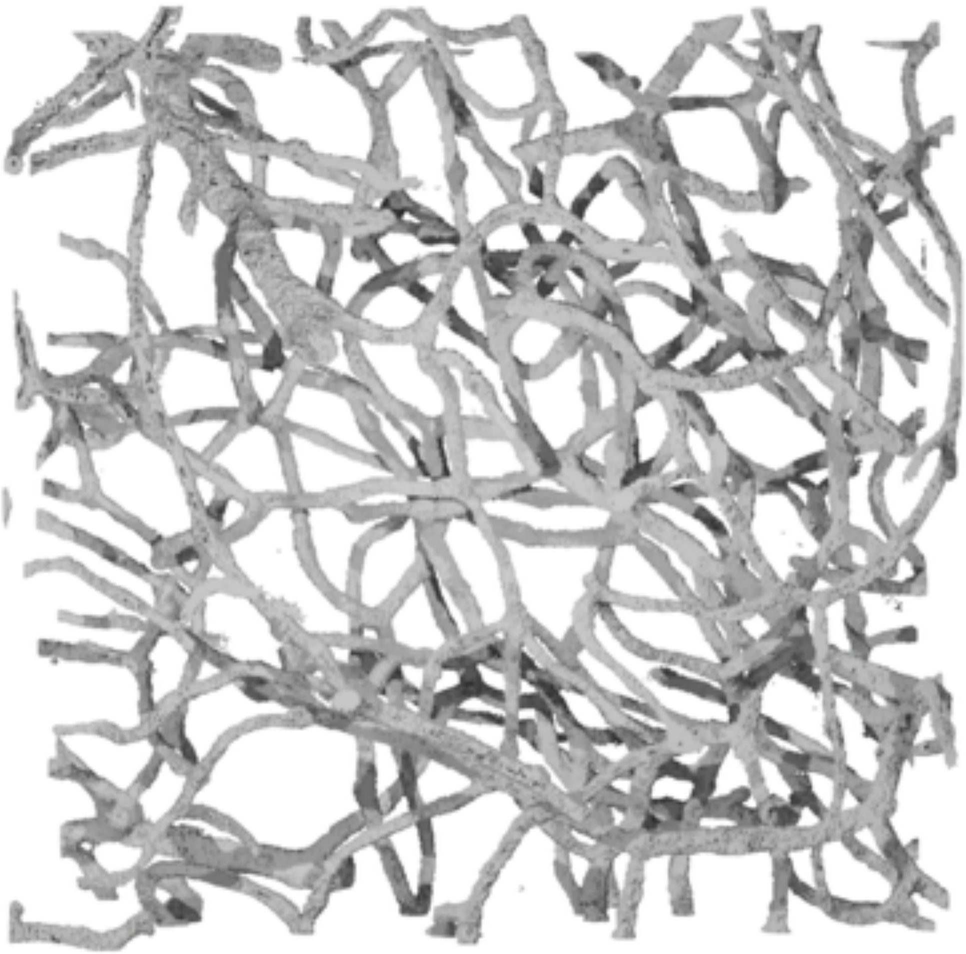
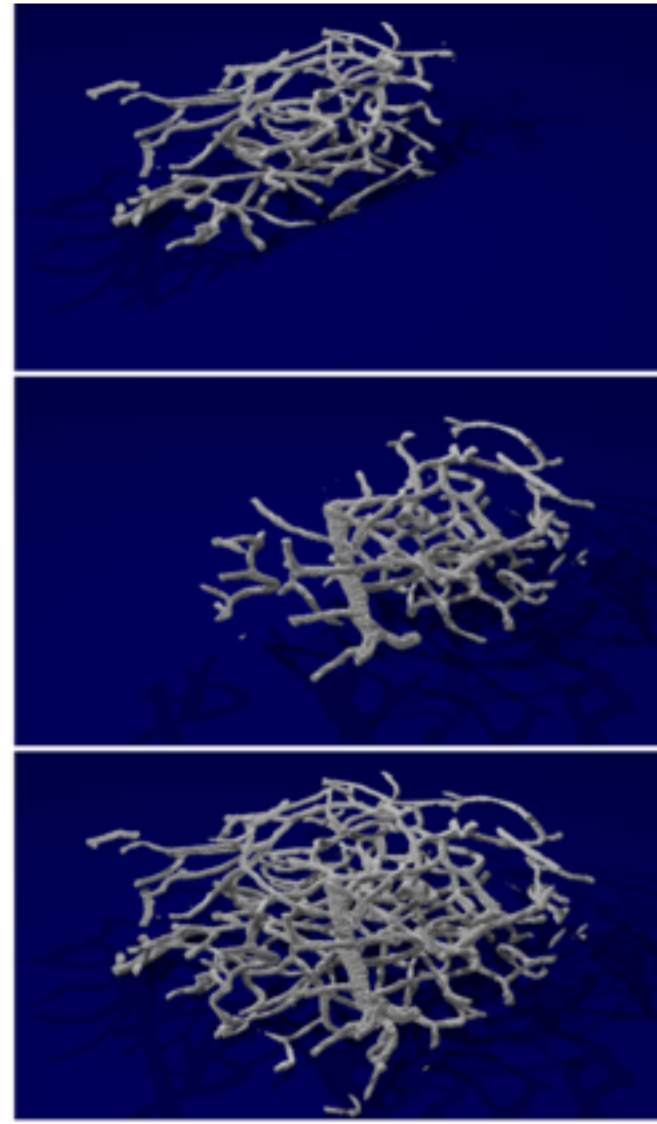
Progressive extraction of neural models from high-resolution 3D images of brain

CAD '16
Vancouver, Canada



Topologically exact solid model of microvessels between neurons



Contents

- **Motivation**
- **Background**
- **First experiments**
- **Method formulation**
- **Conclusion**

Motivation

Marching-cubes (Siggraph, 1987)

Find the 0-surface (or any iso-surface) of a discrete 3D field

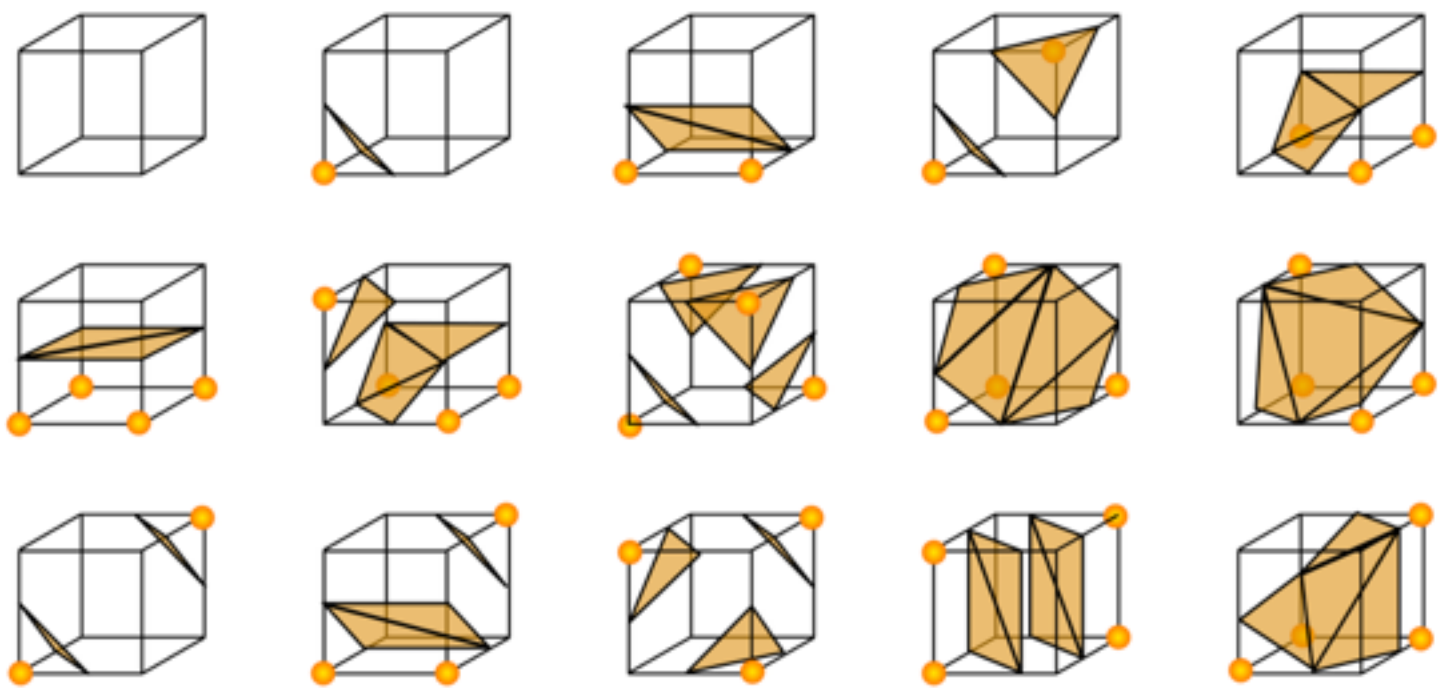


Figure 1: Marching cubes is a method for triangulating an iso-surface of a function, by subdividing the field into equally sized small cubes and triangulating each cube in a way that enforces continuity (if possible)

Progress of high-resolution imaging technology

William E. Lorensen & Harvey E. Cline, "Marching Cubes: a High Resolution 3D Surface Construction Algorithm", Siggraph, 1987

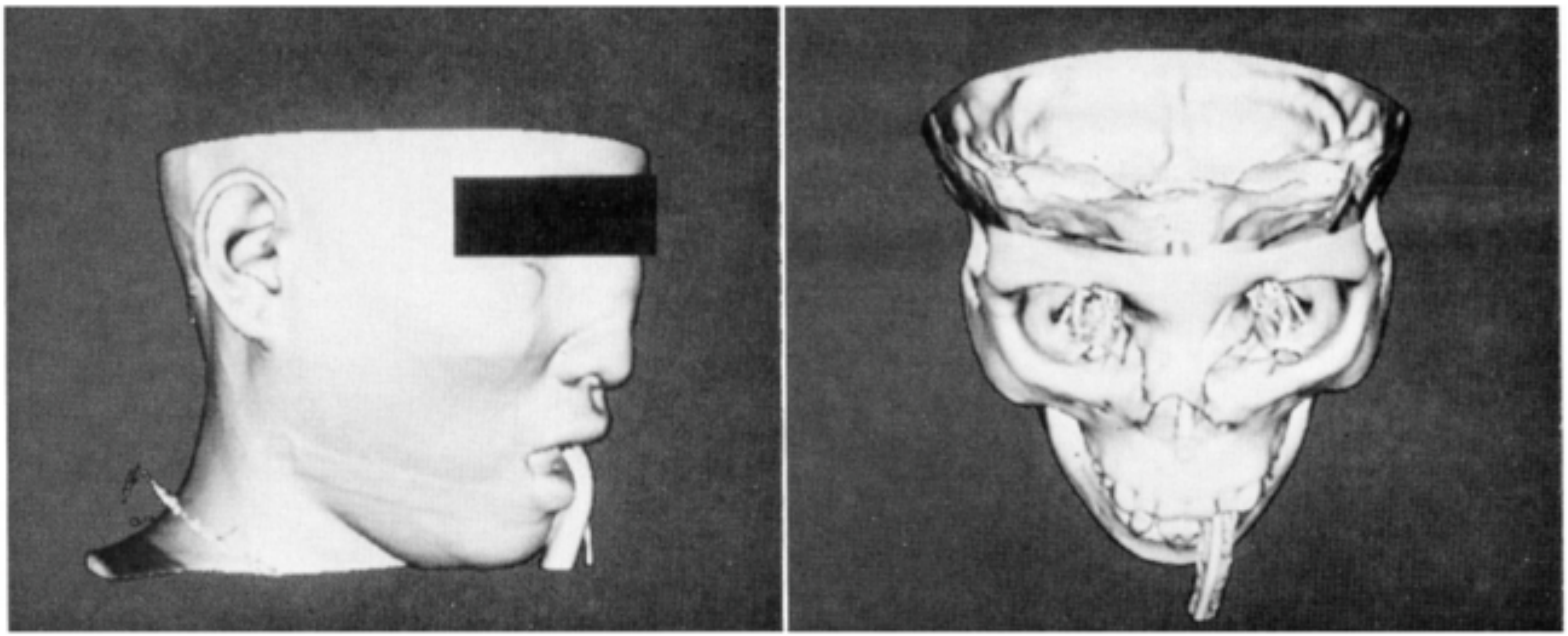


Figure 2: $260 \times 260 \times 93$ CT images. The 93 axial slices are 1.5 mm thick, with pixel dimensions of 0.8 mm (1987)

Progress of high-resolution imaging technology

Nobel Prize in Chemistry 2014 awarded to Betzig, Moerner & Hell for "development of super-resolved fluorescence microscopy," which brings "optical microscopy into the nanodimension"

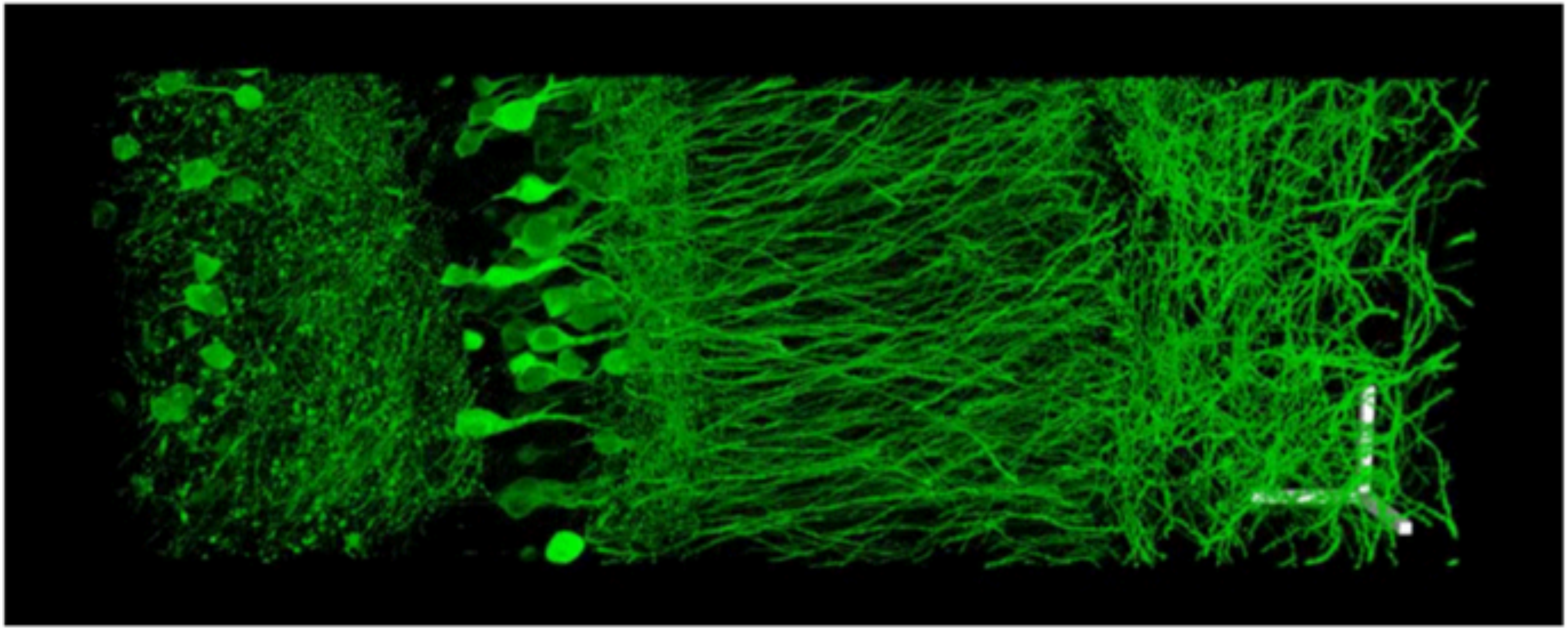


Figure 3: Volume rendering (i.e. dynamic images) of a portion of hippocampus showing neurons and synapses

Source: Fei Chen, Paul W. Tillberg, Edward S. Boyden. Expansion microscopy. Science, January 15 2015; DOI: 10.1126/science.1260088

Progress of high-resolution imaging technology

Source: Fei Chen, Paul W. Tillberg, Edward S. Boyden. Expansion microscopy. Science, January 15 2015; DOI: 10.1126/science.1260088

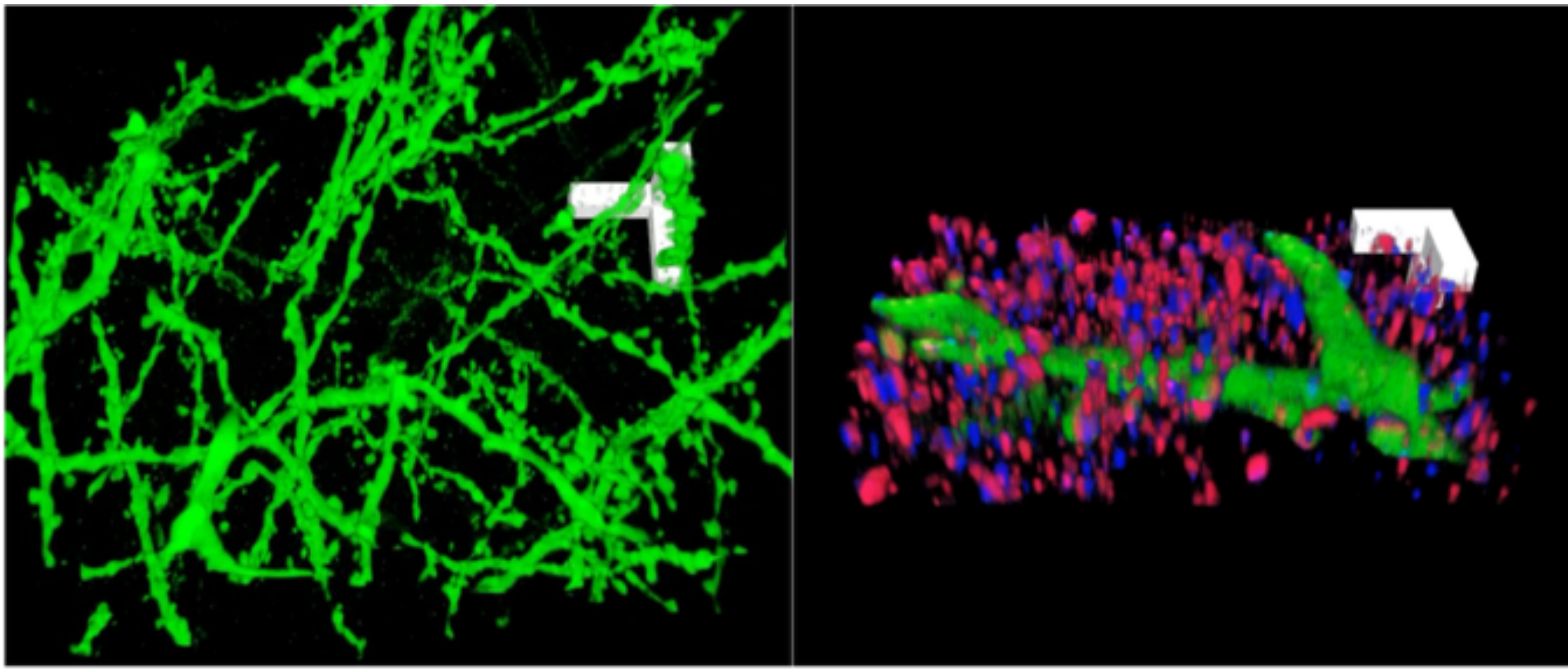


Figure 4: Volume rendering (i.e. images) of dendrites

(A) Volume rendering of dendrites in CA1 stratum lacunosum moleculare (slm): 52.7 μm (x), 42.5 μm (y), and 35.2 μm (z). (B) Volume rendering of dendritic branch in CA1 slm: 13.5 μm (x), 7.3 μm (y), and 2.8 μm (z) 1 μm.

Progress of high-resolution imaging technology

The ViSUS software framework was designed to allow the interactive exploration of massive scientific models on a variety of hardware, even geographically distributed

Neuroscientist Alessandra Michelucci (Utah) with ViSUS power wall and neurones image

"Houston: We've had a problem"

Search for detailed geometric models from hi-res imaging

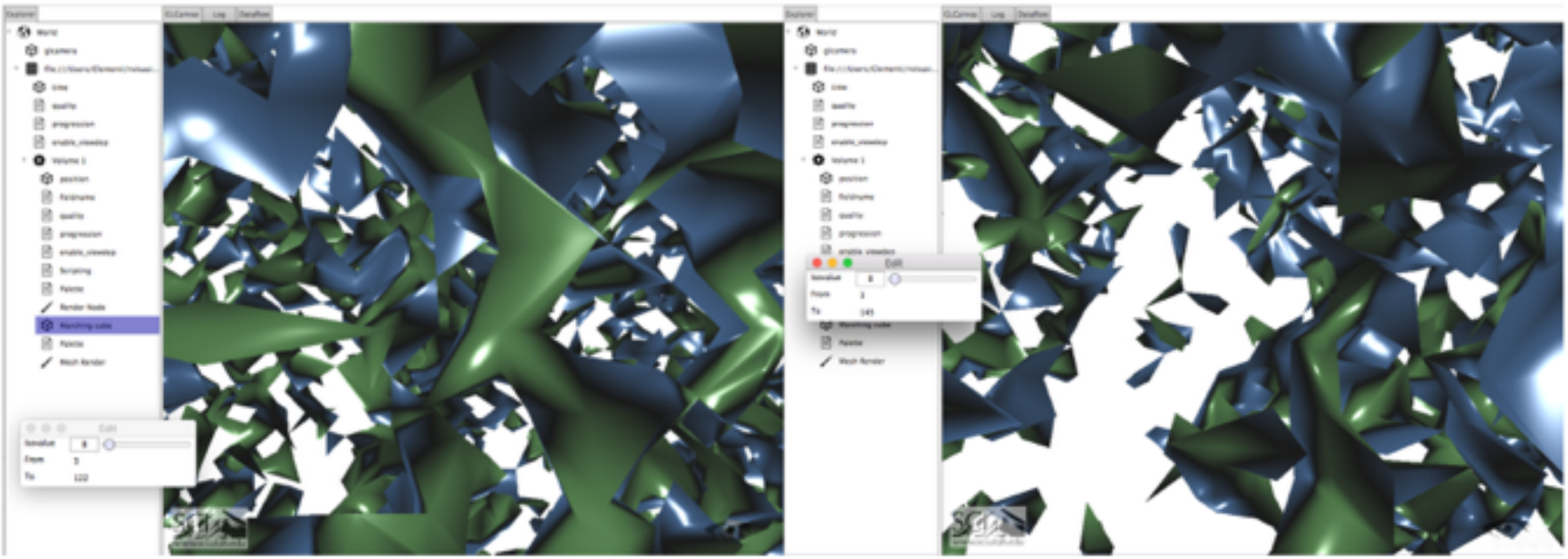


Figure 6:Marching cubes on small structures, with compressed voxels

WHY ?

because the luminance field of compressed voxels is not continuous for diffused small structures . . .

"Apollo 13: We have a solution"

Look for geometric models using extreme hi-res imaging

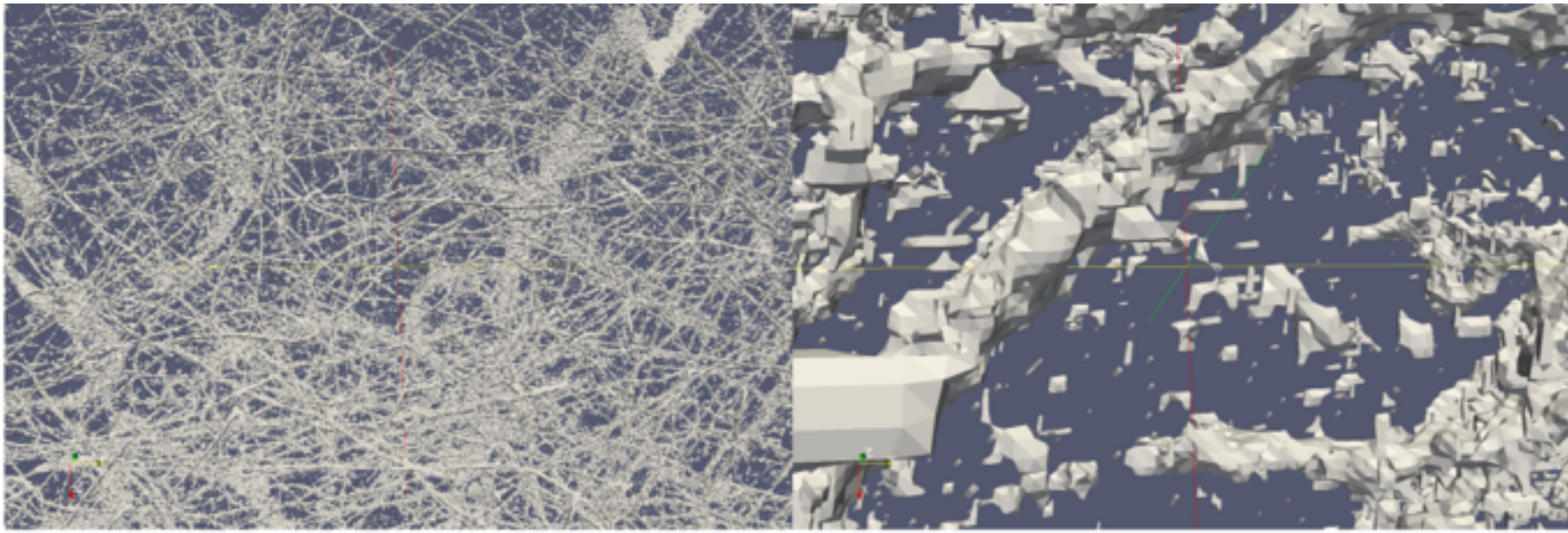
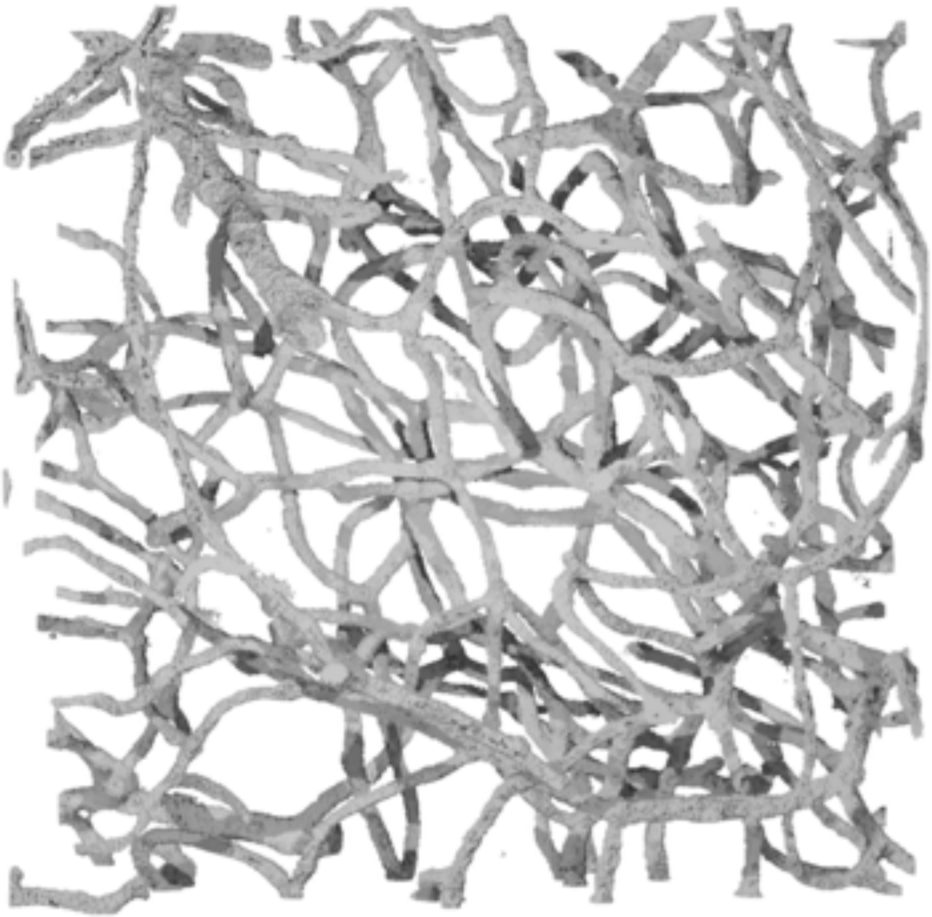


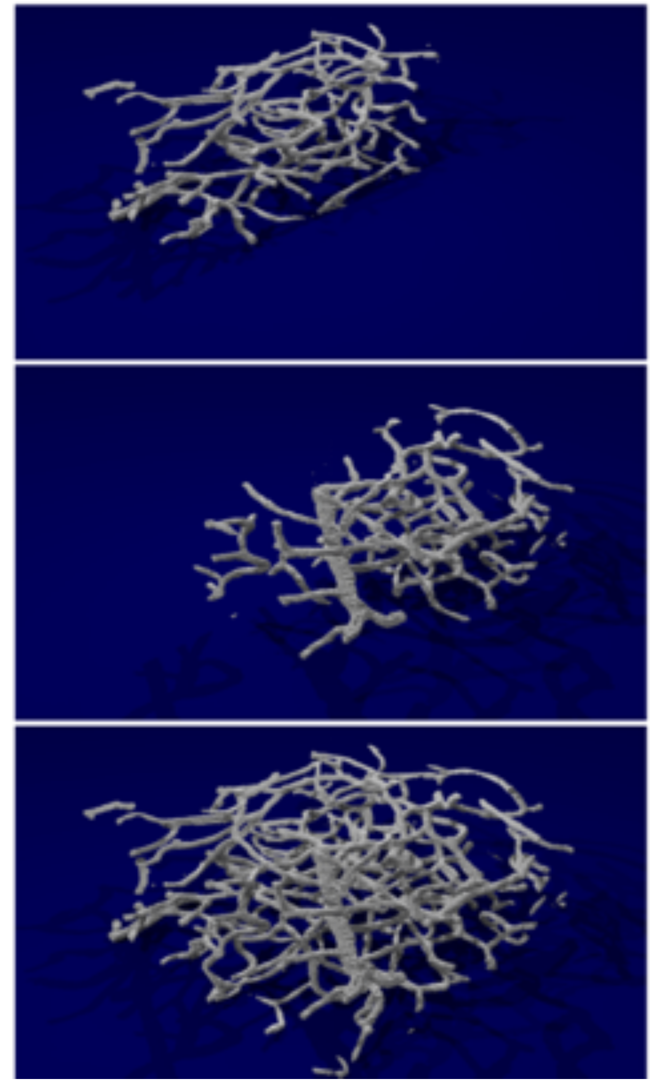
Figure 7:LAR extraction of solid models (B-reps), topologically exact at the resolution of the image

REMARK

All boundary surfaces are closed, including noise !!

And continuing the travel . . .





Topologically exact solid model of microvessels between neurons

notice the shadows in these ray-traced images . . . (using 24 GB of memory!)

Background

LAR: novel topological representation scheme

$\boxed{\text{Models: (co)chain complexes}} \rightarrow \boxed{\text{Reprs: sparse binary matrices}}$

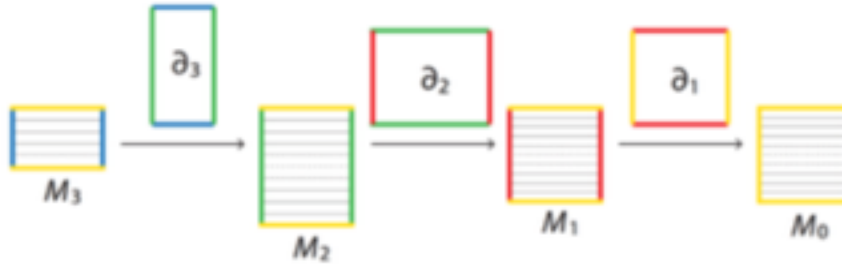
cochains (all maps, discrete fields) and coboundary maps (δ^d operators)

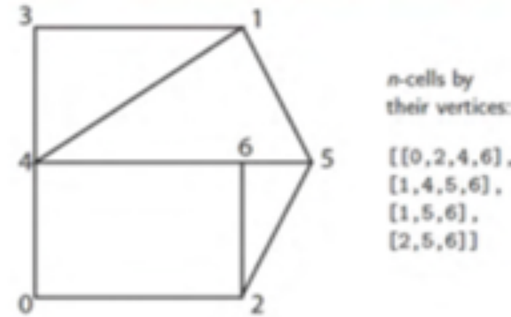
$$
\begin{tikzcd}
C^d \arrow[d, leftrightarrow, "\cong"] & C^{d-1} \arrow[l, "\delta^{d-1}"'] \arrow[d, leftrightarrow, "\cong"] & \cdots\cdots \arrow[l, "\delta^{d-2}"'] & C^1 \arrow[l, "\delta^1"'] \arrow[d, leftrightarrow, "\cong"] & C^0 \arrow[l, "\delta^0"'] \arrow[d, leftrightarrow, "\cong"] \\
C_d \arrow[r, "\partial_d"] & C_{d-1} \arrow[r, "\partial_{d-1}"] & \cdots\cdots \arrow[r, "\partial_2"] & C_1 \arrow[r, "\partial_1"] & C_0
\end{tikzcd}
$$

chains (linear spaces of model subsets) and boundary maps (∂_d operators)

Linear Algebraic Representation:

from cellular models to sparse binary matrices





Remark (Input and long-term storage)

space(LAR)= $|FV| = 2|E|$!!!

Remark (Full topology representation)

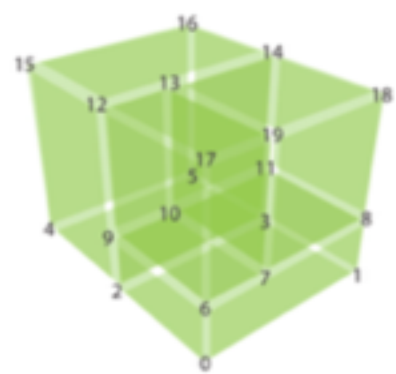
$|VE| + |VF| = 4|E|$

Remark (Any topological queries)

single SpMV multiplication

Remark (Sparse Matrix-Vector Multiplication)

is one of the most important computational kernels, for very effective iterative solution methods



$$M_3 = \begin{pmatrix}
0&0&1&1&1&1&0&0&0&1&1&1&1&1&1&1&1&0&0&0\\
1&1&1&1&0&0&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&1&1&0&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&1&1&0&1&1&0&1&1&0&0&0&1&1
\end{pmatrix}$$

$$M_2 = \begin{pmatrix}
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&0&0&0&0&1&0&0&1&0&0&0&0&0&0&0&0\\
0&0&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&1&0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0\\
0&0&0&1&0&1&0&0&0&0&0&1&0&0&1&0&1&0&0&0\\
0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0\\
0&0&0&0&0&0&1&1&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&0&1\\
0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&1&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0&1&1\\
0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1&1&0&1&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&0&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0&0&0&1&1
\end{pmatrix}$$

Sparse matrices: COO, CSR and CSC formats

COO: coordinate format: (val, row, col)

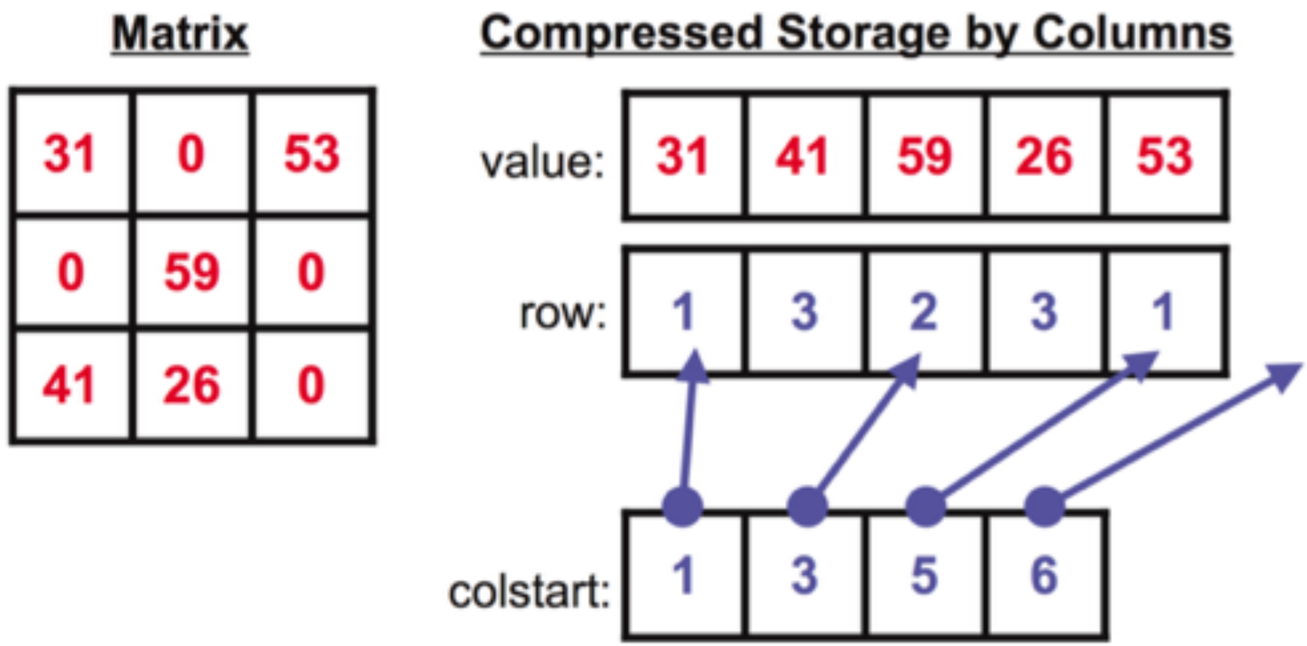
is a fast format for constructing sparse matrices. Once a matrix is constructed, convert to CSR or CSC format for fast arithmetic and matrix vector operations

CSR: Compressed Sparse Row: ((val, col) row)

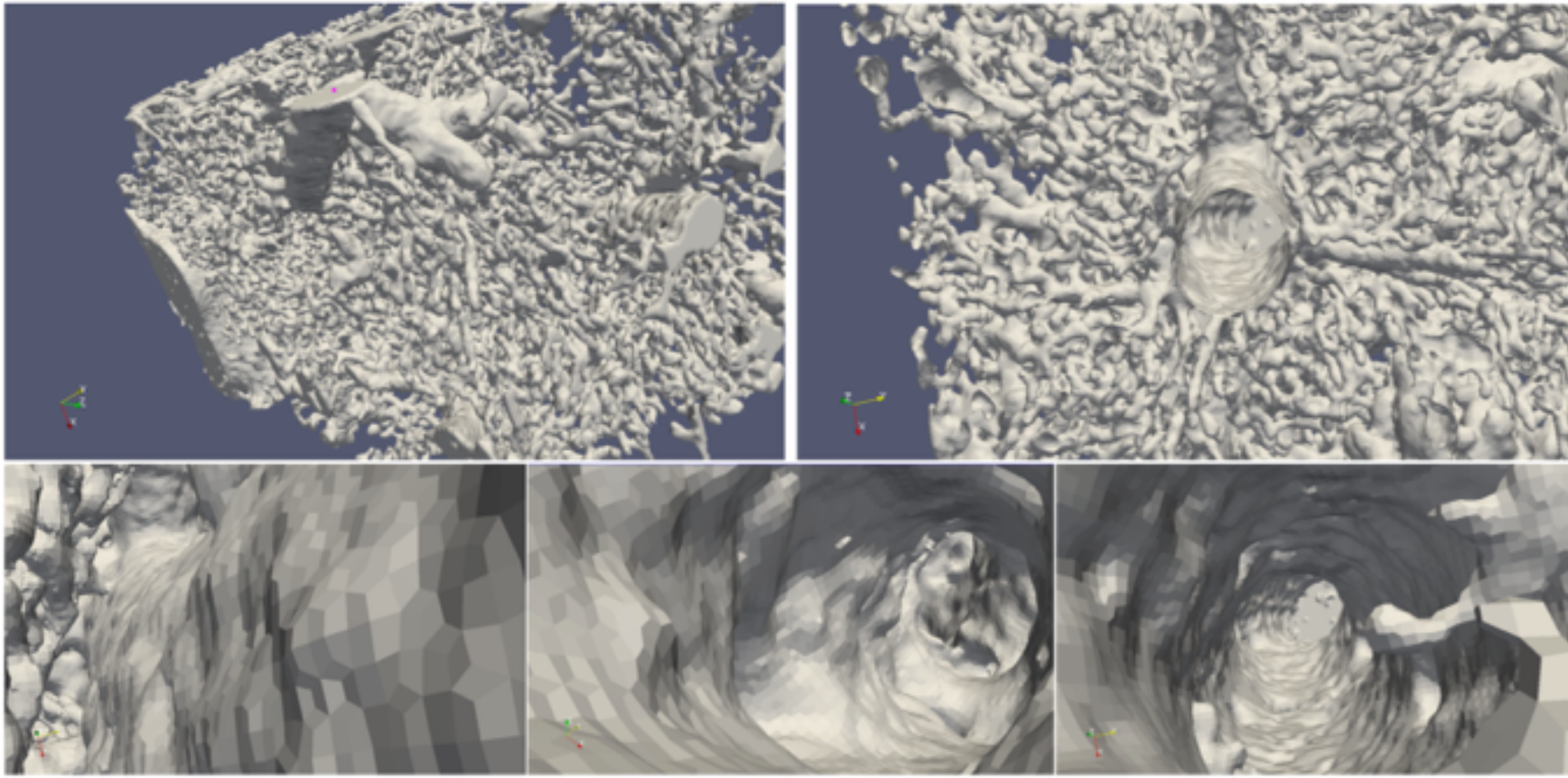
efficient arithmetic operations, efficient row slicing, fast matrix vector products

CSC: Compressed Sparse Column: ((val, row) col)

efficient arithmetic operations, efficient column slicing, fast matrix vector products

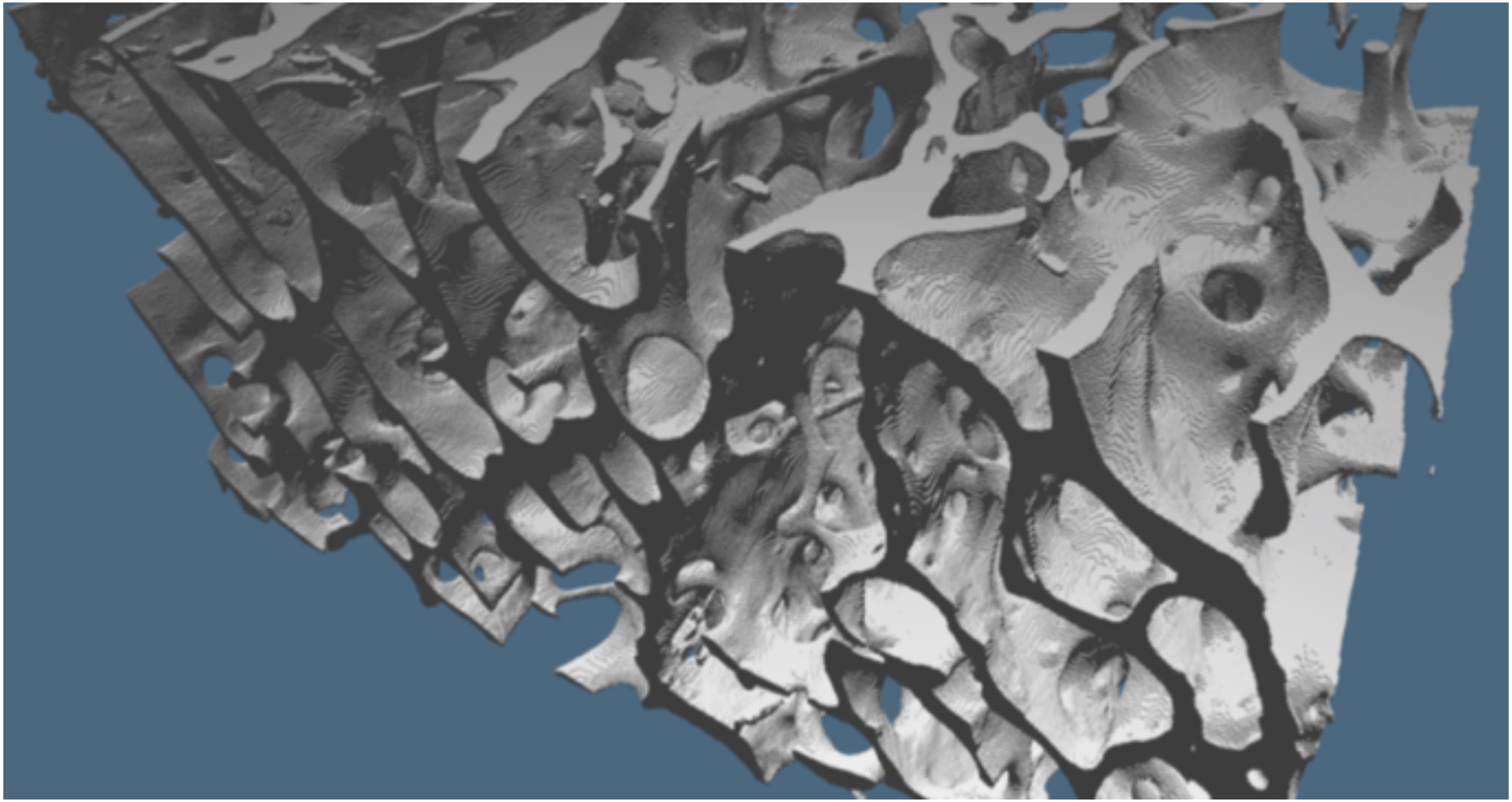


LAR examples (1/2)



Paoluzzi, DiCarlo, Furiani, Jirik, CAD models from medical images using LAR, Computer-Aided Design and Applications, 2016. doi: 10.1080/16864360.2016.1168216

LAR examples (2/2)



Algebraic extraction of topology and geometry from 3D images: proof-of-concept. IEEE P3333.2 WG meeting, c/o SK Telecom, Feb 14/15 2013, Seoul, KOREA

Algorithm: matrix of ∂_2 operator

For a surface (topological space) made of all quads (2-cells)

```
function LAR_CSC_BOUNDARY(FV, EV)
    n, m ← |EV|, |FV|
    edict ← {tuple(e) : k for (k, e) in enumerate(EV)}
    data, row, col ← [], [], []
    data ← [data.extend( [1, 1] ) for k ∈ range(n)]
    for f ∈ FV do
        for i ∈ range(4) do
            for j ∈ range(i + 1, 4) do
                if (f[i], f[j]) ∈ edict then
                    row.extend( edict[ (f[i], f[j]) ] )
                end if
            end for
        end for
    end for
    col ← [col.extend( [k, k, k, k] ) for k ∈ range(m)]
    return coo_matrix( (data, (row, col)), shape = (n, m) ).tocsc()
end function
```

Fast construction of the CSC sparse matrix form

Topological incidence operators

Binary topological relations between LAR cells

	C	F	E	V
C	CC	$\boxed{\text{CF}}$	$\boxed{\text{CE}}$	CV
F	$\boxed{\text{FC}}$	FF	$\boxed{\text{FE}}$	FV
E	$\boxed{\text{EC}}$	$\boxed{\text{EF}}$	EE	EV
V	VC	VF	VE	VV

	C	F	E	V
C	$\mathbf{1}_C^\top \circ \mathbf{1}_C$	∂_3	$\partial_2 \oplus \partial_3$	$\mathbf{1}_C$
F	δ_2	$\mathbf{1}_F^\top \circ \mathbf{1}_F$	∂_2	$\mathbf{1}_F$
E	$\delta_2 \oplus \delta_1$	δ_1	$\mathbf{1}_E^\top \circ \mathbf{1}_E$	$\mathbf{1}_E$
V	$\mathbf{1}_C^\top$	$\mathbf{1}_F^\top$	$\mathbf{1}_E^\top$	$\mathbf{1}_V$

and corresponding topological operators on ∂◦ chains

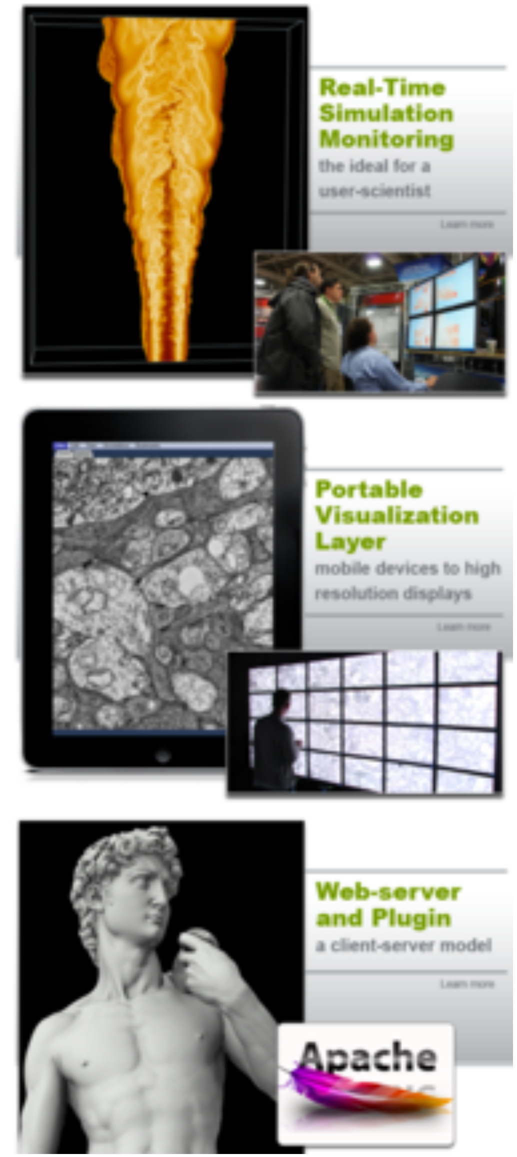
Remark

Operators may be applied to chains (sets of cells)

Goal: integrating LAR with ViSUS

The ViSUS software framework was designed to allow the interactive exploration of massive scientific models on a variety of hardware, even geographically distributed

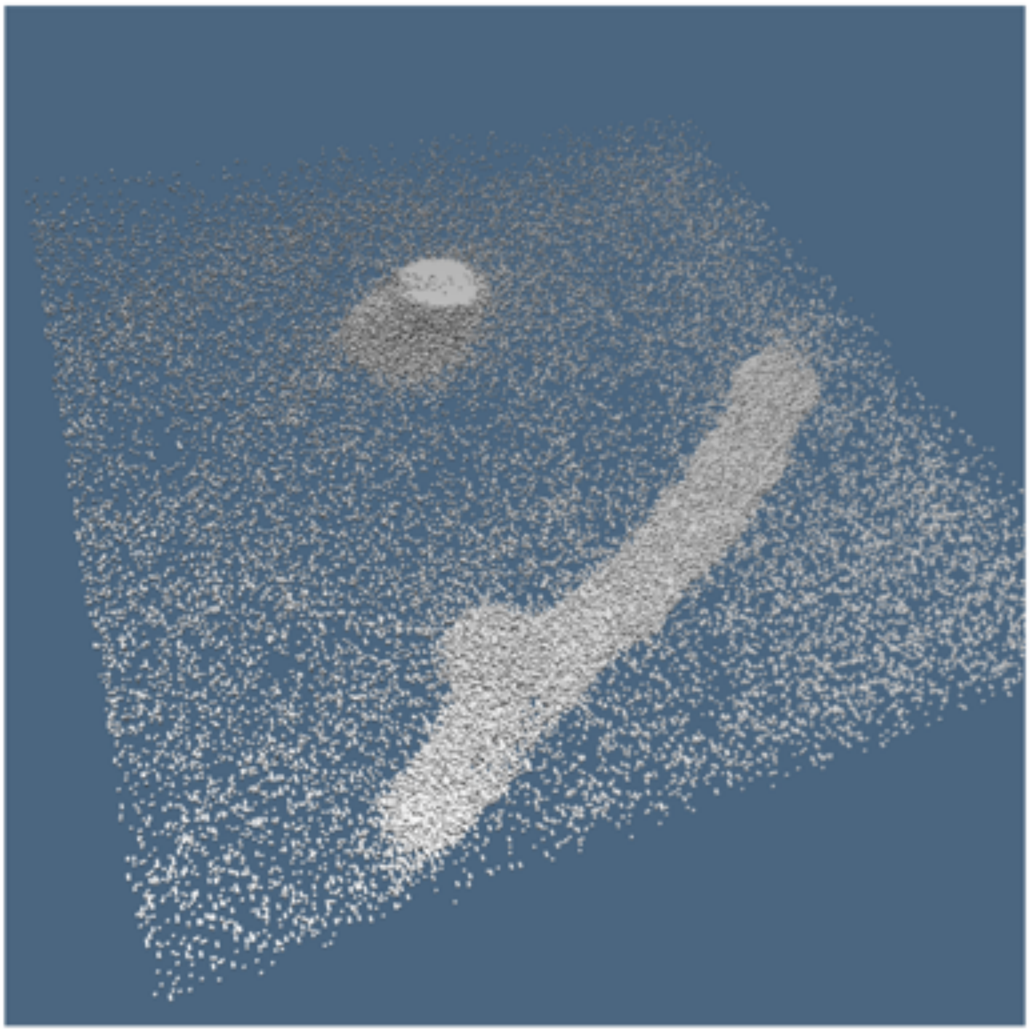
1. A lightweight and fast out-of-core data management framework using multi-resolution space-filling curves
2. A dataflow framework that allows data to be processed during movement
3. A portable visualization layer which was designed to scale from mobile devices to powerwall displays with same code base.



First experiments

First experiments

STEP 1: surface extraction (512 × 512 × 128)

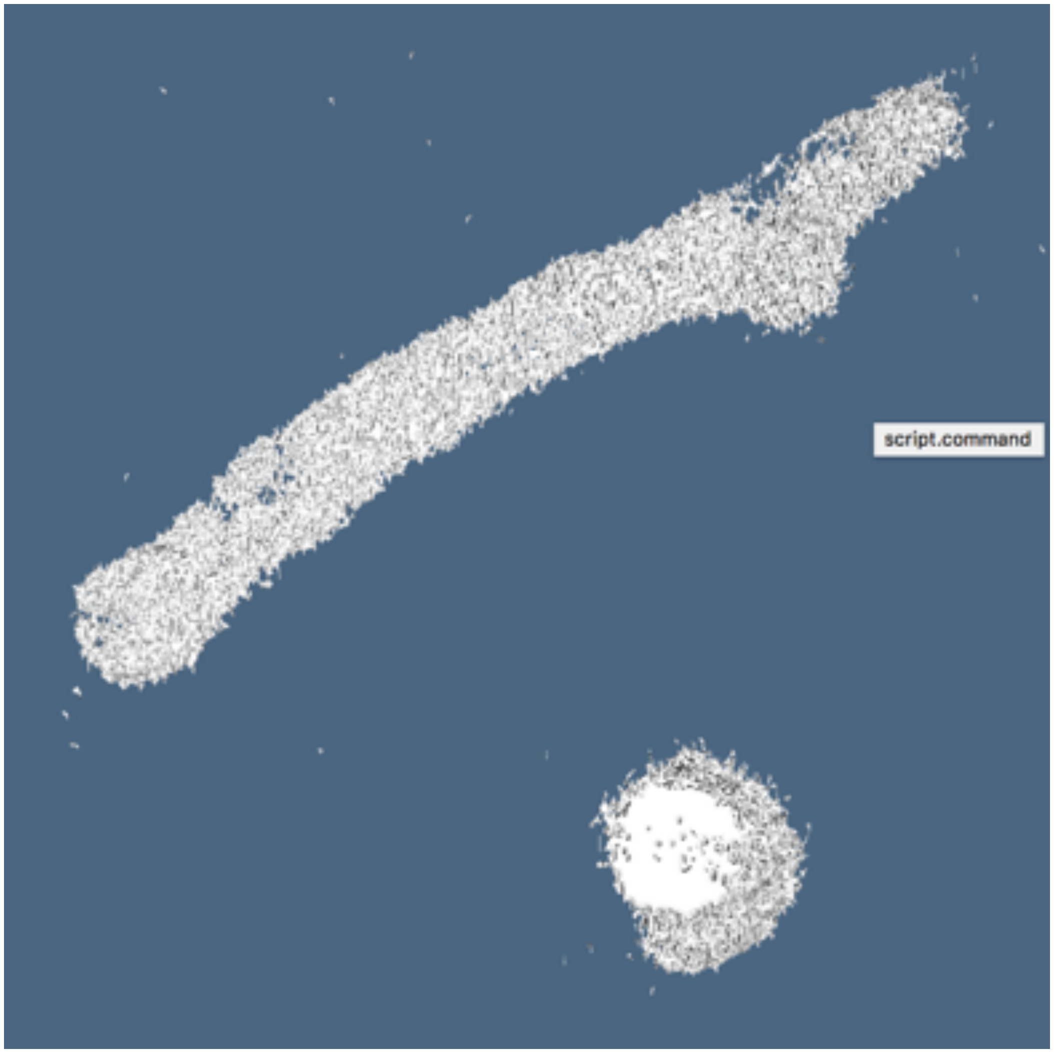


B-rep model, including noise

Test Data Set: B-rep of the chain of voxels above a luminance threshold

First experiments

STEP 1: surface extraction (512 × 512 × 128)

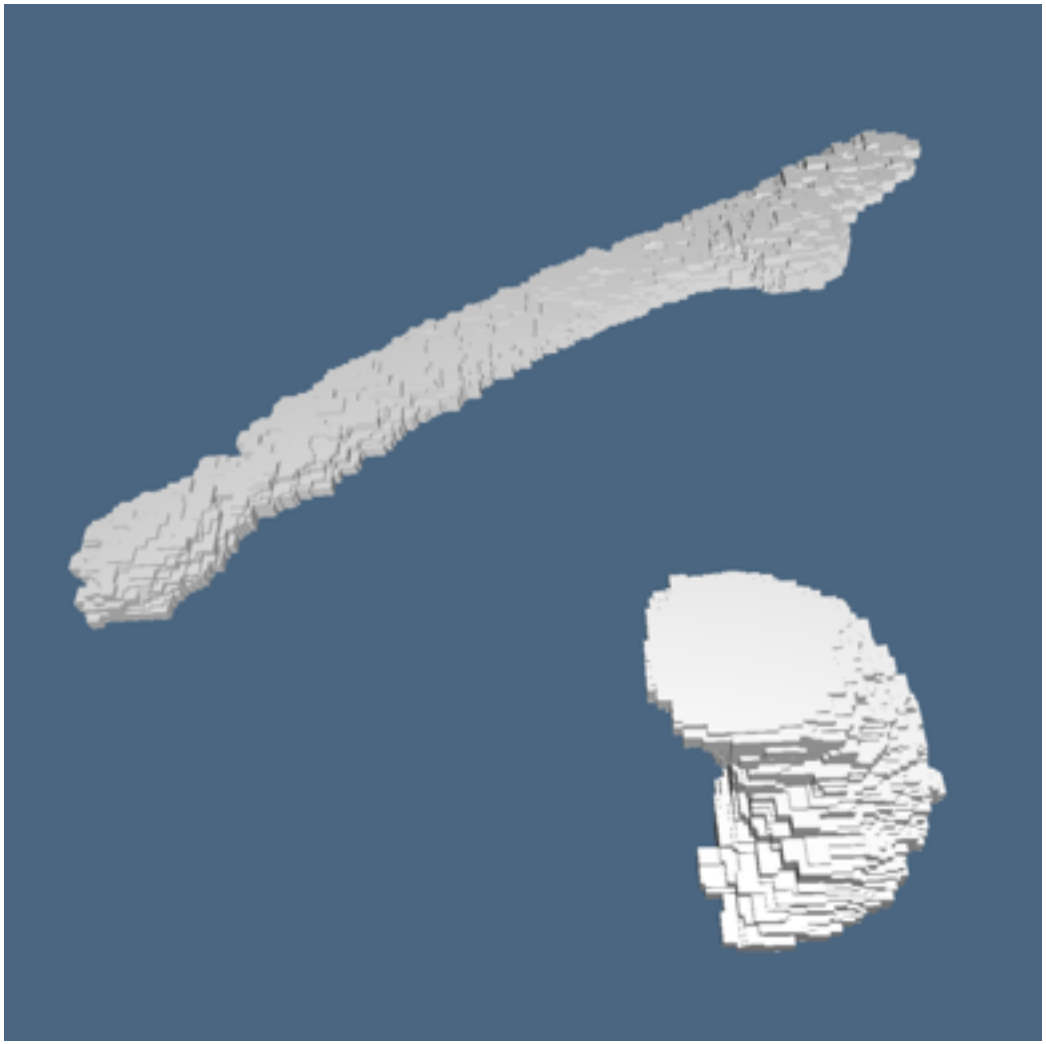


model after noise filtering

Test Data Set: B-rep of the chain of voxels above a luminance threshold

First experiments

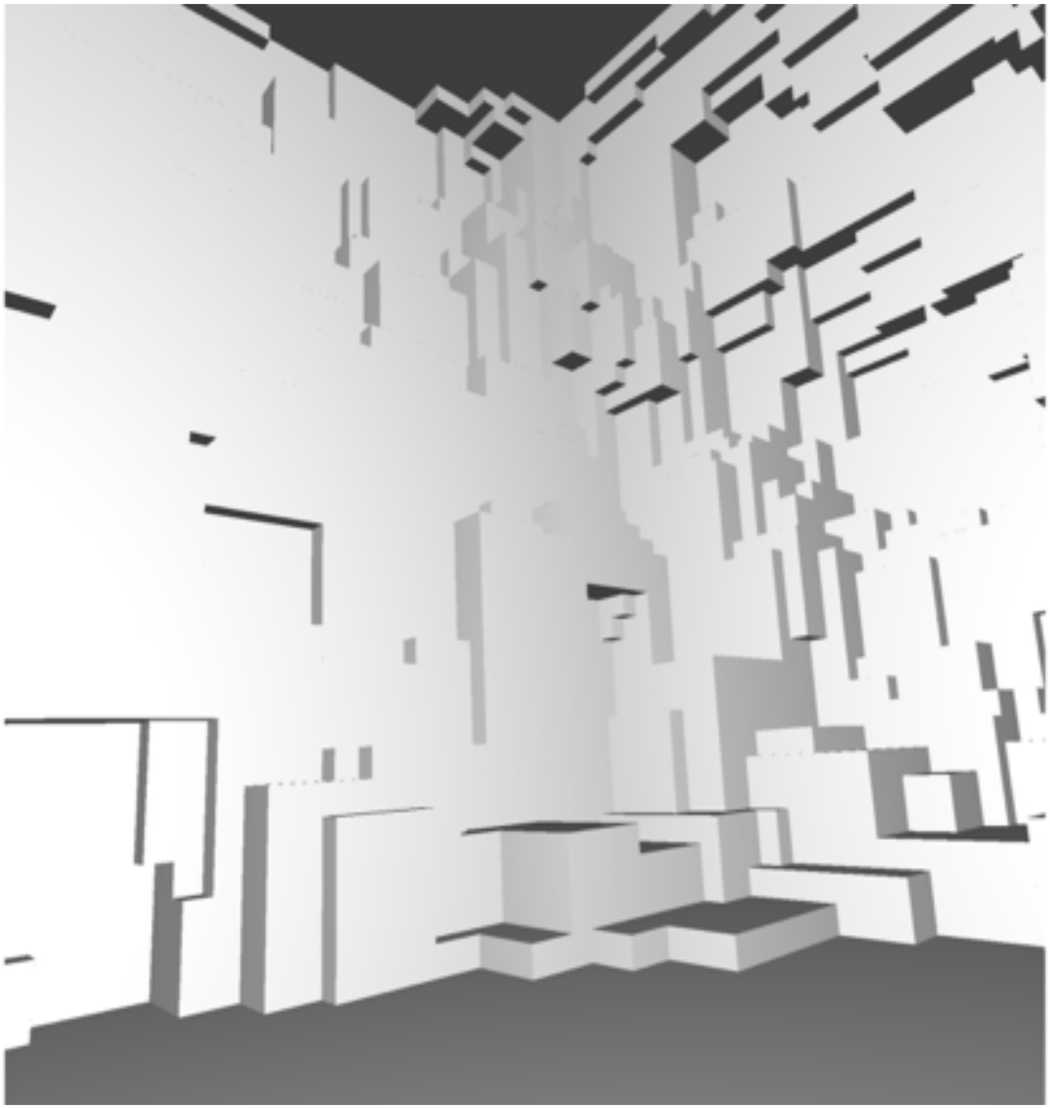
STEP 1: surface extraction (512 × 512 × 128)



model after *closing* and *opening* mathematical morphology operators

Test Data Set: B-rep of the chain of voxels above a luminance threshold

Surface extraction (512 × 512 × 128)



interior view of a closed shell (portion of data chain)

Test Data Set: B-rep of the chain of voxels above a luminance threshold

Method formulation

IDEA: compute a family of intrinsic curves

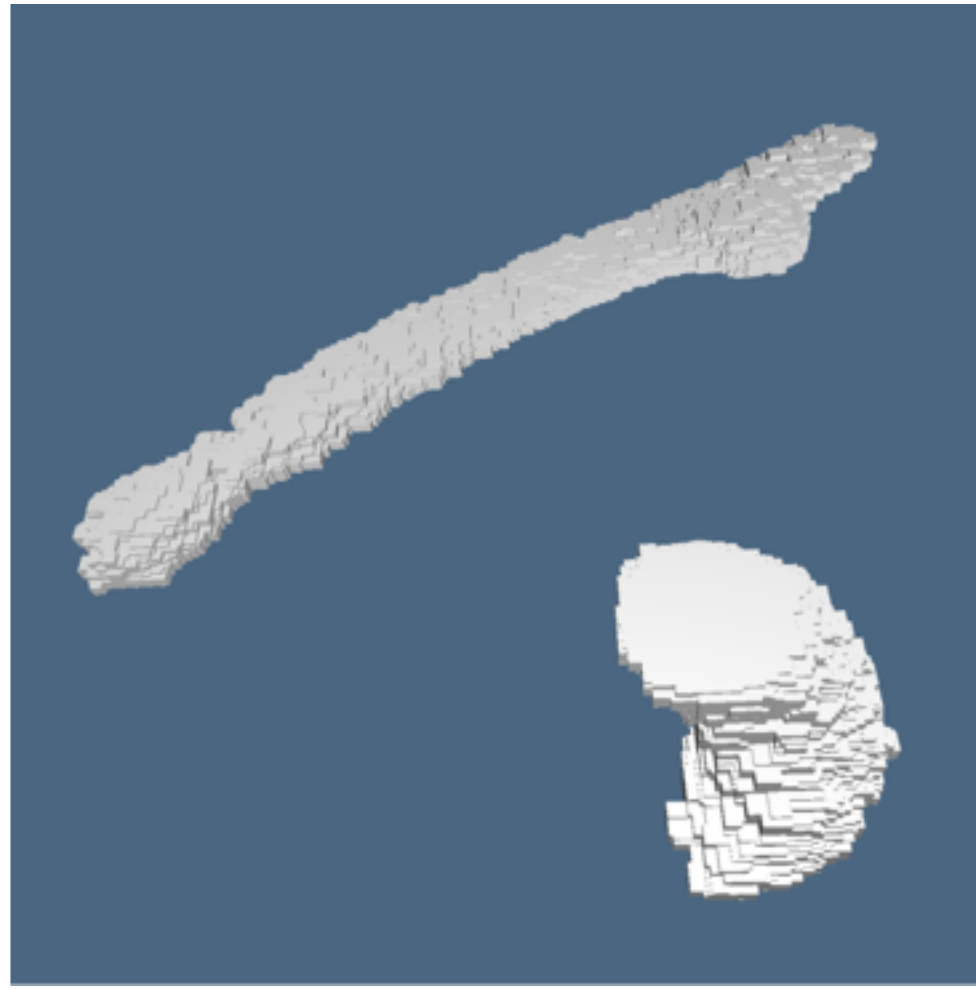
For each Vol in a block-partition of a 3-image:

Initialize

$$Vol \in C_3$$
$$S_0 := \partial_3 Vol \in C_2$$
$$\mathcal{C}_0 := \partial_2 S_0 \in C_1$$

While $\partial_2 S_k \neq 0$:

$$S_{k+1} := S_k + \delta_1(\partial_2 S_k)$$
$$\text{if } (k+1 \bmod d) = 0:$$
$$\mathcal{C}_{k+1} := \mathcal{C}_k + \partial_2 S_{k+1}$$



Something wrong here?

I wrote:

$$Vol \in C_3$$
$$S_0 := \partial_3 Vol \in C_2$$
$$\mathcal{C}_0 := \partial_2 S_0 \in C_1$$

hence

$$\mathcal{C}_0 = \partial_2(\partial_3 Vol) \neq 0 \ ??$$

ERROR !?!

NO.

Because ∂_2 and ∂_3 are computed over different topological spaces:

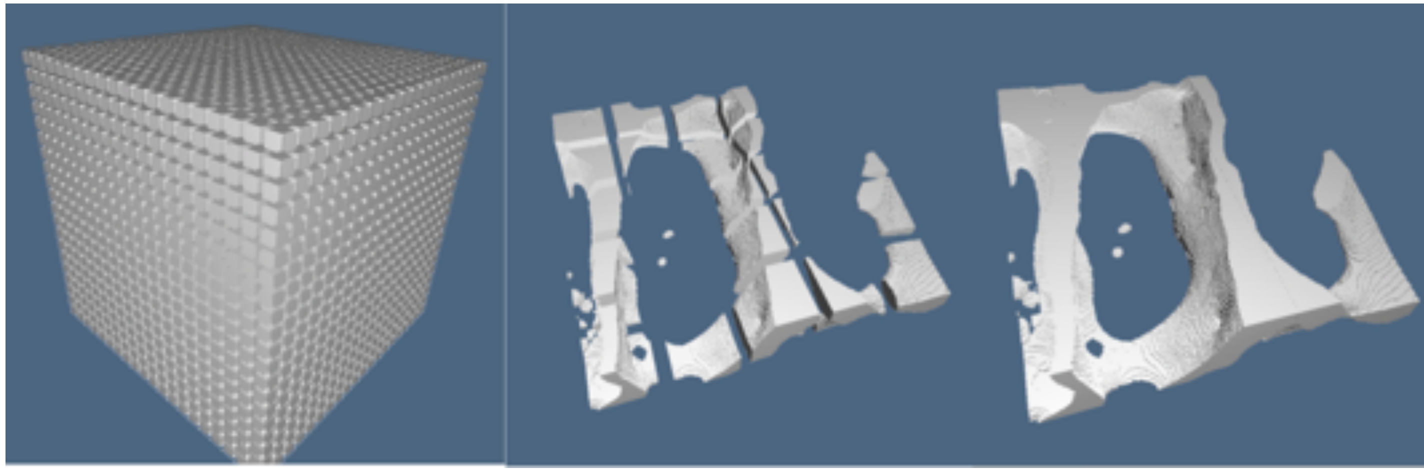
$$\partial_3^B : C_3(B) \to C_2(B)$$
$$\partial_2^S : C_2(S) \to C_1(S),$$

where B is a block of the 3-image,
i.e. a 3-array of voxels,
whereas S is the boundary surface of a 3-chain inside B:
i.e. a 2-chain of voxel faces,

$$S = \partial_3^B Vol, \quad Vol \in C_3(B)$$

Algorithm: *map_image_2_b-rep*

Extract boundary models from 3D "bricks" of voxels



```
function MAP_IMAGE_2_B-REP(imageArray, threshold, const ∂3(256^3) )
    n, m, p ← shape(imageArray)
    bricks ← split(imageArray) into subimages of shape(256^3)
    for brick ∈ bricks do
        brick ← brick.extract_B-Rep( threshold )
        brick ← brick.remove( frontier )
        models.extend( brick.simplify_B-Rep( par ) )
    end for
    return join([ m for m ∈ models ])
end function
```

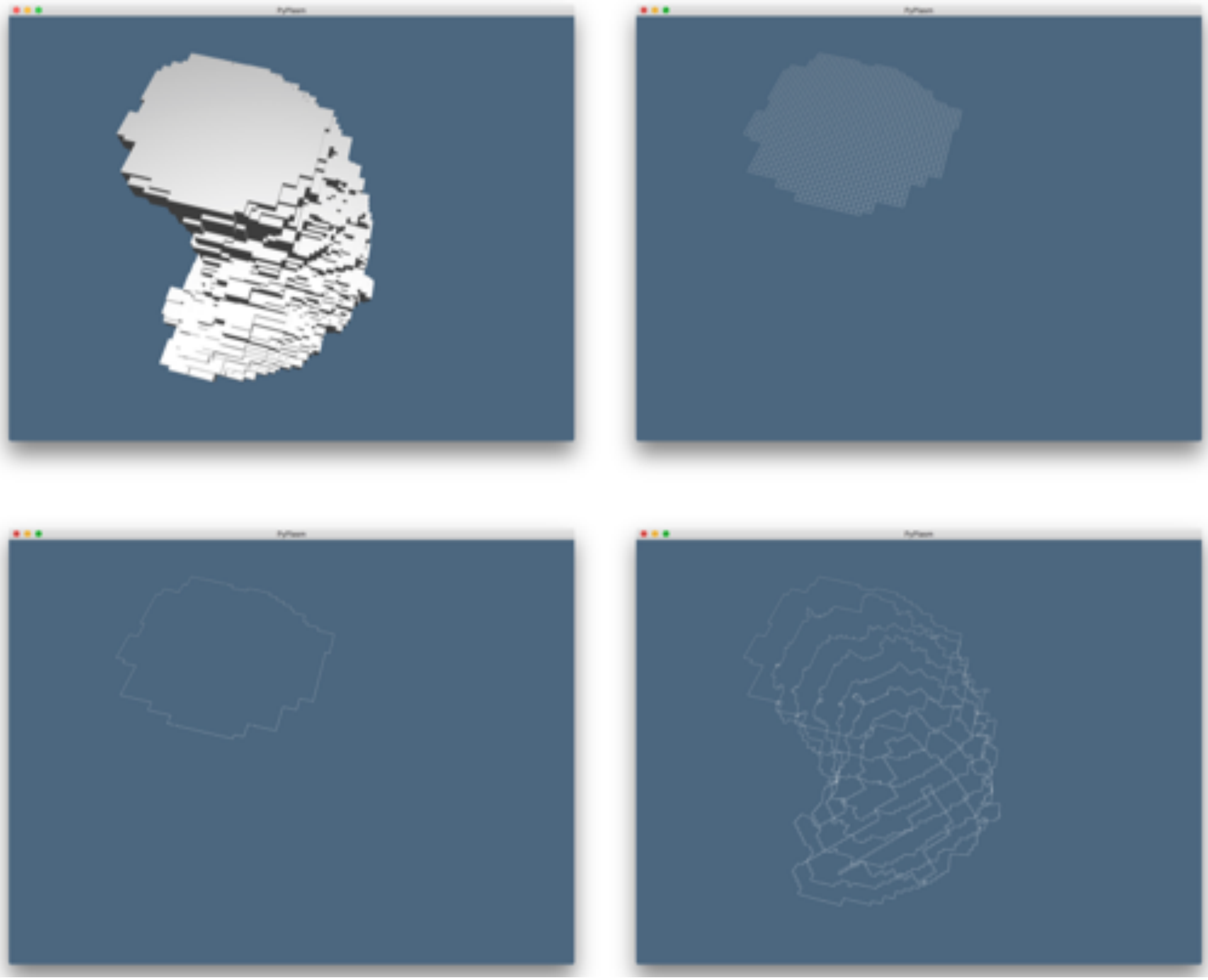
Algorithm: *simplify-B-Rep*

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models

function $\textsc{simplify\_B-Rep}(\ models, distance\ )$
    for all $LAR\_model \in models$ **do**
        $FV, EV \leftarrow LAR\_model$
        $\partial_2 \leftarrow compute\_boundary\_operator(\ FV, EV\ )$
        $S_0 \leftarrow box(FV)$ ▷ open surface (2-chain in $\mathcal{C}_2$)
        $C_0 \leftarrow \partial_2\, S_0$ ▷ boundary cycle (1-chain in $\mathcal{C}_1$)
        $\mathcal{C} = simplify(C_0, distance)$
        while $\partial_2\, S_k \neq 0$ **do** ▷ while S_k is open
            $S_{k+1} \leftarrow S_k + \delta_1\,(\partial_2\, S_k)$ ▷ updated surface
            $C_{k+1} \leftarrow C_k + \partial_2\, S_{k+1}$ ▷ new boundary cycle
            if $(k+1) \mod distance = 0$ **then**
                $\mathcal{C}_{k+1} = simplify(C_{k+1}, distance)$
            end if
        end while
    end for
    return $join([\ m$ *for* $m \in models\ ])$
end function

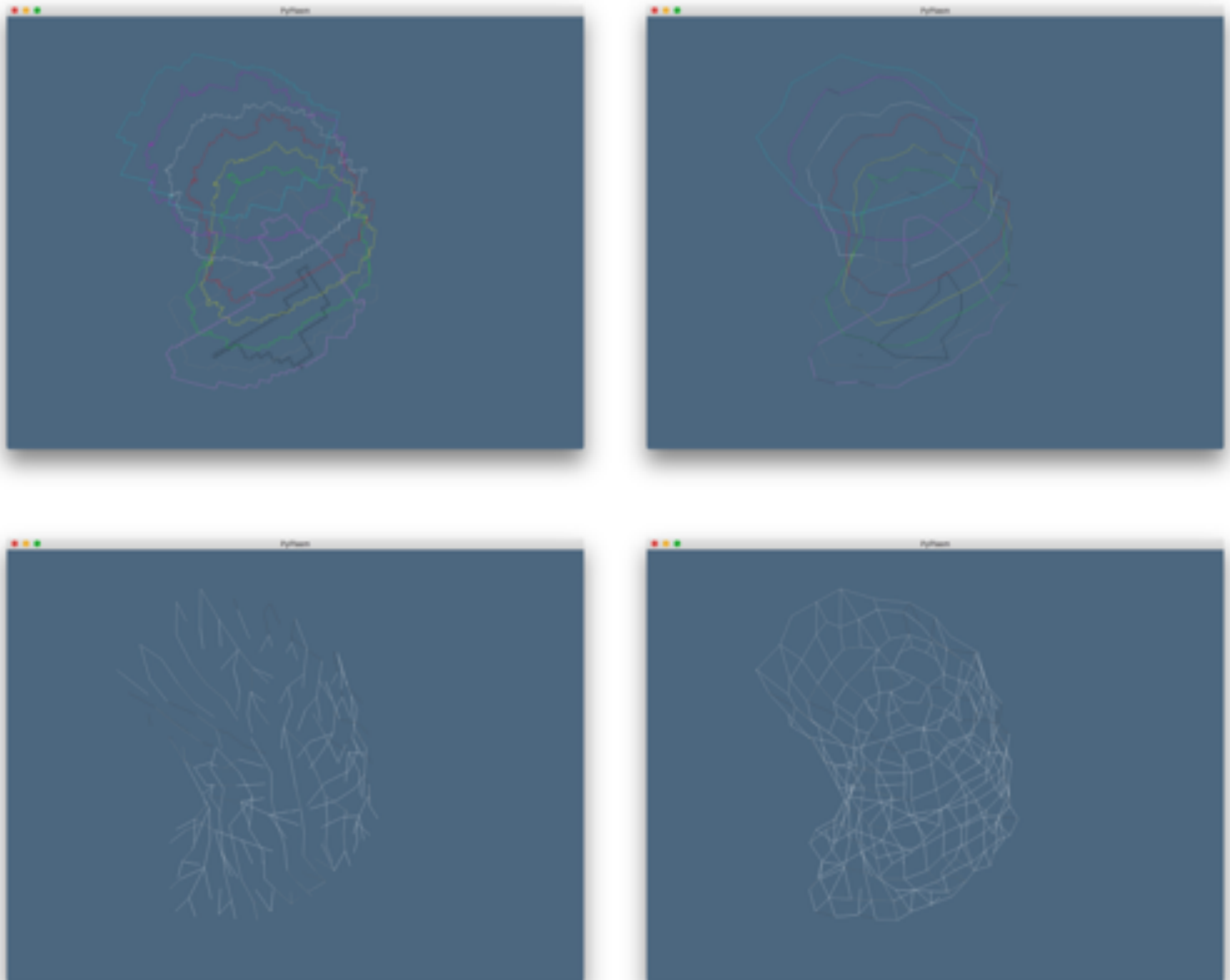
Algorithm: *example* 1

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models



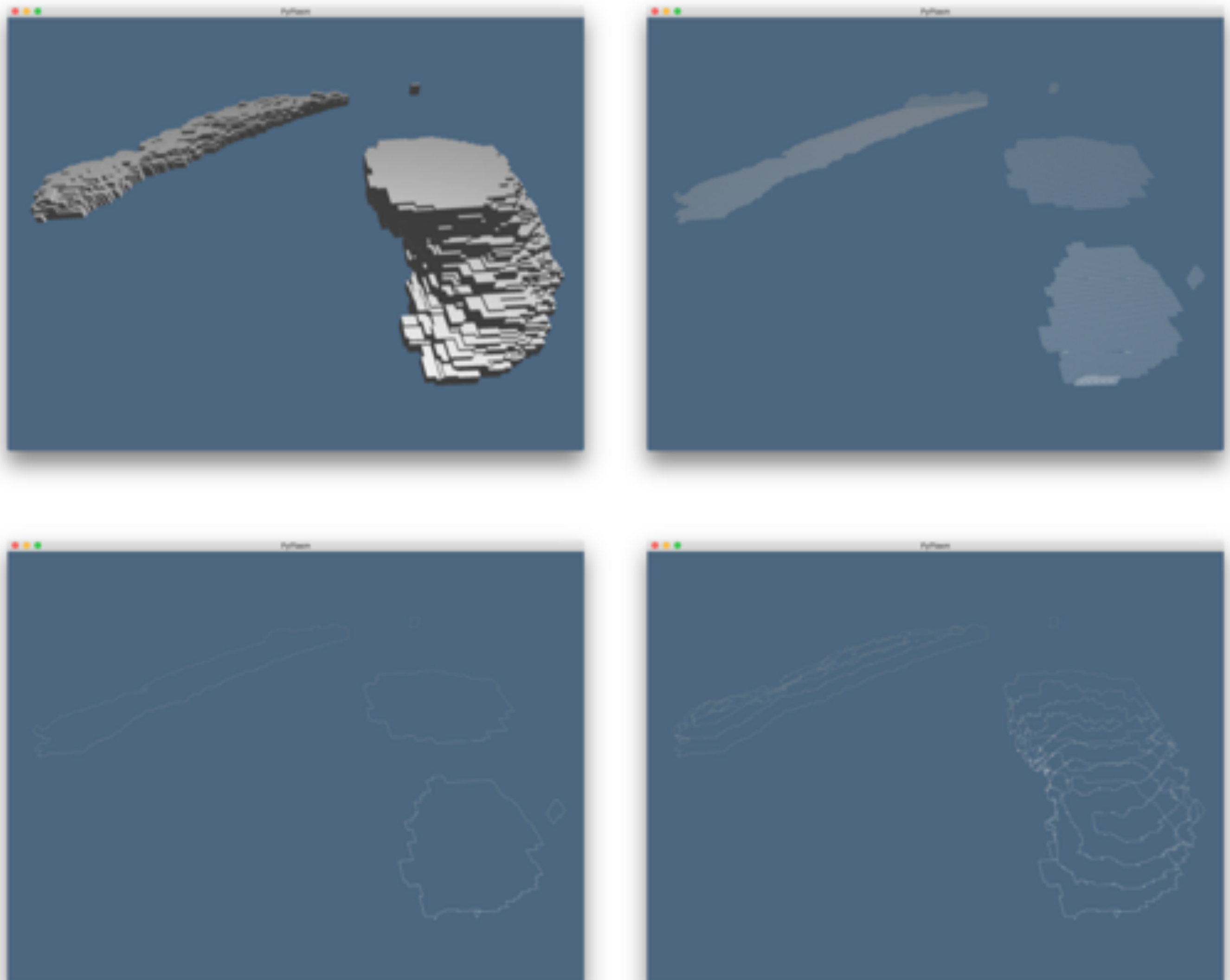
Algorithm: *example* 1

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models



Algorithm: *example* 2

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models

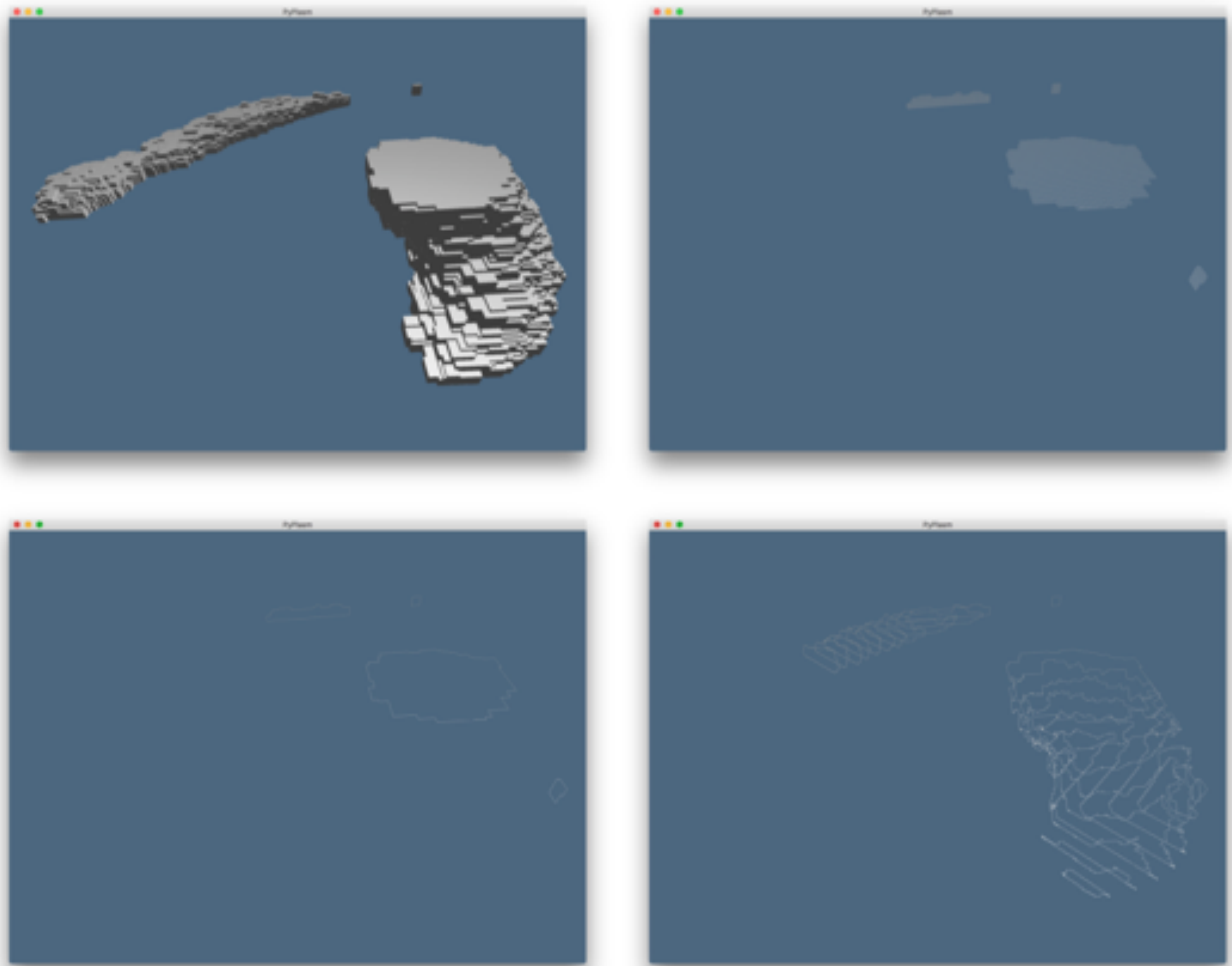


Algorithm: *example* 2

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models

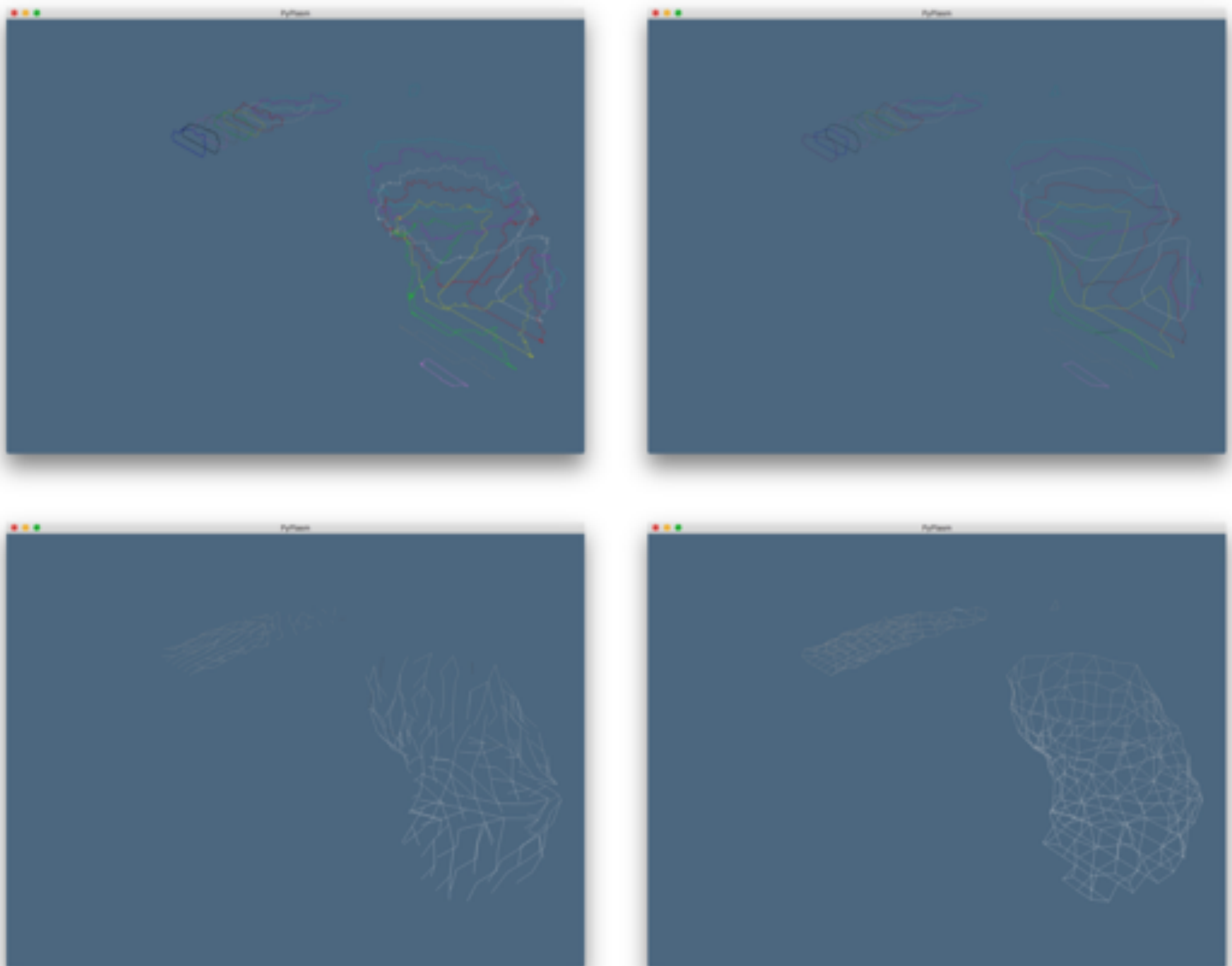
Algorithm: *example* 3

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models



Algorithm: *example* 3

Extract "intrinsic" curves and Generate homologically equivalent & strongly reduced models



Next steps

Parallel MAP implementation

- using C++ with `parallel sparse matrix` libraries CombBLAS/GraphBLAS

Apply MAP to new neuron data

- get from neurobiologists at max resolution

Complete SIMPLIFY implementation

- and port to CombBLAS/GraphBLAS

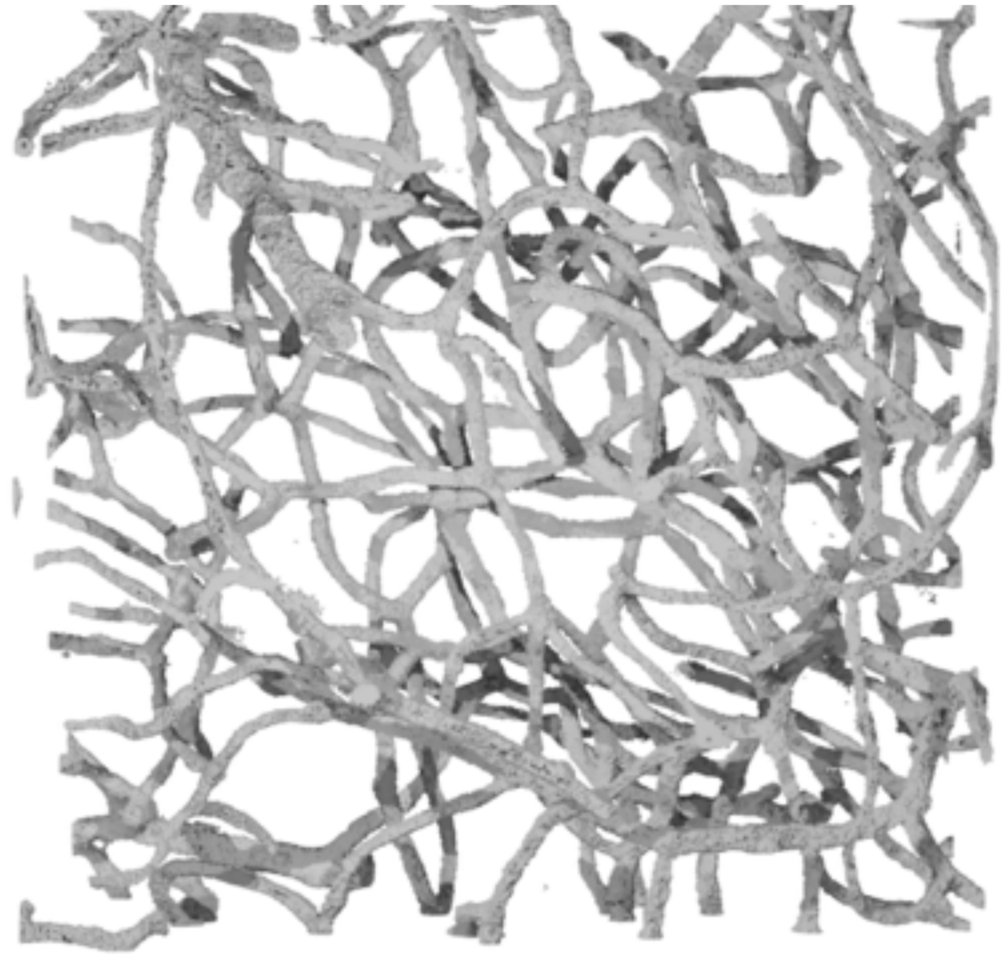
Integrate the method as ViSUS add-on

- to give the biologist an interactive tool

Conclusion

(Partial) results obtained and further advances

- Experiments with this method where quite successful.
- Will provide first real solid-medels (B-reps) of both neurons and microvessels in brain tissue allowing engineering simulations
- Further demonstration of LAR expressive power, ranging from graphics, to meshes, to images, with bot standard (simplicial, cuboidal, convex) and non standard (puntured) cellular decompositions
- Going to implement with parallel libraries for sparse matrices and graphs



Topologically exact solid model of microvessels between neurons

Thanks for your attention

QUESTIONS ?