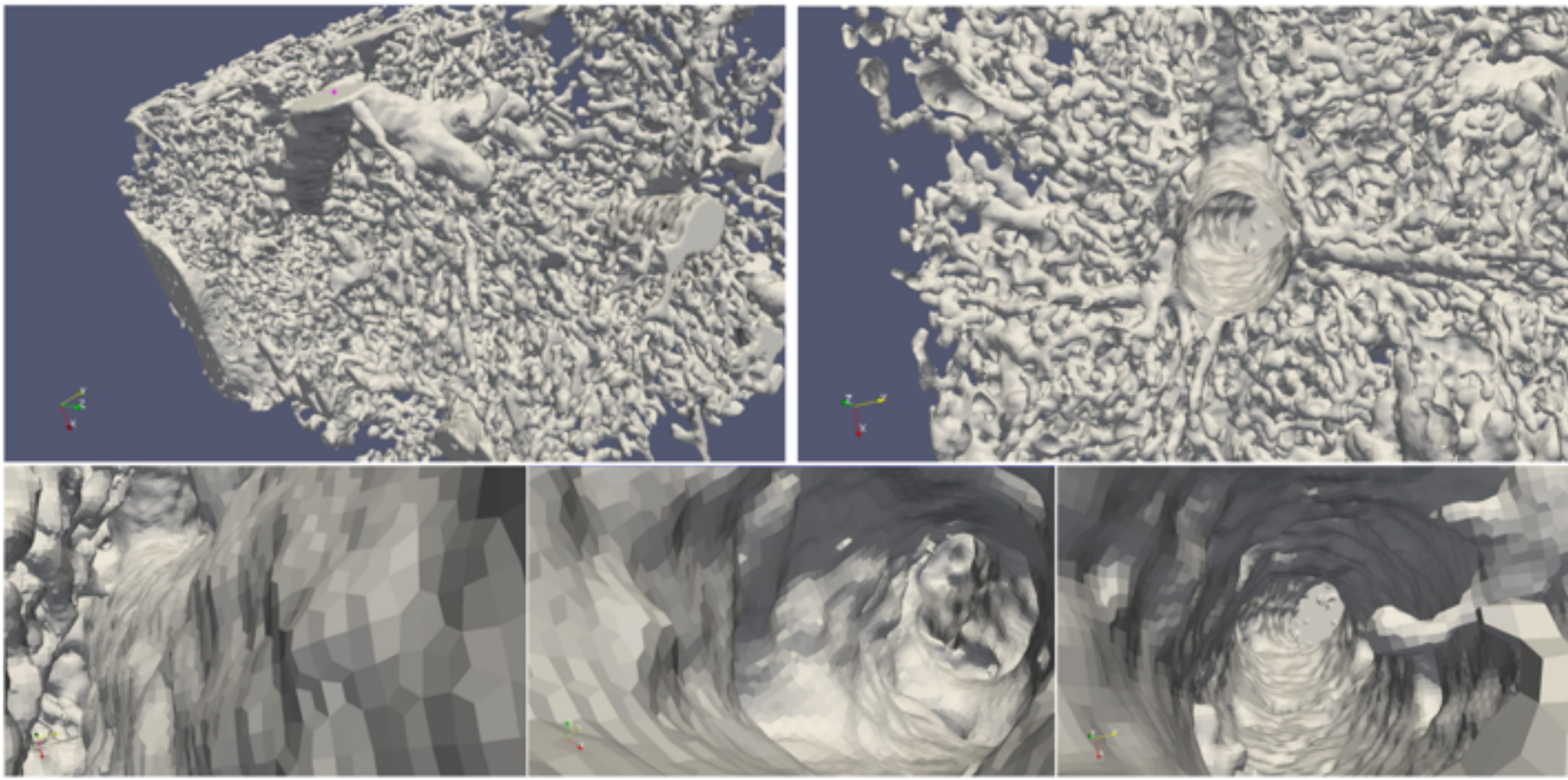
CAD Models from Medical Images Using LAR

CAD'15

London, UK



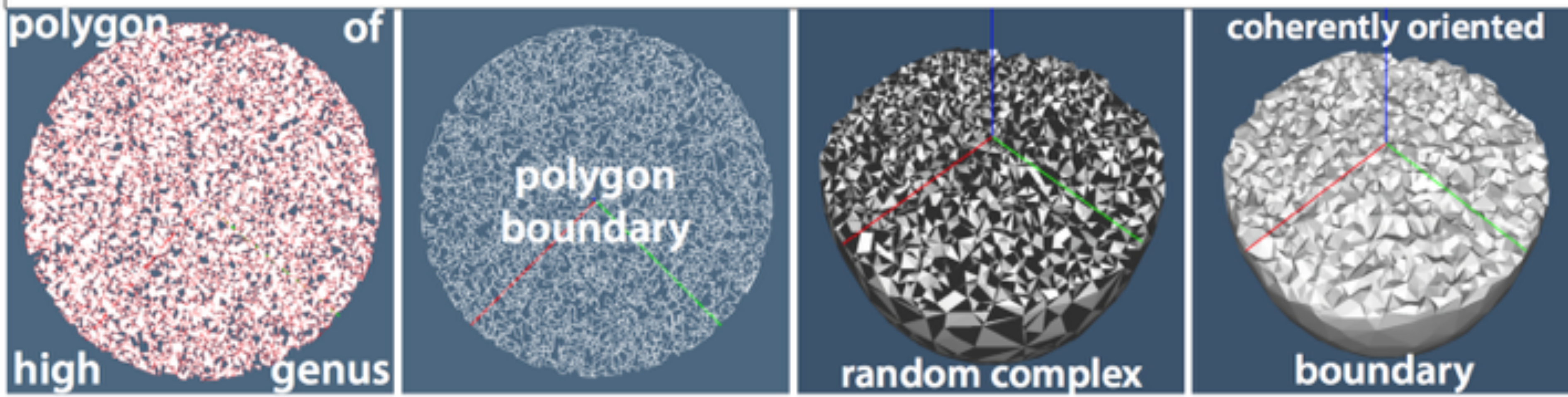
Content

- Linear Algebraic Representation
- Basic Definitions and Examples
- Some operations and algorithms
- Lar of images
- Extraction of the liver portal vein system
- Project aims & Conclusion

Linear Algebraic Representation

If there is anything like a unifying aesthetic principle in mathematics, it is this: simple is beautiful.
"A Mathematician's Lament", **Paul Lockhart**

Simplicity is the ultimate sophistication. **Leonardo Da Vinci**
Numquam ponenda est pluralitas sine necessitate.
(paraphrased as Occam's razor) **William of Occam**



Models: (co)chain complexes $\rightarrow$ Reprs: sparse binary matrices

Chains and Cochains

$C^0, C^1, C^2, C^3 \equiv \mathcal{V}, \mathcal{E}, \mathcal{F}, \mathcal{P}$ $\Delta_p \equiv \mathit{Laplacian}$ $\delta^0, \delta^1, \delta^2 \equiv$ **grad**, **curl**, div

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)

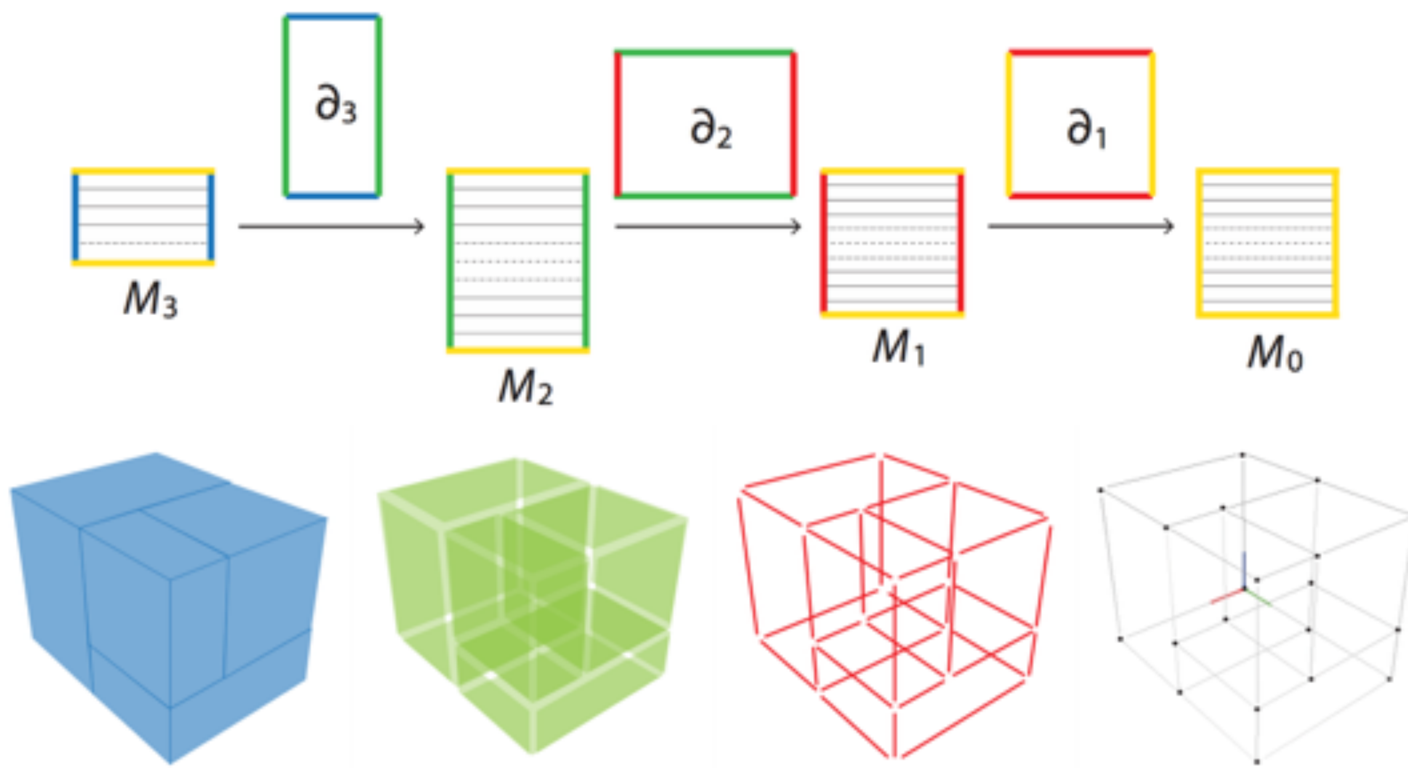
$$\boxed{\delta^p = \partial_{p+1}^\top} \qquad \boxed{\Delta_p = (\delta^p)^\top \delta^p + (\delta^{p-1})^\top \delta^{p-1}}$$

$$C^3 \xleftarrow{\text{div}} C^2 \xleftarrow{\textbf{curl}} C^1 \xleftarrow{\textbf{grad}} C^0$$

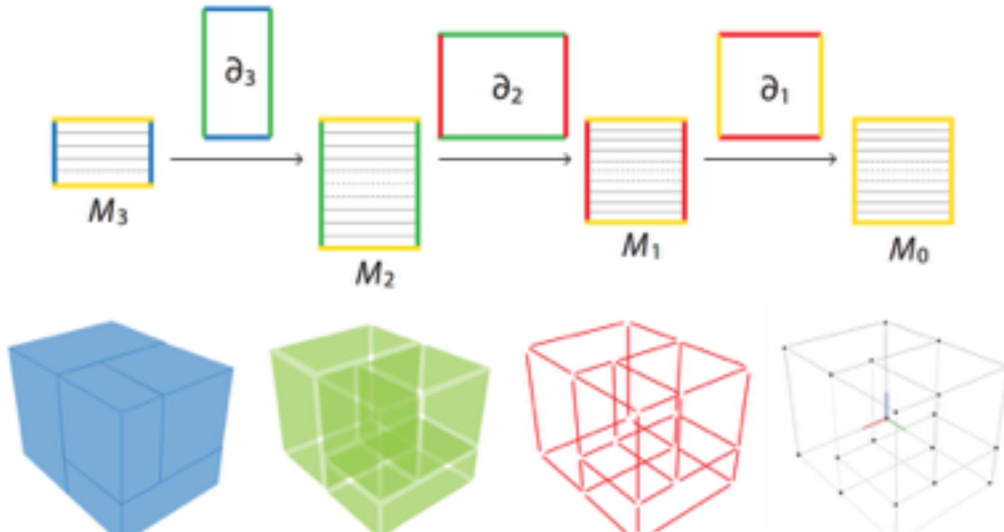
Chains and Cochains

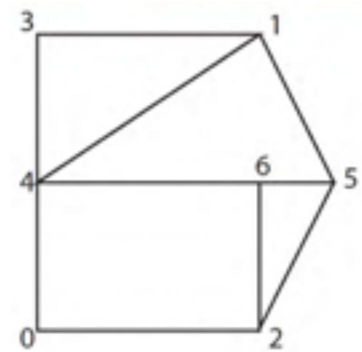
Sequence of linear spaces (over $\mathbb{Z}_2$) of d-cell **subsets**

Unit d-chains (single d-cell subsets), give the standard bases (M_d rows) of d-chain spaces



Binary Compressed Representation





n-cells by their vertices:

[[0,2,4,6],
[1,4,5,6],
[1,5,6],
[2,5,6]]

Remark (Input and **long-term storage**)

$space(\mathrm{LAR}) = |FV| = 2|E|$!!!

Remark (**Full** topology representation)

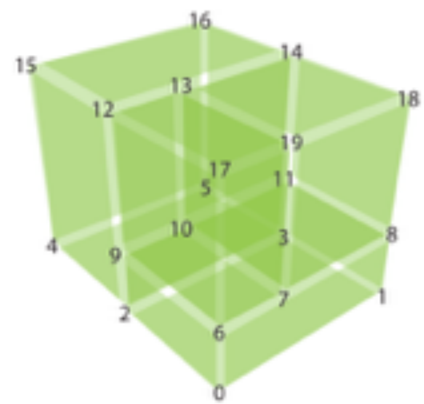
$|VE| + |VF| = 4|E|$

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



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

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

Compressed Sparse Row

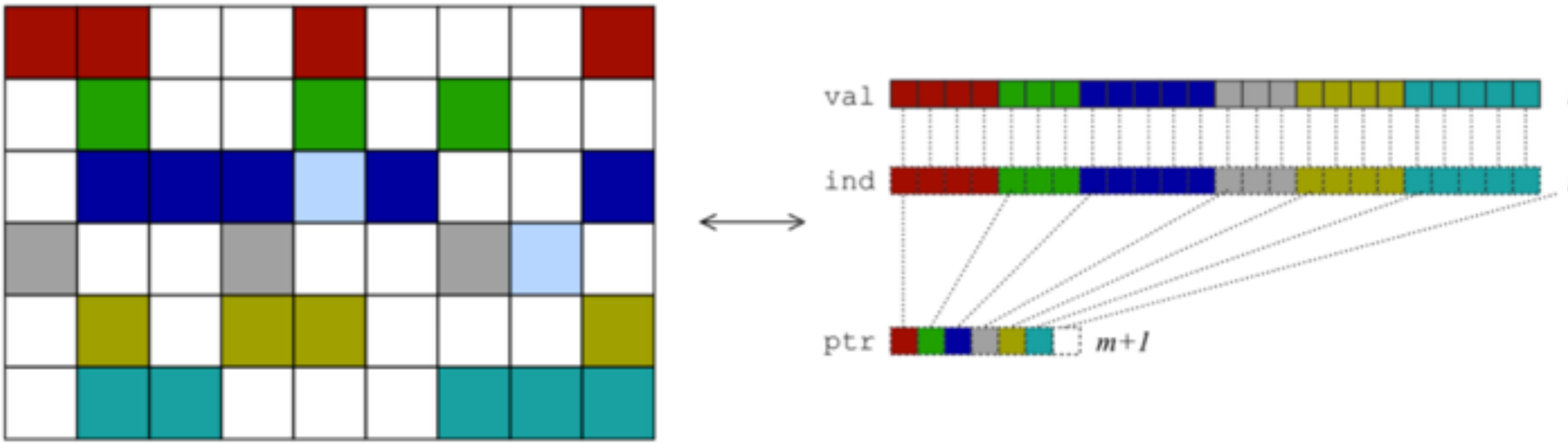


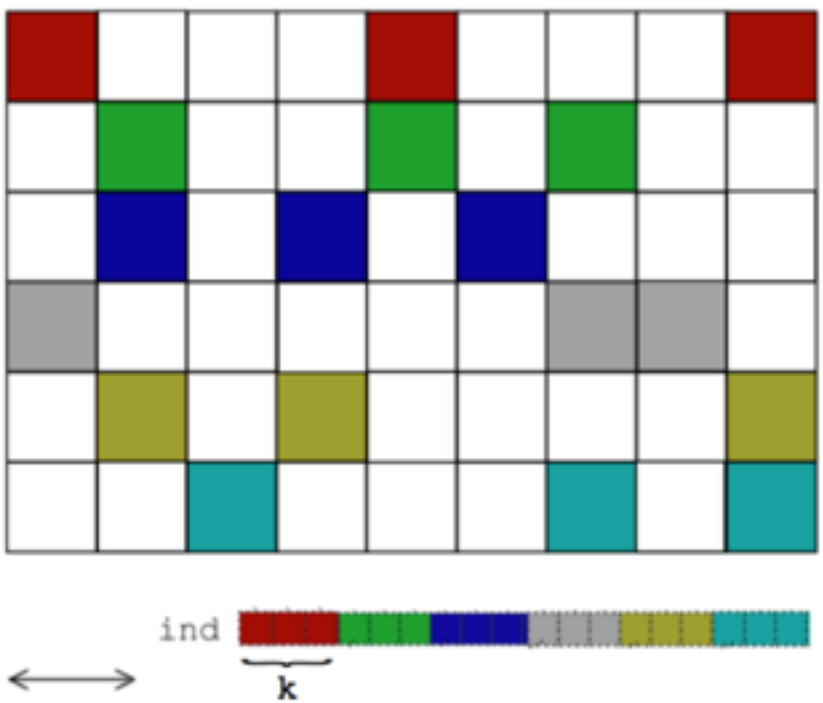
image from

Samuel Williams, Leonid Oliker, Richard Vuduc, John Shalf, Katherine Yelick, and James Demmel, Optimization of sparse matrix-vector multiplication on emerging multicore platforms, Proceedings of the 2007 ACM/IEEE conference on Supercomputing (New York, NY, USA), SC '07, ACM, 2007, pp. 38:1–38:12.

Compressed Sparse Row

Amazingly compact storage of a **solid** model

REDUCED LAR



simplicial d-complexes: $k = d + 1$
cuboidal d-complexes: $k = 2^d$

Remark (Input and **long-term storage**)

$$space(\text{LAR}) = |FV| = 2|E| \ !!!$$

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

Boundary and Coboundary operator

(i) Compute $\mathrm{CSR}(M_{p-1}^{p}) := \mathrm{CSR}(M_{p-1})\,\mathrm{CSR}(M_p)^t$ as the product of sparse matrices.

(ii) Filtering procedure: for each $0 \leq i \leq k_{p-1} - 1$,

(a) compute the number $k := \sharp\mu_{p-1}^{i}$ of non-zero elements stored in row i of $\mathrm{CSR}(M_{p-1})$;

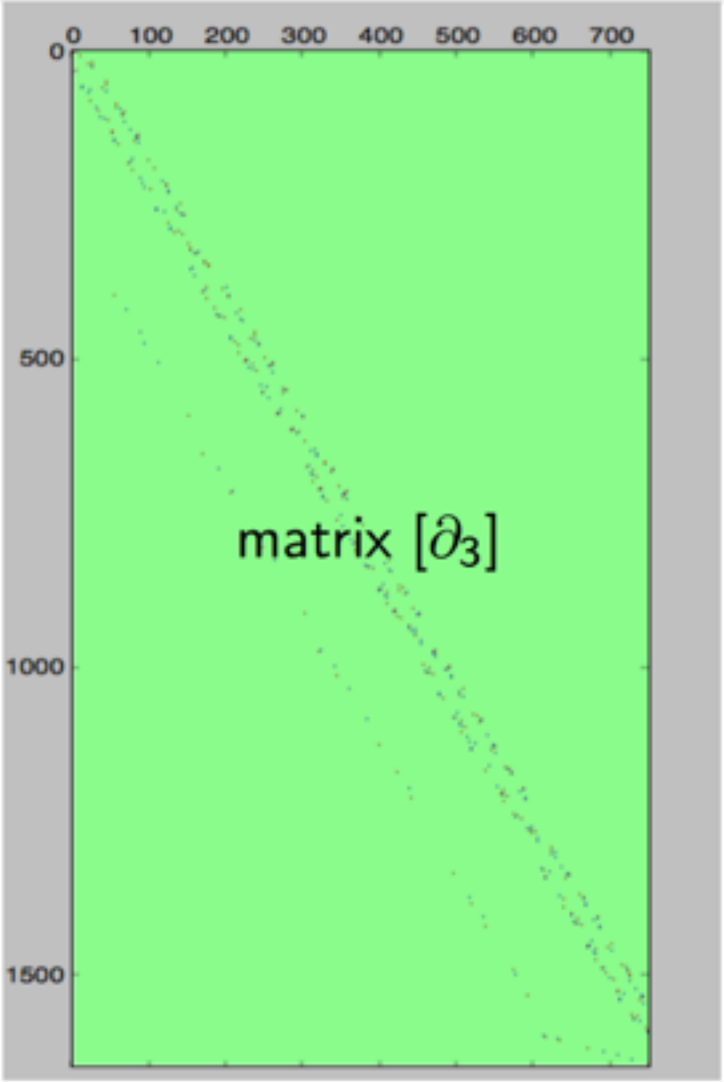
(b) for each $0 \leq j \leq k_p - 1$:

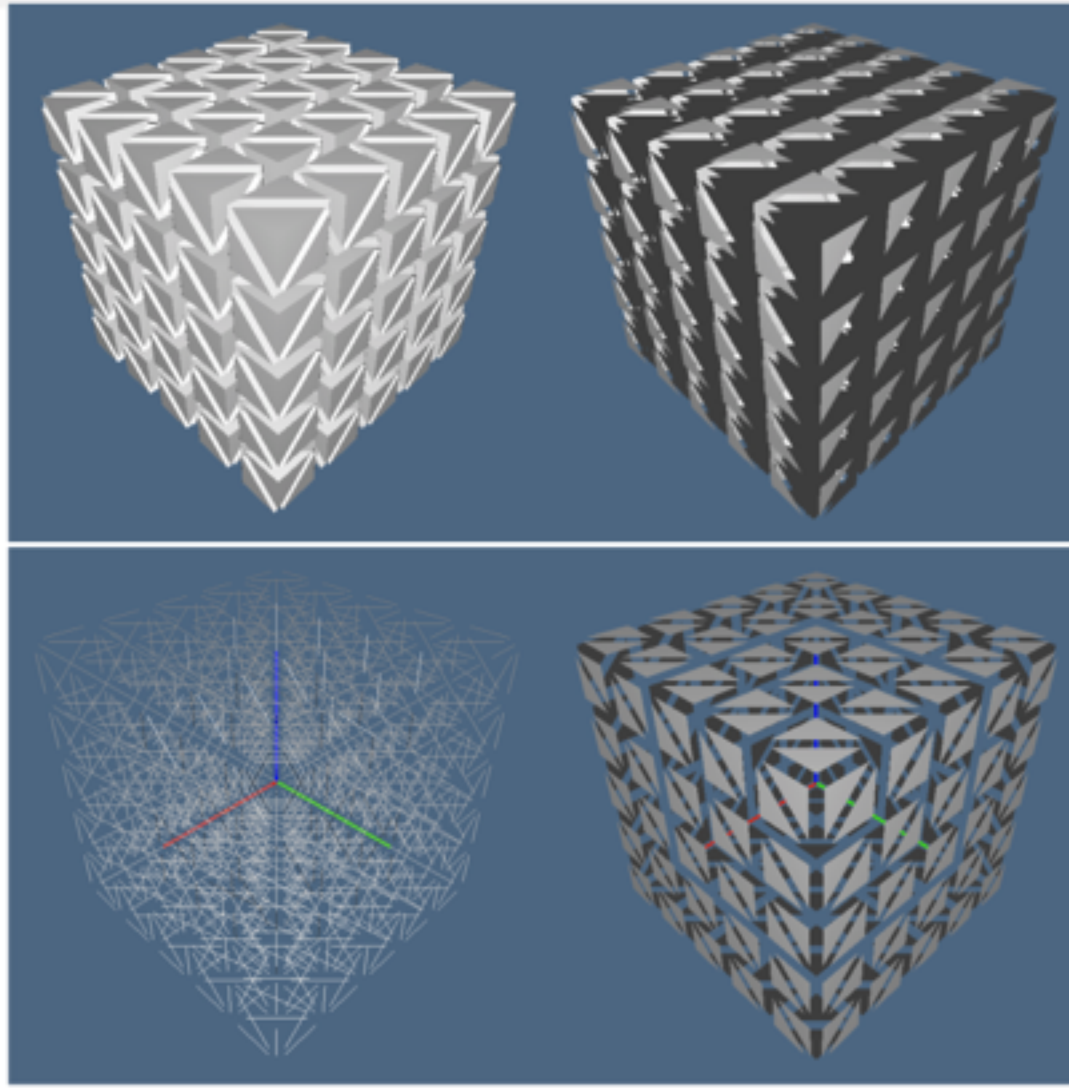
$$[\partial_p](i, j) := 1 \quad \text{if } M_{p-1}^{p}(i, j) = k \quad \text{else} \quad [\partial_p](i, j) := 0.$$

By duality, the same procedure may be used to compute the $(p-1)$-coboundary operator δ_{p-1} as the transpose of the boundary ∂_p.

Boundary extraction

Boundary computation via a single SpMV product: $[c_2] = [\partial_3]\,\mathbf{1}$

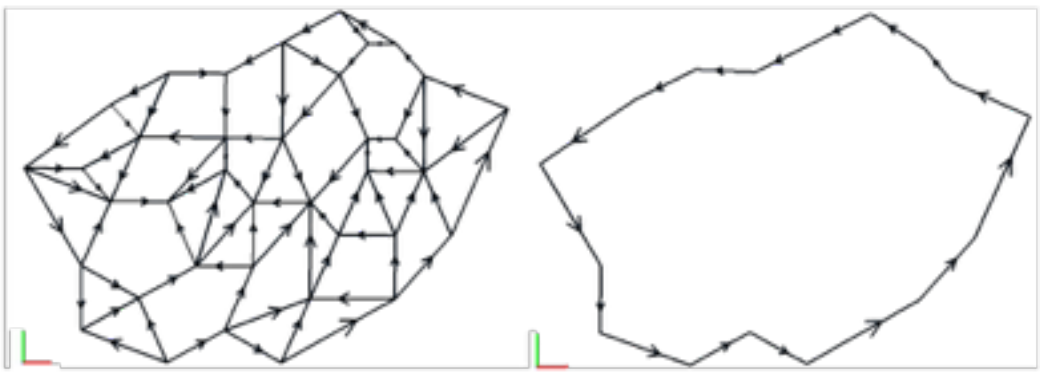


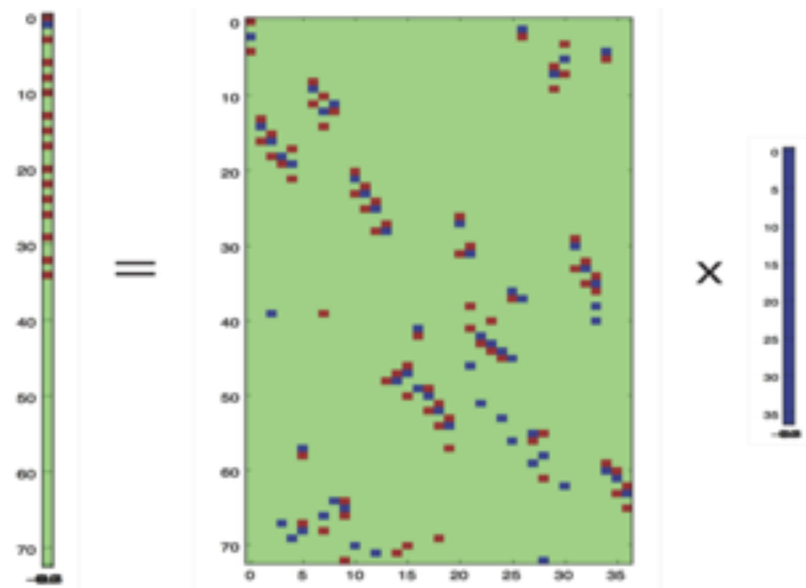


Boundary extraction

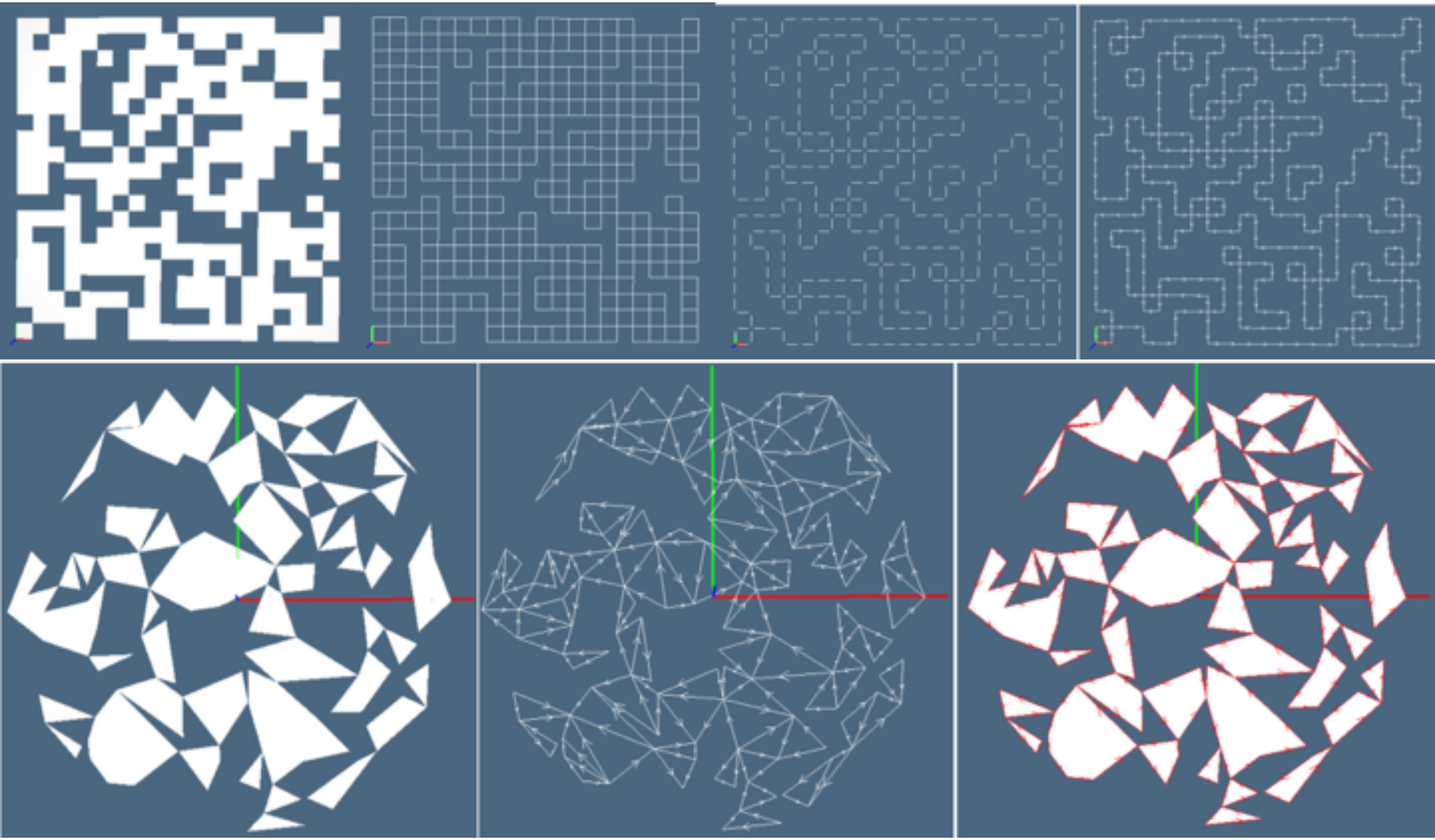
```
V = [[5.,0.],[7.,1.],[9.,0.],[13.,2.],[15.,4.],[17.,8.],[14.,9.],[13.,10.],[11.,11.],[9.,10.],[5.,9.],[7.,
9.],[3.,8.],[0.,6.],[2.,3.],[2.,1.],[8.,3.],[10.,2.],[13.,4.],[14.,6.],[13.,7.],[12.,10.],[11.,9.],[9.,7.],
[7.,7.],[4.,7.],[2.,6.],[3.,5.],[4.,2.],[6.,3.],[11.,4.],[12.,6.],[12.,7.],[10.,5.],[8.,5.],[7.,6.],[5.,5.]]

FV = [[0,1,16,28,29],[0,15,28],[1,2,17],[1,16,17,33],[2,3,17],[3,4,18,19],[3,17,18,30],[4,5,19],[5,6,19],
[6,7,20,21,22,32],[6,19,20],[7,8,21],[8,9,21,22],[9,11,23,24],[9,22,23],[10,11,24,25],[10,12,25],[12,13,25,
26],[13,14,27],[13,26,27],[14,15,28],[14,27,28,29,36],[16,29,34],[16,33,34],[17,30,33],[18,19,31],[18,30,
31],[19,20,31,32],[22,23,32,33],[23,24,34,35],[23,33,34],[24,25,27,36],[24,35,36],[25,26,27],[29,34,35],[29,
35,36],[30,31,32,33]]
```





Boundary extraction



Other ops: incidences

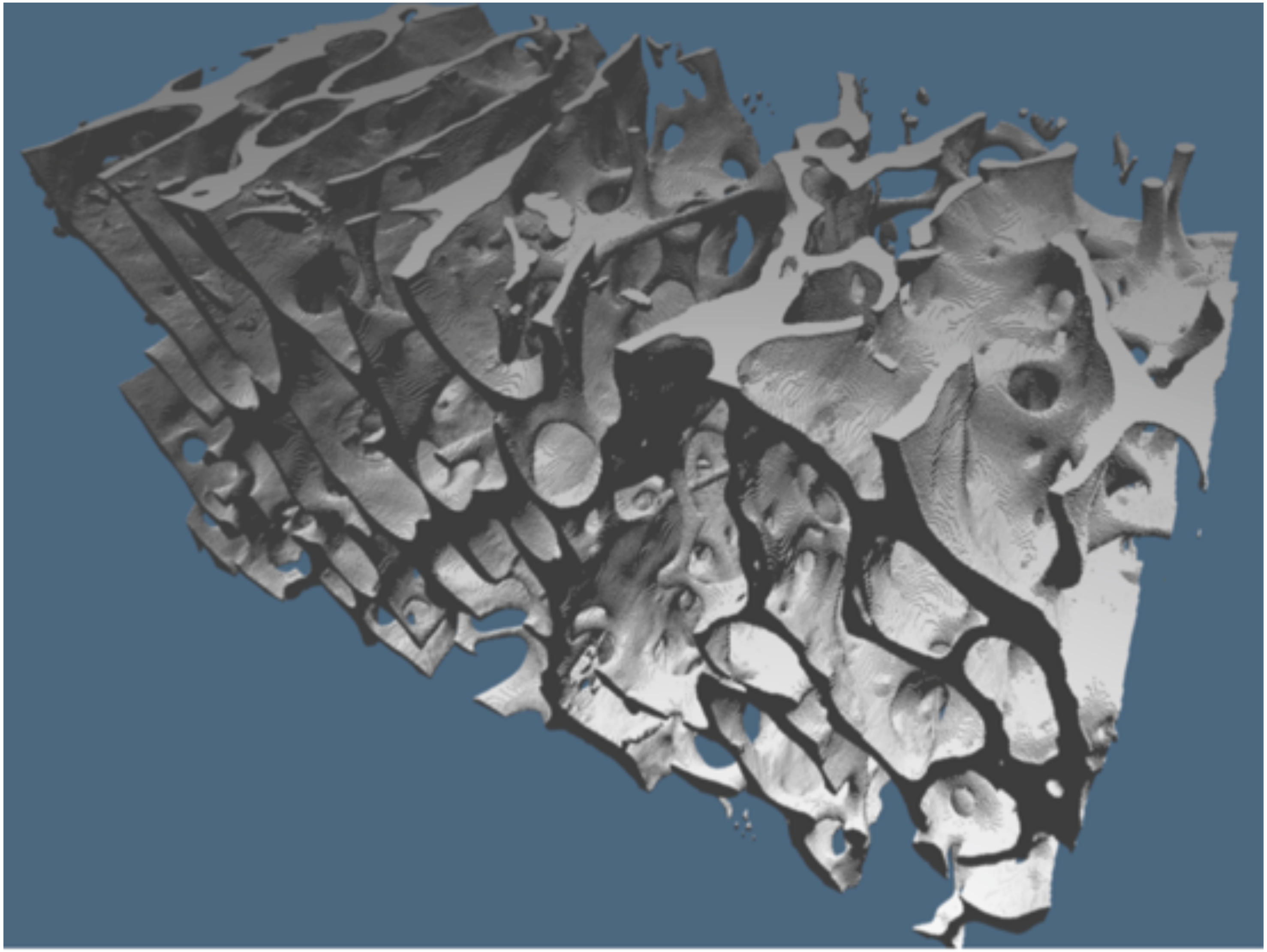
The chain *operators* corresponding to the incidence relations $VV \subset V \times V$, $VE \subset V \times E$, and $VF \subset V \times F$ are given below:

$$\mathcal{VV} : C_0 \to C_0, \qquad \mathcal{EV} : C_0 \to C_1, \qquad \mathcal{FV} : C_0 \to C_2;$$
$$\mathcal{VE} : C_1 \to C_0, \qquad \mathcal{EE} : C_1 \to C_1, \qquad \mathcal{FE} : C_1 \to C_2;$$
$$\mathcal{VF} : C_2 \to C_0, \qquad \mathcal{EF} : C_2 \to C_1, \qquad \mathcal{FF} : C_2 \to C_2.$$

The corresponding CSR matrices are readily computed:

$$\mathcal{VV} = \mathcal{VE} \circ \mathcal{EV} = \mathcal{EV}^{\top} \circ \mathcal{EV} \Rightarrow [\mathcal{VV}] = M_1^t M_1$$
$$\mathcal{VE} = \mathcal{EV}^{\top} \Rightarrow [\mathcal{VE}] = M_1^t$$
$$\mathcal{VF} = \mathcal{FV}^{\top} \Rightarrow [\mathcal{VF}] = M_2^t$$
$$\mathcal{EV} \quad [\mathcal{EV}] = M_1$$
$$\mathcal{EE} = \mathcal{EV} \circ \mathcal{VE} = \mathcal{EV} \circ \mathcal{EV}^{\top} \Rightarrow [\mathcal{EE}] = M_1 M_1^t$$
$$\mathcal{EF} = \mathcal{EV} \circ \mathcal{VF} = \mathcal{EV} \circ \mathcal{FV}^{\top} \Rightarrow [\mathcal{EF}] = M_1 M_2^t$$
$$\mathcal{FV} \quad [\mathcal{FV}] = M_2$$
$$\mathcal{FE} = \mathcal{FV} \circ \mathcal{VE} = \mathcal{FV} \circ \mathcal{EV}^{\top} \Rightarrow [\mathcal{FE}] = M_2 M_1^t$$
$$\mathcal{FF} = \mathcal{FV} \circ \mathcal{VF} = \mathcal{FV} \circ \mathcal{FV}^{\top} \Rightarrow [\mathcal{FF}] = M_2 M_2^t.$$

Imaging Applications



Mapping images to chains

Grid of hyper-cubes: Let $N_h := (0, 1, \dots, n_h - 1)$ be an ordered set of integers. Then $S := N_0 \times N_1 \times \cdots \times N_{d-1}$ is the set of *multi-indices* of elements of a d-image.

Definition 3 (d-image shape). The shape *of a d-image, with* $N = n_0 \times n_1 \times \cdots \times n_{d-1}$ *elements, called* voxels *or d-*cells, *is the ordered set* $(n_0, n_1, \cdots, n_{d-1})$.

Definition 4 (d-dimensional row-major order). *Given a d-image of shape* $S = (n_h)$, $(0 \le h \le d-1)$ *and number of d-cells* $N = \prod_h n_h$, *the* mapping $\mu : S \to \{0, 1, \dots, N-1\}$ *is a combination of multi-indices with integer weights* $(w_0, w_1, \dots, w_{d-2}, 1)$, *such that*:

$$(i_0, i_1, \dots, i_{d-1}) \mapsto i_0 w_0 + i_1 w_1 + \cdots + i_{d-1} w_{d-1},$$

with $w_k = n_{k+1} n_{k+2} \cdots n_{d-1}$ *for* $(0 \le k \le d-2)$.

Example 2 (LAR voxels). *The general hexahedral 3-cell (with 8 vertices), depending on three indices h, i, j* (page, row, column) *is obtained as the convex combination of the vertices indexed as integers via the mapping:*

$$\mu : N_0 \times N_1 \times N_2 \to M = \{m \mid m \in Z, 0 \le m \le n_0 n_1 n_2 - 1\},$$
$$(h, i, j) \mapsto h\, n_1 n_2 + i\, n_2 + j, \quad 0 \le h \le n_0, 0 \le i \le n_1, 0 \le j \le n_2,$$

where (n_0, n_1, n_2) *correspond respectively to the number of pages, rows, and columns of a 3-dimensional array, called the array* shape, and $N_h = \{0, 1, \dots, n_h - 1\}$, $(0 \le h \le 2)$. *Therefore we have, as LAR representation of a 3-cell (voxel):*

$$cell[\mu(i, j, k)] = [\mu(i, j, k), \mu(i+1, j, k), \mu(i, j+1, k), \mu(i, j, k+1), \mu(i+1, j+1, k),$$
$$\mu(i+1, j, k+1), \mu(i, j+1, k+1), \mu(i+1, j+1, k+1)]$$

Divide et Impera

Bottleneck of GPGPU: moving data from global to local memory

Solution: store the (sparse) $[\partial_3]$ of n^3 voxels in device Constant Memory, and move the (binary) coordinate vectors of chains in Private Memory

LAR on OpenCL

Morphological operators

linear operators $\mathtt{U} : C_d \to C_{d+1}$ and $\mathtt{D} : C_d \to C_{d-1}$

morphological gradient operator

$$\eta = (\oplus\gamma) - (\ominus\gamma)$$

d-chain γ being given as input,

(i) compute its boundary $\partial_d(\gamma)$;
(ii) extract the (d–2)-chain $\varepsilon = (\mathtt{D} \circ \partial_d)(\gamma)$;
(iii) single-out the (*d–1*)-chain returned from its coboundary $\delta_{d-2}(\varepsilon)$;
(iv) finally compute the *d*-chain $\eta := (\mathtt{U} \circ \delta_{d-2})(\varepsilon) \subset C_d$.

we obtain $\oplus\gamma = \gamma \cup \eta$, and $\ominus\gamma = \gamma - \eta$

Dilation and erosion

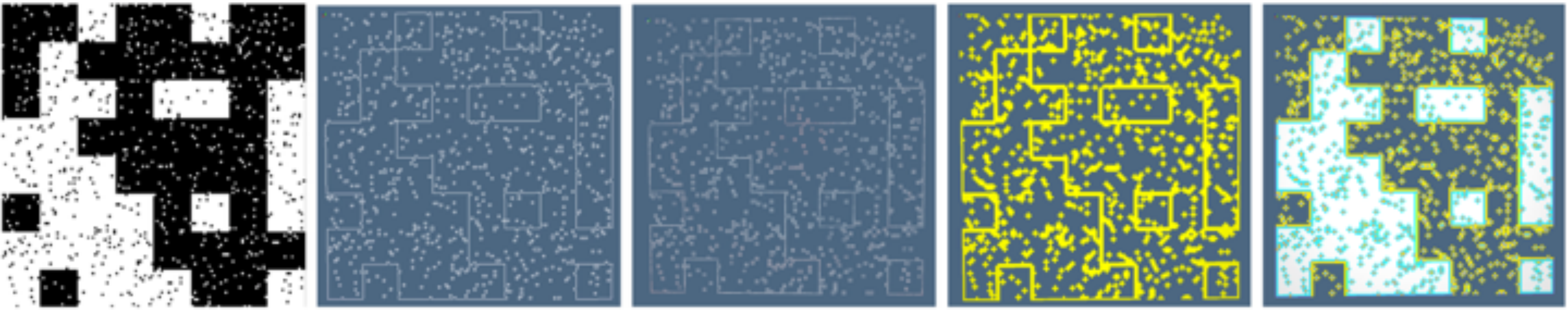


Figure 11: Subimage 128 × 128 example: (a) input chain $\gamma \in C_2$ (white) ; (b) extraction of the boundary chain $\beta = \partial_2(\gamma) \in C_1$; (c) chain $\eta = \mathtt{VE}(\beta) \in C_0$; (d) chain $\beta_2 = \mathtt{FV}(\eta) \in C_2 \equiv (\mathtt{FV} \circ \mathtt{VE} \circ \partial_2)(\gamma)$; (e) from the chain β_2 the dilation component $\beta_2 - \gamma$ (yellow), and the erosion component $\beta_2 \cap \gamma$ (cyan) are obtained.

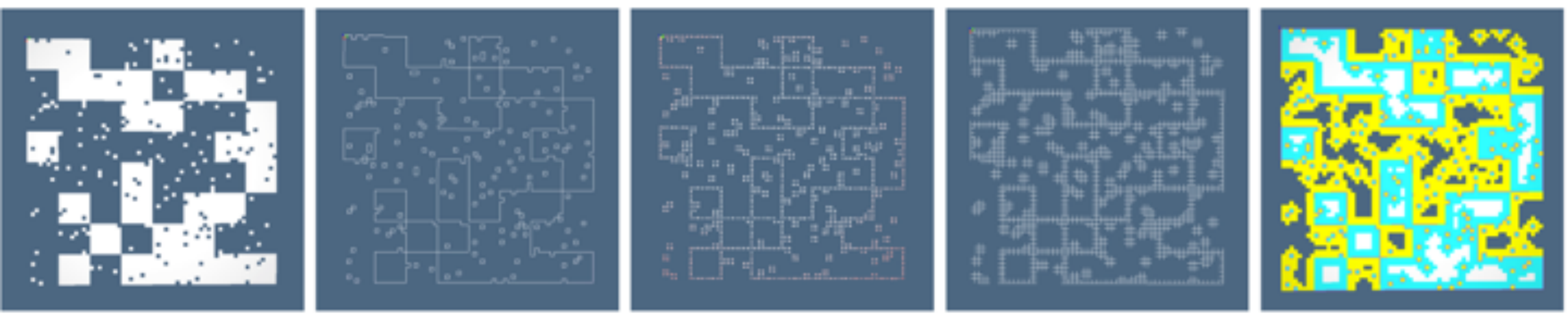
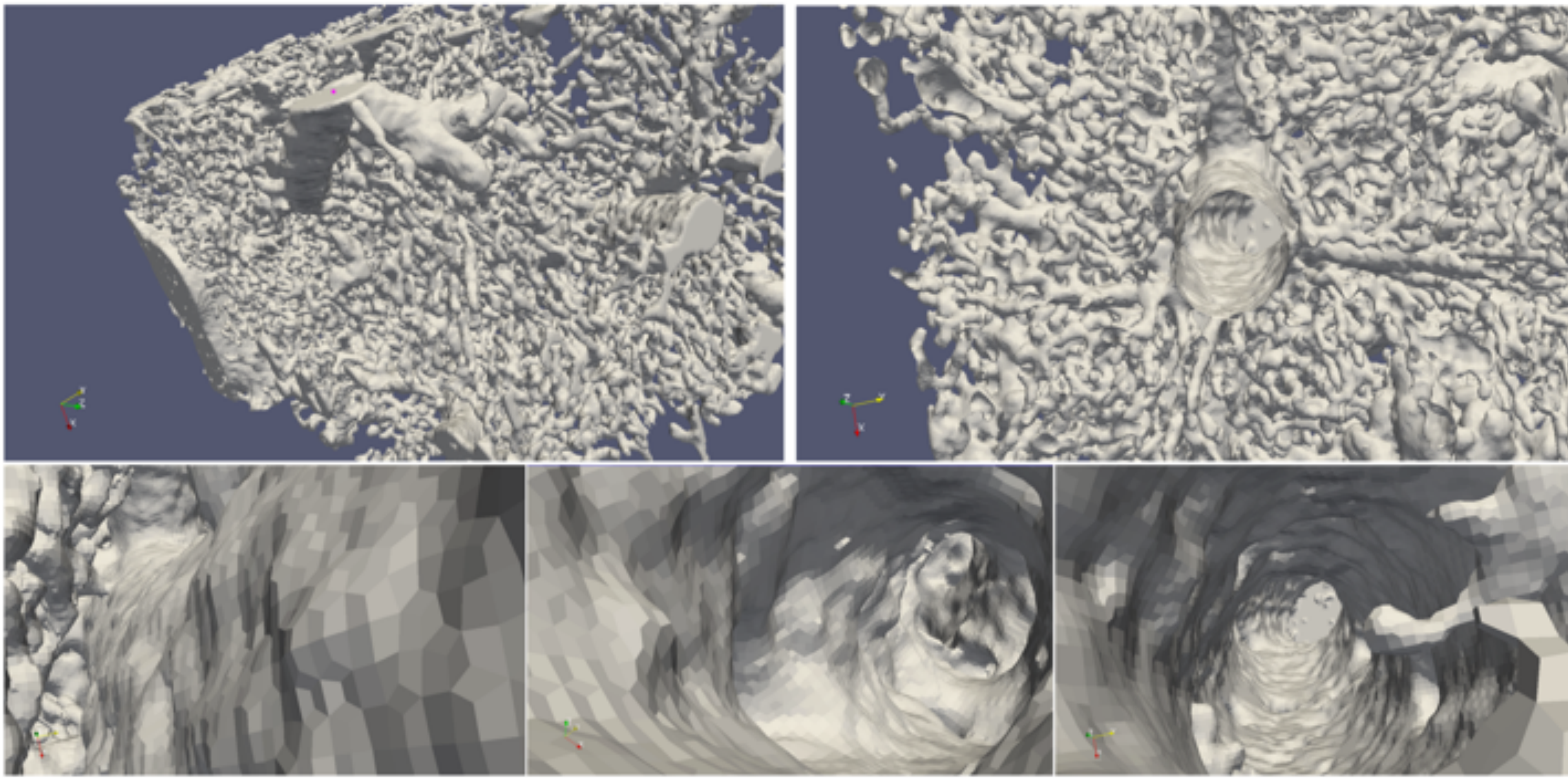
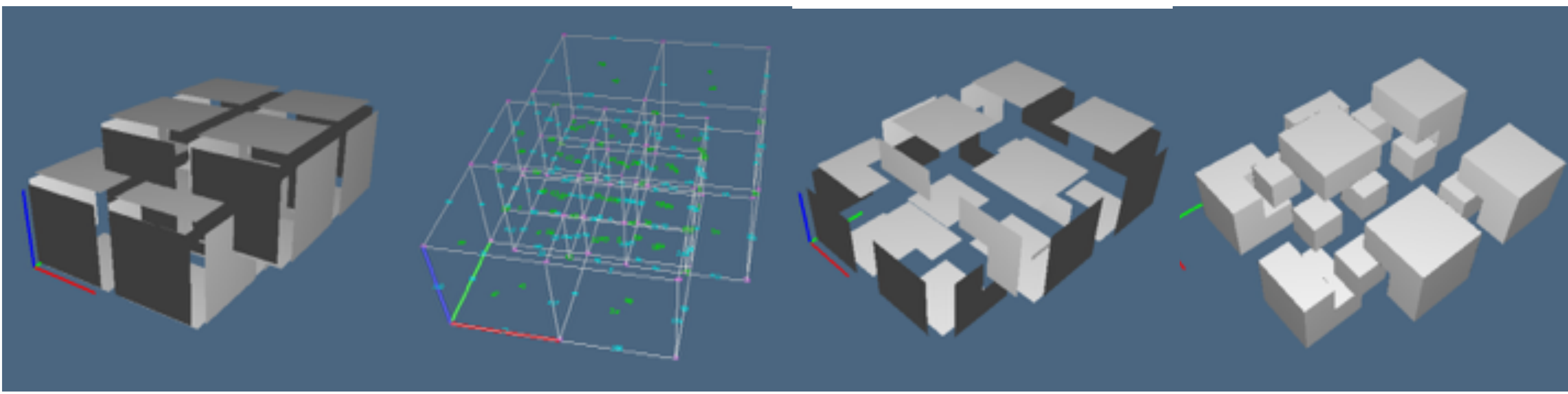


Figure 12: Subimage 64 × 64: (a) input chain $\gamma \in C_2$; (b) chain $\beta = \partial_2(\gamma) \in C_1$; (c) chain $(\mathtt{VE} \circ \partial_2)(\gamma) \in C_0$; (d) $(\mathtt{EV} \circ \mathtt{VE} \circ \partial_2)(\gamma) \in C_1$; (e) chain $\beta_2 = (\mathtt{FE} \circ \mathtt{EV} \circ \mathtt{VE} \circ \partial_2)(\gamma) \in C_2$, exhibiting the *dilation* component chain $\mathtt{DIL}(\beta_2)(\gamma) = \beta_2 - \gamma$ (yellow), and the *erosion* component chain $\mathtt{ERO}(\beta_2)(\gamma) = \beta_2 \cap \gamma$ (cyan).

Extracting boundary models from liver images



Next steps:
Boolean cochains on arrangements



Conclusion

dealing with Big Data and scalable architectures | **Google's map-reduce**

Emerging applications (e.g. space, nano & bio technology, medical 3D) require the convergence of shape synthesis and analysis from:

- computer imaging
- computer graphics
- computer-aided geometric design
- discrete meshing of domains
- physical simulations

The goals of unification, scalability, and massively parallel distributed computing

call for **rethinking the foundations** of geometric and topological computing

GOAL: 10^3 **times faster and** 10^4 **times bigger**