iso2mesh: an image-based mesh generation toolbox

Automatic surface and volumetric mesh generation from 3D binary or gray-scale images

`http://iso2mesh.sourceforge.net`

Massachusetts General Hospital, Harvard Medical School

Qianqian Fang and David Boas

Martinos Center for Biological Imaging

Motivation: To generate triangular surface or tetrahedral volumetric meshes is typically an important preprocessing step in many popular numerical modelling methods, such as the boundary-element (BE) or finite element (FE) methods. One can find abundant meshing tools, freely or commercially available, to process simple geometries. However, there is an increasing need to produce meshes from complex geometries represented by 3D volumetric images, for example, the tissue structures in a segmented or raw medical MRI or CT images. This need becomes critical for many emerging multi-modality medical imaging approaches where the structural images fuse in the functional imaging data analysis. Iso2mesh is a free mesh generation toolbox and is capable of producing high quality surface and volumetric meshes directly from 3D binary or grayscale images. This toolbox was written in Matlab language and is compatible with Matlab(7.0+) and GNU Octave (3.0+) on multiple platforms.

highlights of iso2mesh

» **Free**
iso2mesh is free and open-source under the terms of GPL.
» **Designed for simplicity**
many meshing tasks can be done in only a few lines of code
» **Matlab and GNU Octave compatible**
iso2mesh can be used in Matlab or its free clone GNU Octave
» **Cross platform**
iso2mesh can be run on Linux, Windows (32/64bit) and Mac OS X for both PowerPC or Intel processors
» **Modularized**
the toolbox is highly structured and one can cascade the atomic meshing operations to achieve more complicated meshing tasks

» **Versatile**
this tool can be used to mesh binary, gray-scale images or surface patches; this can satisfy most of the needs for mesh generation
» **Fast**
all Matlab scripts were optimized by vectorization and profiling; many of the underlying tools were compiled with optimization flags
» **Meshing multiple regions**
it is able to mesh structures containing multiple regions with fully automatic procedures
» **Multiple mesh format support**
support reading/writing of several mesh format

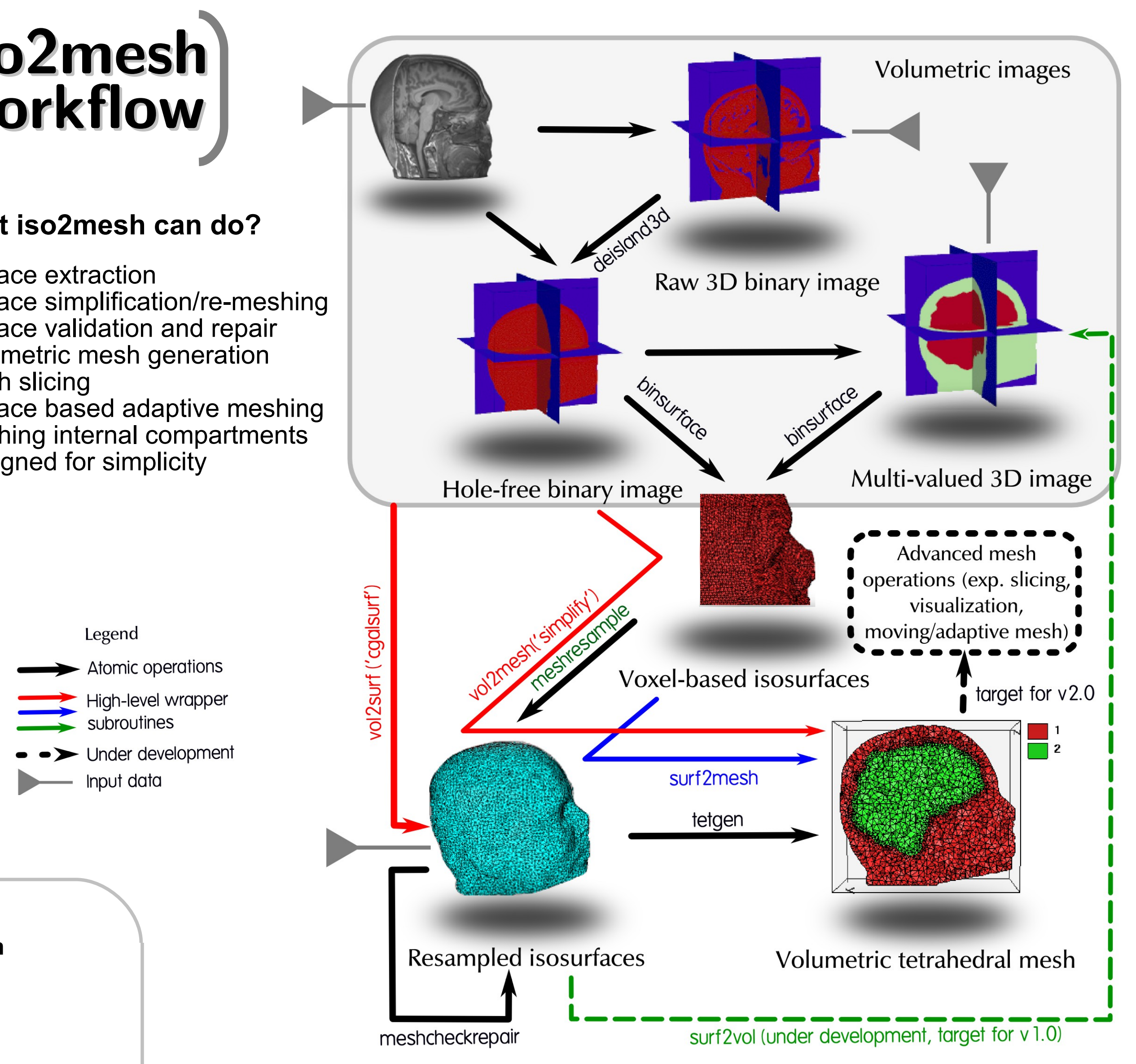
iso2mesh was built on top of several free software external modules. These tools including CGAL library (mesh simplification and surface mesh extraction tools), tetgen by Hang Si for 3D mesh generation, and JMeshLib by Macro Attene for automatic mesh validation and repairing.

CGAL http://www.cgal.org/
Tetgen http://tetgen.berlios.de/
JMeshLib http://jmeshlib.sf.net

iso2mesh workflow

What iso2mesh can do?

- Surface extraction
- Surface simplification/re-meshing
- Surface validation and repair
- Volumetric mesh generation
- Mesh slicing
- Surface based adaptive meshing
- Meshing internal compartments
- Designed for simplicity



iso2mesh cheat-sheet

» **Streamlined Mesh Generation (wrapper)**
```
[no,el,regions,holes]=vol2surf(img,ix,iy,iz,opt,dofix,method,isovalues)
[no,el,regions,holes]=v2s(img,isovalues,opt,method)
```
Converting a 3D volumetric (binary or gray-scale) image to surfaces at the specified isovalues. The variable 'method' can be either 'simplify' or 'cgalsurf'. The surface maximum element diameter can be specified by opt.

```
[node,elem,bound]=surf2mesh(v,f,p0,p1,keepratio,maxvol)
[node,elem,face]=s2m(v,f,keepratio,maxvol)
```
Generating quality tetrahedral mesh from isosurfaces. The maximum element volume is specified by 'maxvol'. If keepratio is less than 1, iso2mesh will downsample the surface before the volumetric mesh generation.

```
[node,elem,bound]=vol2mesh(img,ix,iy,iz,opt,maxvol,dofix,method,isovalues)
[node,elem,face]=v2m(img,isovalues,opt,maxvol,method)
```
Converting a binary, segmented or gray-scale volume to tetrahedral mesh by calling vol2surf and surf2mesh sequentially.

```
img=surf2vol(node,face,xi,yi,zi)
img=s2v(node,face,xi,yi,zi)
```
Convert a surface mesh to a 3D binary image with all voxels enclosed by the surface as 1 and otherwise 0 (under construction).

» **Mesh simplification**
```
[node,elem]=meshresample(v,f,elemnum)
```
meshresample: resample mesh using CGAL mesh simplification utility

```
seg=bbxflatsegment(node,loop)
```
bbxflatsegment: decompose edge loops into flat segments alone x/y/z planes of the bounding box

» **Mesh repair**
```
[node,elem]=meshcheckrepair(node,elem,opt)
```
check and repair a surface mesh

```
elem=removedupelem(elem)
newnode=removedupnode(node,elem)
```
remove evenly duplicated elements (cancel duplicated elements) or nodes

```
[no,el]=removeisolatednode(node,elem)
```
remove isolated nodes: that are not included in any element

```
[no,el]=removeisolatesurf(node,face,maxdiameter)
```
remove disjointed surface components using maxdiameter as threshold

```
elem=delendelem(elem,mask)
```
delete elements whose nodes are all edge nodes

```
f=surfaceclean(f,v)
```
remove surface patches that are located inside the bounding box faces

```
nodenew=smoothsurf(node,mask,conn,iter,method)
```
smooth a surface mesh by 3 smoothing algorithms: method= 'laplacian','laplacianhc' or 'lowpass'

» **Mesh decomposition and query**
```
facecell=finddisconnsurf(face)
```
subroutine to extract disjointed surfaces from a cluster of surfaces

```
f=maxsurf(facecell)
```
return a surface which contains the most elements in a collection of surfaces (stored in a cell array)

```
mask=flatsegment(node,edge)
```
decompose edge loops into flat segments alone the facets of the bounding box

```
p=internalpoint(v,aloop)
```
internalpoint: imperical function to find an internal point of a planar polygon

```
[conn,connnum,count]=meshconn(elem,nn);
[conn,connnum,count]=neighborelem(elem,nn);
```
create node/element neighbor list for each node in a surface/volumetric mesh

```
edges=surfedge(f)
```
find the edge of an open surface

```
loops=extractloops(edges)
```
extract individual loops from an edge table of a loop collection

```
newedge=orderloopedge(loops)
```
order the node list of a simple loop based on connection sequence

```
[cutpos,cutvalue,facedata]=qmeshcut(elem,node,value,plane)
```
Slicing a tetrahedral/surface mesh at any plane and get the cross-sectional cut.

» **Binary image pre-processing (hole-filling and island removing)**
```
islands=bwislands(img)
```
decompose a 2D binary image into isolated regions
and return the indices of the region pixels as a cell array

```
cleanimg=deislands2d(img,sizelim)
cleanimg=deislands3d(img,sizelim)
```
remove all small islands smaller than specified size from a binary image

```
edgeimg=imedge3d(img)
```
exract all voxels where are located on the boundary of a segmented volume

» **File IO**
```
[node,elem]=readoff(fname)
saveoff(v,f,fname)
```
Reading/writing Geomview Object File Format

```
[node,elem]=readsmf(fname)
savesmf(v,f,fname)
```
Reading/writing simple model format

```
[node,elem]=readasc(fname)
```
Reading ASCII mesh format

```
saveinr(vol,fname)
```
Writing a volume array, vol, to INR format

```
[node,elem,bound]=readtetgen(fstub)
```
read tetgen output files

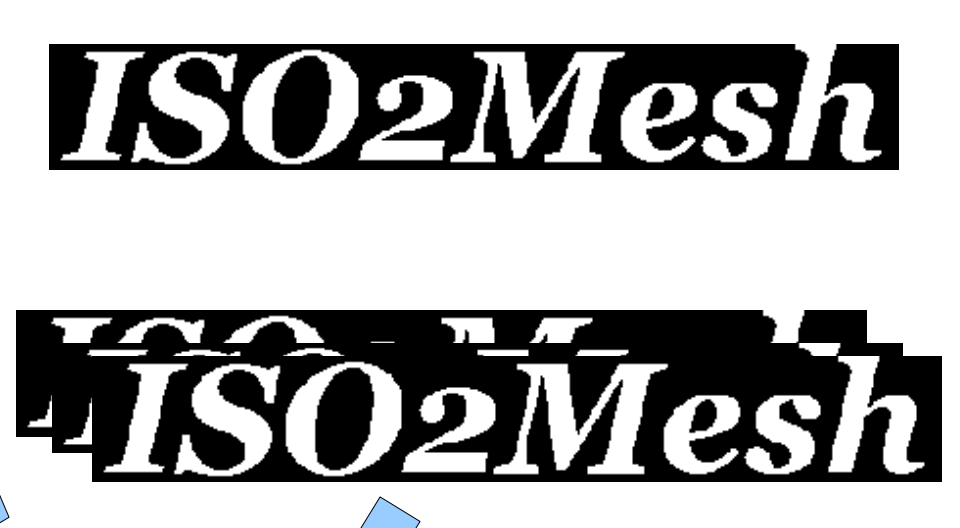
```
savesurfpoly(v,f,p0,p1,fname)
```
create node neighbor list from a mesh

```
fullname=mwpath(fname)
```
get full temp-file name by prepend working-directory and current session name.

```
fullname=mcpath(cmdname)
```
get full executable path by prepending a command directory path

one-line example for beginners

iso2mesh one-liner demonstration

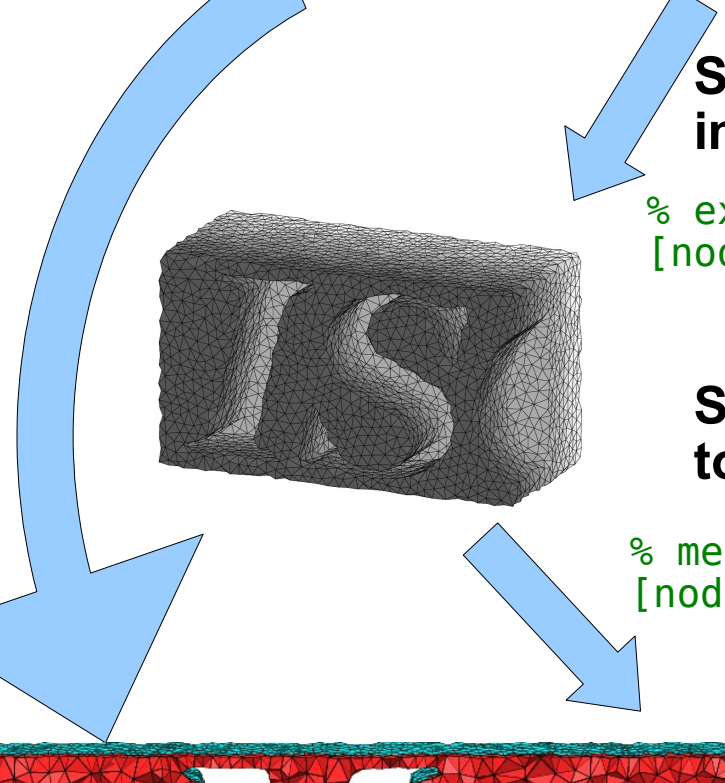


Step 0. (preparation of the binary image)
Create a complex 3D binary object from images
```
img=imread('iso2mesh_bar.tif');

fullimg=repmat(1-img,[1,1,30]);
fullimg(:,:,31:60)=repmat(ones(size(img)),[1,1,30]);
```

Step 1a: convert the binary image to volumetric mesh in 1 line

Binary image · Threshold · Volumetric elem size
```
[node,elem,face]=v2m(fullimg,0.5,3,100);
```
Surface mesh density



Step 1b.A: convert the binary image to triangular surfaces
```
% extract surface at threshold=0.5
[node,face]=v2s(fullimg,0.5,3);
```

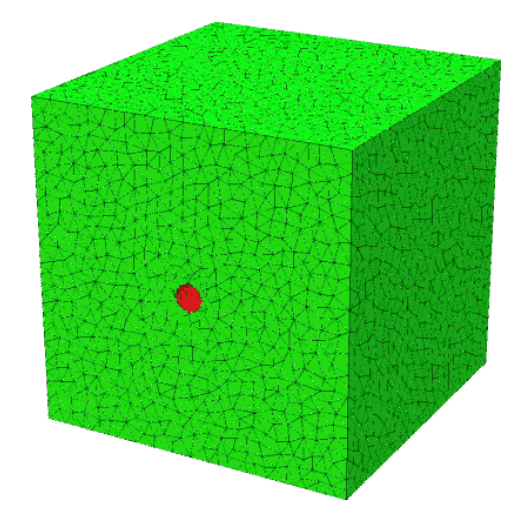
Step 1b.B: convert the surface to tetrahedral mesh
```
% mesh with max elem volume=100
[node,elem,face]=s2m(node,face,1,100);
```

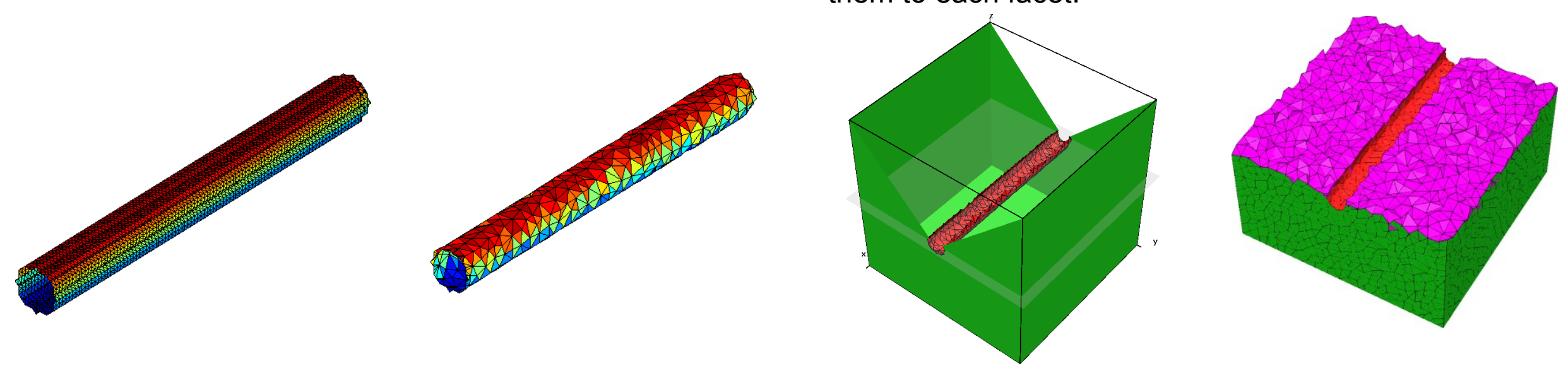
do more with iso2mesh

Mesh generation from open surfaces

Iso2mesh is able to generate a water-tight surface from a open surface assuming that the edges of the open surface locate inside a bounding box. The following example is to demonstrate the workflow for processing this

1. user input raw surface mesh is an open surface; we first identify the nodes on the edges (loops).
2. we resample the surface to the user specified density while keeping the edge nodes unchanged (as we want to keep them in the same planes as in the original).
3. we add the bounding box facets cut out by the open edges of the mesh. If an edge is located in more than one facets, we decompose it into planar segments and put them to each facet.
4. finally, we are able to produce tetrahedral mesh from the closed surface produced from step 3.

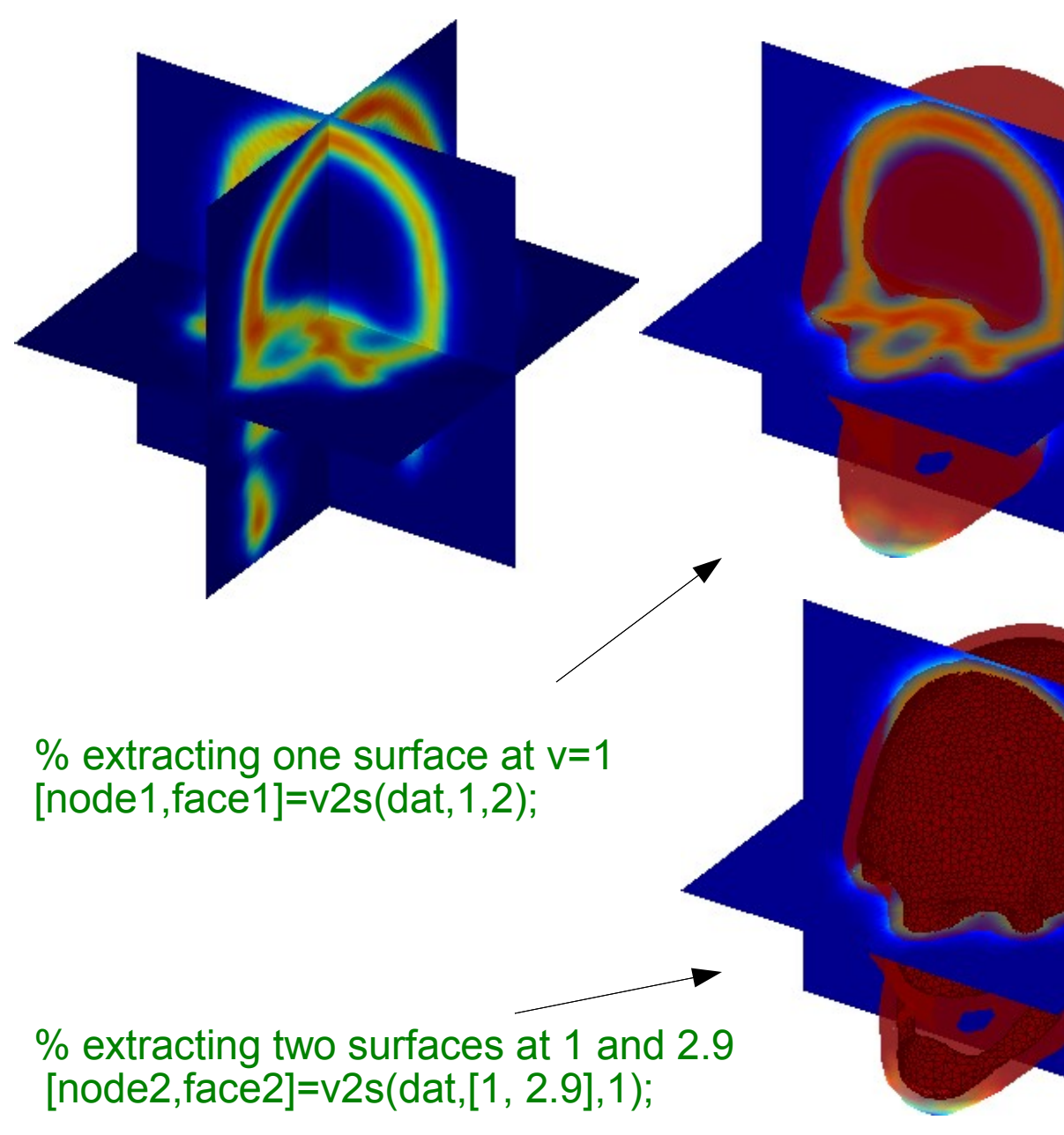




Mesh generation from gray-scale images

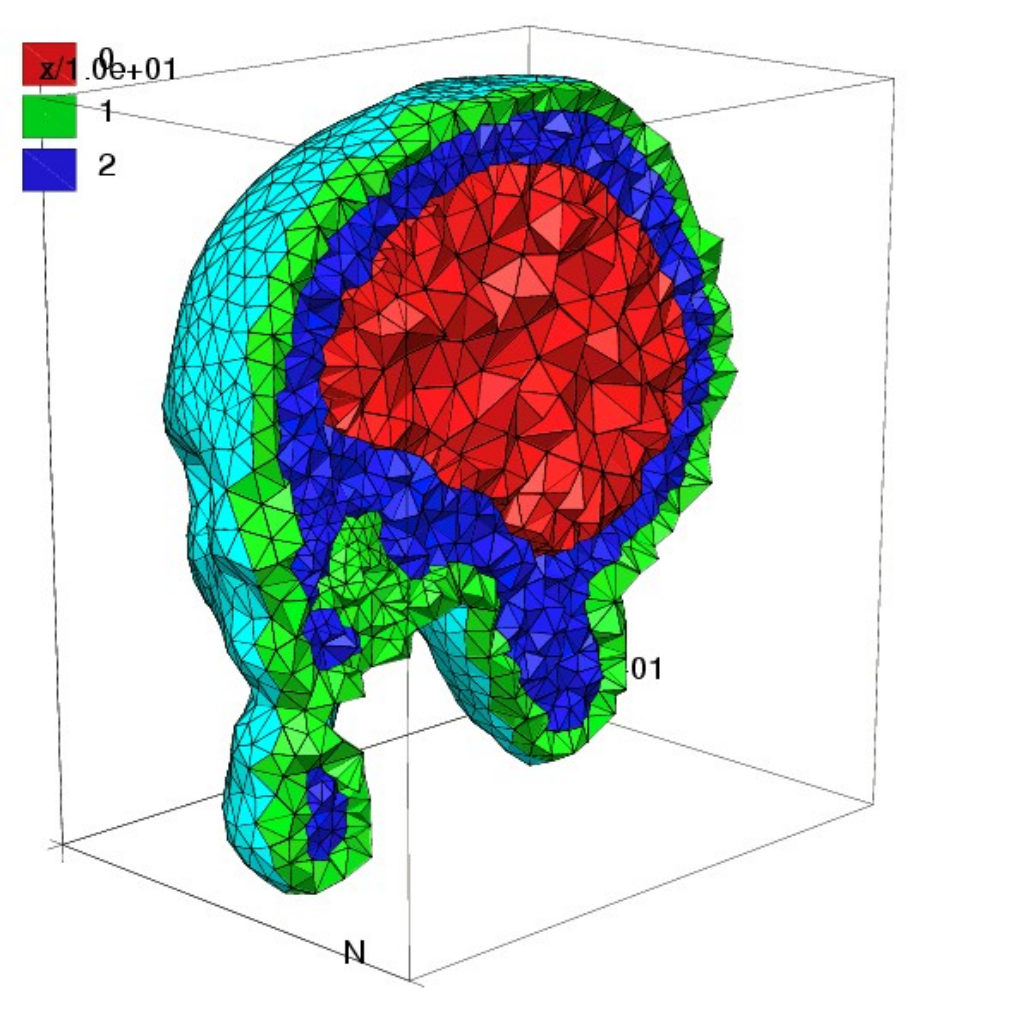
To process a gray-scale image is as straightforward as binary images. The only thing the users need to specify is the isovalues at which one want to create a levelset. One can set surface mesh density via the opt(i).radbound for each levelset. The following is a demonstration for creating volumetric mesh from a gray scale images (data comes from CGAL 3.4).

```
opt(1).radbound=2; % set the first levelset resolution to 2
opt(2).radbound=1; % set the 2nd levelset resolution to 1
[node,elem,face]=v2m(dat,[1,2.9],opt,10);
```



```
% extracting one surface at v=1
[node1,face1]=v2s(dat,1,2);
```

```
% extracting two surfaces at 1 and 2.9
 [node2,face2]=v2s(dat,[1, 2.9],1);
```

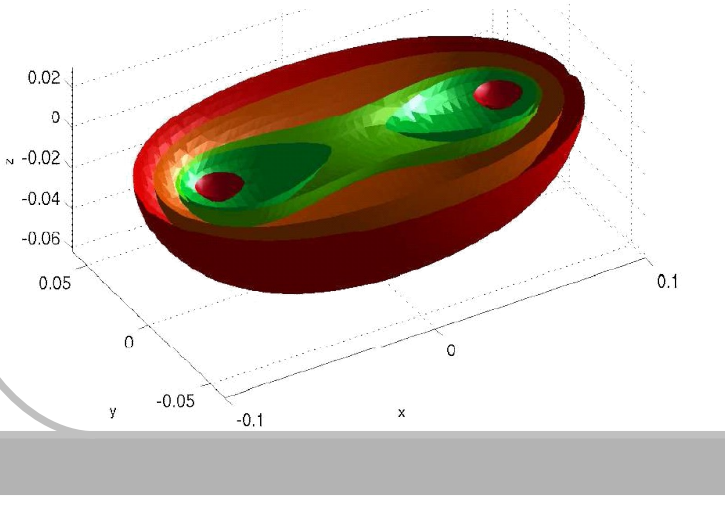


Mesh generation from implicit functions

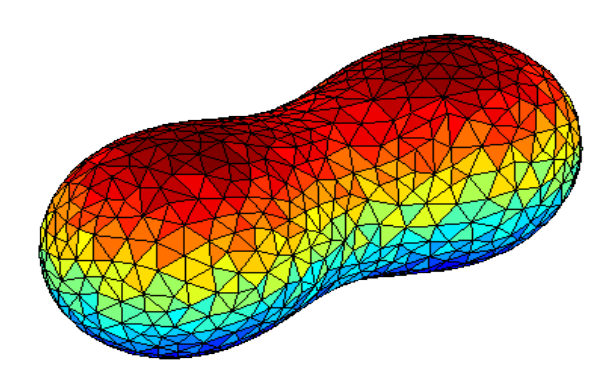
Similar to the gray-scale image based mesh generation, any implicit functions, one can create surface meshes at specified isovalues and subsequently create volumetric mesh. The following is an example to show

3D Green's function for the Helmholtz equation

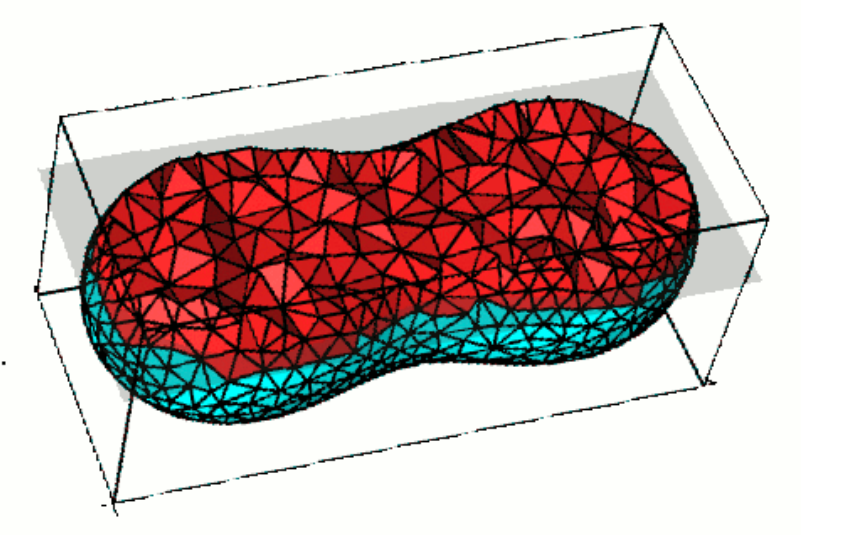
$$g(\vec{r},\vec{r}_s)=\frac{\exp jk|\vec{r}-\vec{r}_s|}{4\pi|\vec{r}-\vec{r}_s|}$$



An iso-surface extracted at implicit function log10(g(r))= -7.6 with maximum element size of 2



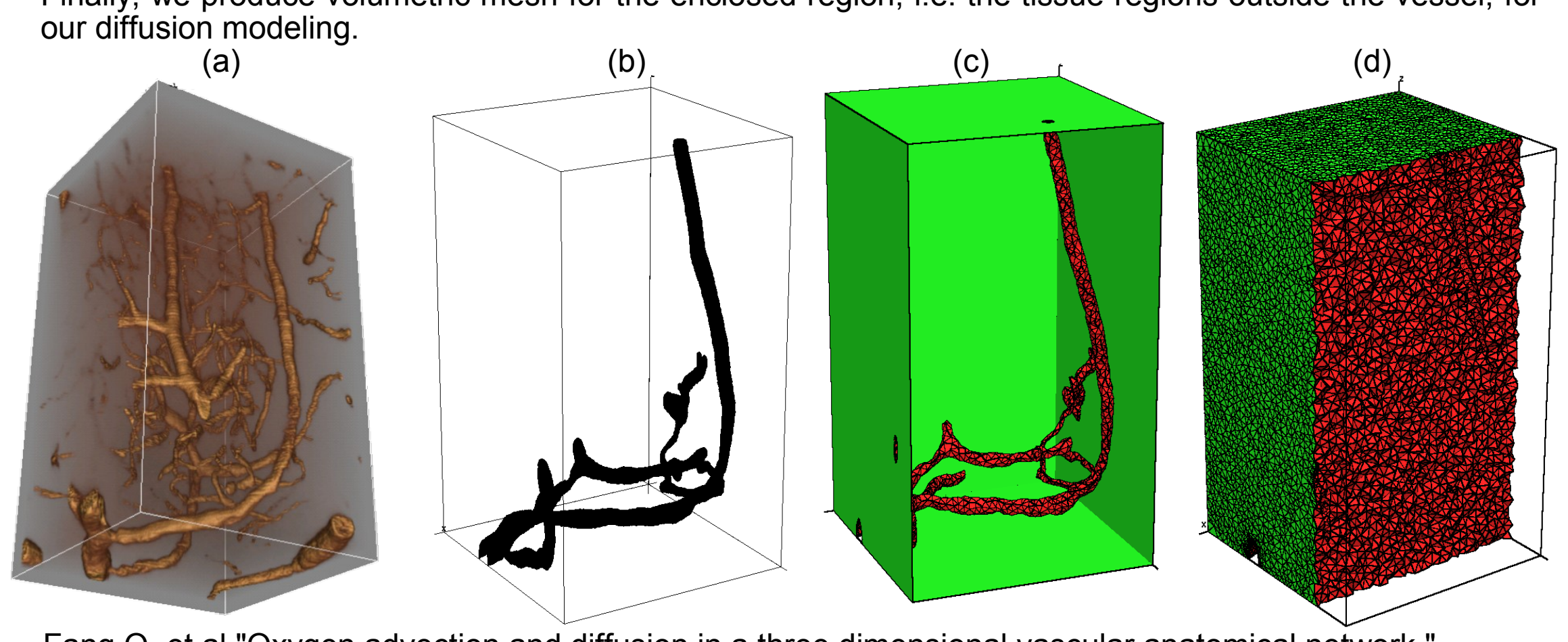
Cross cut view of the generated 3D volumetric mesh



application 1: vessel network

In this project, we attempt to model the oxygen transport, diffusion and consumption in a realistic brain cortex tissue region. Using two-photon microscopy, we obtained high resolution scan of the somasensory cortex of a rat, as in (a). From this image, we identify the arterioles and venules and capillary vessels. A subset of these vessels were used for a finite-element based modeling.

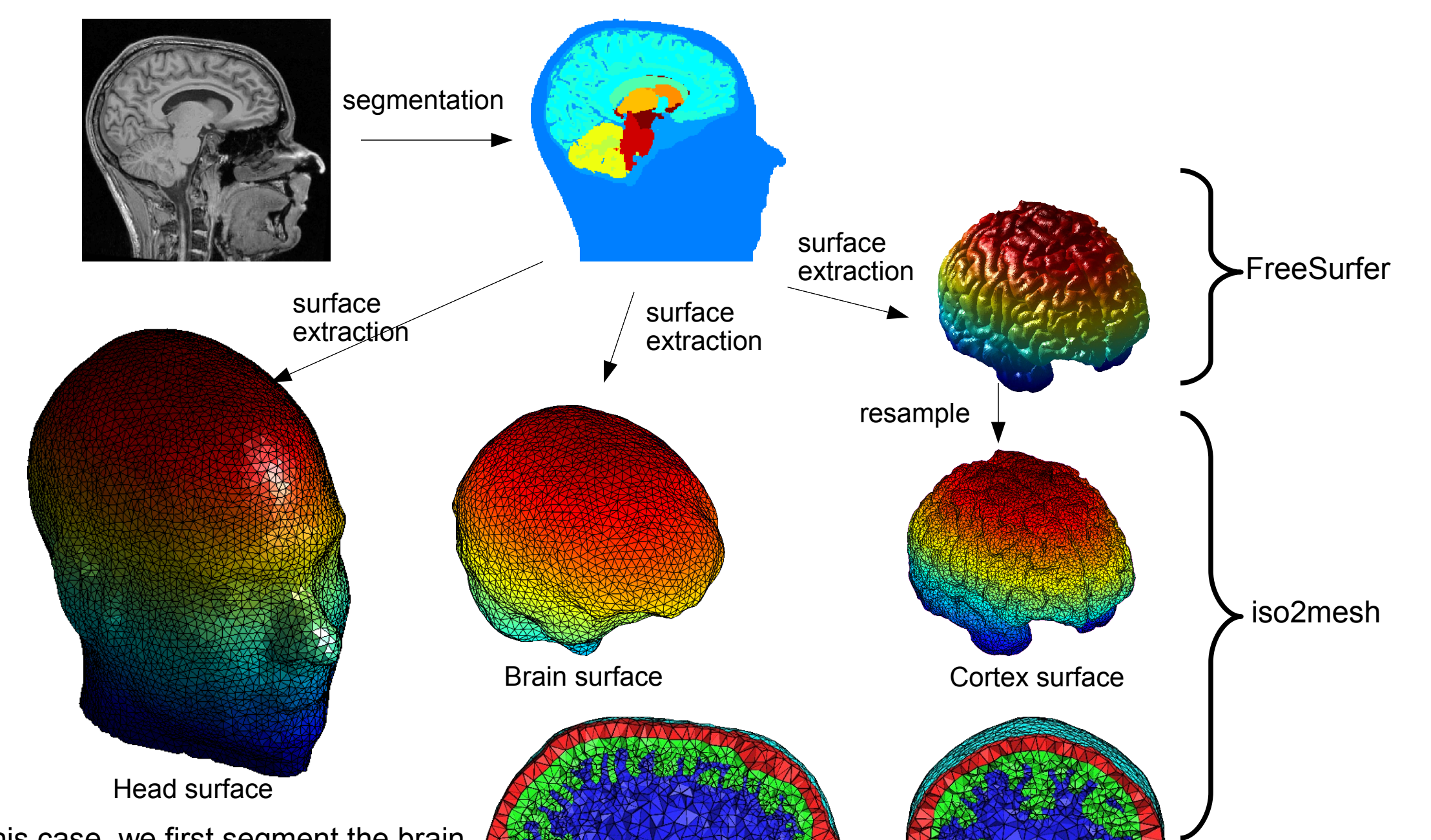
We first produce a binary mask for the voxels inside the vessel, and then produce the voxel-based isosurface, as shown in (b). From this surface mesh, we decimate 98% of the elements, and produce a coarse surface mesh. The bounding box facets are then added to form a water-tight surface (in (c)). Finally, we produce volumetric mesh for the enclosed region, i.e. the tissue regions outside the vessel, for our diffusion modeling.



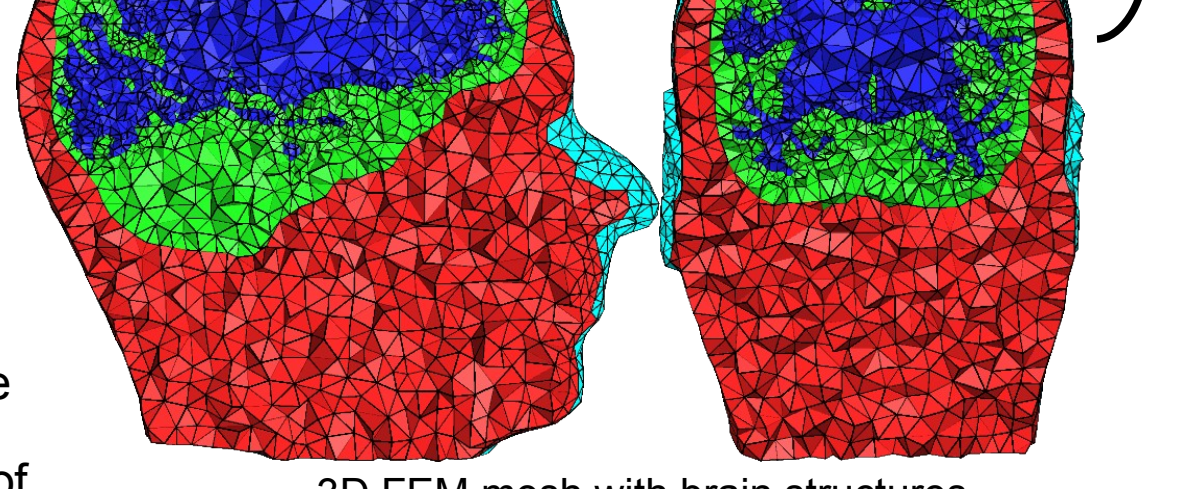
Fang Q, et al,"Oxygen advection and diffusion in a three dimensional vascular anatomical network," Optical Express, vol. 16, Issue 22, pp. 17530-17541, 2008.

application 2: multi-model brain functional imaging

In this project, we combine the structural MRI image of human head with MEG and optical measurements of brain activations to access multi-modality enhancement in the data analysis.



In this case, we first segment the brain MRI image using a widely used tool, FreeSurfer, and produce the cortex surface mesh. From the segmentation, we produce the surface meshes of the head and brain. The FreeSurfer cortex surface is down-sampled to reduce unknown size. From these surfaces, we produced a multi-layer FE mesh of the head and compute the sensitivity map of the brain. These map will be used for a join image reconstruction using optical and MEG data simultaneously.



3D FEM mesh with brain structures

what's next

From the work-flow diagram, you can see our next step is to add a surface-to-volume route to the processing flow. This will be based on our fast mesh slicing subroutine – `qmeshcut`. We will also work on advanced mesh operations, such as moving mesh or adaptive mesh. These features will make iso2mesh an even more attractive option for solving dynamic problems or multi-resolution modeling.

We will also focus on solving two major limitations for the current version of iso2mesh. First, iso2mesh is unable to model non-manifold structures. The iso-surfaces for different values intersect to each other. The other limitation is that the mesh re-sampling process sometimes produces self-intersecting elements. We expect both of these issues can be solved in version 1 or 2 of this toolbox, with advanced algorithms or incorporating proper external tools.

We will keep you posted on the progress. Please visit iso2mesh's homepage at

`http://iso2mesh.sourceforge.net/`
or `http://iso2mesh.sf.net/`