grouped in sections sorted longitudinally. Each quarter of a section

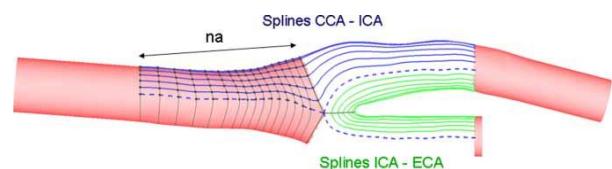


Figure 3. Longitudinal Bezier splines reproduce the vessel boundary, spanning from one branch to another. The seven blue lines represent one quarter of the CCA and the ICA whereas the seven green lines represent one quarter of the ICA and the ECA. The dotted lines represent the three top splines at the centre of the branches which converge to the bifurcation centre. The centrelines are located under the dotted lines. The number of curved profiles used to cut the Bezier splines of the CCA, na, is the number of cells aligned longitudinally in the CCA in the final mesh and therefore represents a parameter of longitudinal mesh resolution.

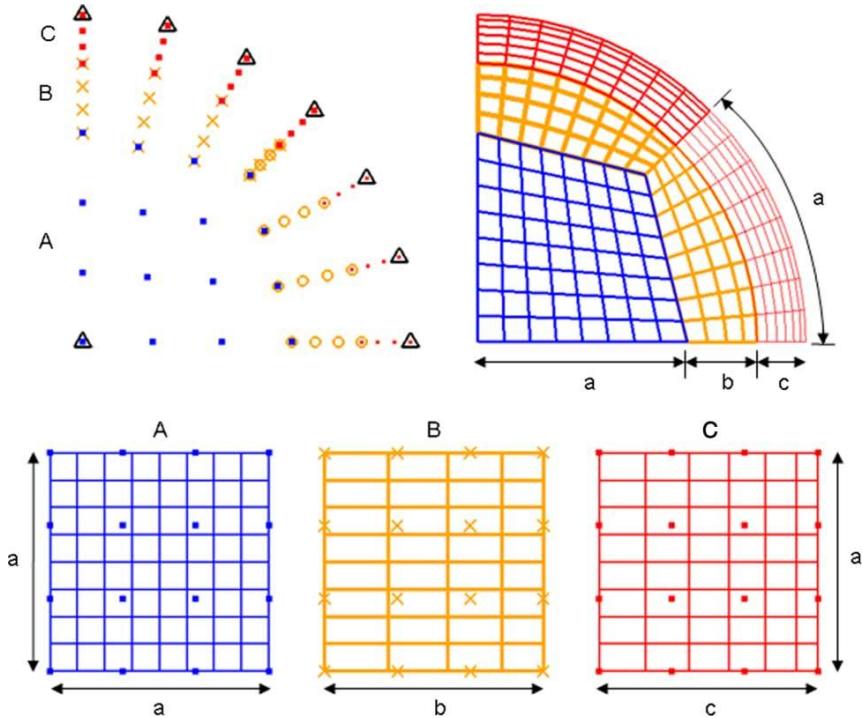


Figure 4. The seven points on the boundary and the single point on the centreline describe one quarter of a section. On the basis of these initial points (triangles), 5 sets of 16 points each are created (coloured points). Three parametrically defined quadrilateral patterns (A, B, C) are mapped into the 5 regions by using an isoparametric transformation. Three parameters (a , b , c) describe the number of cells in the inner-squared pattern (blue, $a \cdot a$ cells), in the two-transition region (yellow, $2 \cdot a \cdot b$ cells) and in the boundary layer (red, $2 \cdot a \cdot c$ cells), defining the cross-sectional mesh density. The cells become gradually thinner moving towards the wall, in order to better resolve the flow patterns near the wall for the calculation of the WSS.

contains 8 points, 7 on the lumen and 1 on the centreline, which are used to create 5 sets of 16 points. These sets of points are used to control a bi-cubic surface isoparametric transformation which maps a planar quadrilateral surface (original surface mesh) into a cubic surface according to the coordinates of the 16 control points (Figure 4). The resulting stack of 3D quadrilateral surface meshes can be used to generate a 2.5 D volume mesh by performing a sweeping operation along the sections (all the sections have the same connectivity; Figure 5, top).

Mesh quality and geometrical accuracy

The relative difference of volume and area of the final mesh with respect to the original surface has been taken as a measure of geometrical accuracy. Local accuracy has been quantified by measuring the signed distance between the two surfaces (positive values corresponding to measurement points located outside the STL model – functionality provided by vmtk) relative to the vessel diameter (the smallest diameter has been chosen in order to define an upper limit for the error) (Antiga et al. 2008). The parameter equiangle skew (Qeas) has been used to describe the mesh quality, being a normalised measure of

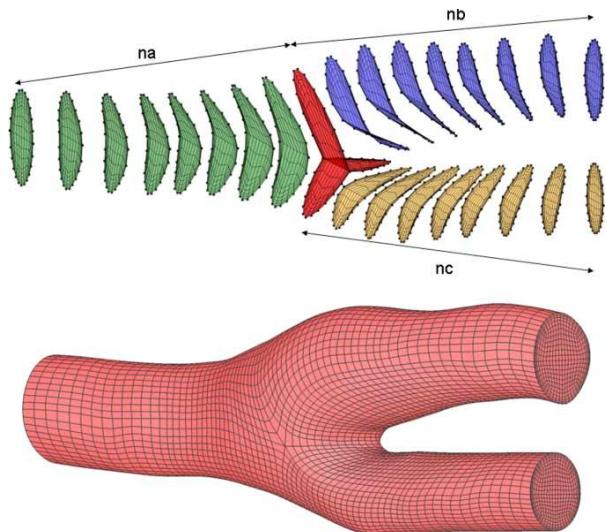


Figure 5. Top: three stacks of sections originate from the three-sided central section (only some sections are shown for clarity). Each section comprises four quarters (i.e. quadrilateral meshes) which have a cross-sectional resolution dependent on the parameters (a , b , c) described in Figure 4. The number of sections aligned longitudinally for each branch (na , nb , nc) defines the longitudinal mesh resolution (Figure 3). Bottom: the final 3D volume mesh is obtained by sweeping the sections stacked along the three branches.

Table 1. Three different sets of mesh parameters are used to generate coarse, medium and fine meshes. The mesh has high quality in terms of skewness (mean Qeas \approx 0.25 and max Qeas \approx 0.75) and accurately reproduces the initial vessel volume (relative differences of volume and area are below 0.3% and upper limit of the surface-to-surface distance is two orders of magnitude lower than the smallest diameter).

Cross-sectional Mesh parameters: a-b-c ^a	Longitudinal parameters: na-nodes ^a	Cells ^b	Qeas mean (max)	Relative jDarea _j /Volume _j	Relative distance mean (max)
Coarse-2-2	24-18-18	8910	0.24 (0.73)	2.7×10^{-23}	1.98×10^{-25} (1.8×10^{-22})
Medium-4	59-50-44	78,009	0.23 (0.74)	2.9×10^{-23}	1.21×10^{-25} (2.7×10^{-22})
Fine 13-10-10	130-120-113	1,023,847	0.21 (0.74)	2.9×10^{-23}	2.78×10^{-25} (2.8×10^{-22})

^a The cross-sectional parameters (a, b, c) are described in Figure 4 while the longitudinal parameters (na, nb, nc) are described in Figures 3 and 5.

^b Because the mesh is structured, the number of cells is determined a priori by the selected set of parameters and can be quantified using the following equation: Number of cells \approx $4a \cdot nb \cdot 2b \cdot 2c \cdot nc$.

skewness ranging from 0.0 (perfect equiangular cell) to 1.0 (maximally distorted cell).

The described user friendly methodology has also been applied to a stenosed carotid bifurcation of an 83-year-old male patient, segmented using Mimics, in order to show how a single non-tubular bifurcation can be meshed using the Graphical User Interface within one minute (Movie-1 and Movie-2 in Supplementary material).

Results

From the original triangulated surface mesh of a patient-specific healthy carotid bifurcation, a parametrically defined conformal structured hexahedral mesh is generated (Figure 5, bottom).

By setting different values for the three cross-sectional (Figure 4) and the three longitudinal (Figures 3 and 5) mesh parameters, meshes with different resolution are obtained. High mesh quality (mean Qeas \approx 0.25 and max Qeas \approx 0.75) and high geometrical accuracy (relative differences of volume and area $<$ 0.3%) are maintained both in coarse and fine computational grids (see Table 1). Local distances between the original STL model and the outer surface of the hexahedral mesh are in average five orders of magnitude smaller than the vessel diameter with peak values occurring at the carina (from \approx 0.14 to \approx 0.058 mm, average 0.00014 mm). Such peaks are not an artefact of the meshing process but are related to the mesh resolution of the original STL model: the carina, being the region with highest curvature, would require a finer mesh, whereas, due to the homogeneity size of the triangles (edges from 0.19 to 0.71 mm), its representation is suboptimal, with angles between adjacent triangular cells reaching 288.

Discussion

A novel methodology for generating high-quality hexahedral meshes for CFD analysis is proposed. These meshes are conformal, structured and accurately match the surface of vascular districts reconstructed from medical images. The new mesher is robust and automatic, because the operator interaction is performed at high level and is limited to the definition of the vessel topology; and parametrical, because both the cross-sectional and the longitudinal mesh resolution can be defined by setting six parameters.

In a structured hexahedral mesh, the cells can be oriented along one direction, thus allowing alignment of the mesh gridlines (i.e. the cell edges) with the velocity field. It is well known that such alignment is beneficial for CFD calculations because it reduces the computational errors due to the numerical diffusion which occurs during the integration of the flow equations. Tetrahedral meshes

(and unstructured meshes in general), having randomly distributed cells and not aligned edges, introduce relevant numerical diffusion errors in CFD calculation, resulting in lower accuracy of the results and longer computational times. Moreover, the difference between structured hexahedral with respect to unstructured meshes is amplified in mainly directional flow systems like blood vessels and lung airways (Longest and Vinchurkar 2007; Vinchurkar and Longest 2008; De Santis et al. 2010).

Despite their limitations, mainly unstructured meshes have been used in patient-specific CFD, because they can be automatically generated in nearly any geometry (using common software packages such as Gambit, TGrid and Tetgen). Generation of hexahedral meshes is more difficult, generally requiring extensive operator interaction which can be prohibitively time consuming and needs to be repeated de novo to mesh a new vessel (Antiga et al. 2008). With respect to our previous methodology (i.e., generation of structured meshes starting from tubular vessel geometries described by differently oriented circular sections aligned along centrelines (De Santis et al. 2010)), the vessels surface can now be non-tubular and present asymmetric stenosis, as shown in Movie-2. Moreover, a graphical user interface is provided in order to address the meshing of a single bifurcation from any triangulated surface model to a structured volume mesh within one minute, as shown in Movie-1. An automatic hexahedral mesher for vascular bifurcations has been developed by Verma and collaborators but it requires the numerical solution of three additional heat conduction problems, whereas our methodology is purely geometrical. Moreover, solving these additional problems requires the generation of unstructured grids and can be non-trivial because the computational cost considerably increases with the desired hexahedral mesh resolution (Verma et al. 2005).

The new proposed tool allows a straightforward and semi-automatic generation of structured hexahedral meshes and it is therefore suitable for large population studies. Remarkably, it is not as versatile as unstructured mesh generators, because it has been designed and optimised for vascular districts, including non-tubular vessels, stenoses and bifurcations.

Conclusion

High-quality hexahedral meshes in patient-specific vascular fluid domains are required to reduce the numerical errors and the computational time associated with CFD analyses. Being user friendly (it is managed by a graphical user interface) and open source based, the developed innovative tool may improve the current standard in vascular patient-specific CFD. Current development towards coupling the new pyFormex mesher with the

open source vascular modelling package vmtk will allow automatic hexahedral meshing of multiple bifurcations, extending its applications to more complex districts, such as coronary arteries and the cerebral circulation.

Acknowledgements

The Authors thank Luca Antiga, PhD, and ir. Michele Conti for kindly providing the triangulated surfaces used to describe the new meshing methodology, and ir. Peter Mortier for his support and for carefully proofreading the manuscript.

References

- Antiga L, Piccinelli M, Botti L, Ene-Iordache B, Remuzzi A, Steinman D. 2008. An image-based modeling framework for patient-specific computational hemodynamics. *Med Biol Eng Comput.* 46(11):1097–1112.
- Carlier SG, van Damme LCA, Blommerde CP, Wentzel JJ, van Langehove G, Verheyen S, Kockx MM, Knaapen MWM, Cheng C, Gijsen F et al., 2003. Augmentation of wall shear stress inhibits neointimal hyperplasia after stent implantation – Inhibition through reduction of inflammation? *Circulation.* 107(21):2741–2746.
- De Santis G, Mortier P, De Beule M, Segers P, Verdonck P, Verhegge B. 2010. Patient specific computational fluid dynamics: structured mesh generation from coronary angiography. *Med Biol Eng Comput.* 48(4):371–380.
- Katritsis D, Kaittsis L, Chaniotis A, Pantos J, Efstatopoulos EP, Marmarelis V. 2007. Wall shear stress: theoretical considerations and methods of measurement. *Prog Cardiovasc Dis.* 49(5):307–329.
- Longest PW, Vinchurkar S. 2007. Effects of mesh style and grid convergence on particle deposition in bifurcating airway models with comparisons to experimental data. *Med Eng Phys.* 29(3):350–366.
- Malek AM, Alper SL, Izumo S. 1999. Hemodynamic shear stress and its role in atherosclerosis. *JAMA.* 282(21):2035–2042.
- Morbiducci U, Ponzini R, Rizzo G, Cadioli M, Esposito A, De Cobelli F, Del Maschio A, Monteverchi FM, Redaelli A. 2009. In vivo quantification of helical blood flow in human aorta by time-resolved three-dimensional cine phase contrast magnetic resonance imaging. *Ann Biomed Eng.* 37(3):516–531.
- Ponzini R, Lemma M, Morbiducci U, Monteverchi FM, Redaelli A. 2008. Doppler derived quantitative flow estimate in coronary artery bypass graft: a computational multiscale model for the evaluation of the current clinical procedure. *Med Eng Phys.* 30(7):809–816.
- Swillens A, De Schryver T, Lovstakken L, Torp H, Segers P. 2009. Assessment of numerical simulation strategies for ultrasonic color blood flow imaging, based on a computer and experimental model of the carotid artery. *Ann Biomed Eng.* 37(11):2188–2199.
- Verma CS, Fischer PF, Lee SE, Loth F. 2005. An all-hex meshing strategy for bifurcation geometries in vascular flow simulation, Proceedings of the 14th International Meshing Roundtable; San Diego, CA. p. 363–375.
- Vinchurkar S, Longest PW. 2008. Evaluation of hexahedral, prismatic and hybrid mesh styles for simulating respiratory aerosol dynamics. *Comput Fluids.* 37(3):317–33.

