vmtk - the Vascular Modeling Toolkit

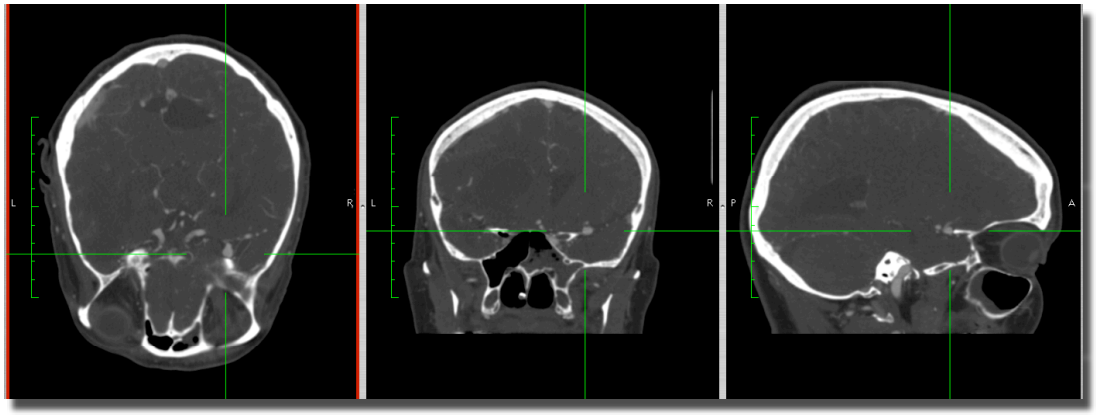
Hands-on workshop

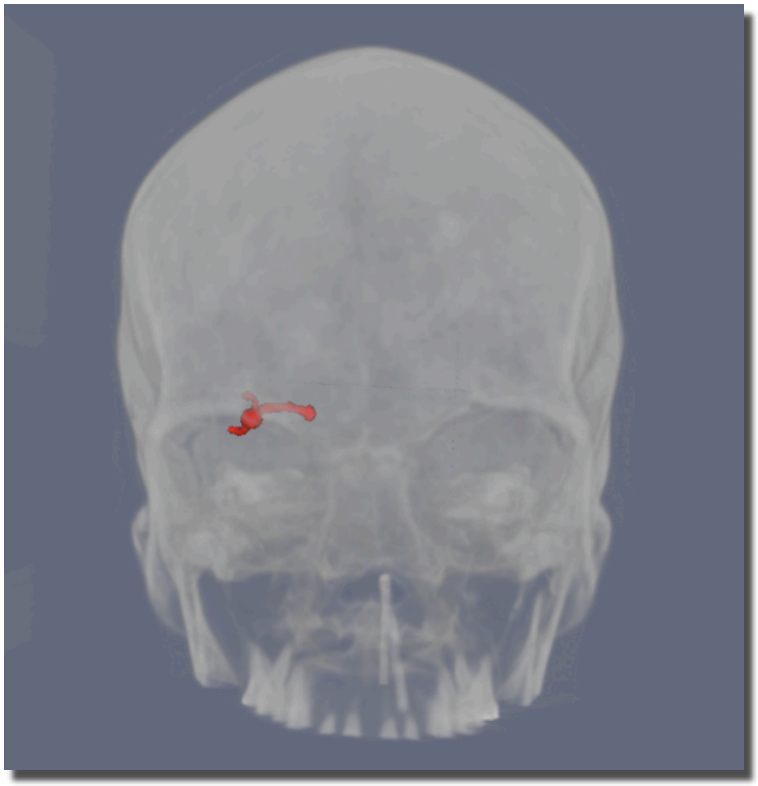
ASME Summer Bioengineering Conference 2008
Marriot Resort, Marco Island, FL

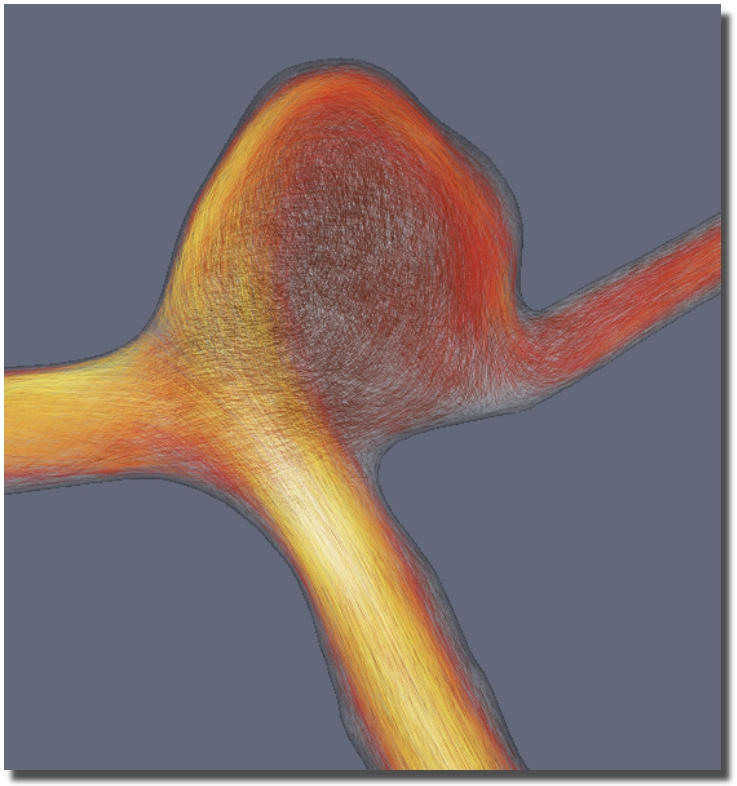
Luca Antiga, PhD
Medical Imaging Unit
Biomedical Engineering Department
Mario Negri Institute, Bergamo, Italy
antiga@marionegri.it

David A Steinman, PhD
Biomedical Simulation Lab
Mechanical and Industrial Engineering
University of Toronto, Canada
steinman@mie.utoronto.ca

Image-based computational hemodynamics: a reality







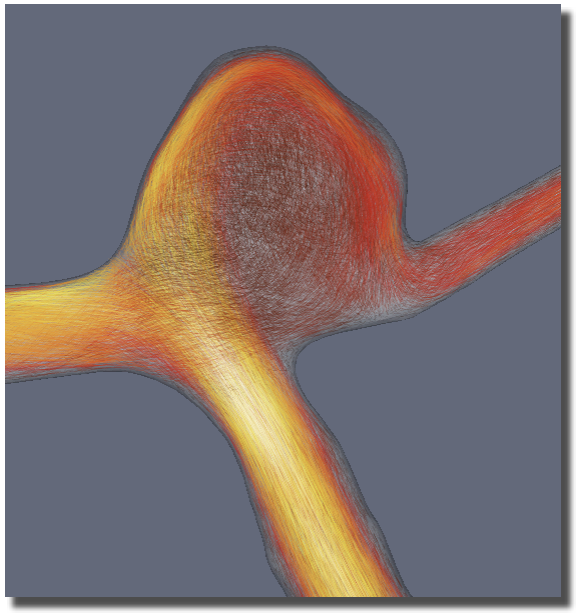
Current challenges

Creating better (more sophisticated) models:

- fluid-structure interaction
- non-Newtonian rheology
- multi-scale modelling
- solute transport
- chemical reactions
- cell biology
- ...

Reaching clinical relevance:

high-throughput patient-specific simulations



Ways towards evidence

Image-based computational hemodynamics is still dependent on many "manufacturing" steps, often not well documented

image segmentation
model editing
mesh generation
CFD
post-processing

Their impact on results in terms of accuracy, reproducibility and operator-dependence is often unknown and likely non-negligible

High-throughput computational hemodynamics:

user decisions vs user dependence
reproducibility
automated mesh generation
robust solvers
geometry-savy post-processing

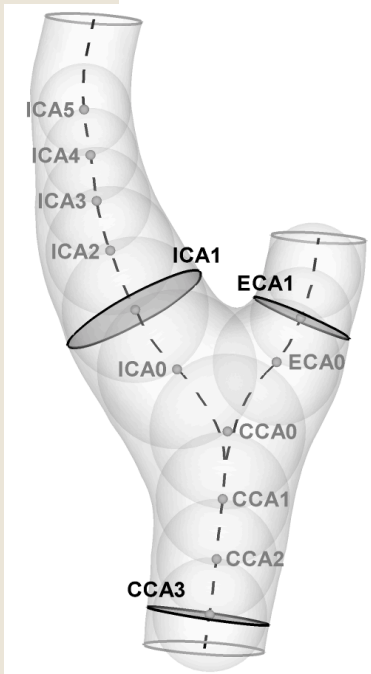
<u>Shared tools, shared data</u>

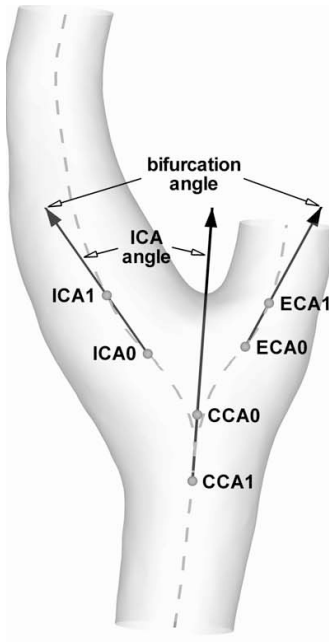
Ways towards evidence

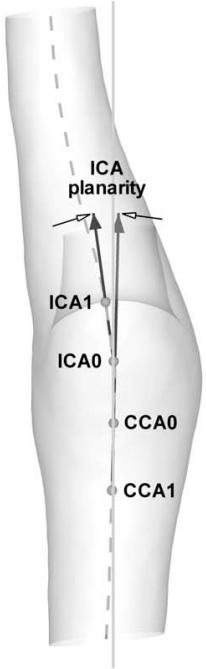
If we can only observe only gross hemodynamics, can't we find simple criteria that anticipate it?

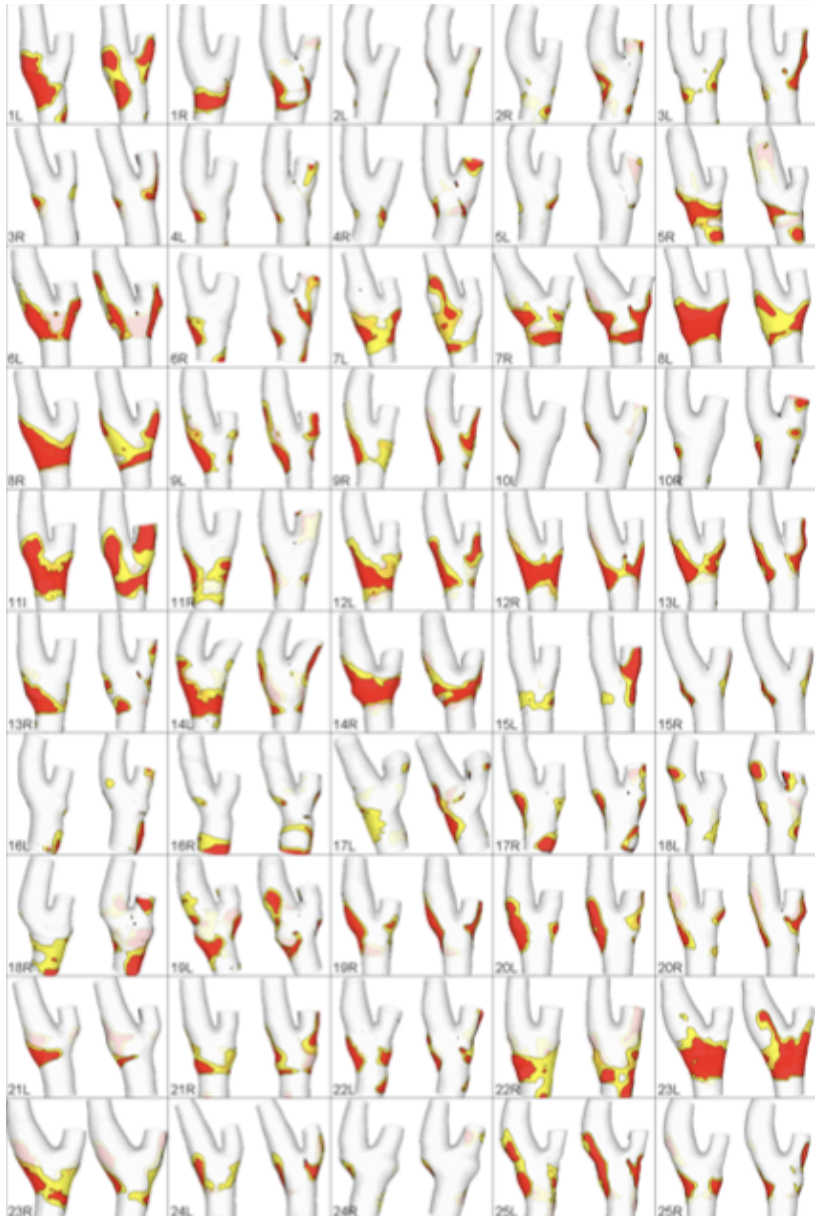
Is geometry capable of anticipating gross hemodynamics?

Can we find geometric surrogates to use in large-scale clinical studies?







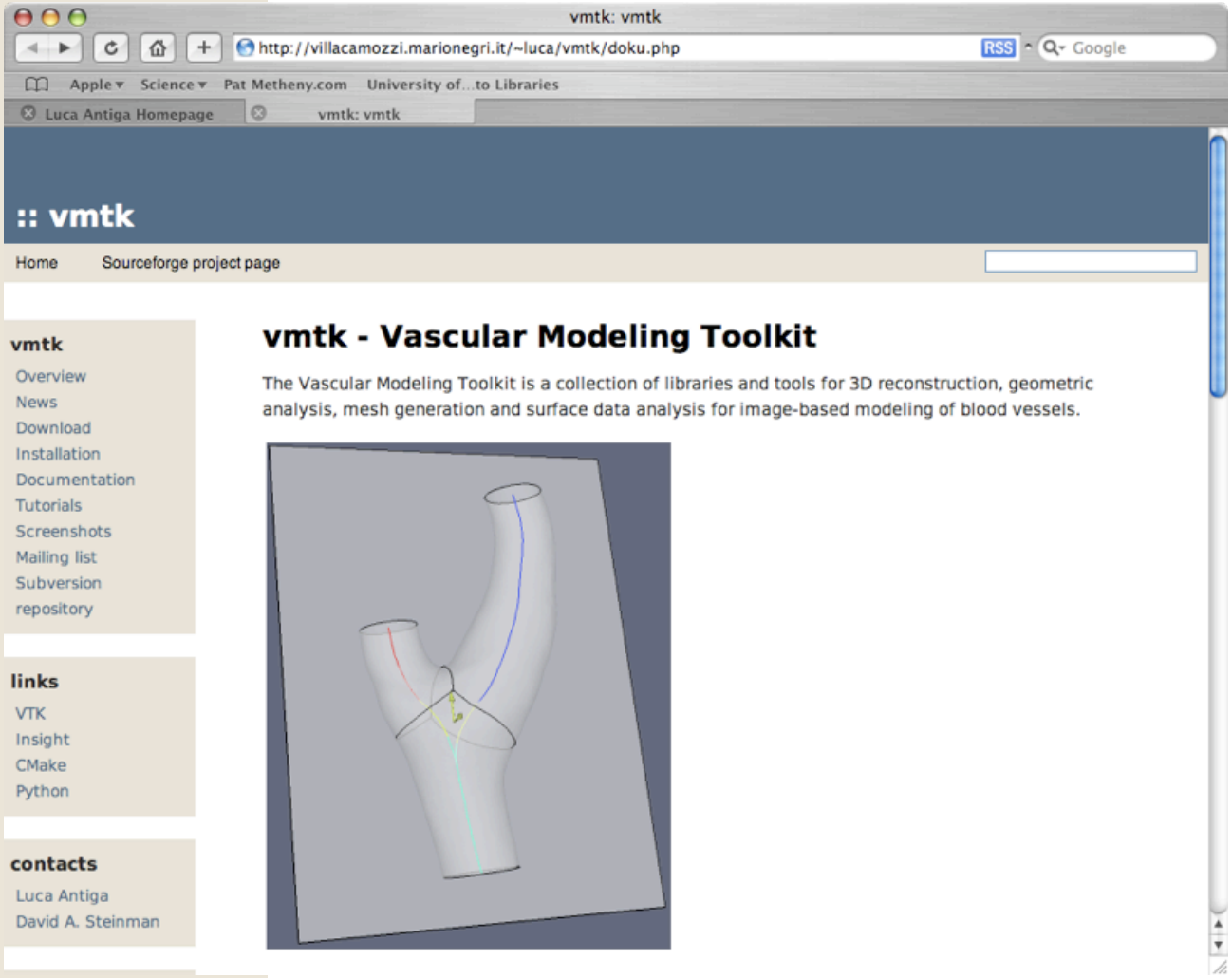


Lee et al, Stroke Aug 2008 (front cover)

vmtk - the Vascular Modeling Toolkit

image segmentation, geometric analysis, mesh generation, post processing

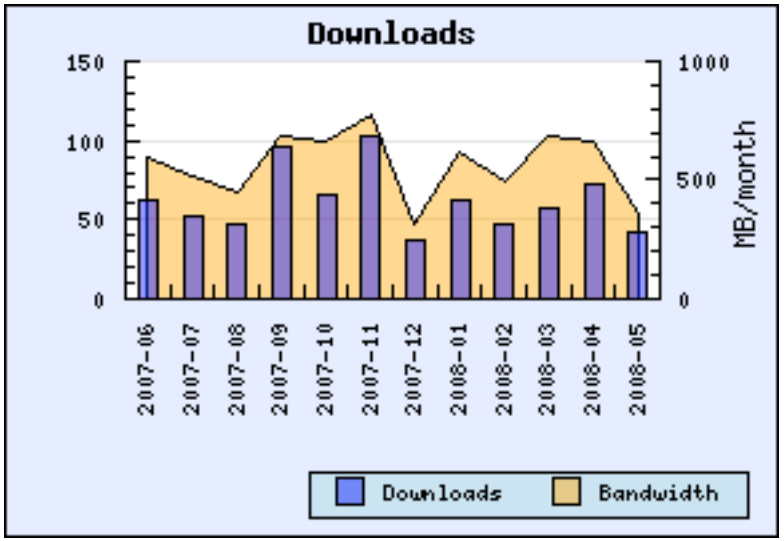
http://vmtk.sourceforge.net



Luca Antiga, Mario Negri Institute
David Steinman, University of Toronto

platforms: Linux, Mac OSX, Windows
licence: BSD
based on: VTK, ITK, Tetgen

coming soon: vmtk 0.7



vmtk - Main features

- Level-set and surface deformable model segmentation
- Branch-wise interactive initialization
- Image processing and enhancement
- Centerline computation
- Geometric analysis of vessels, shapes, bifurcations
- Surface processing (decimation, smoothing, healing, capping)
- CFD pre-processing (flow extensions, model editing)
- Mesh generation
- Finite element framework for surface mapping and CFD post-processing
- Surface mapping and patching for population studies

vmtk Architecture

vmtk

PypeS	vmtkScripts	vtkVmtk
Python pipeable scripts framwork	High-level Python scripts	C++ algorithms

VTK

Visualization Toolkit

ITK

Insight Segmentation and Registration Toolkit

Tetgen

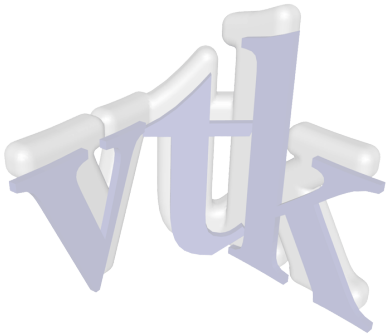
Quality Mesh Generator

Base libraries

VTK - Visualization Toolkit

www.vtk.org

Data structures (surface, mesh, image)
Surface and mesh processing algorithms
Image processing algorithms
Contouring (Marching Cubes)
Visualization and interaction
I/O functionality



ITK - Insight Segmentation and Registration Toolkit

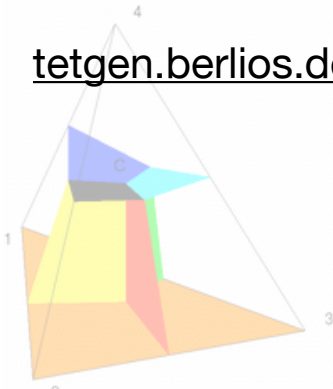
www.itk.org

Data structures (image, mesh)
Advanced image processing algorithms
Segmentation algorithms (e.g. level sets)
Registration algorithms
Spatial objects
I/O functionality

Tetgen - Quality Tetrahedral Mesh Generator

tetgen.berlios.de

Tetrahedral mesh generation
Delaunay tessellation



Downloading vmtk

Release version

Go to the vmtk page and follow the download page, or go directly to

http://sourceforge.net/projects/vmtk

and download vmtk-0.7.tar.gz

Development version

Install Subversion and run

```
svn co https://vmtk.svn.sourceforge.net/svnroot/vmtk vmtk
```

Compiling vmtk

Install CMake (http://www.cmake.org)

(official packages are available for most linux distributions)

Install Python (http://www.python.org)

Compile and install VTK (http://www.vtk.org)

(official packages are available for some linux distributions)

CMake flags: BUILD_SHARED_LIBS = ON
VTK_WRAP_PYTHON = ON

Compile and install the Insight Toolkit (http://www.itk.org)

CMake flags: BUILD_SHARED_LIBS = ON

Compile and install vmtk

Set environment variables

```
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/vtk-5.2
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/InsightToolkit
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib/vmtk
export LD_LIBRARY_PATH
PYTHONPATH=$PYTHONPATH:/usr/local/lib/vmtk
export PYTHONPATH
```

Binary vmtk packages for Ubuntu 8.04 (32 and 64 bit)

Packages provided by Johannes Ring, Simula Research Laboratory, Norway

Add two repositories

```
sudo wget http://packages.simula.no/hardy.list -O /etc/apt/sources.list.d/simula.list

sudo wget http://apt.paulnovo.org/hardy.list -O /etc/apt/sources.list.d/paulnovo.list
```

Add the respective public keys

```
wget http://packages.simula.no/pubring.gpg -O- | sudo apt-key add -

wget http://apt.paulnovo.org/549EC7E2.key -O- | sudo apt-key add -
```

Update the packages

```
sudo apt-get update
```

Install the vmtk package

```
sudo apt-get install vmtk0-bin
```

this will install a binary vmtk with all the dependencies (VTK, Insight, etc)

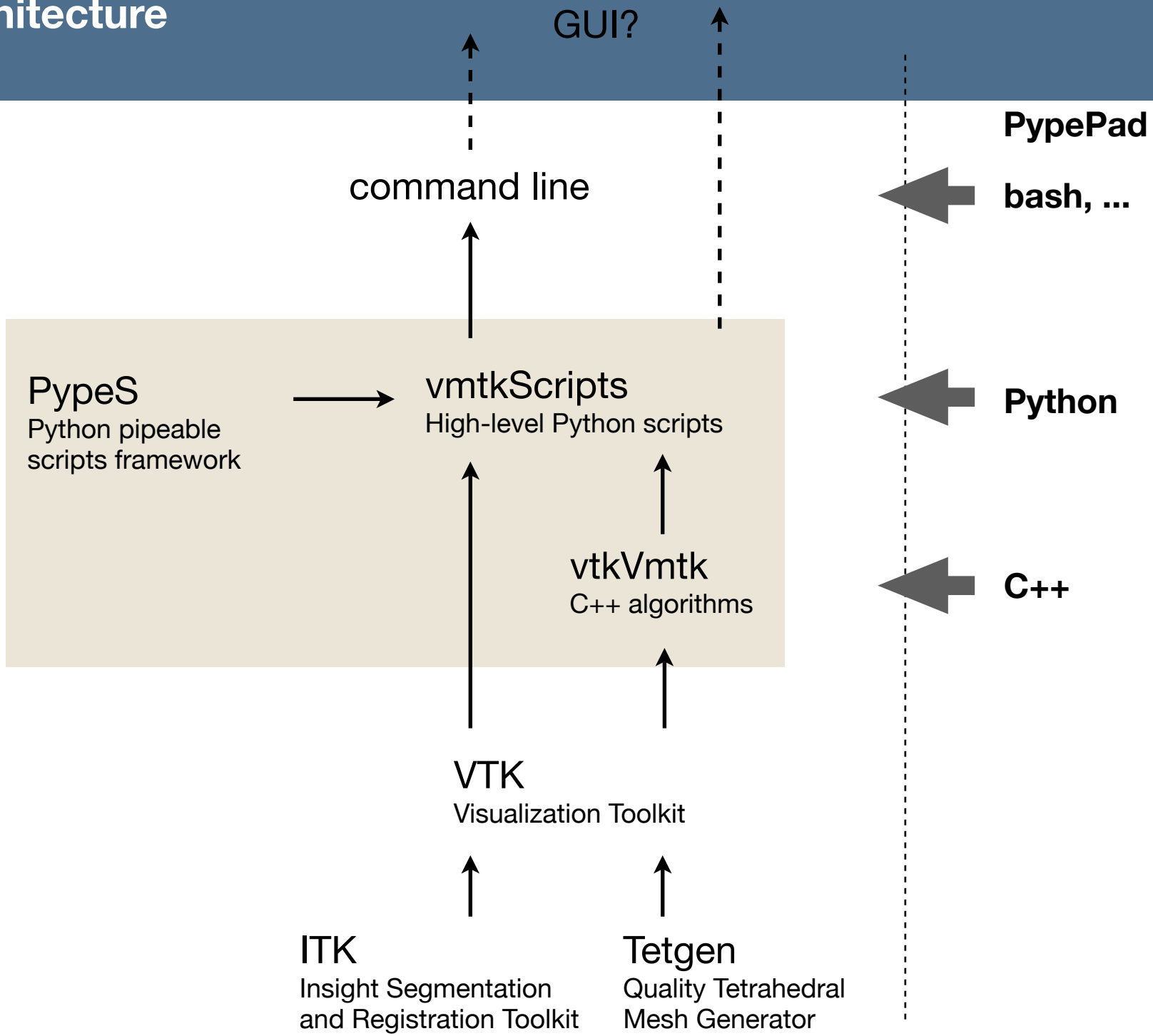
Binary packages for Windows

...volunteer anybody?

Binary packages for Windows

...I'm serious!

vmtk Architecture



vmtk - Rationale for the design

C++: algorithm efficiency

Python: algorithm workflow management and extensibility

Pypes: high-level flexibility

```
vmtkimagereader -ifile foo.vti --pipe vmtkmarchingcubes -l 200 --pipe
vmtksurfacewriter -ofile bar.vtp

vmtkimagereader -ifile foo.vti --pipe vmtkmarchingcubes -l 200 --pipe
vmtksurfacesmoothing -iterations 30 -passband 0.1 --pipe
vmtksurfacewriter -ofile bar.vtp
```

Possibility of GUI layer (without changing the underlying code)

The curse of script proliferation

vtkXMLImageDataReader.h

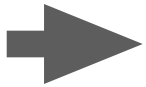
```
class vtkXMLImageDataReader :
     public vtkXMLDataReader
{
public:
     vtkXMLImageDataReader();
     ~vtkXMLImageDataReader();
     void SetFileName(const char* filename);
     void Update();
};
```

vtkMarchingCubes.h

```
class vtkMarchingCubes :
     public vtkPolyDataAlgorithm
{
public:
     vtkMarchingCubes();
     ~vtkMarchingCubes();
     void SetInput(const vtkImageData* input);
     void Update();
};
```

vtkXMLPolyDataWriter.h

```
class vtkXMLPolyDataWriter :
     public vtkXMLDataWriter
{
public:
     vtkXMLPolyDataWriter();
     ~vtkXMLPolyDataWriter();
     void SetInput(const vtkPolyData* input);
     void SetFileName(const char* filename);
     void Write();
};
```



```
#!/usr/bin/env python

import vtk

reader = vtk.vtkXMLImageDataReader()
reader.SetFileName('foo.vti')
reader.Update()

mc = vtk.vtkMarchingCubes()
mc.SetInput(reader.GetOutput())
mc.Update()

writer = vtk.vtkXMLPolyDataWriter()
writer.SetInput(mc.GetOutput())
writer.SetFileName('bar.vtp')
writer.Write()
```

intermediate files and script proliferation:

e.g. “I want a script that smooths the surface with vtkSmoothPolyData after applying Marching Cubes”:

either write a separate script (intermediate files) or write another version of this script (script proliferation)

code duplication:

I/O (formats), option parsing, usage strings, ...

PypeS - Python pypeable scripts

PypeS allows to write scripts that

- do *one* thing each
- can be assembled on the command line (similarly to Unix pipes)
- can pipe multiple arguments (unlike Unix pipes)
- can be used as classes to build other scripts or to develop GUI applications

Each vmtkScript is derived from a PypeScript class that handles option parsing, usage strings and argument piping

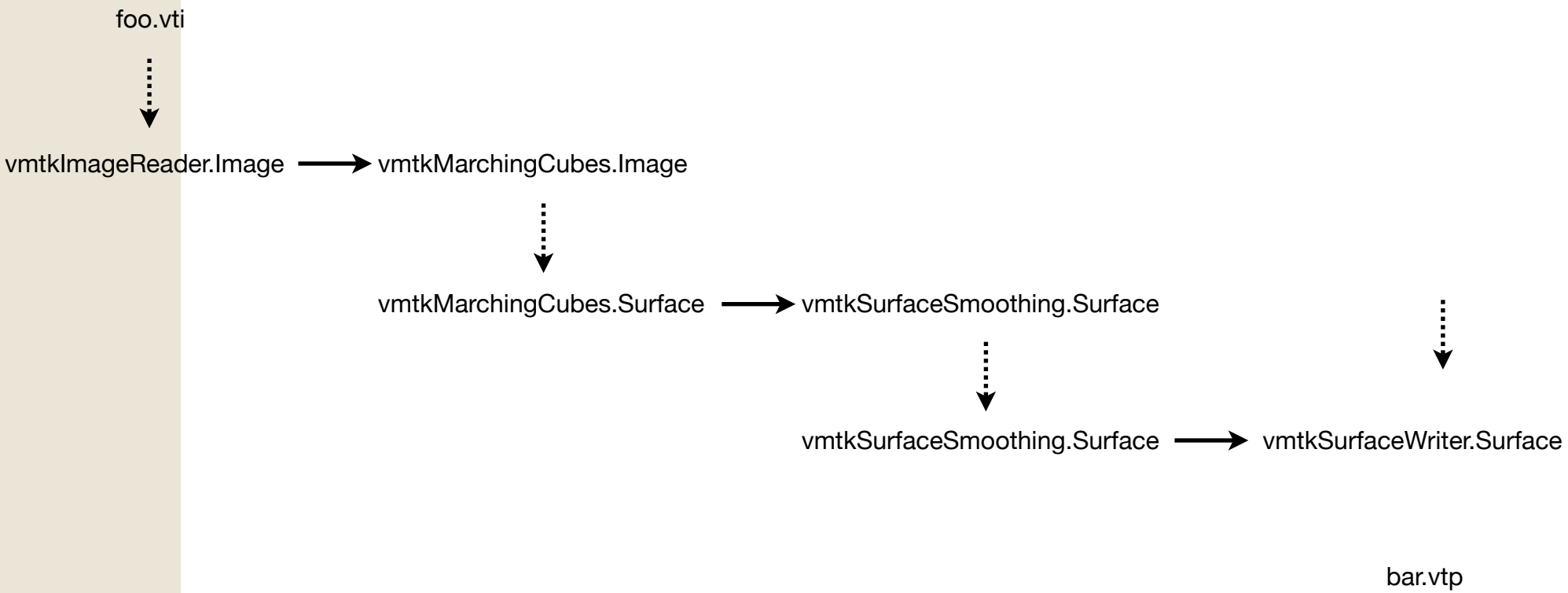
PypeS makes life easier for both the user and the coder

```
vmtkimagereader -ifile foo.vti --pipe vmtkmarchingcubes -l 200 --pipe
vmtksurfacewriter -ofile bar.vtp

vmtkimagereader -ifile foo.vti --pipe vmtkmarchingcubes -l 200 --pipe
vmtksurfacesmoothing -iterations 30 -passband 0.1 --pipe
vmtksurfacewriter -ofile bar.vtp
```

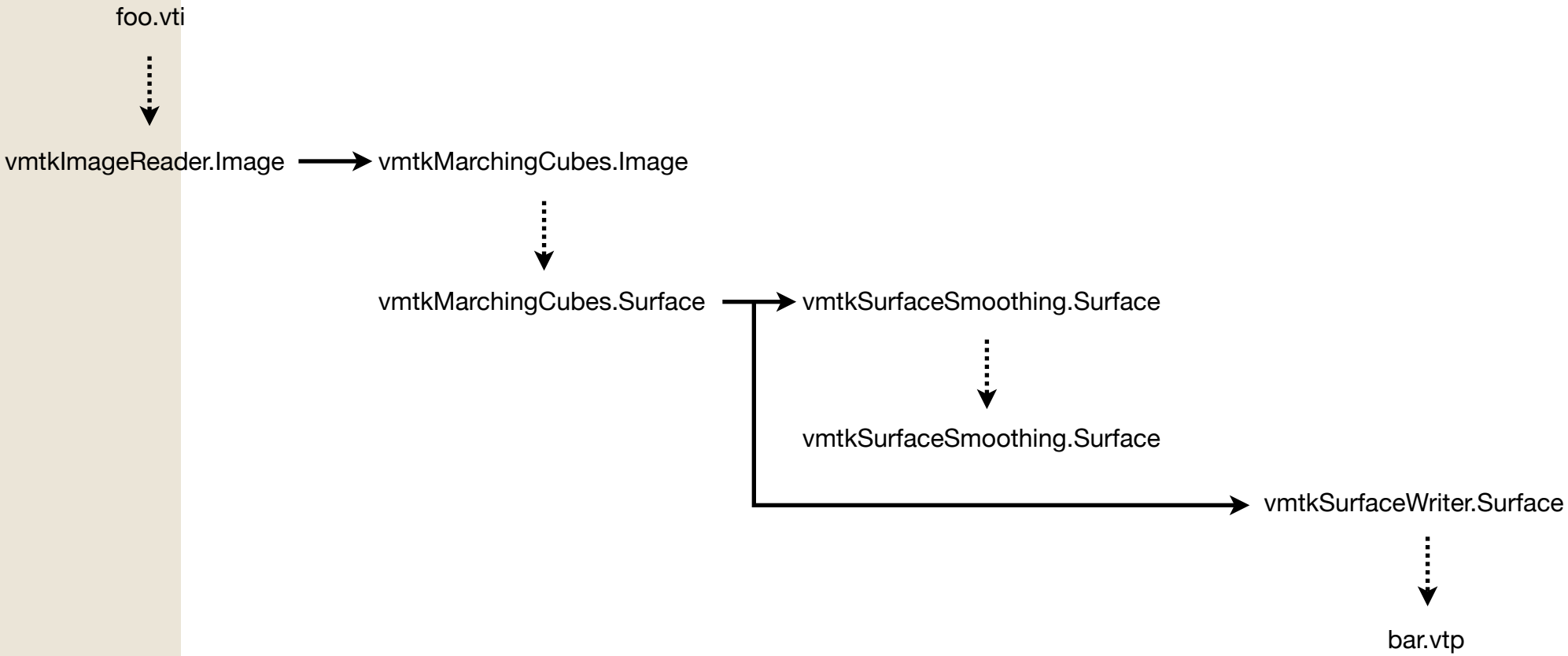
Pypes example

```
vmtkimagereader -ifile foo.vti --pipe
vmtkmarchingcubes -l 200 --pipe
vmtksurfacesmoothing -iterations 30 -passband 0.1 --pipe
vmtksurfacewriter -ofile bar.vtp
```



Pypes example

```
vmtkimagereader -ifile foo.vti --pipe
vmtkmarchingcubes -l 200 --pipe
vmtksurfacesmoothing -iterations 30 -passband 0.1 --pipe
vmtksurfacewriter -i @vmtkmarchingcubes.o -ofile bar.vtp
```



Modeling workflow

- Image segmentation
- Centerline computation
- Geometry analysis
- Model editing
- Mesh generation
- CFD
- Post-processing and data analysis

Image segmentation

Isosurface extraction is (most of the times) not a robust way of defining a surface representing an anatomical structure

Level corresponding to the lumen boundary can depend on:

- scan protocol
- patient's characteristics
- anatomical features
- contrast agent wash-out timing

and can vary within the same scan

Need of more robust tool

Image segmentation

Deformable models: evolution based on image features, rather than intensity alone

Deformable surface

$$S(t) : (R^2 \times R^+) \rightarrow R^3$$

Level sets: evolution of embedding function

$$\phi(\mathbf{x}, t) : (R^3 \times R^+) \rightarrow R$$

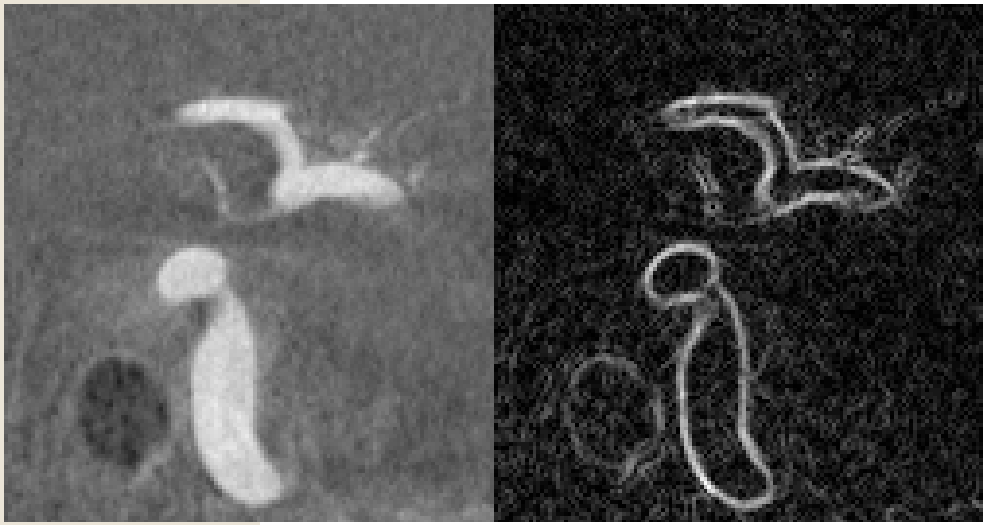
$$S(t) = \{\mathbf{x} \mid \phi(\mathbf{x}, t) = k\}$$

Advantages of level sets:

- large deformations
- topological flexibility
- stable numerical schemes

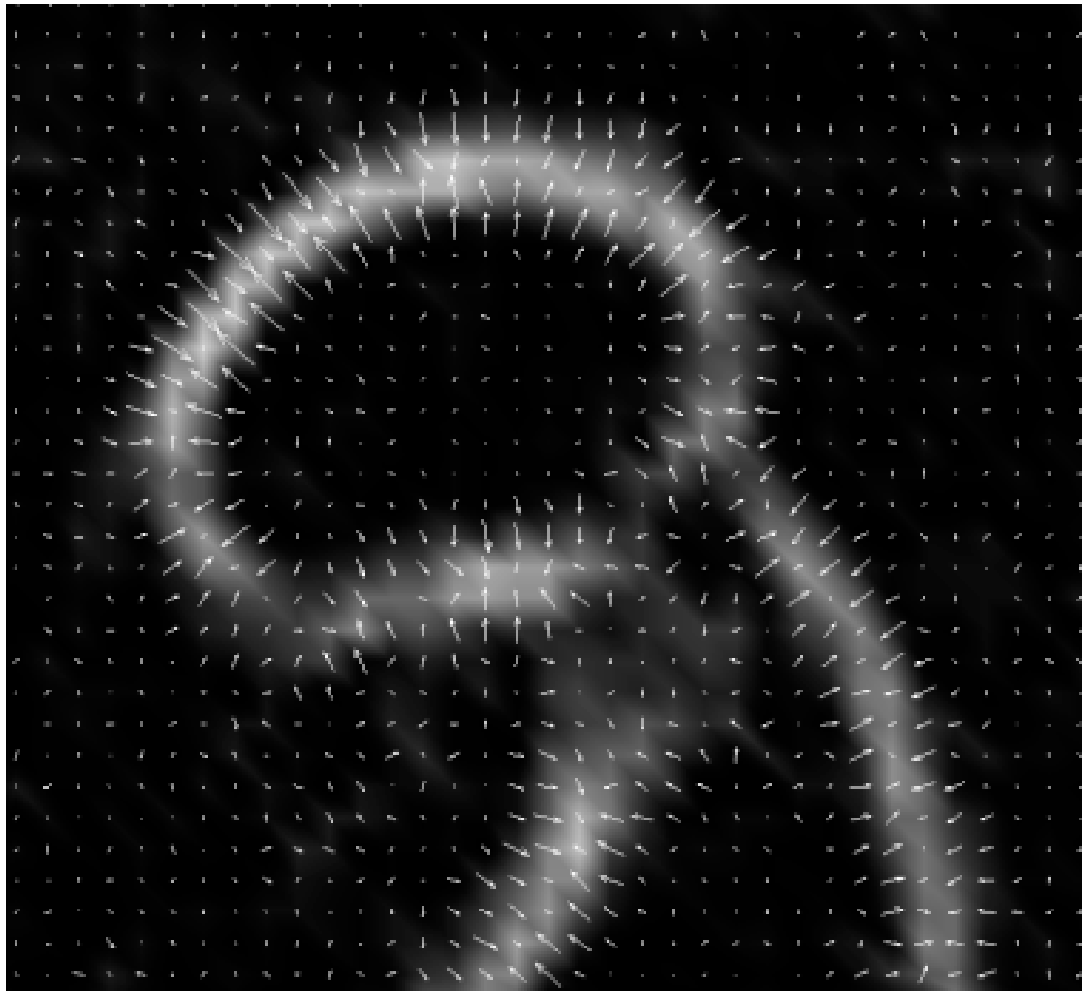
Image segmentation

Image gradient-driven level sets



I

$|\nabla I|$

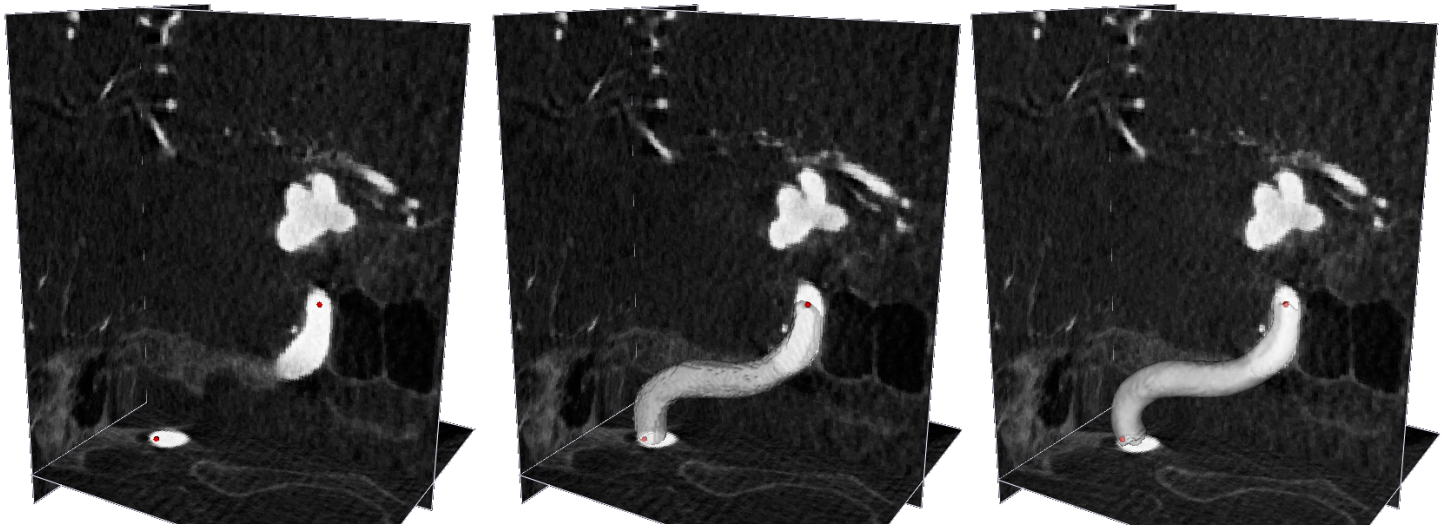


$\nabla|\nabla I|$

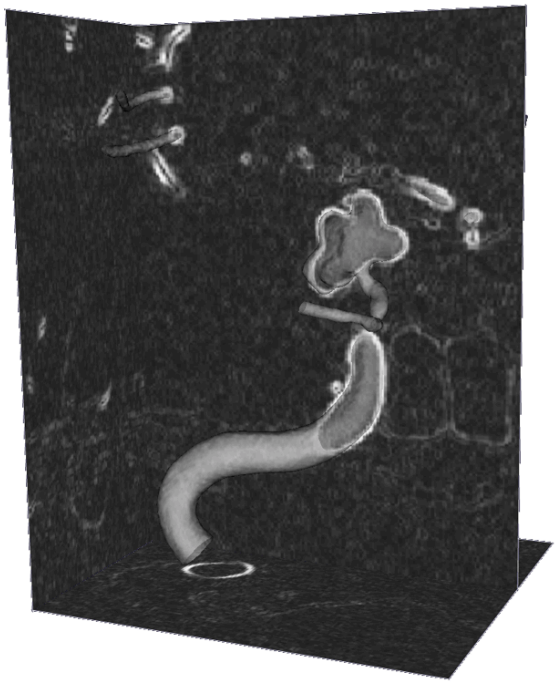
Image segmentation

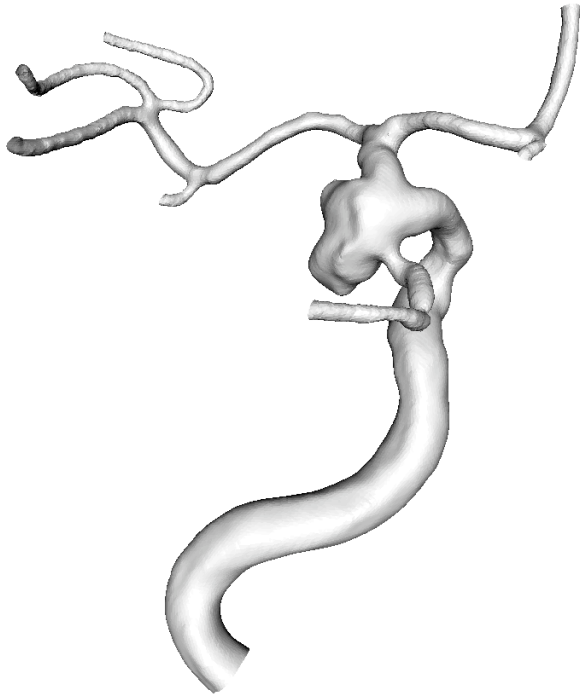
Initialization

Evolution



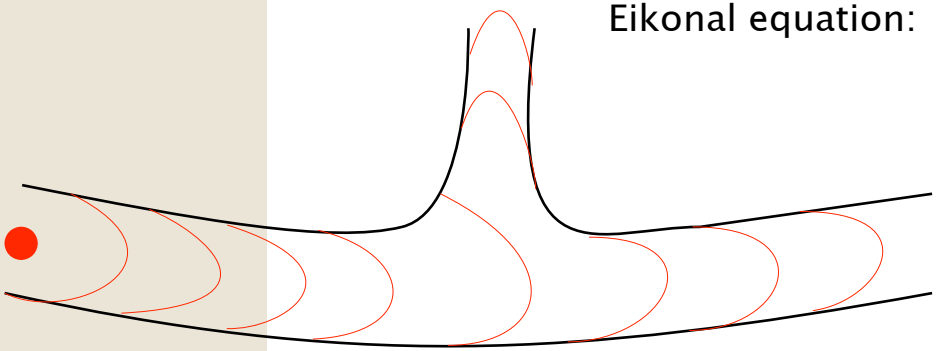
Initialization should bring the surface as close as possible to the features of interest



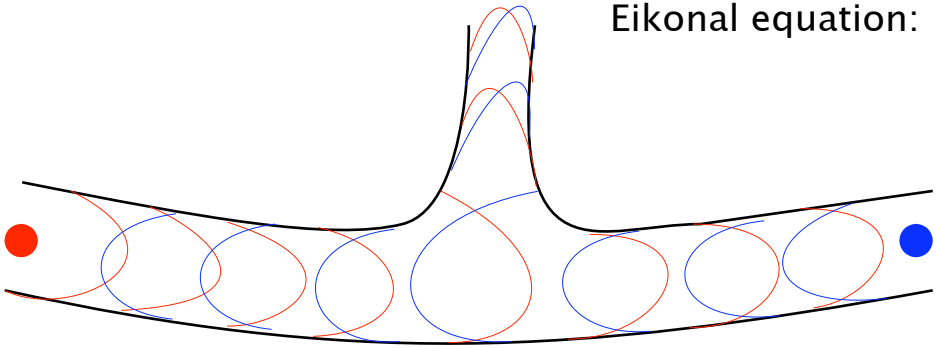


Initialization: colliding fronts

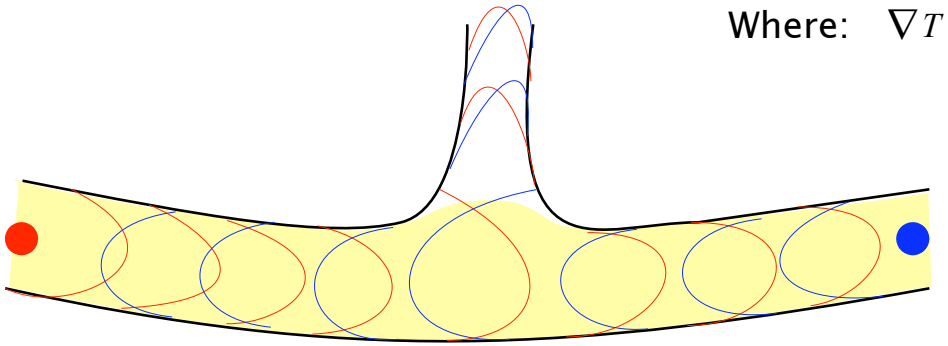
Eikonal equation: $|\nabla T_1| = \frac{1}{1+I}$



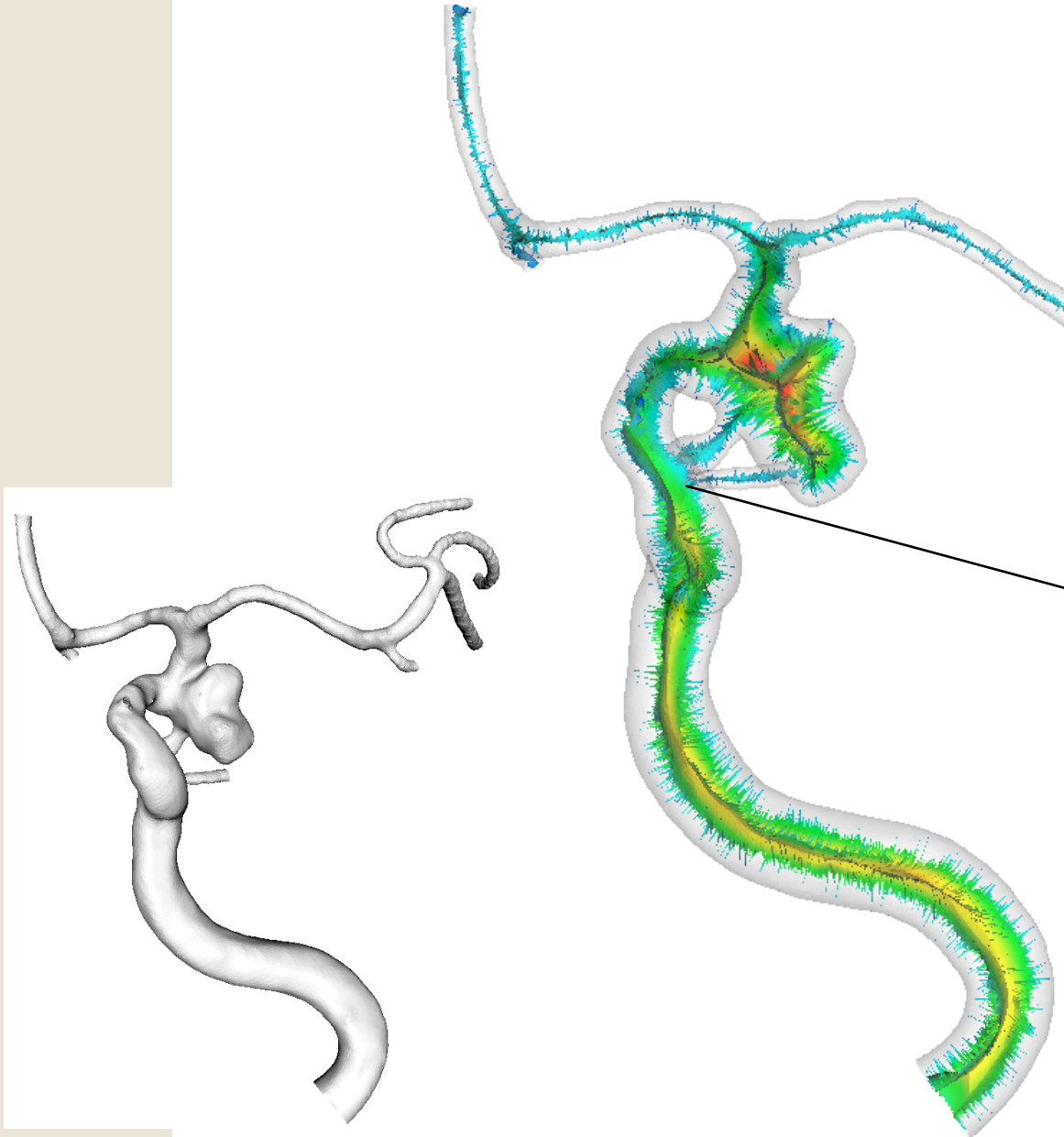
Eikonal equation: $|\nabla T_2| = \frac{1}{1+I}$



Where: $\nabla T_1 \cdot \nabla T_2 < 0$



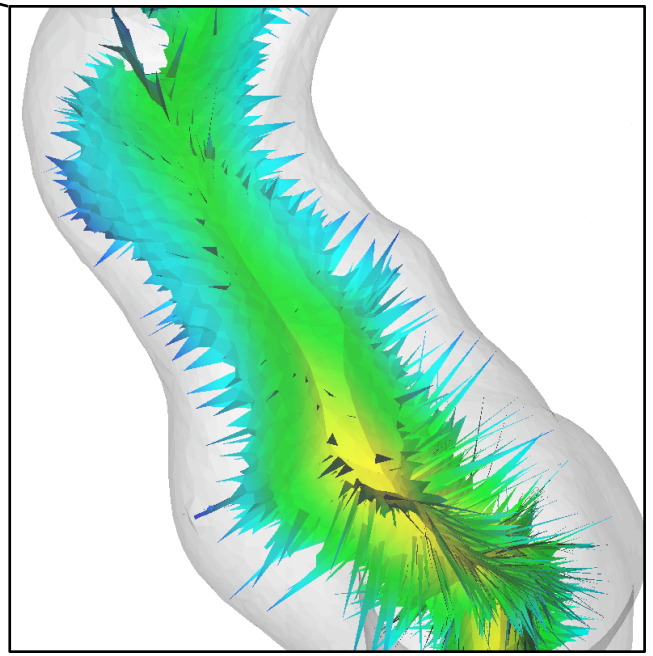
Geometric analysis



Voronoi diagram:

medial axis superset

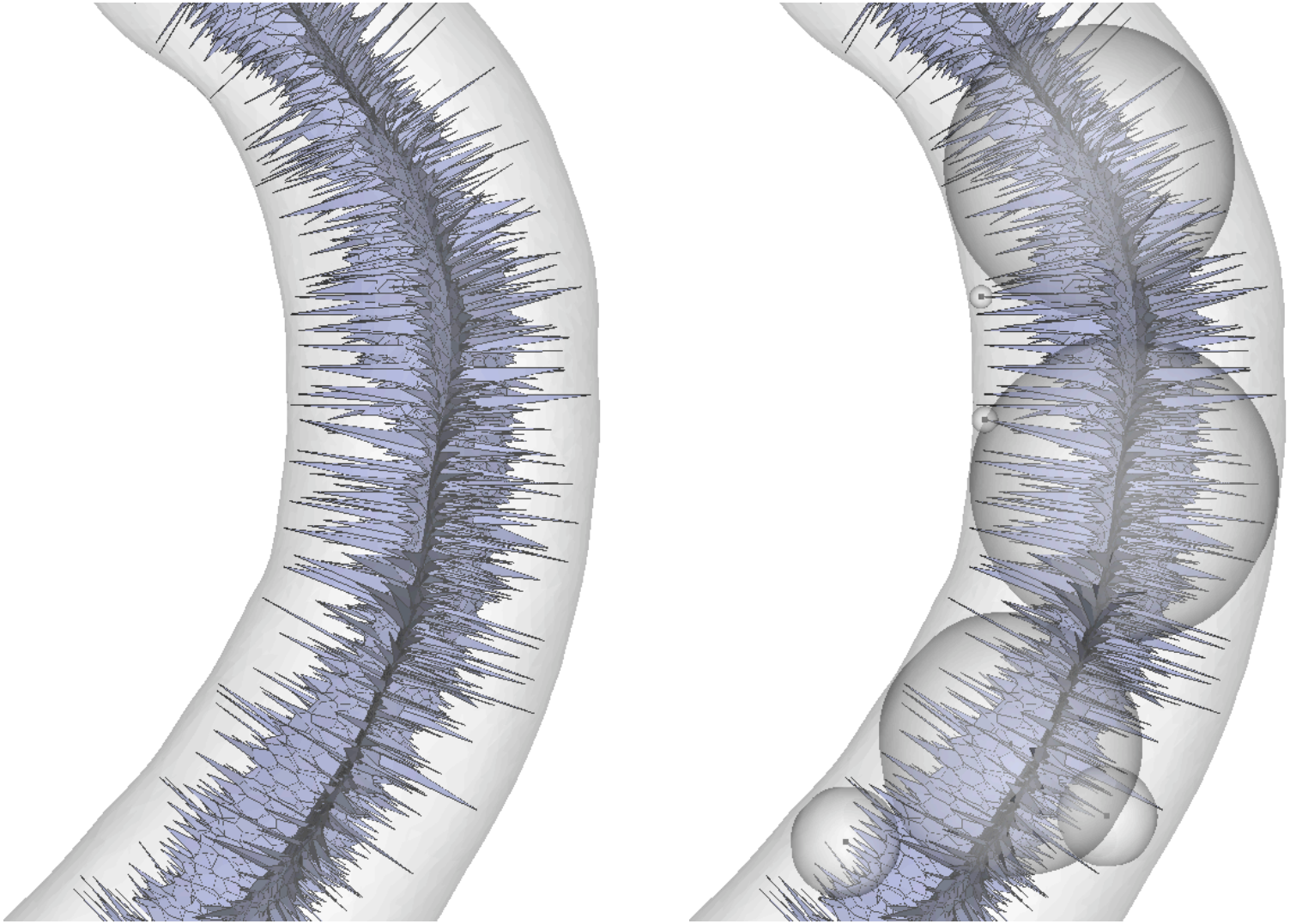
hosts maximal inscribed spheres



Voronoi diagram

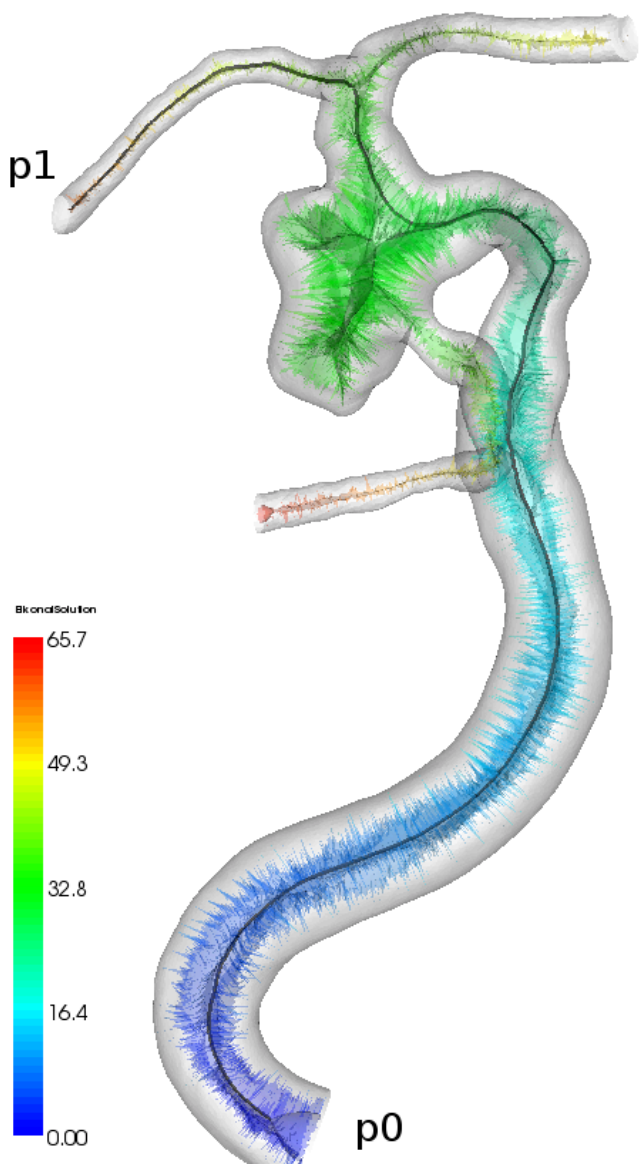
Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University

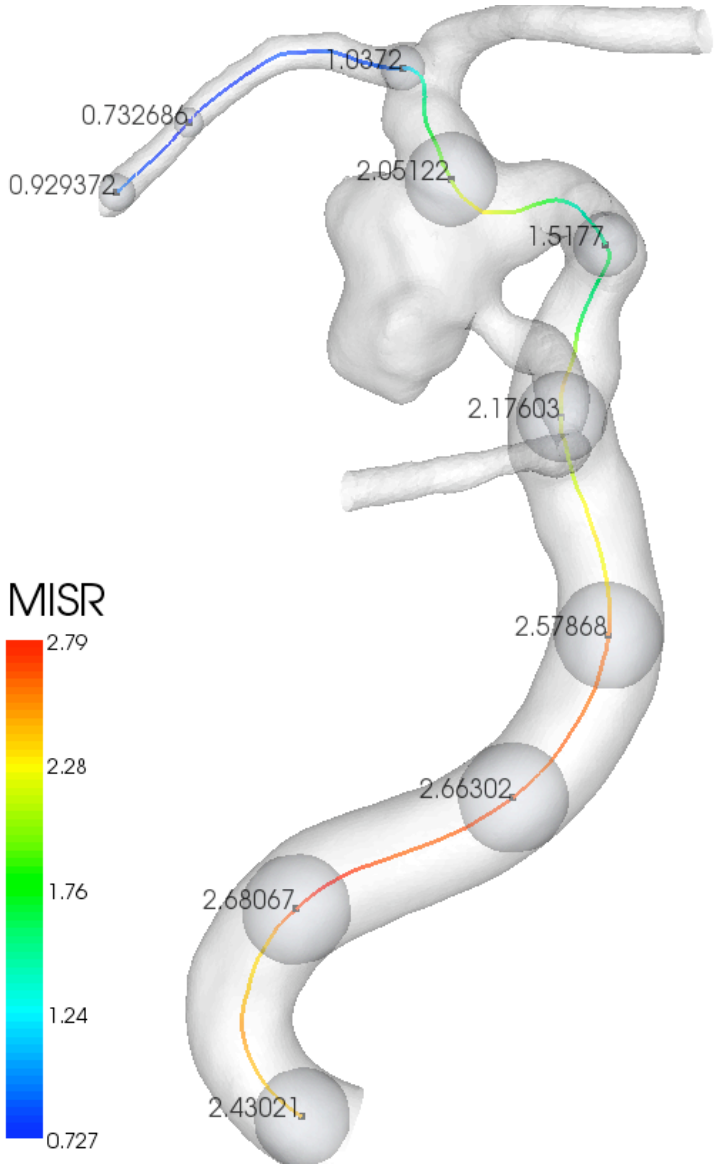
Every point: center and radius of a maximal inscribed sphere



Centerline computation

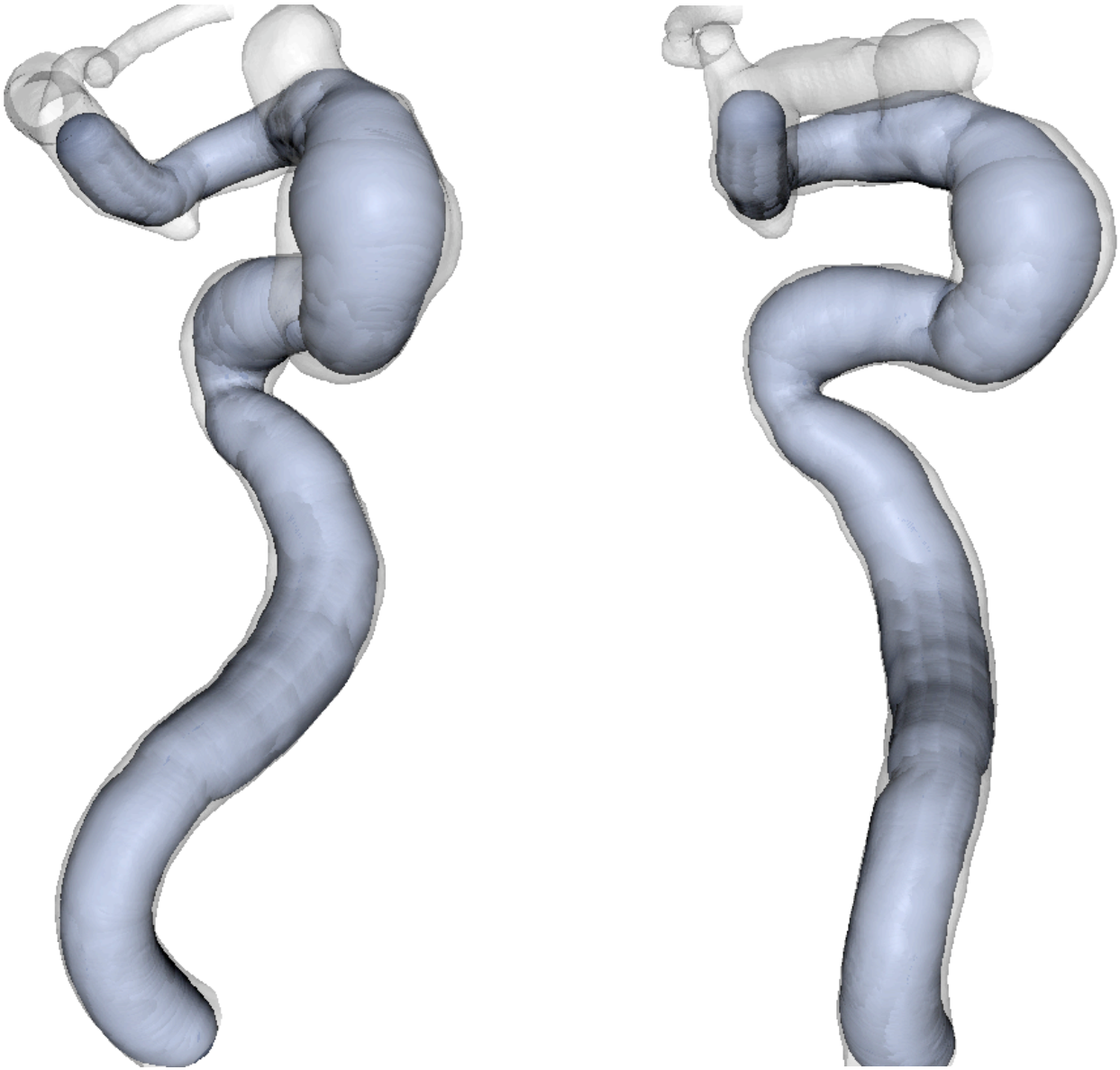
Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University





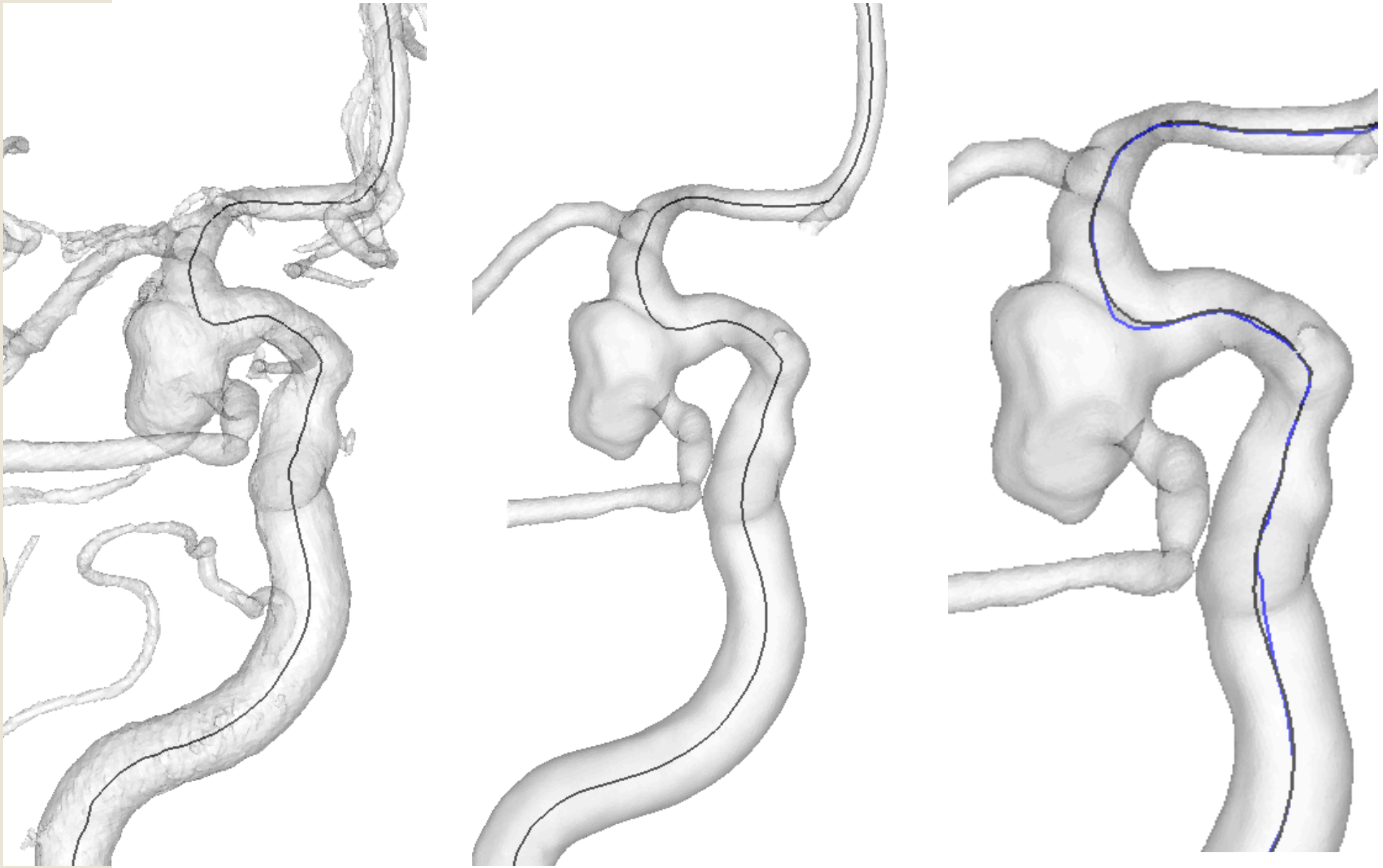
From centerlines to tubes

Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University



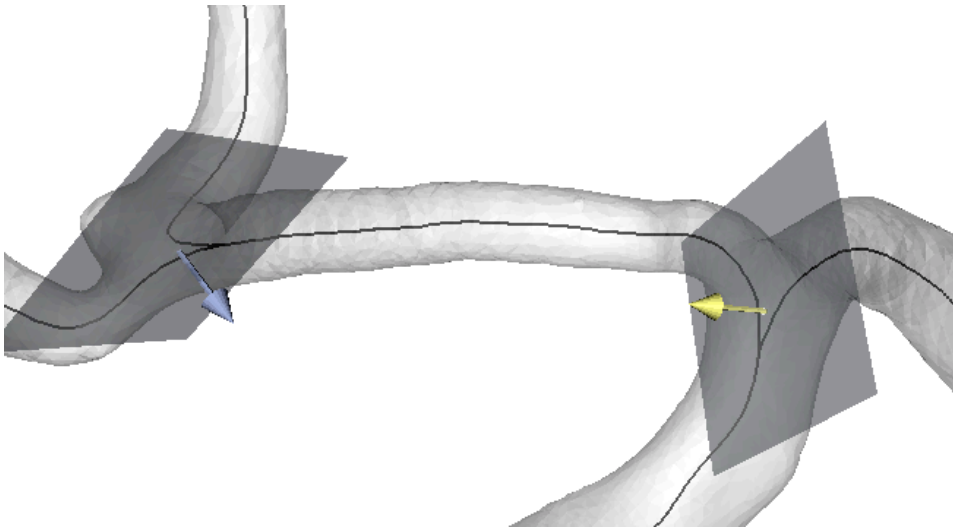
Centerline computation on isosurface

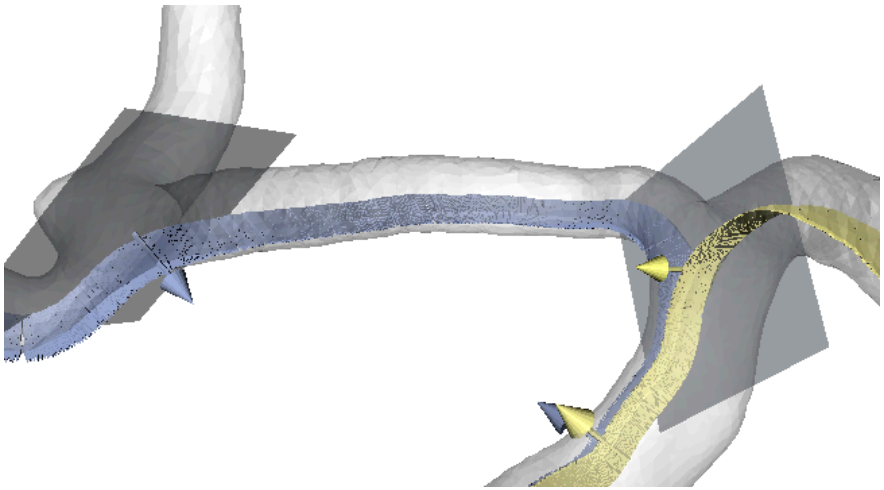
Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University

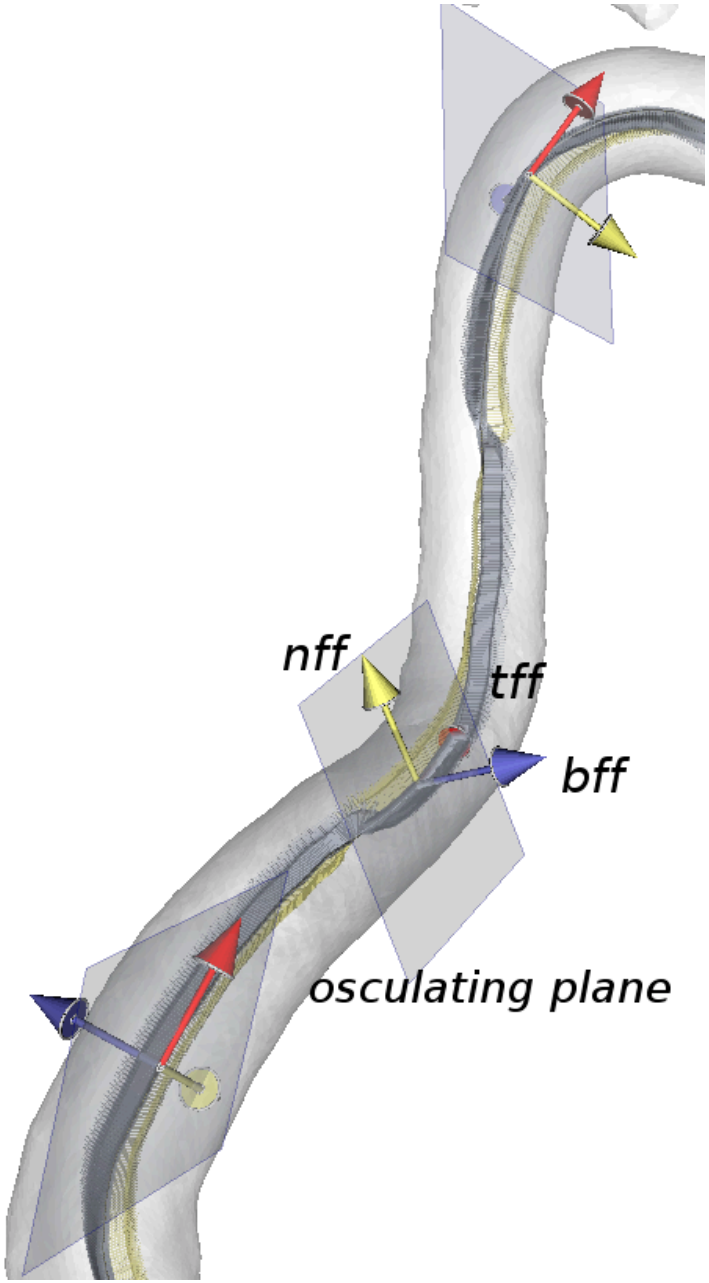


Geometry analysis - framed lines

Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University

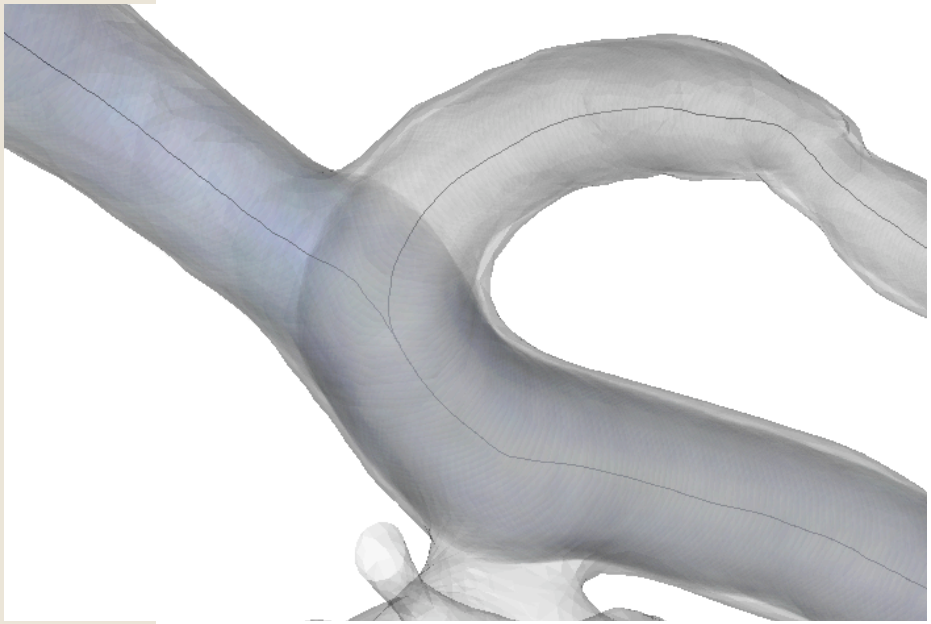


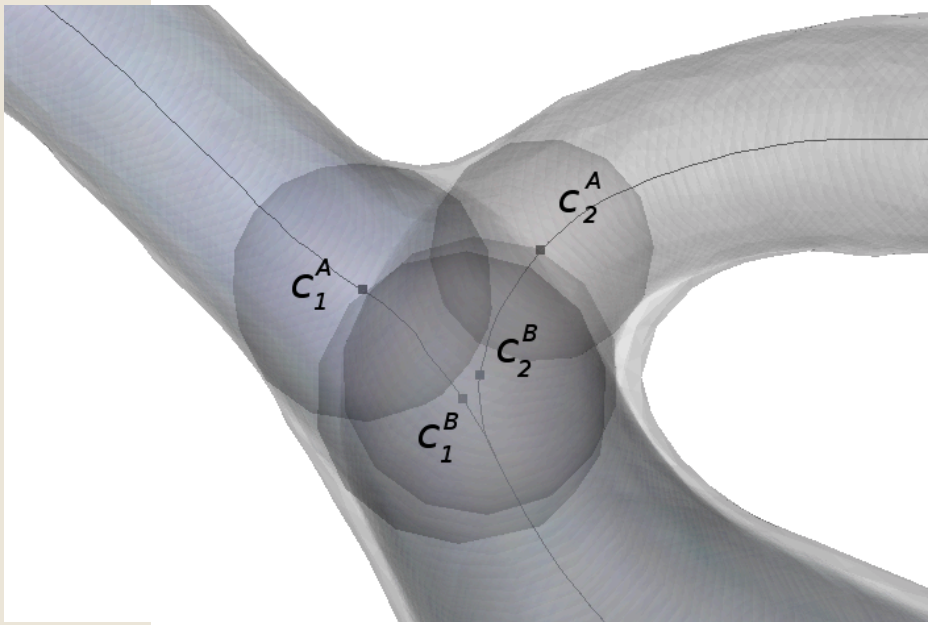


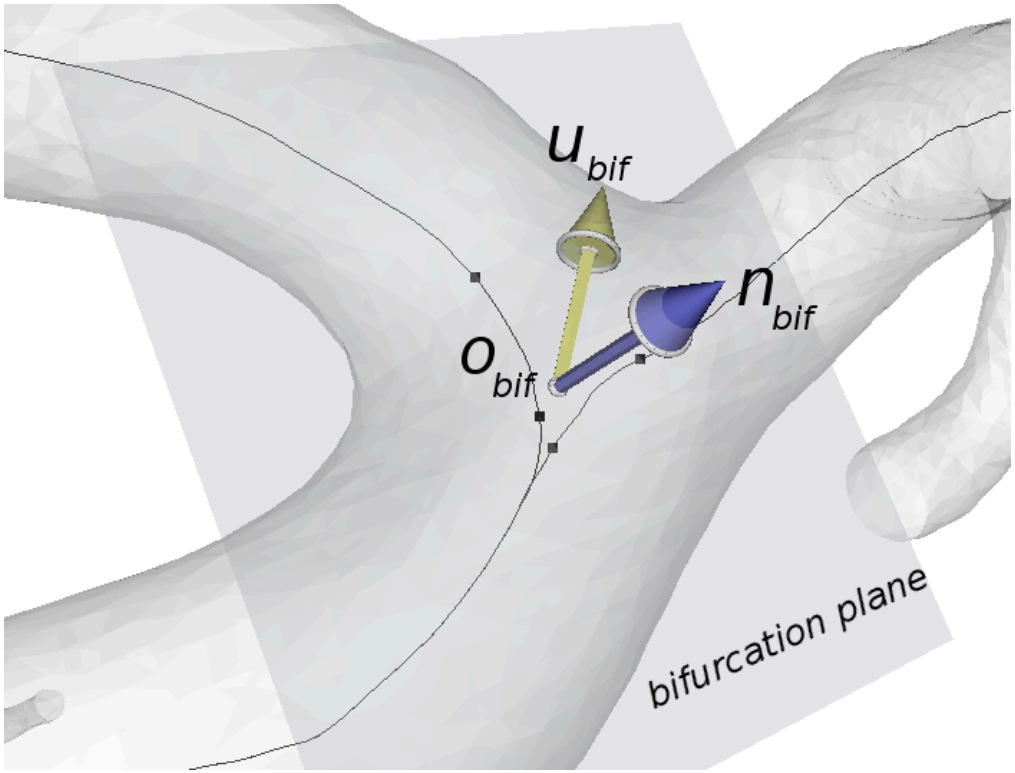


Geometry analysis - bifurcation geometry

Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University

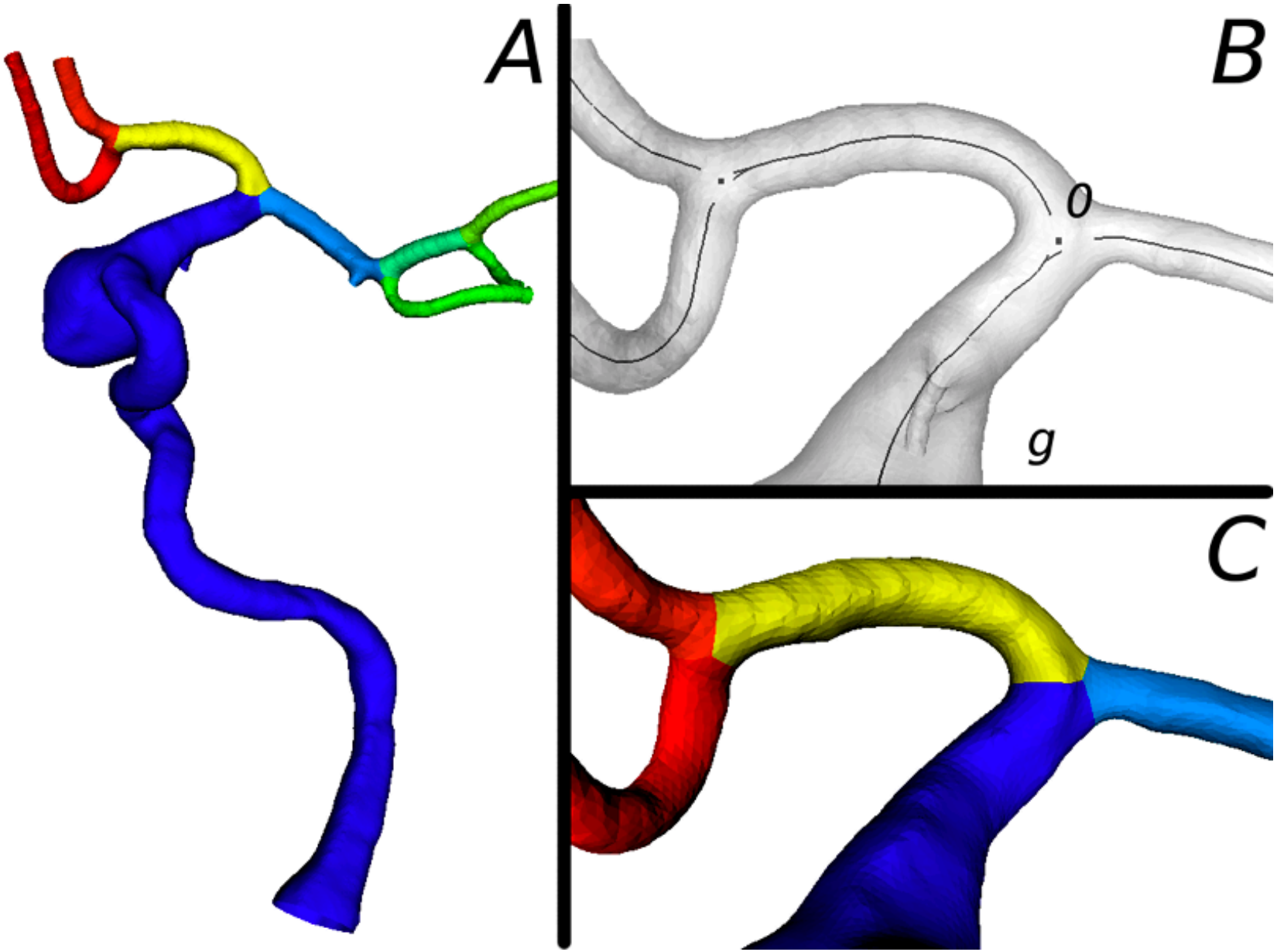




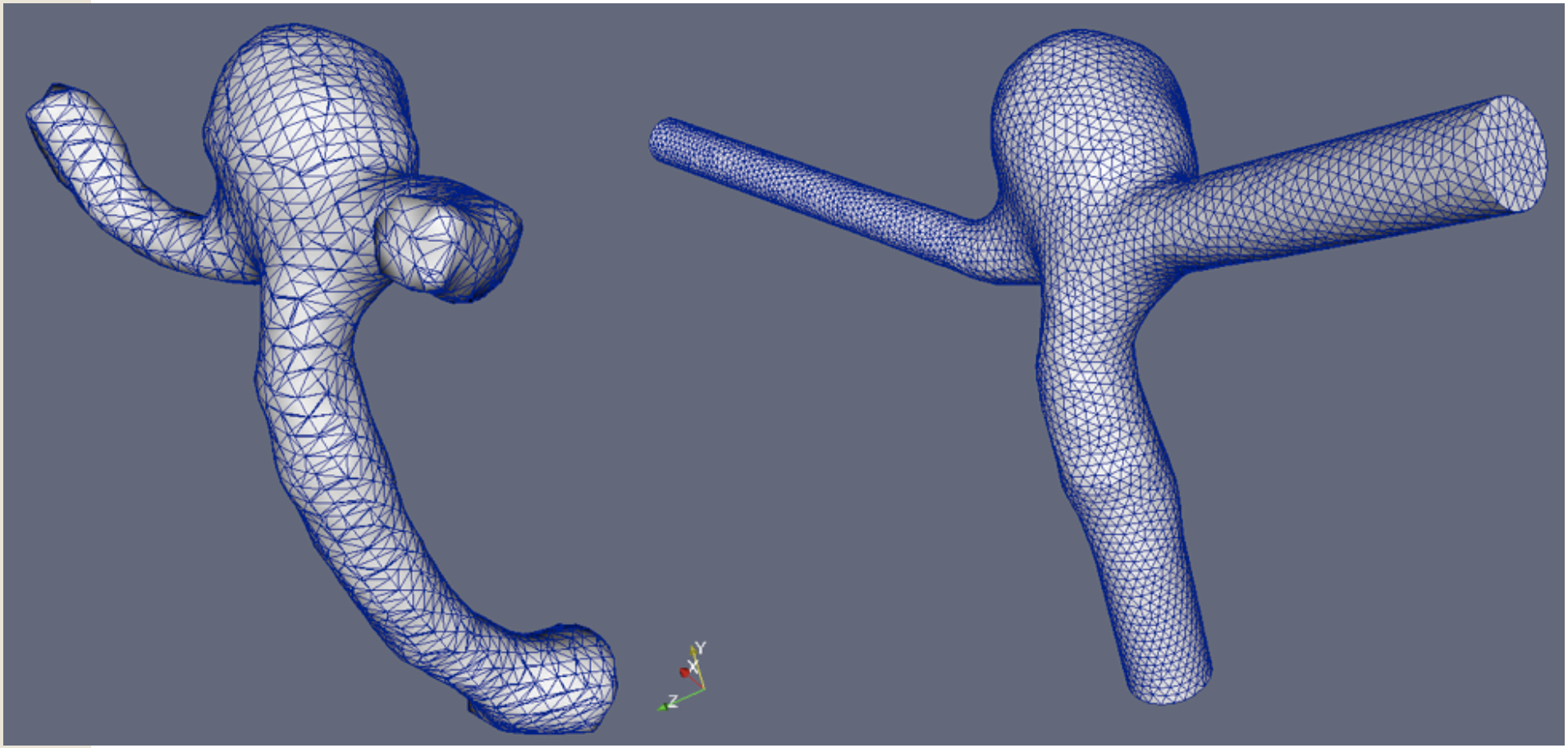


Geometry analysis - bifurcation splitting

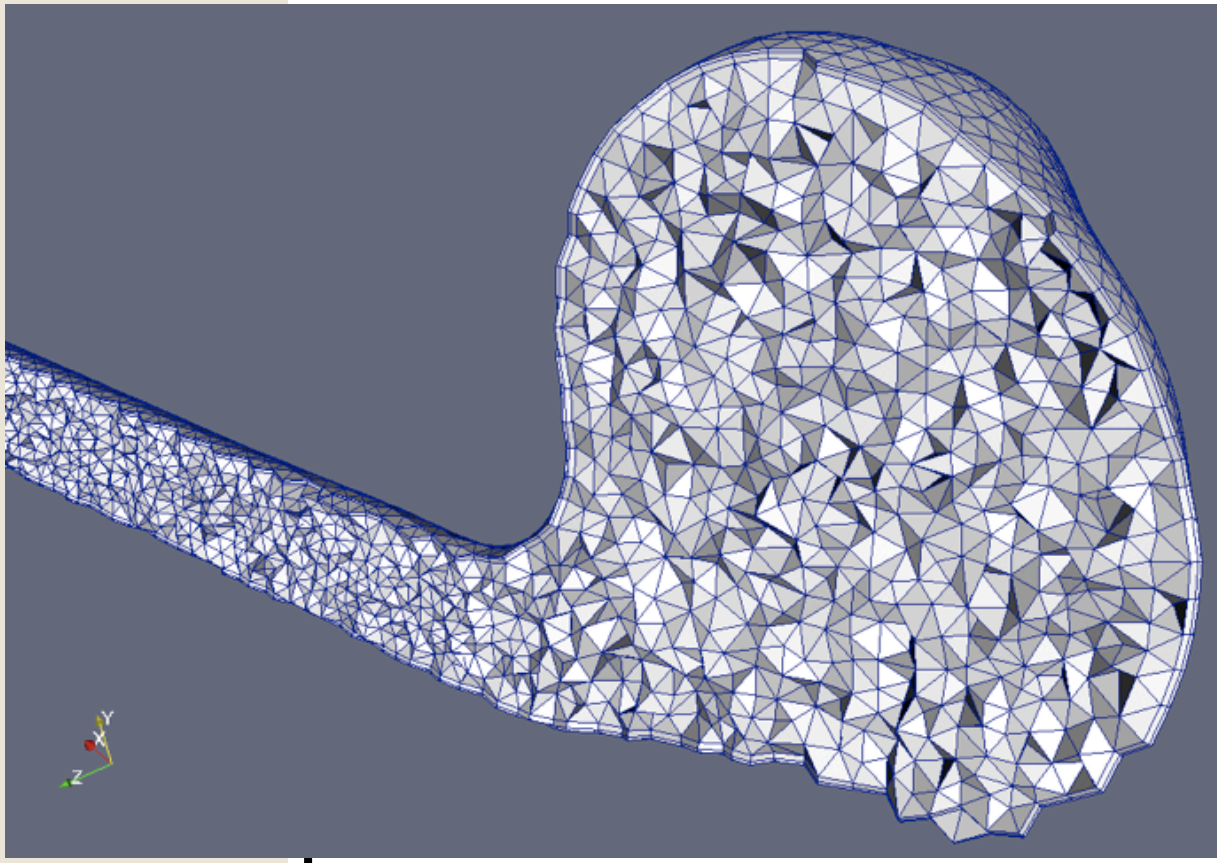
Images courtesy of Marina Piccinelli, Mario Negri Institute and Emory University

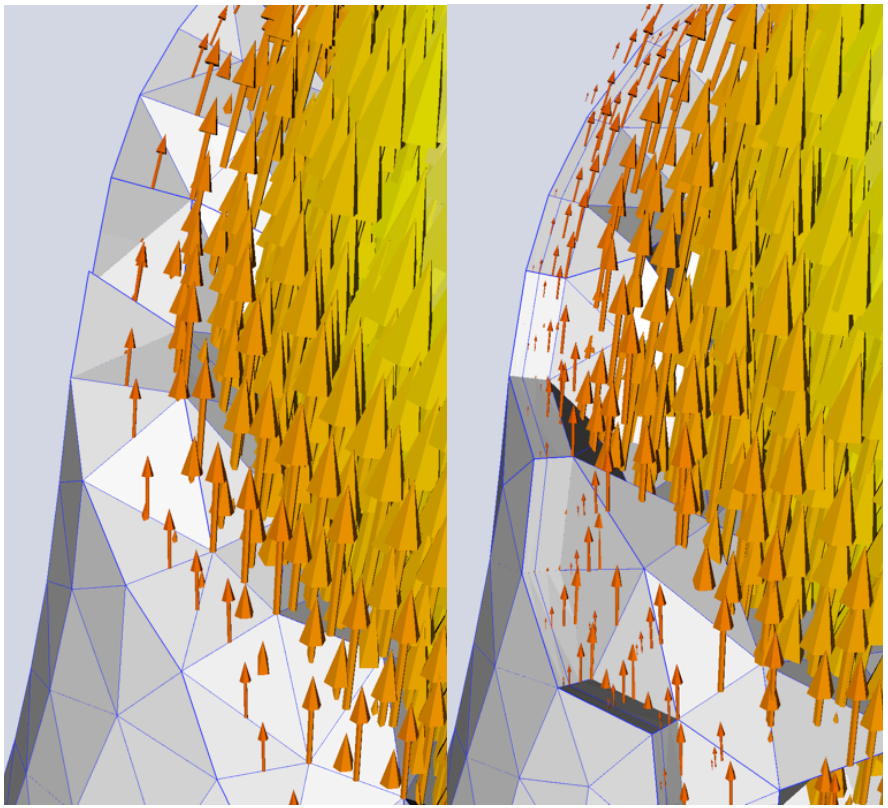


Surface remeshing



Mesh generation





Prismatic boundary layers
with adaptive thickness

Direct export to solvers
(including boundary indicators)

libMesh

FENICS

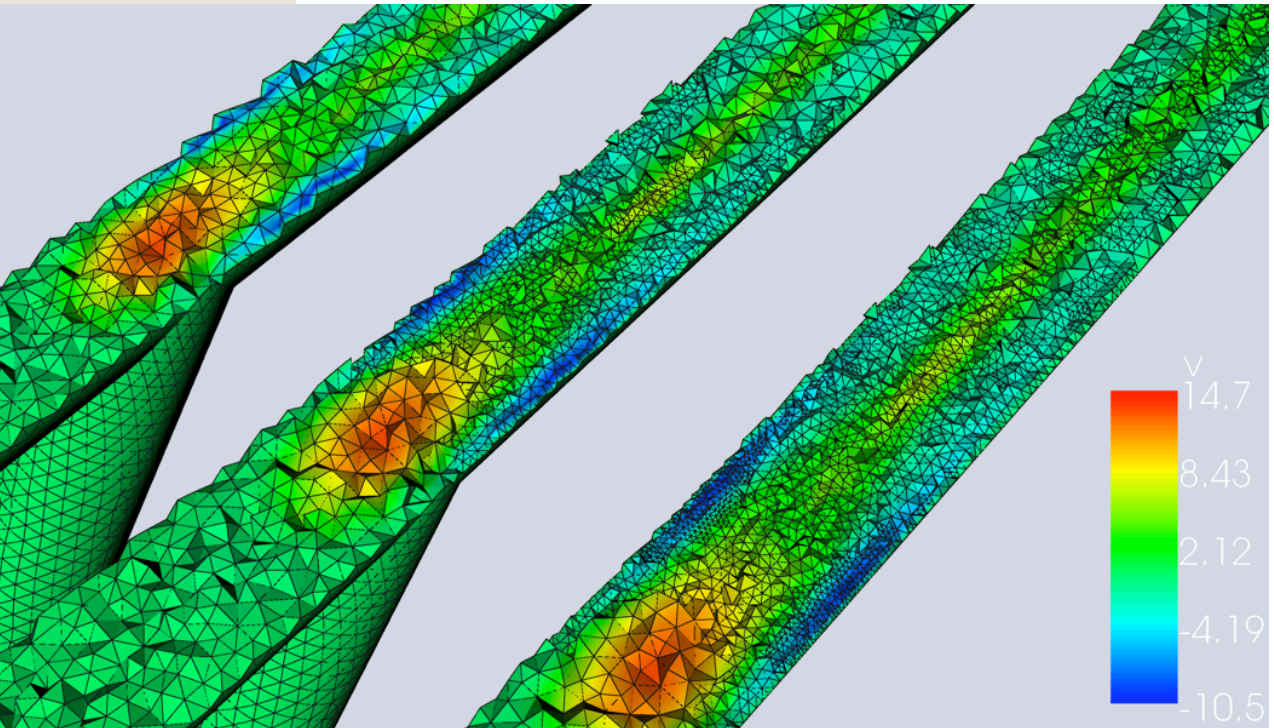
+ Fluent (work in progress)
+ ...

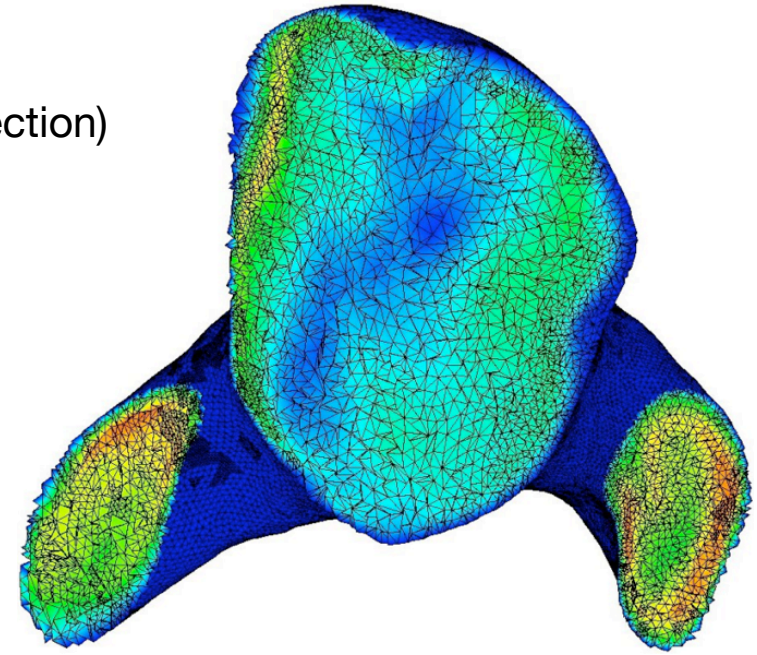
Solver

Incompressible, Newtonian, rigid walls

- simple solution strategy (operator splitting - 2nd velocity correction)
- adaptive non-conformal h-refinement and coarsening
- 1st and 2nd order tetrahedra, prisms and hexahedra
- parallelization

Approach for high-throughput simulations:
simple scheme + aggressive adaptivity





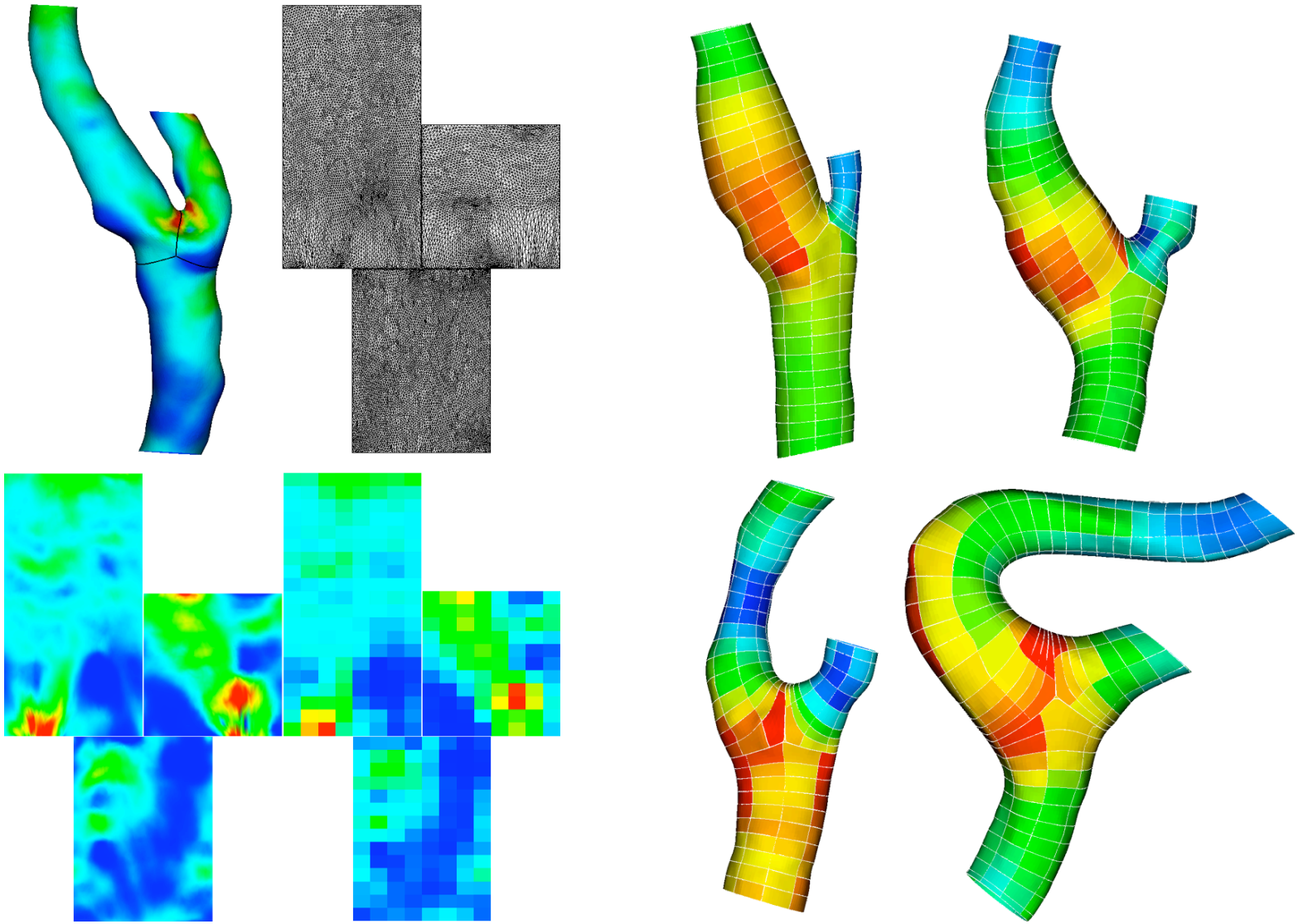
The solver will be presented by Lorenzo Botti (PhD student at Mario Negri Institute) in ESB 2008.

It will be released as open source by the end of the year.

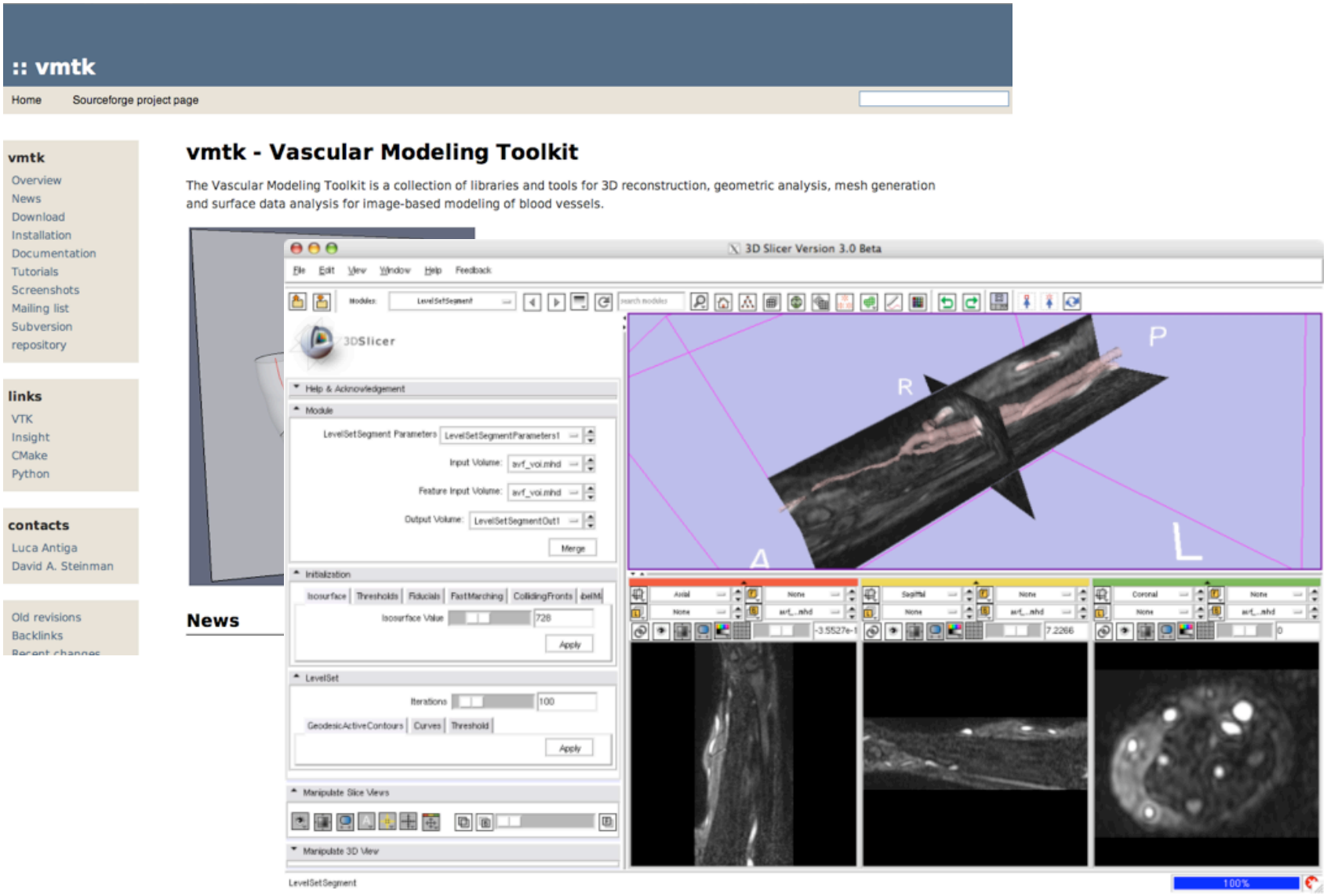
Post-processing

Longitudinal abscissa-based stretched harmonic mapping

Circumferential angle-based mapping



Future developments: vmtk - Slicer integration



National Alliance for Medical Image Computing
A National Center for Biomedical Computing
Funded under the NIH Roadmap Initiative

http://www.slicer.org