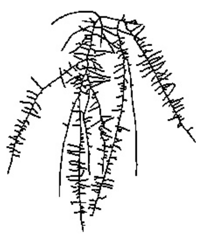
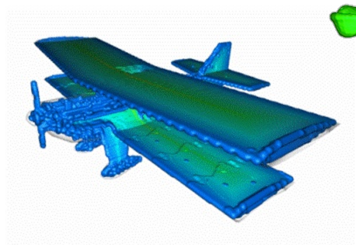
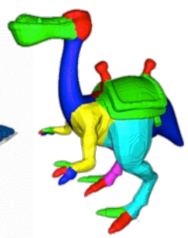
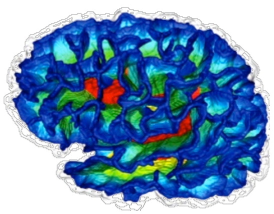
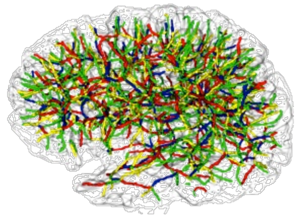
Scalable and Robust Computation of Medial Axes and Surfaces in 2D and 3D

State of the art


Alexandru Telea

Faculty of Mathematics and Natural Sciences
University of Groningen, the Netherlands

Some often-heard statements:

- Medial objects are in general
  - hard to compute
  - sensitive to noise
  - computable only for binary shapes
  - mainly useful for navigation, shape matching/analysis
- 3D medial surfaces are
  - very slow to compute
  - impractical for real-world applications

Some statements:

- Medial objects are in general
  - hard to compute
  - sensitive to noise
  - co
  - ma

Not entirely true!

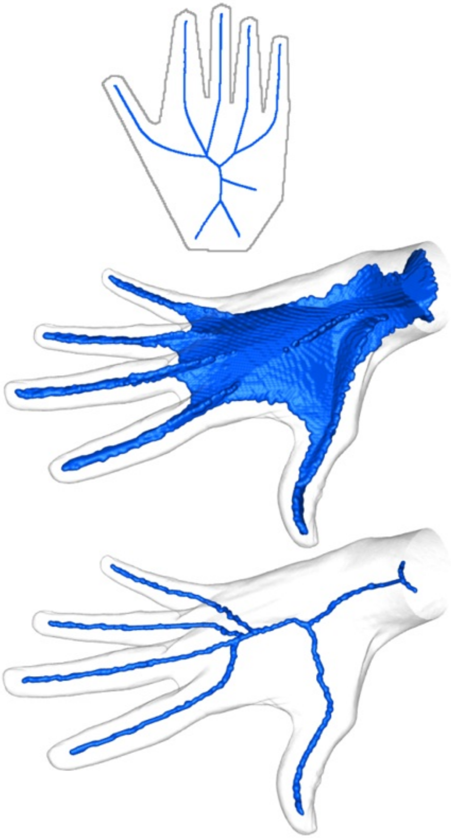
g/analysis

- 3D medial surfaces are
  - very slow to compute
  - impractical for real-world applications

Definition

$$S(\Omega) = \{x \in \Omega | \exists y \neq z \in \partial\Omega, \|x-y\| = \|x-z\| = DT_{\partial\Omega}(x)\}$$

- **2D skeleton:**
  - 1D structure
  - centers of maximally inscribed **discs**
- **3D surface skeleton:**
  - 2D structure
  - centers of maximally inscribed **balls**
- **3D curve skeleton**
  - 1D structure
  - no agreed formal definition

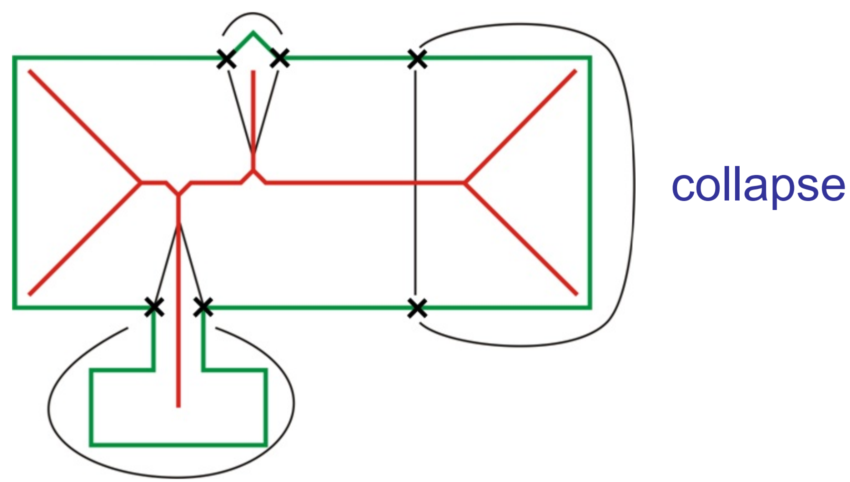


Global 2D detectors

Collapsed boundary metric

[Ogniewicz & Kubler '95]
[Falcao *et al.* '02]
[Telea & Van Wijk '02]

$$\rho(x \in S) = \max_{p,q \in F(x)} \left(\min |p\text{-}q|_{\partial\Omega}\right)$$



- monotonic and continuous on whole shape Ω
- leads to a robust, multiscale skeleton

Implementation

Augmented Fast Marching Method (AFMM)

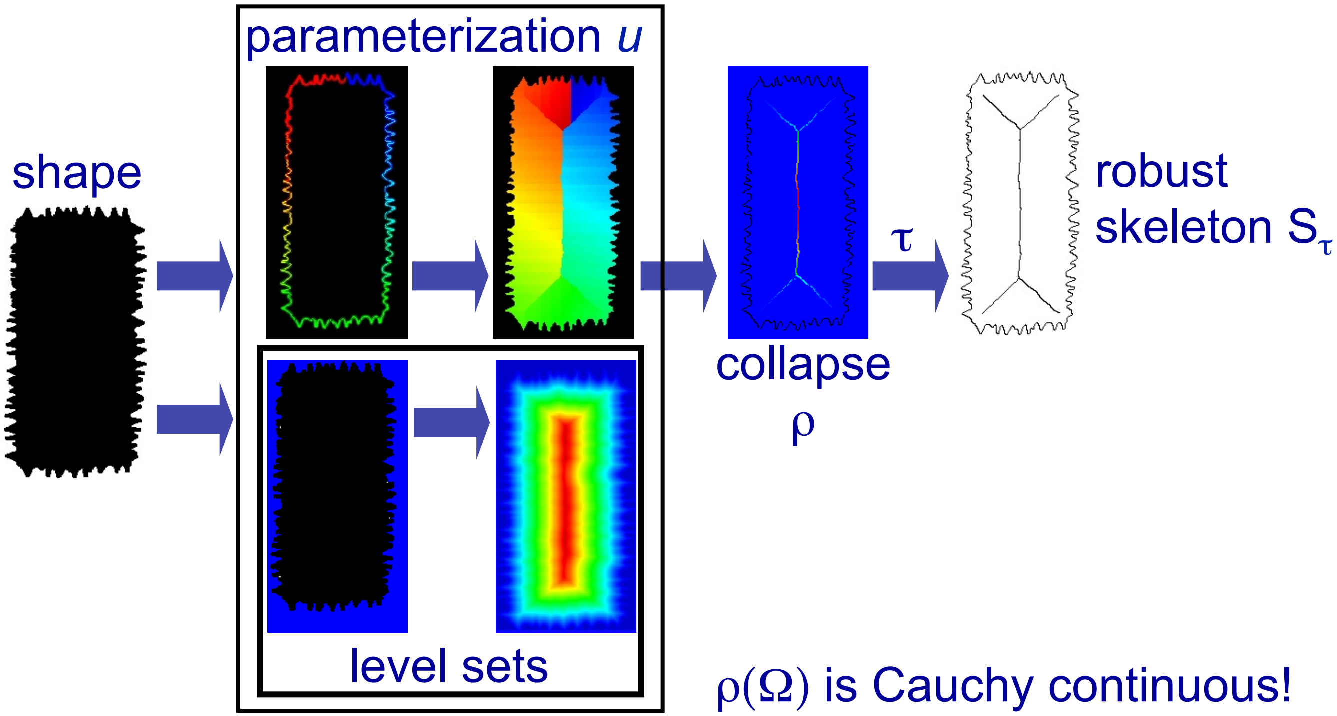
[Telea & Van Wijk'03]

- O(N log $N^{1/2}$), 2 fps @ 1024^2 pixels

CUDA Banding Algorithm

[Hurter *et al.*, TVCG'11]

- O(N), 500 fps @ 1024^2 pixels
- The **fastest**, **simplest**, **most robust** 2D skeletonization method out there

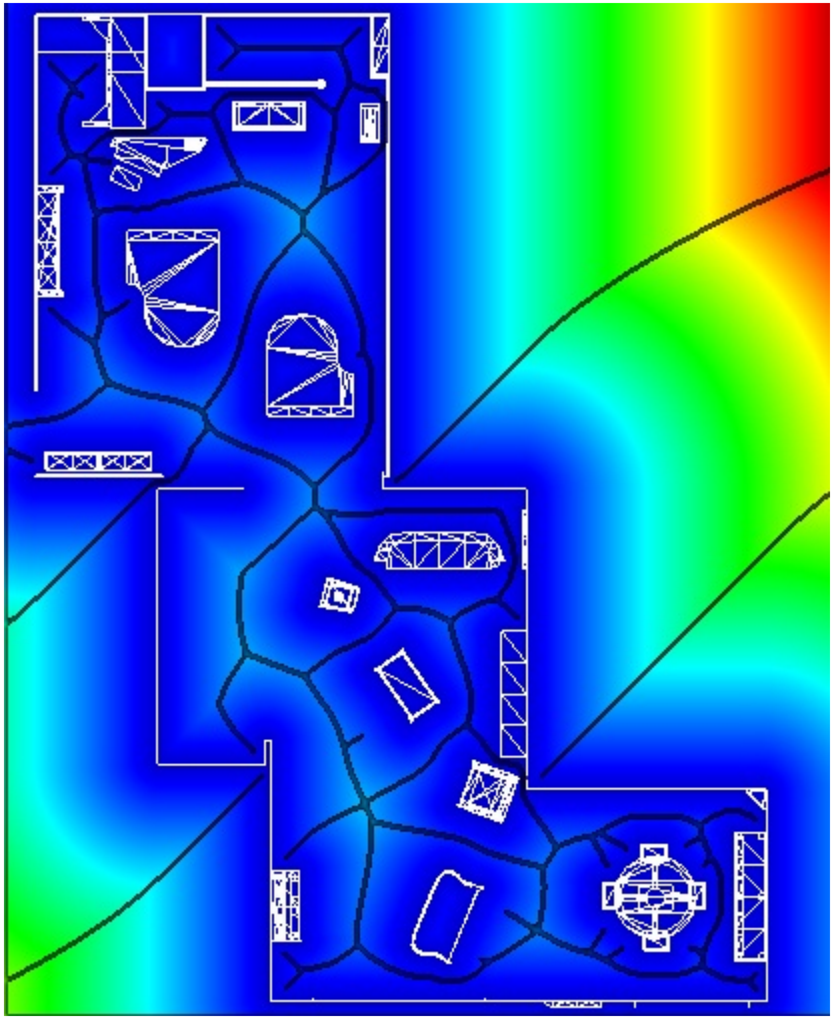


$\rho(\Omega)$ is Cauchy continuous!

Generalized Skeletons

[Strzodka & Telea '04]

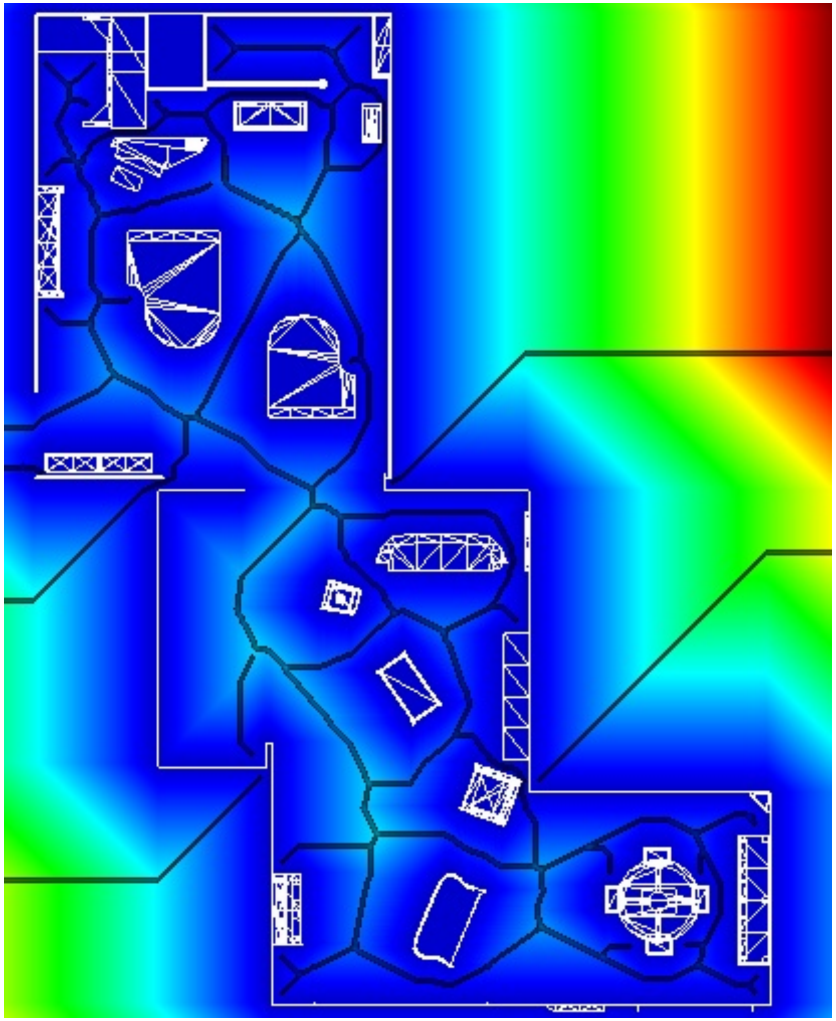
Euclidean Manhattan



Change the distance metric!

Generalized Skeletons

Euclidean Manhattan

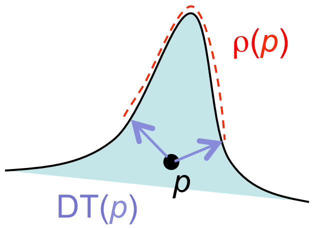


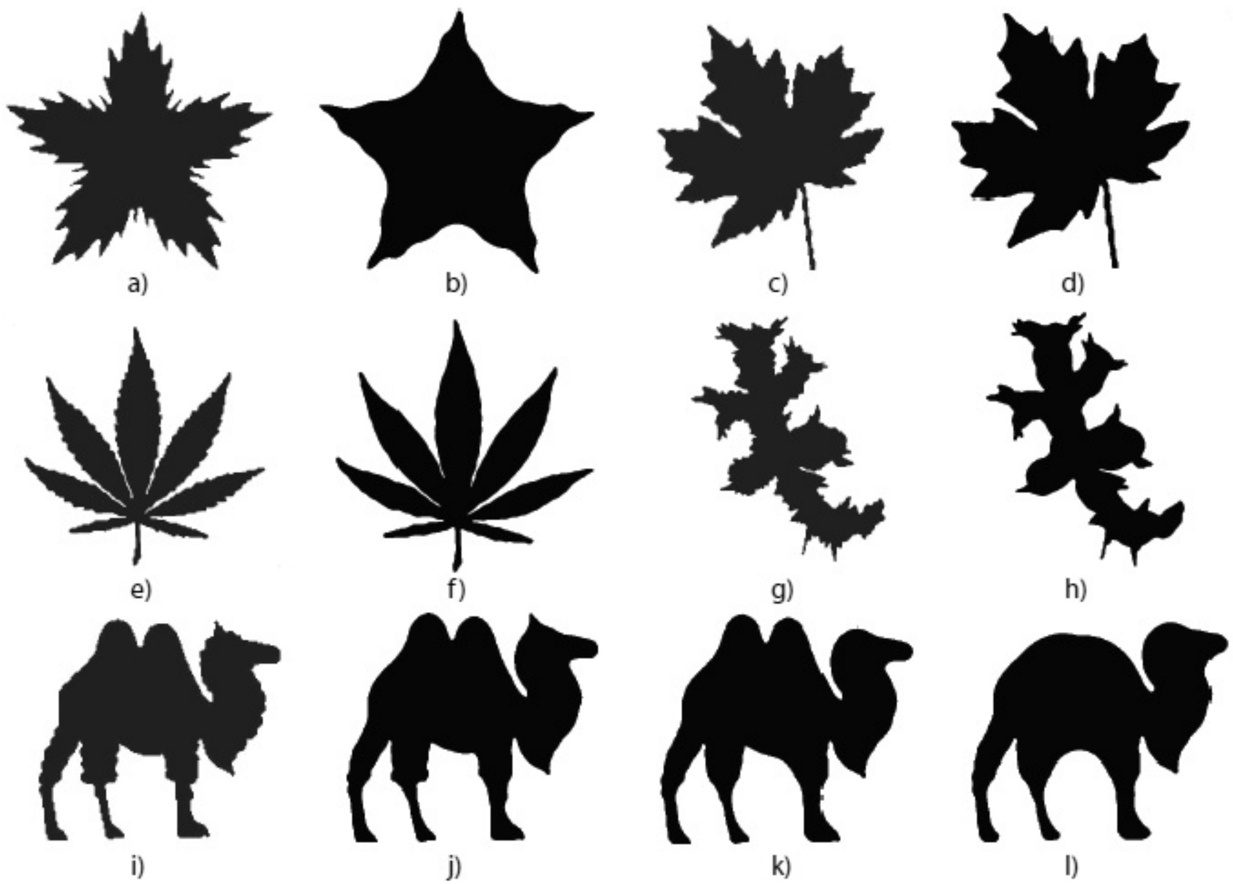
Change the distance metric!

Saliency Skeletons

Saliency metric:

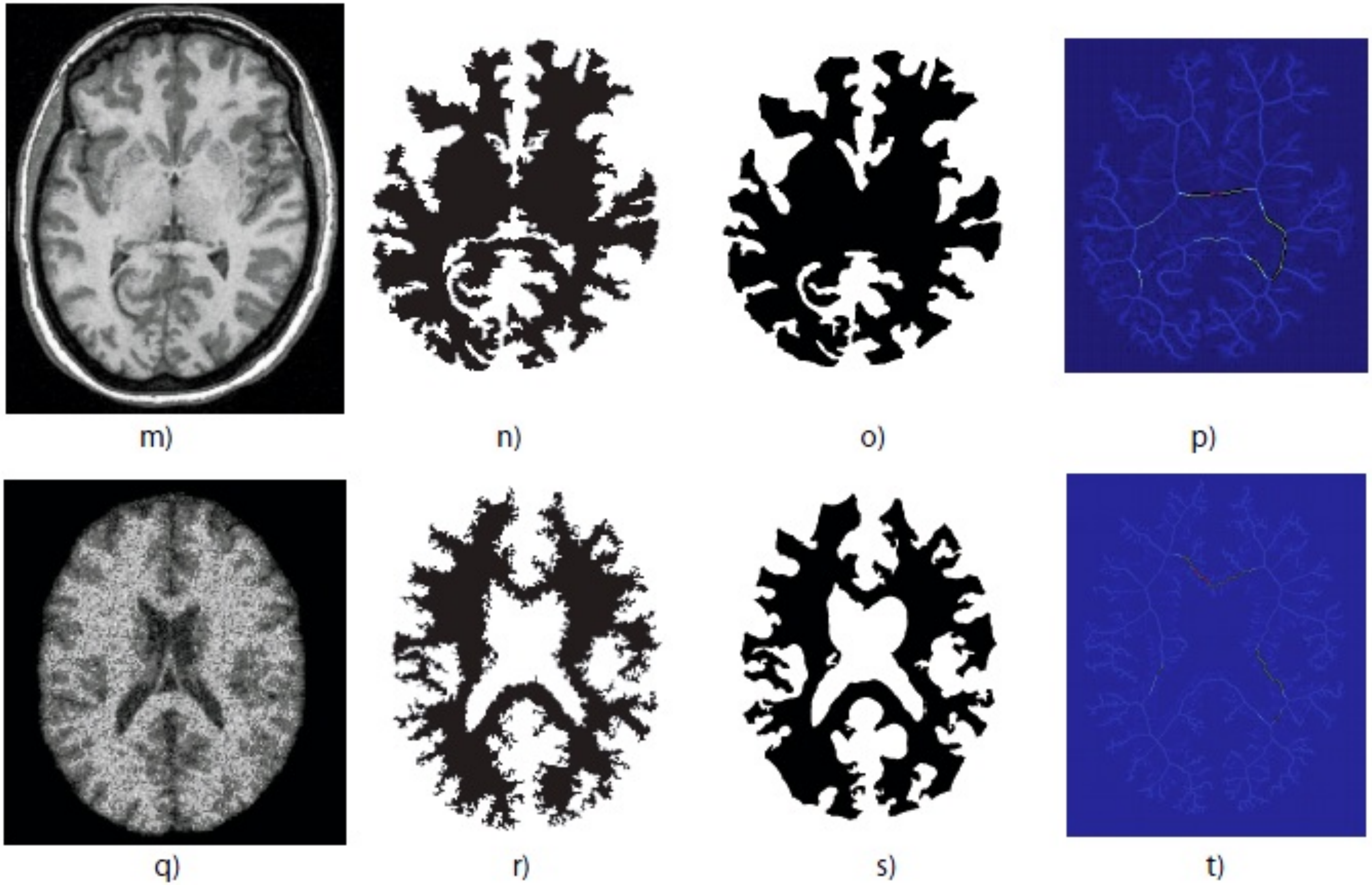
$\sigma(p) = \rho(p) \,/\, DT(p)$





Saliency Skeletons

More complex examples…



Saliency Skeletons

Most challenging example

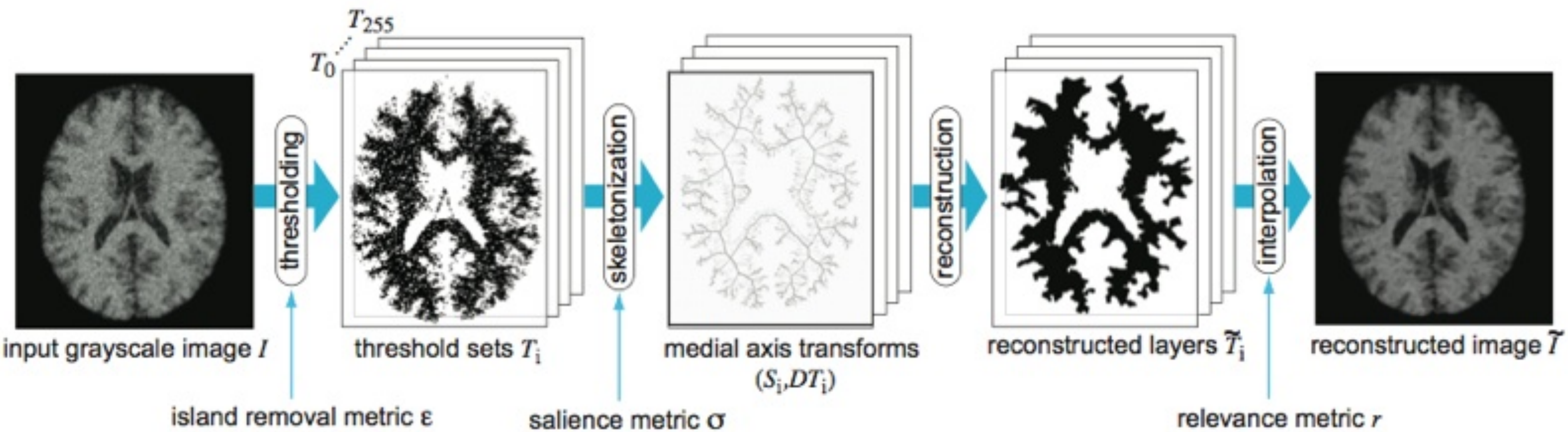
- very noisy CT segmentation
- saliency-based smoothing:
    - connect specks
    - reconstruct *perceived* sharp corners

Dense Skeletons

[Van der Zwan *et al.*, VISAPP'13]

Generalize skeletons

- for a whole color/grayscale image
- not just a binary shape



- this generates a 2-dimensional image **scale-space**

Applications

Image segmentation

- select a few most relevant layers
- simplify each layer (using salience metric)

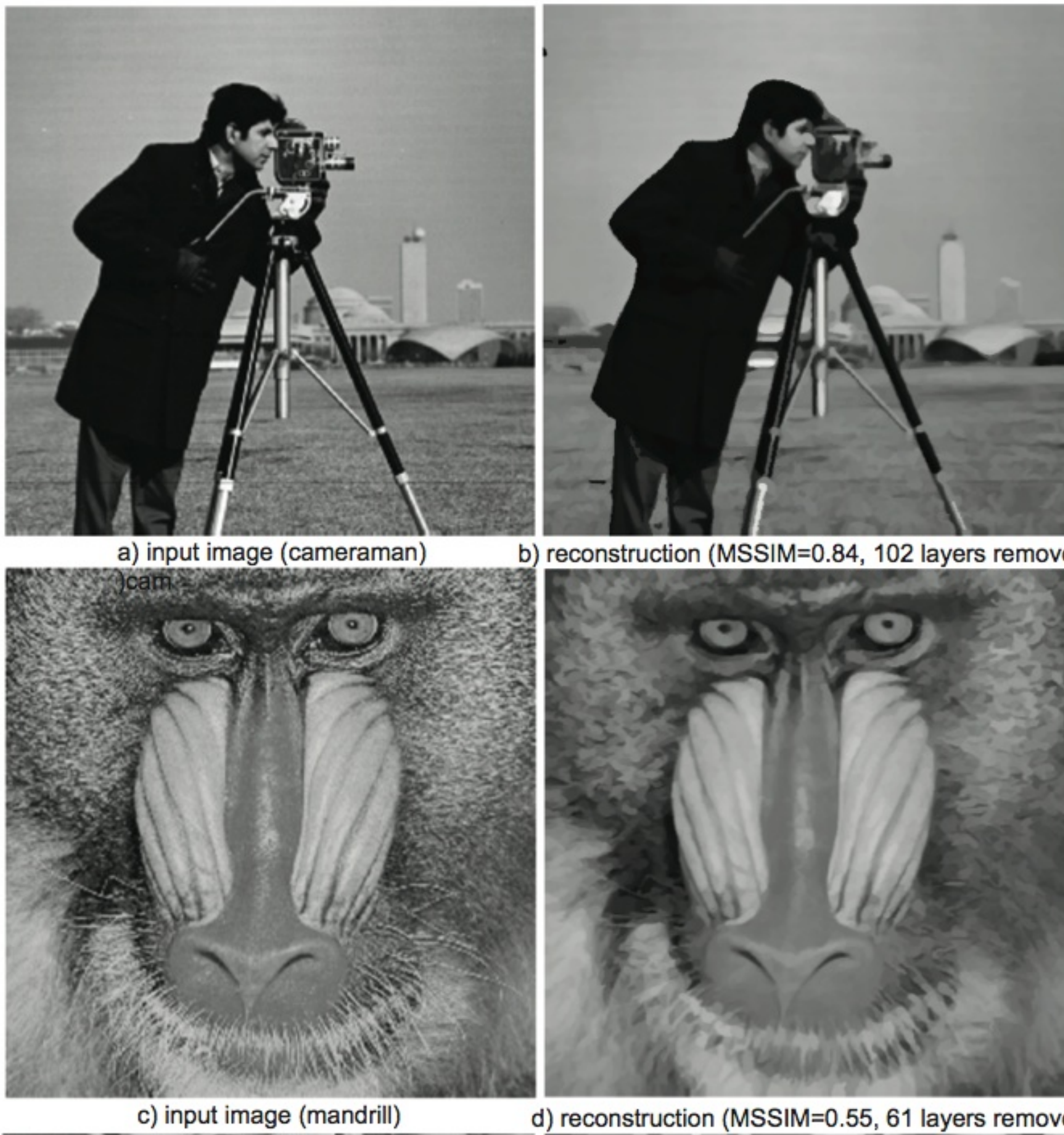
input image | dense skeletons (60% layers) | mean shift segmentation [Comaniciu & Meer '02]

- skeletons: we get less jaggies and we keep sharp corners!

Applications

Image compression

- same procedure as before



a) input image (cameraman)

b) reconstruction (MSSIM=0.84, 102 layers removed)

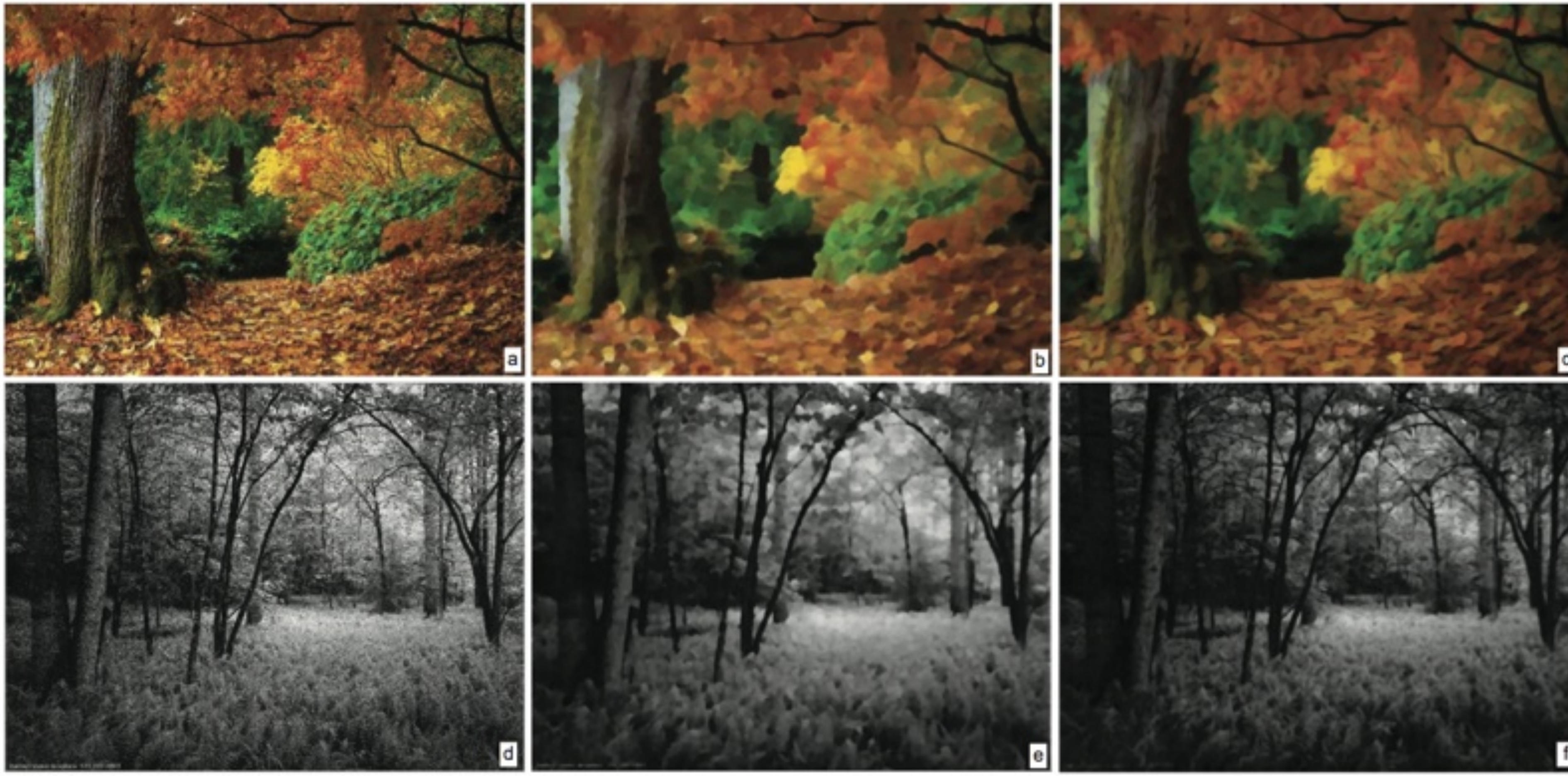
c) input image (mandrill)

d) reconstruction (MSSIM=0.55, 61 layers removed)

Applications

Artistic image manipulation

- keep few, highly simplified, skeletons



input images | Papari *et al.*, TPAMI'07 | our method

3D Skeletons

Generalize the 2D collapse metric to 3D!

Define vector field F and mass ρ on Ω so that

$$F\,|_{\partial\Omega} = n,\ \rho\,|_{\partial\Omega} = 1$$

$$\mathrm{F} = \nabla(\mathrm{DT})\,|_{\Omega \setminus S}$$

$\operatorname{div} \rho F = 0$ on entire Ω (also on S!)

Intuition: Mass...

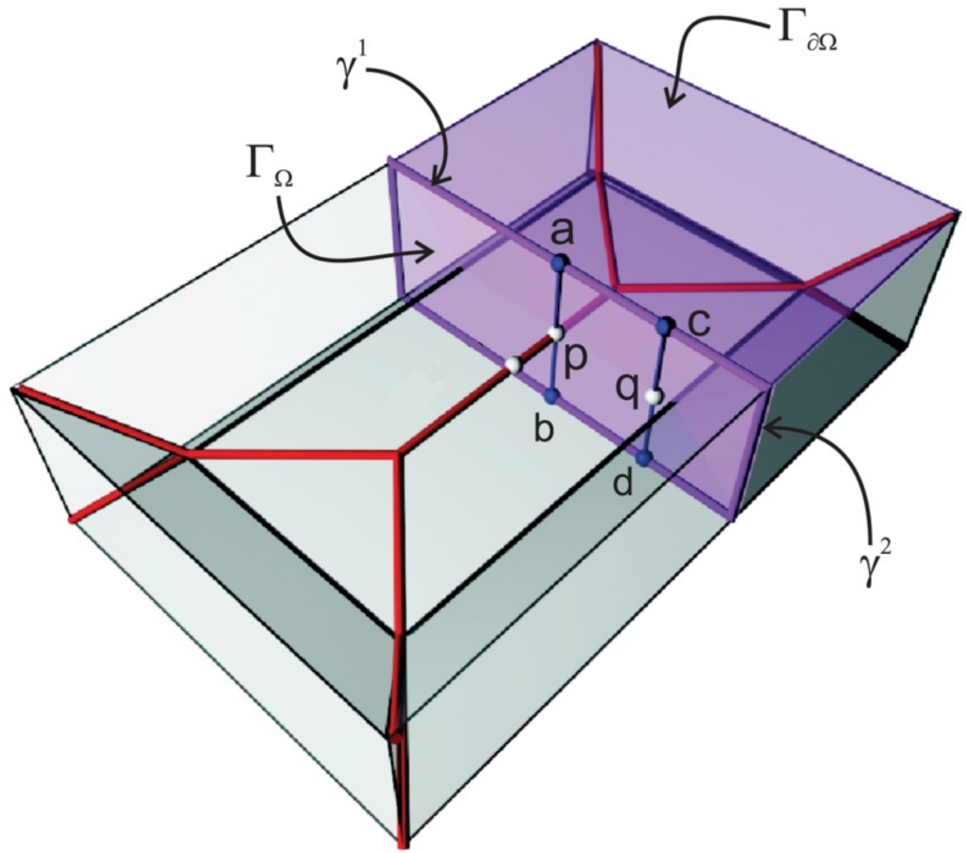
- flows straight from $\partial\Omega$ to surface skeleton S (2D)
- flows on S to curve skeleton C (1D)
- flows on C to a global root-sink R (0D)

Collapse $\rho(x)$: mass passing through x en route to R

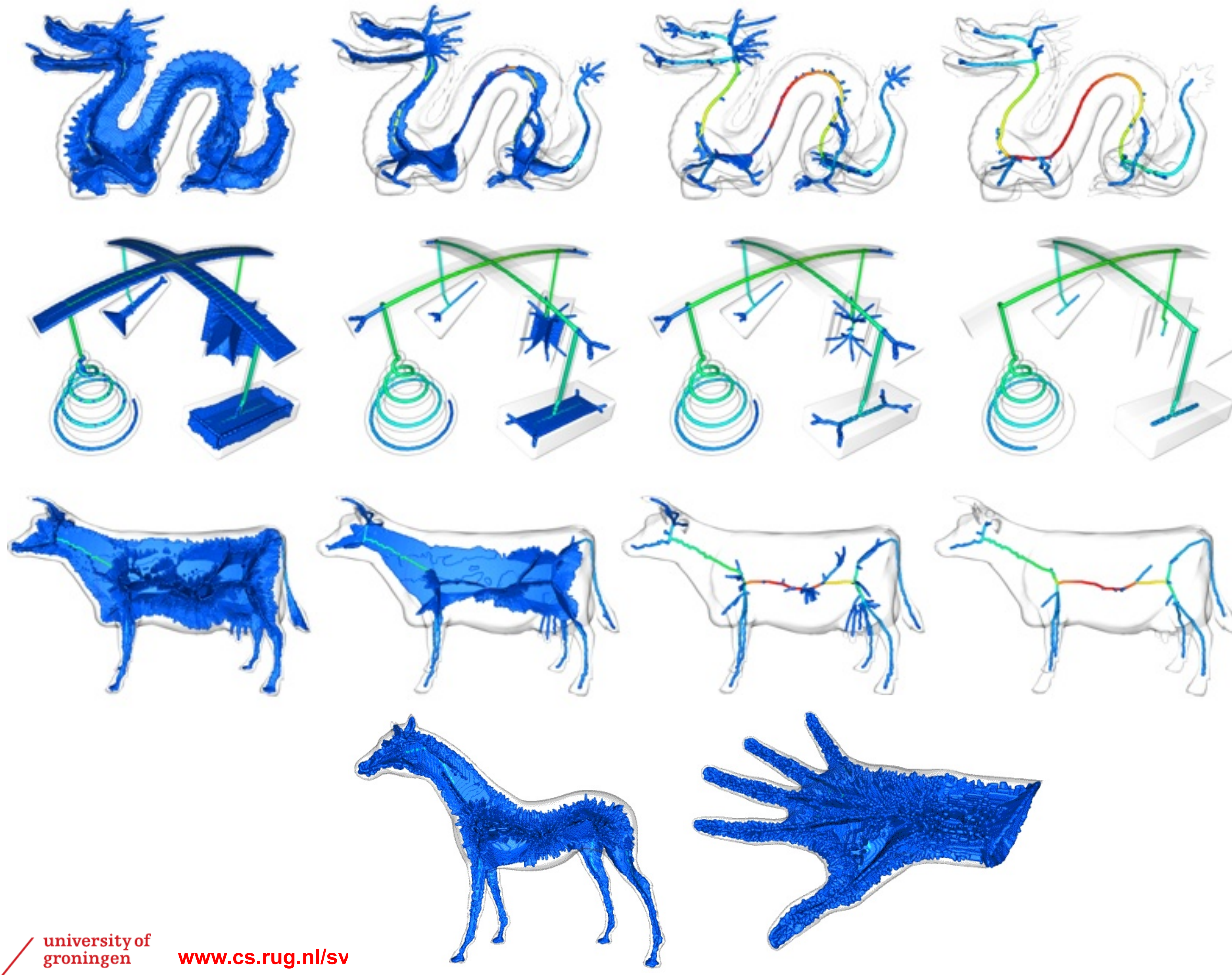
3D Surface and Curve Skeletons

[Reniers *et al.*, TVCG'08]

- **Directly compute collapse:**
  - no advection
- **Curve skeleton formal definition:**
  - $\boldsymbol{x} \in S$ that have two shortest paths between their two feature points
- **Collapse:**
  - $\boldsymbol{x} \in C$: smaller area of the two $\partial\Omega$-components due to the two shortest paths
  - $\boldsymbol{x} \in S$: length of single shortest-path

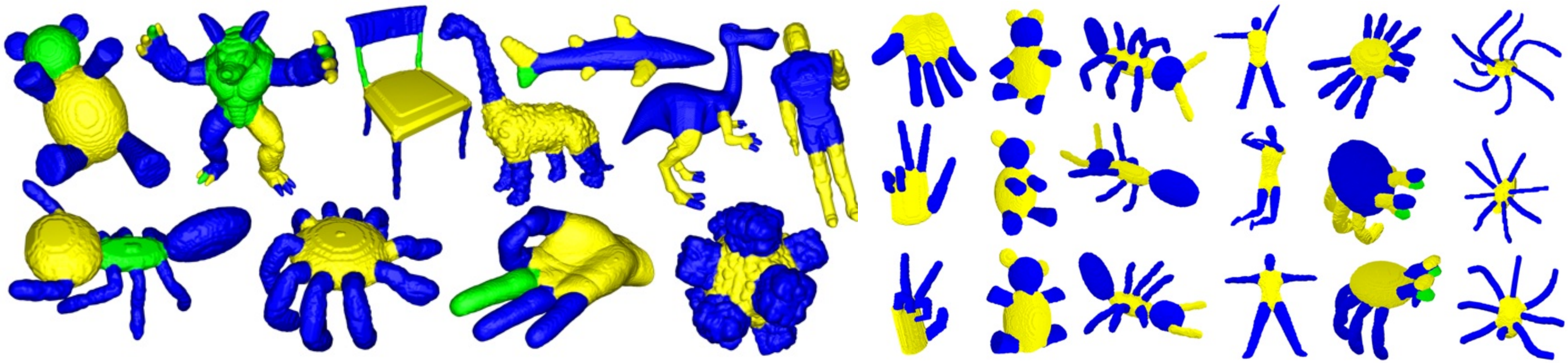


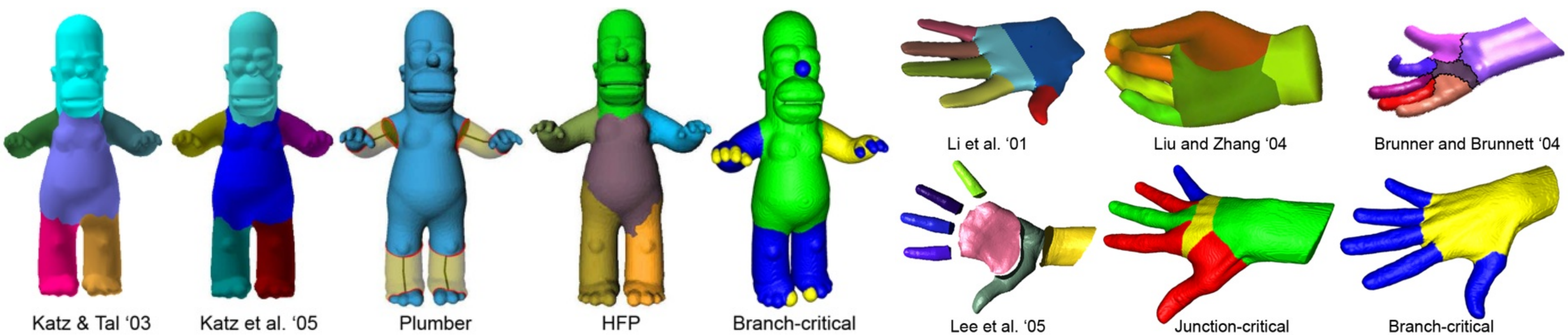
3D Surface and Curve Skeletons



[TVCG'08, DGCI'06, DGCI'08]

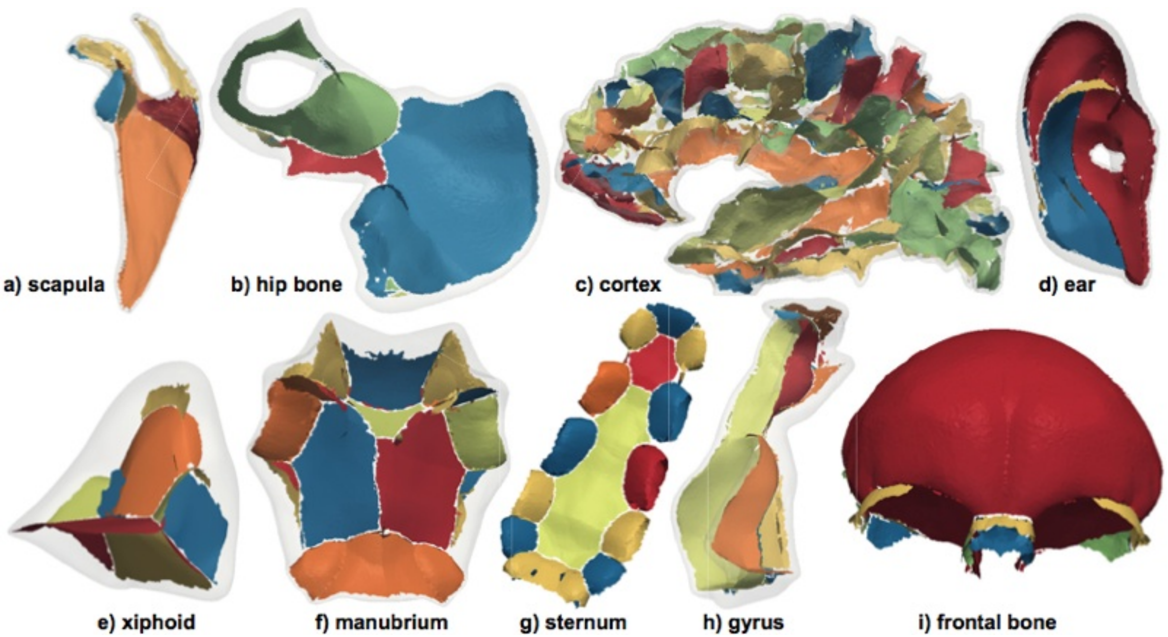
Curve Skeletons: Part Segmentation



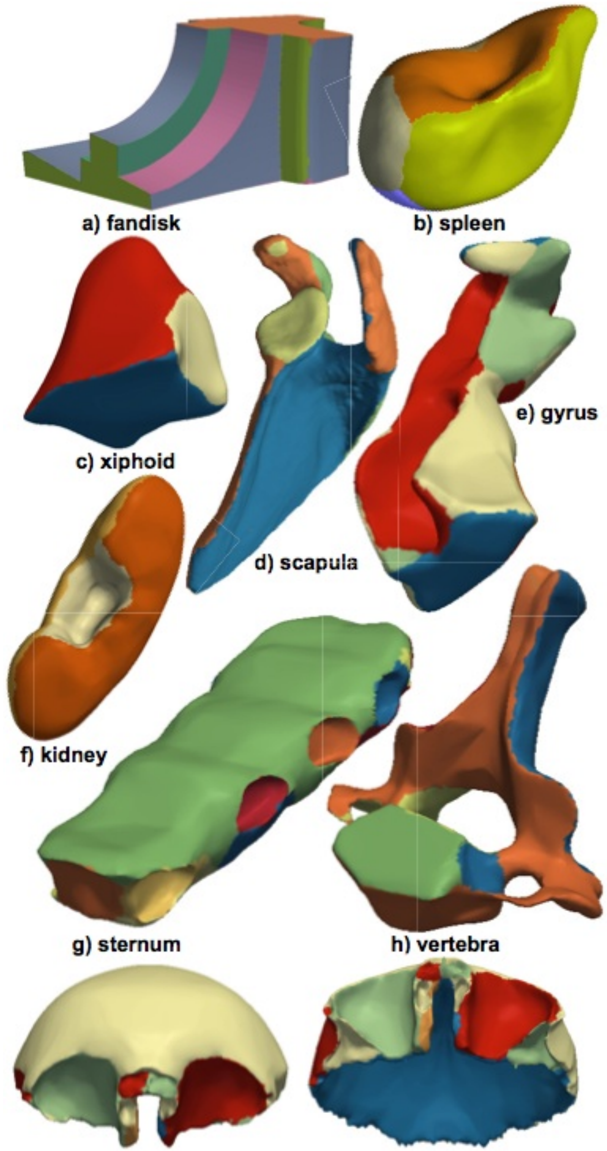


[CGF’08, TVC’08, SMI’07, SMI’08]

Curve Skeletons: Patch Segmentation



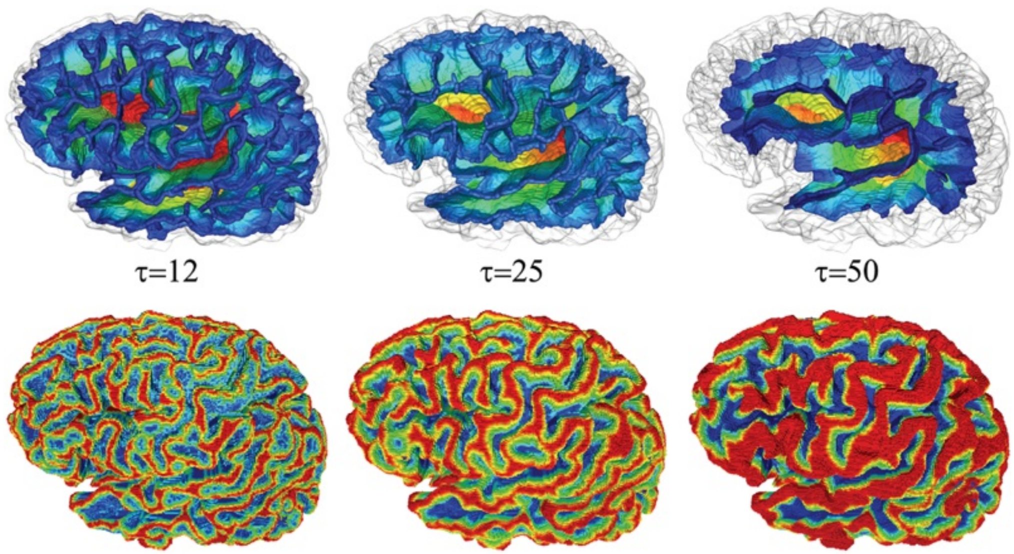
Surface skeleton segmentation



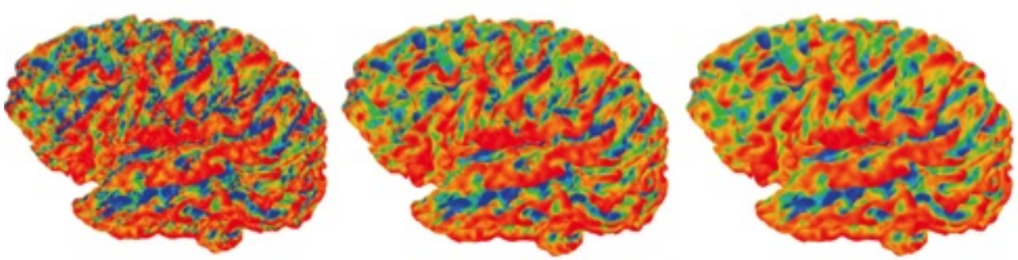
Shape segmentation

Surface Skeletons: Shape Classification

Robust surface classification

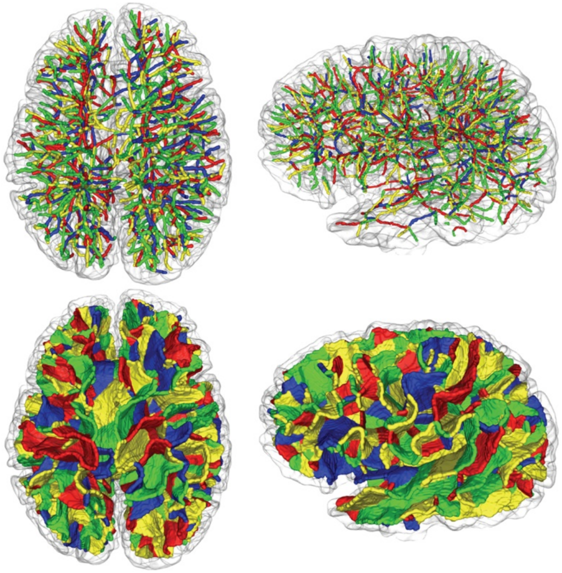


[VCBM'08]



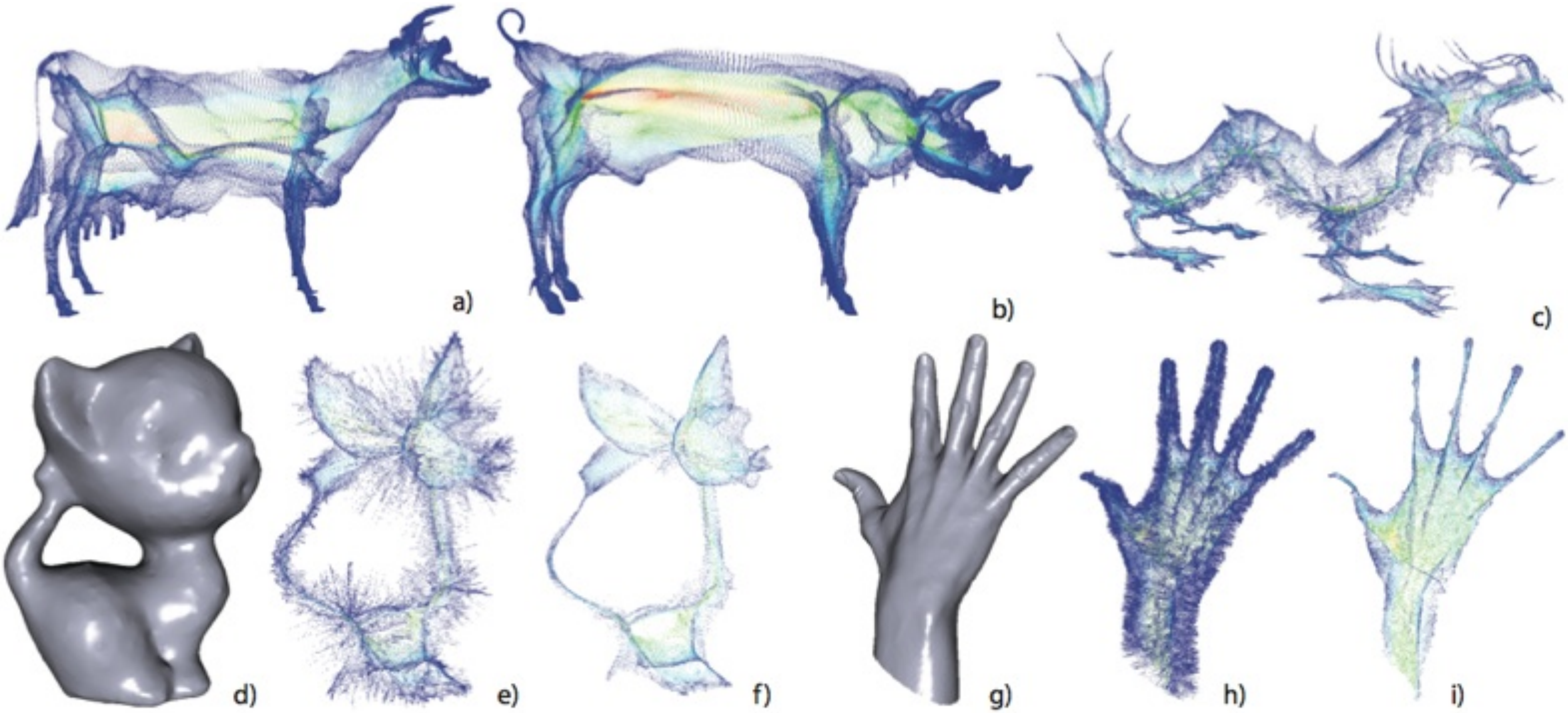
[Taubin'95]

Cortex structure



[VCBM'08]

Scalability



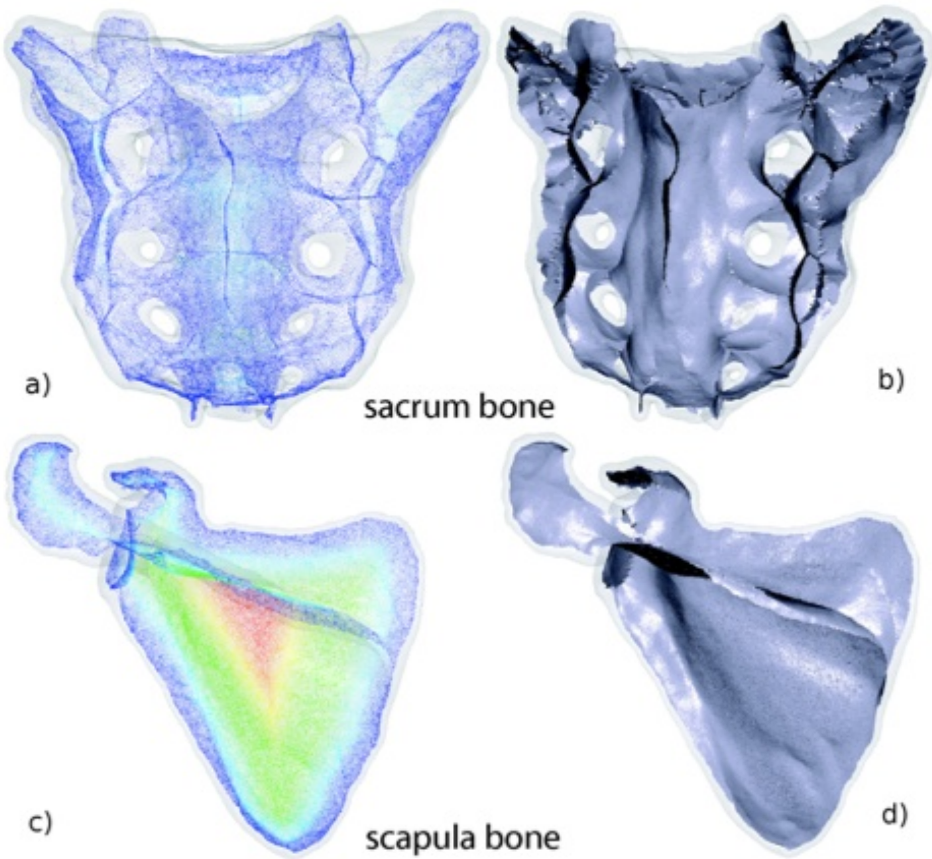
[Jalba *et al.*, TPAMI'12]

GPU implementation

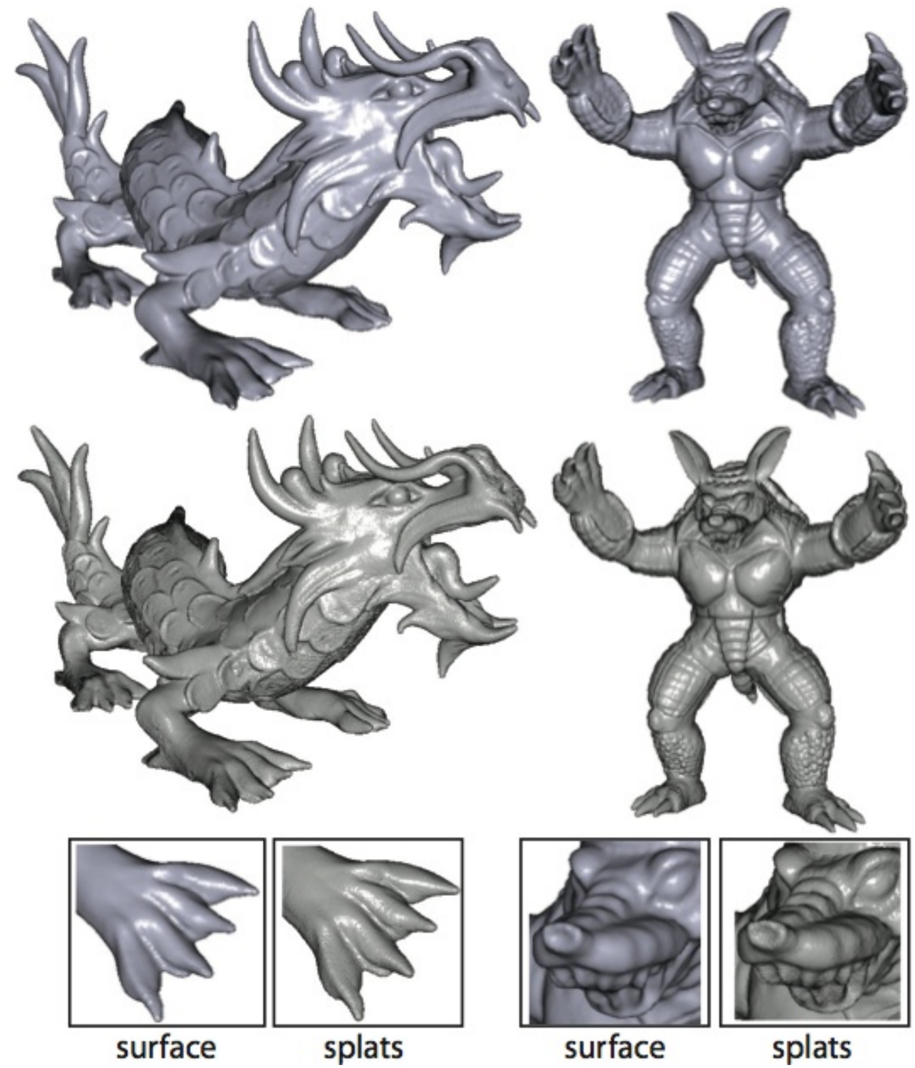
- point clouds & meshes
- **1M points/second** (GTX 280)
- 3D surface skeletons are finally practical

Real-time Reconstruction

Surface skeleton
(technique: mesh projection)



Input shape
(technique: depth splatting)



[Jalba *et al.*, TPAMI'12]

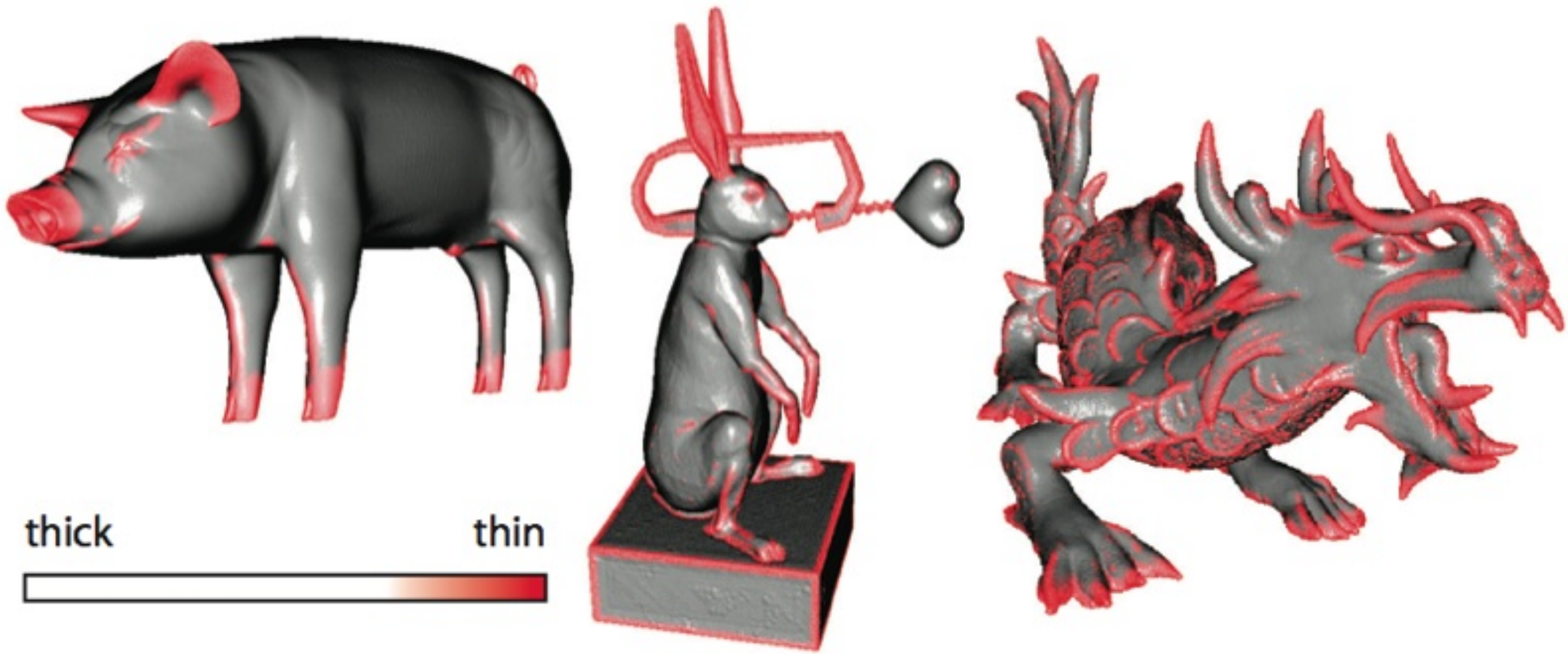
Shape thickness

Find thin shape parts

- important in 3D metrology (e.g. 3D printing)
- how to define/compute shape thickness?
- solution:

$$\mathit{Thickness}(\mathrm{p} \in \partial\Omega) = \min_{\mathrm{q} \in FT^{-1}(\mathrm{p})} DT(\mathrm{q})$$

- easy to implement, real-time to compute



[Jalba *et al.*, TPAMI'12]

On to our unified framework…

Challenge 1: Definition

- what *is* a curve skeleton?
- many algorithms, few formal definitions (except Dey & Sun '06, Reniers *et al.'* 08)

Which is the 'correct' curve skeleton?

Challenge 2: Unification

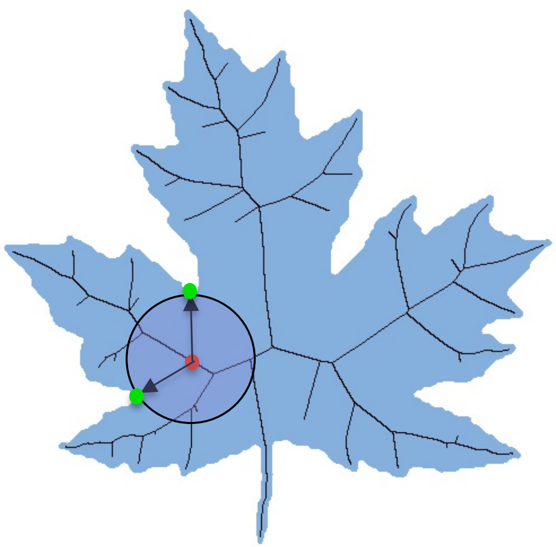
- can we define *and* compute the C-skeleton from the S-skeleton…
- …in the **same** way we compute the S-skeleton from the input shape?

Unified framework: Yes we can☺

Use exactly the same definition for S-skeleton and C-skeleton (!)

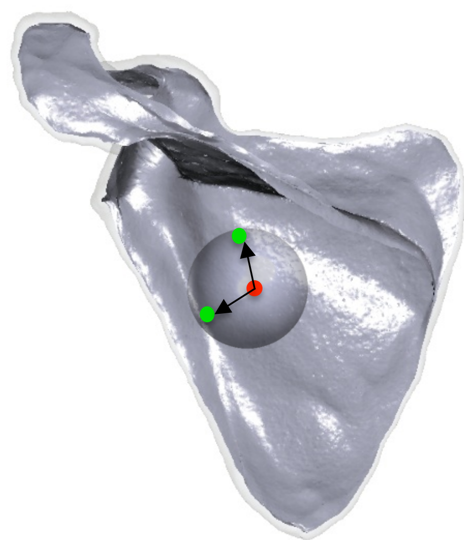
$$S(\Omega) = \{\mathbf{x} \in \Omega \,|\, \exists \mathbf{y}, \mathbf{z} \in \partial\Omega, \mathbf{y} \neq \mathbf{z}, \|\mathbf{x} - \mathbf{y}\| = \|\mathbf{x} - \mathbf{z}\| = DT_{\partial\Omega}(\mathbf{x})\}$$

2D skeleton



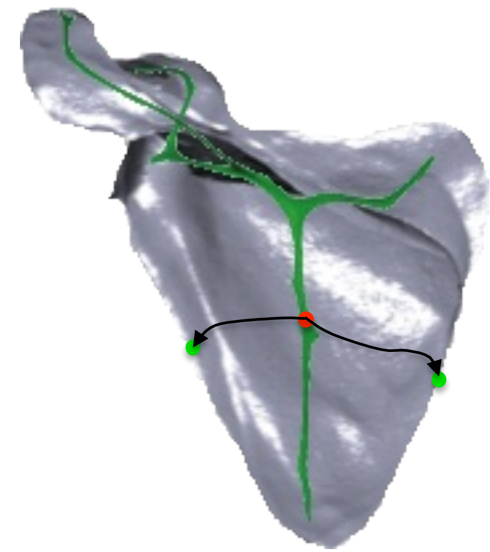
Ω: input shape (2D)
$\|\cdot\|$: Euclidean dist. (Ω)

3D S-skeleton



Ω: input shape (3D)
$\|\cdot\|$: Euclidean dist. (Ω)

3D C-skeleton

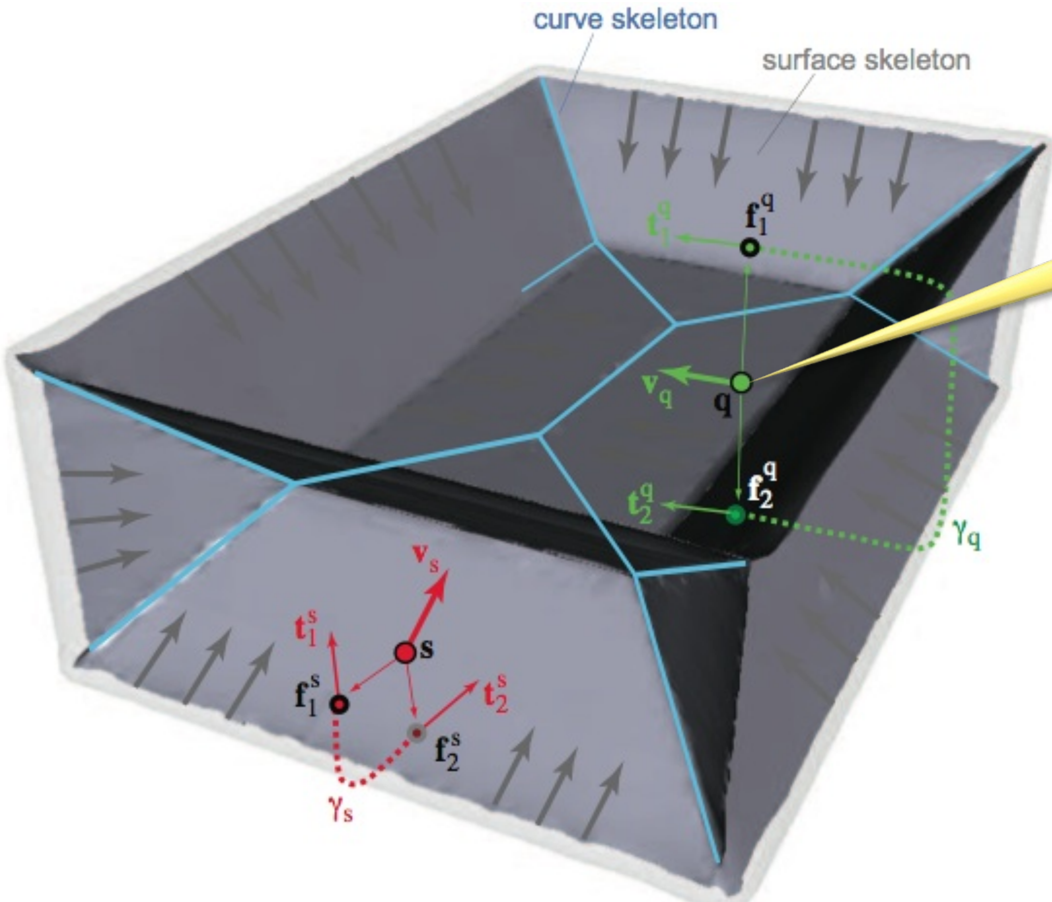


Ω: S-skeleton
$\|\cdot\|$: geodesic distance (S)

[Jalba and Telea, EGUK'12 Best paper]

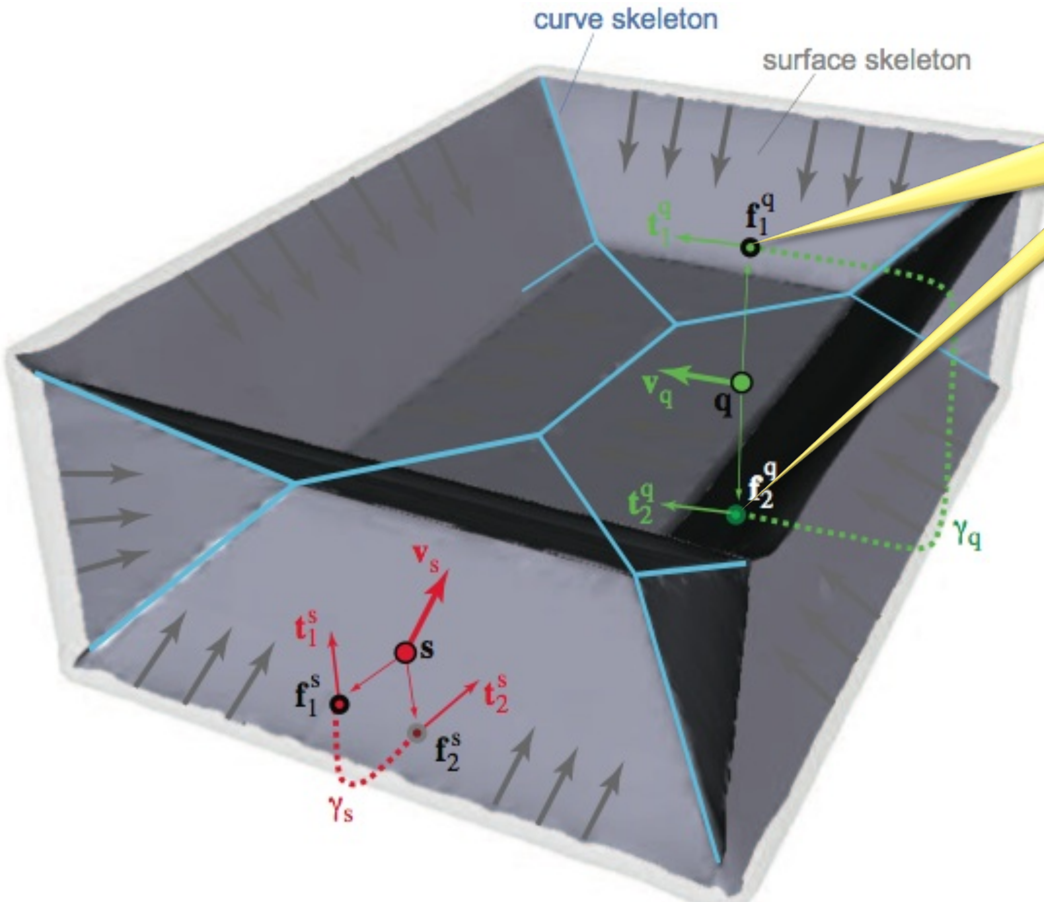
Computation

- advect S in $\nabla DT_{\partial S}$
- compute $\nabla DT_{\partial S}$ analytically using observation in [Reniers *et al.*, TVCG'08]



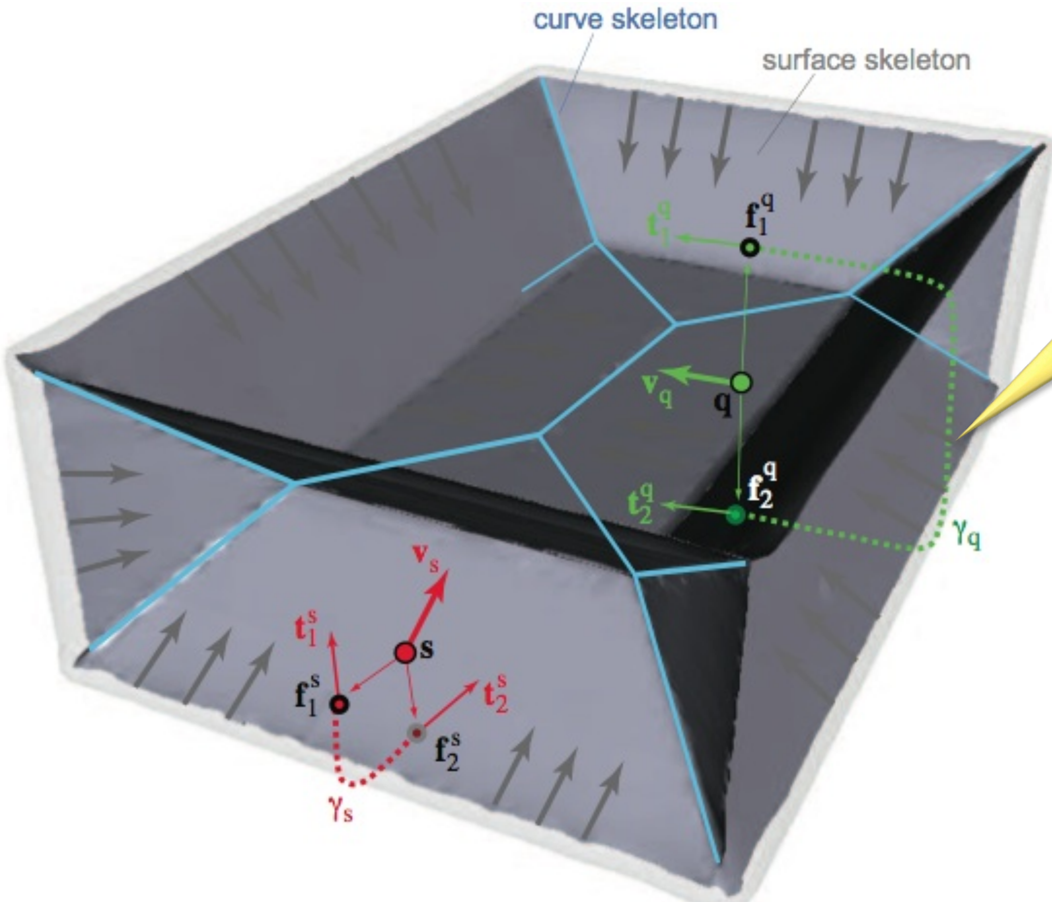
Given a S-skeleton point **q**…

Computation



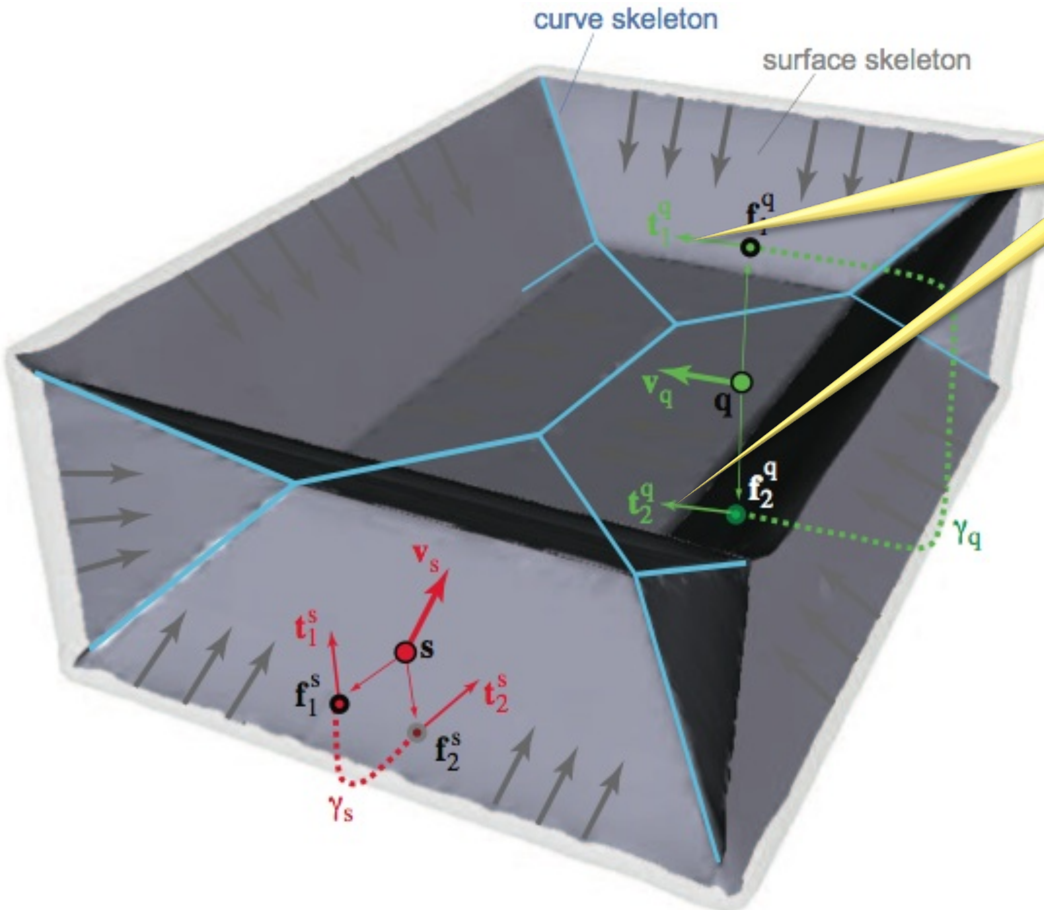
Take its two feature points $\mathbf{f}_1{}^q$ and $\mathbf{f}_2{}^q$…

Computation



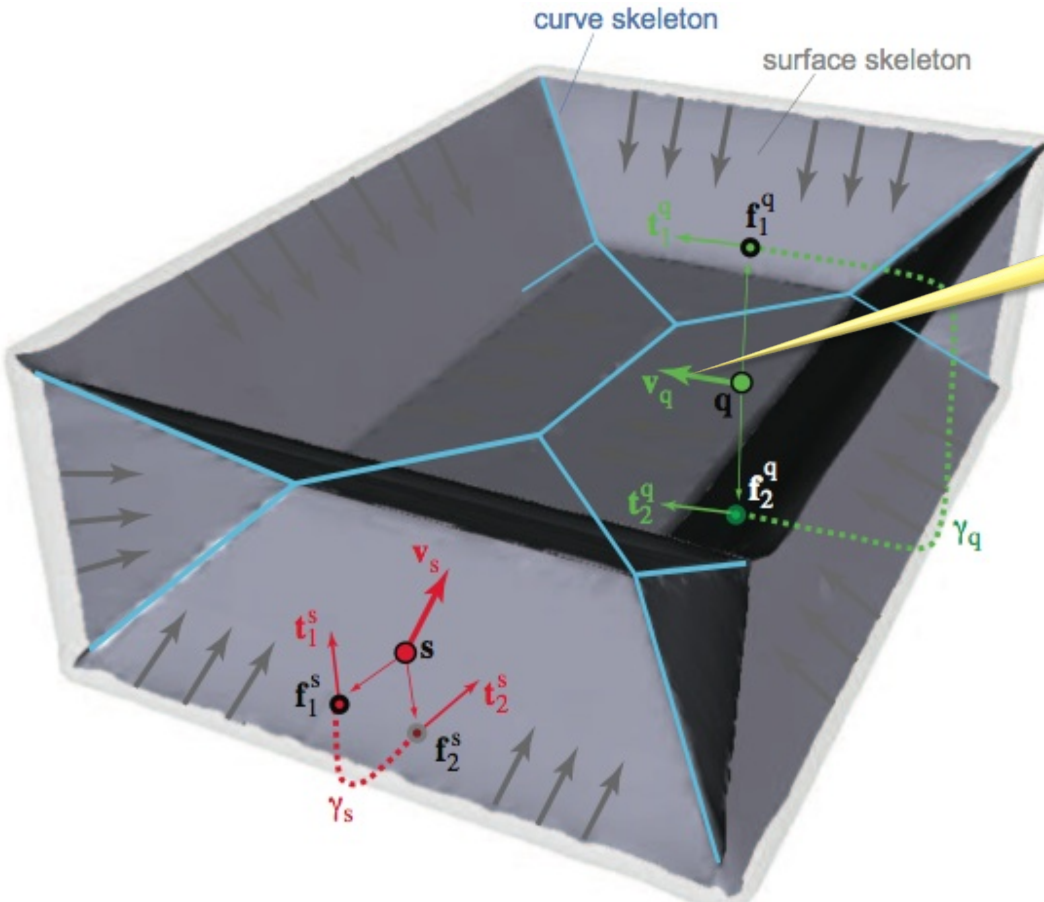
Compute geodesic γ_q on $\partial\Omega$ between $\mathbf{f}_1^q$, $\mathbf{f}_2^q$…

Computation



Take the tangent vectors $\mathbf{t}_1{}^q$, $\mathbf{t}_2{}^q$ on γ_q at $\mathbf{f}_1{}^q$, $\mathbf{f}_2{}^q$…

Computation



Compute $\nabla DT_{\partial S}(q)$ as $(\mathbf{t}_1{}^q+\mathbf{t}_2{}^q)/2$

Computation

Advect S-skeleton into C-skeleton:

$$\mathbf{s}^{i+1} = P_{T(\mathbf{s}^i)}\left(\mathbf{s}^i + \frac{\nabla DT_{\partial S}(\tilde{\mathbf{s}}^i)}{\|\nabla DT_{\partial S}(\tilde{\mathbf{s}}^i)\|}\delta\right)$$

keep advection in S-skeleton (triangle-fan at $\mathbf{s}^i$)

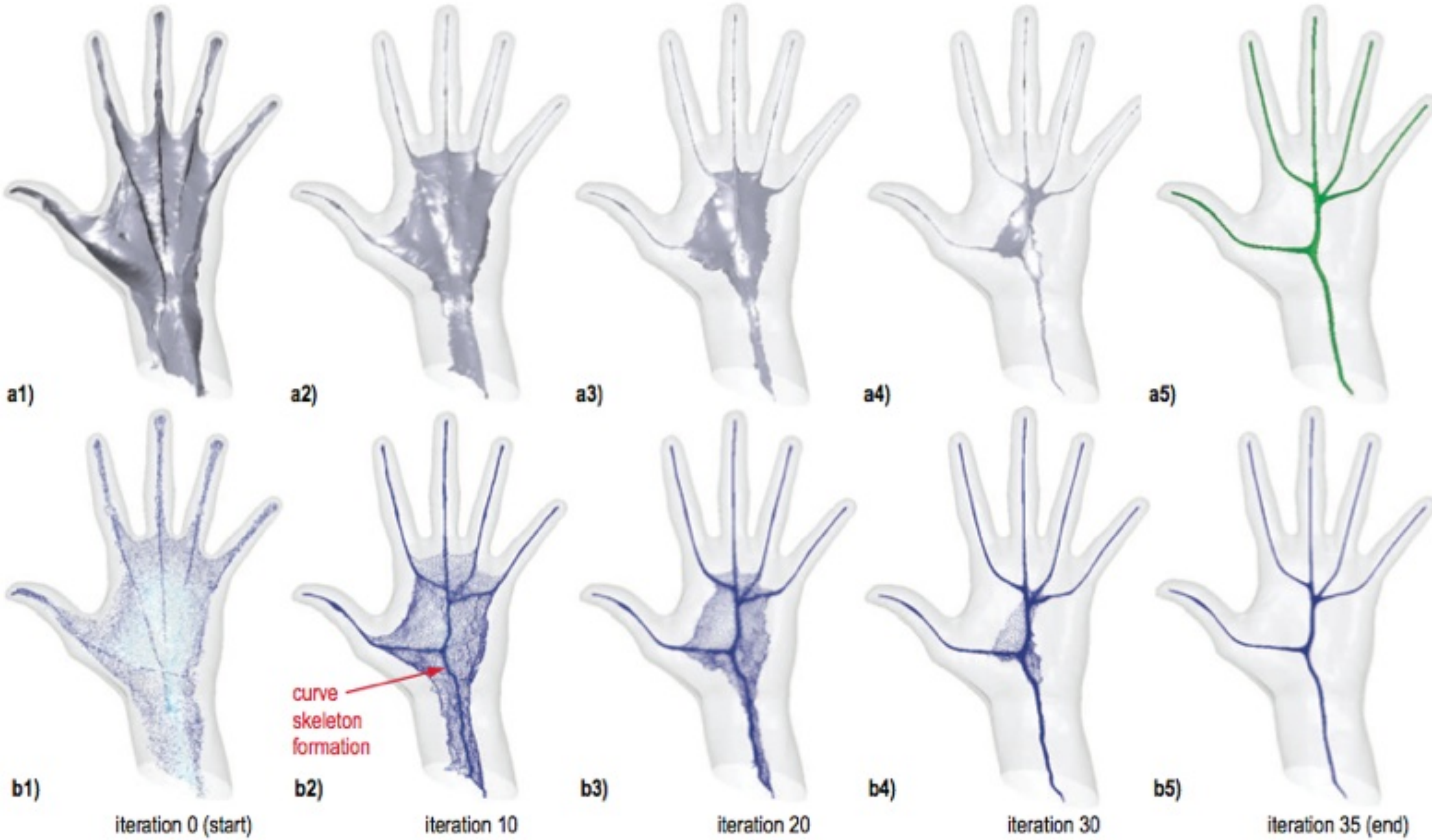
normalize $\nabla DT_{\partial S}$ to control advection speed

nearest-neighbor of $\mathbf{s}^i$ on S-skeleton

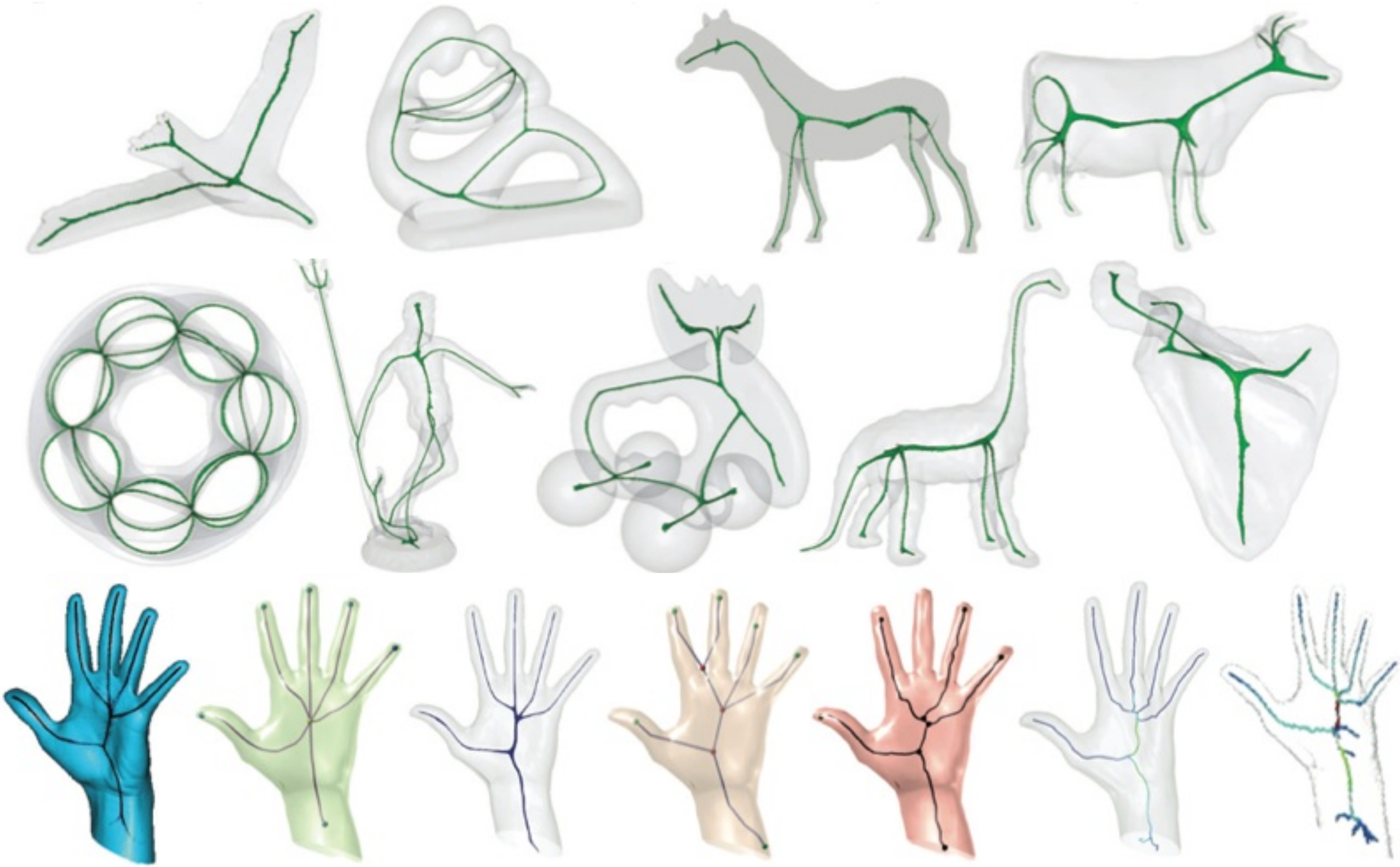
Stop advecting points when $T(\mathbf{s}^i) < \varepsilon$

www.cs.rug.nl/svcg university of groningen

Advection


a1)
a2)
a3)
a4)
a5)
b1)
b2)
b3)
b4)
b5)
curve
skeleton
formation
iteration 0 (start)
iteration 10
iteration 20
iteration 30
iteration 35 (end)

Results

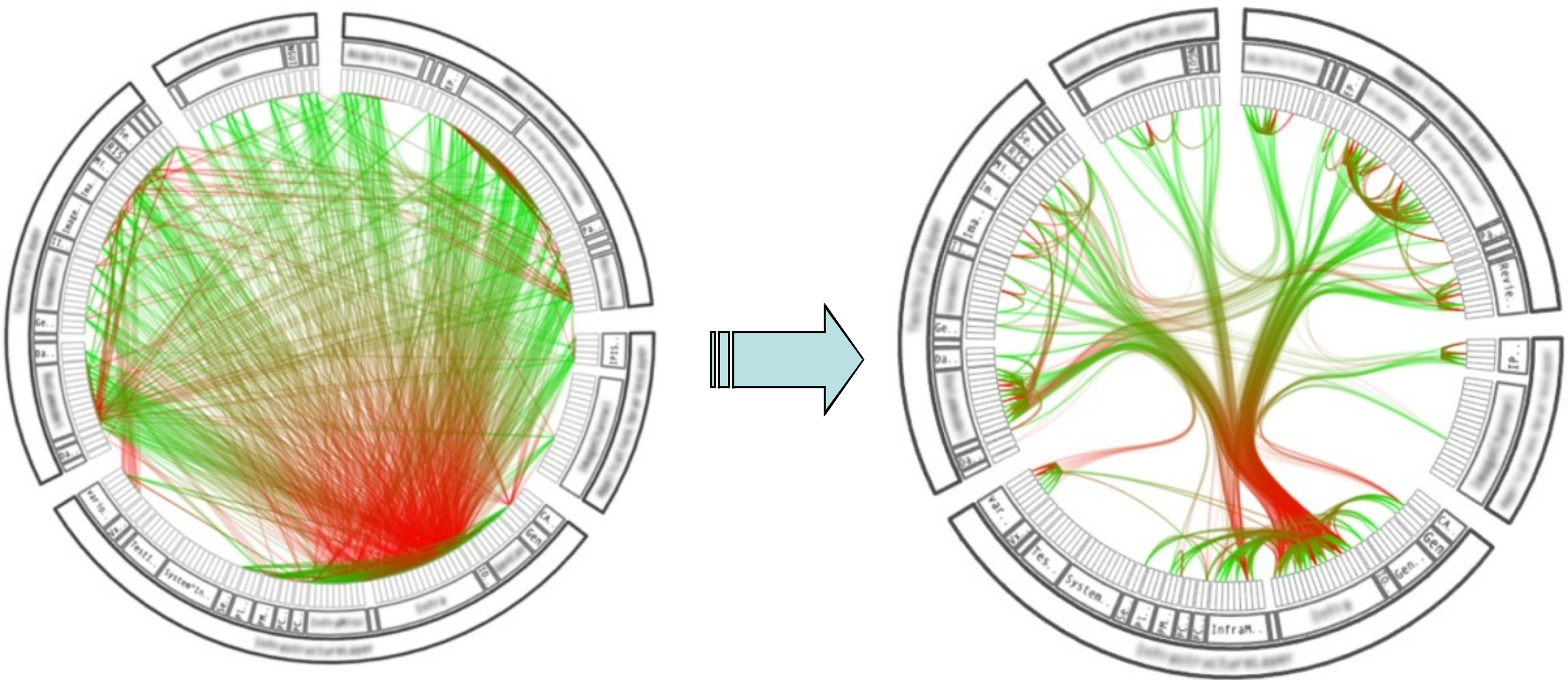


Kustra *et al.*, VISAPP'13 | Livesu *et al.*, TVCG'12 | Our results | Au *et al.*, SIGGRAPH'08 | Dey and Sun, SGP'06 | Jalba *et al.*, TPAMI'12 | Reniers *et al.*, TVCG'08

Applications: Graph visualization

Edge bundling

- how to capture & draw the **essence** of a large graph (>100K edges)
- probably the hottest area in large graph visualization (10..20 top papers/year)



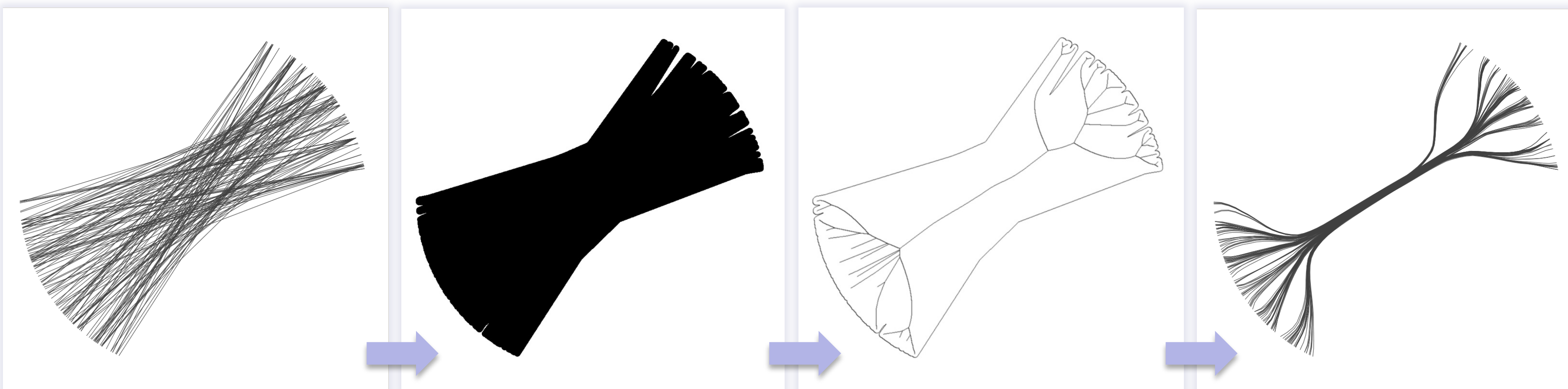
[Holten, InfoVis'06]

Skeleton-based edge bundles (SBEB)

[Ersoy *et al.*, TVCG'11]
[Hurter *et al.*, TVCG'11]

Simple idea

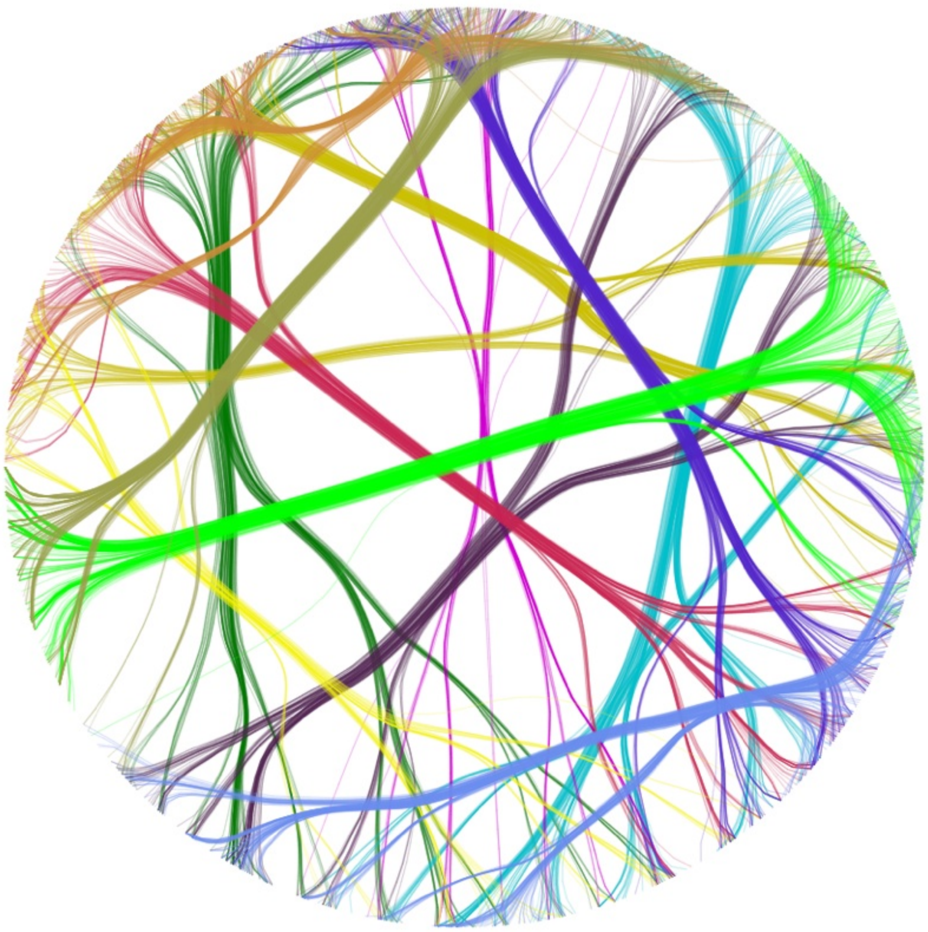
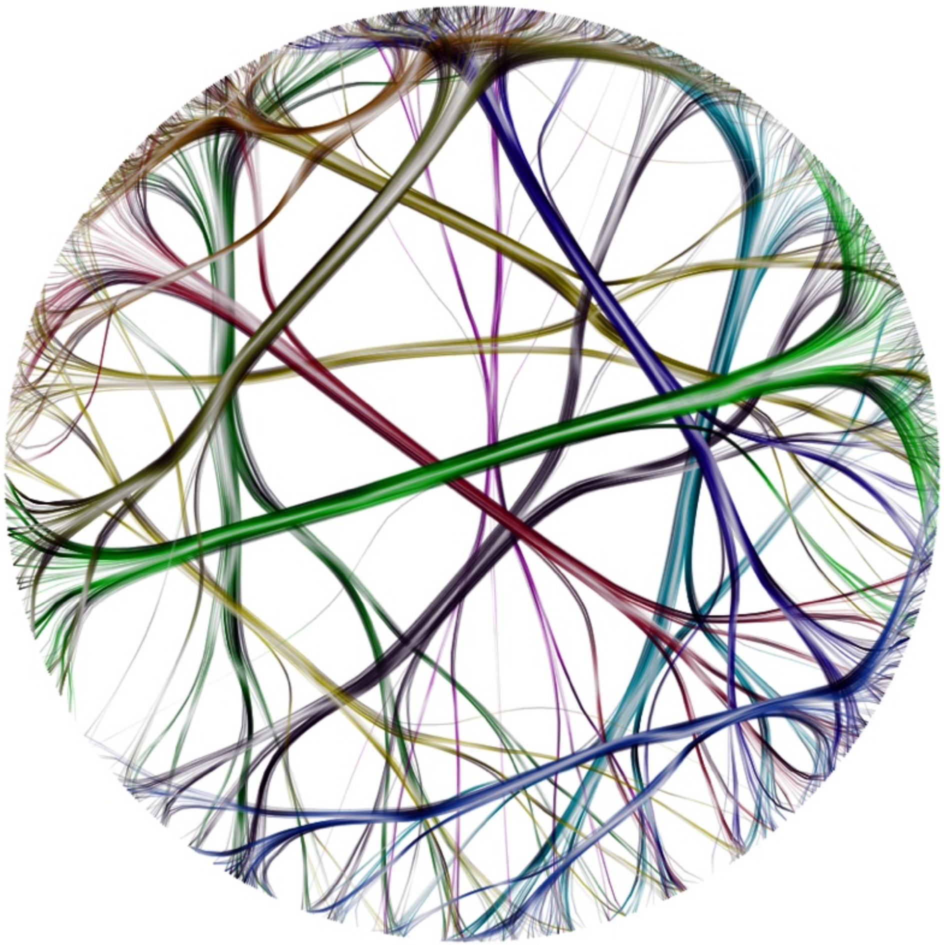
- iterate the following steps:



draw edges — dilate — skeletonize — attract to skeleton

- implementation: fully image-based (CUDA)

Pseudo-shading

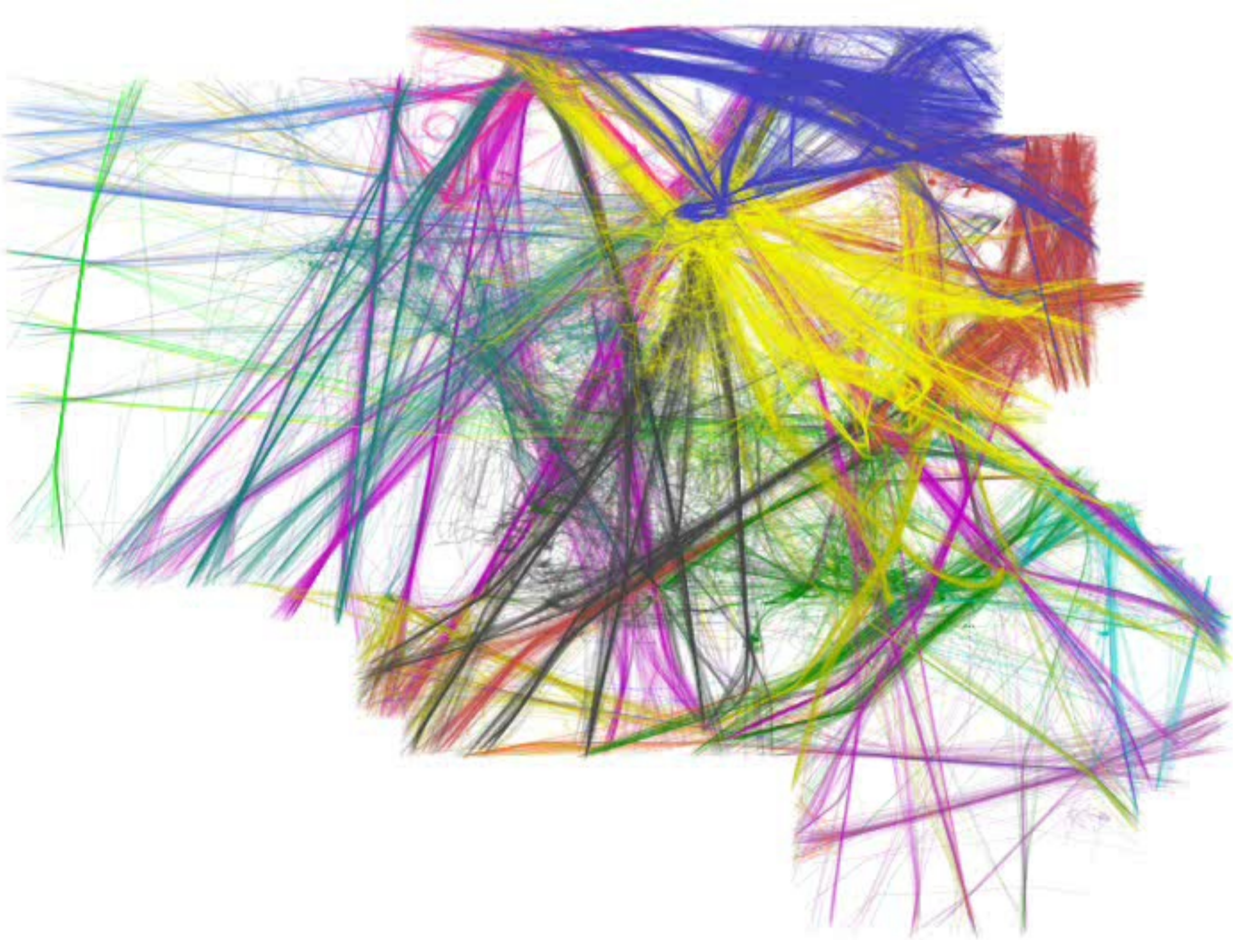
Results

US Migrations graph: 1715 nodes, 9780 edges, 6 clusters, 3 sec.

Results

US Migrations graph: 1715 nodes, 9780 edges, 6 clusters, 3 sec.

Results



France airlines graph: 34550 nodes, 17272 edges, 207 clusters, 27 sec.

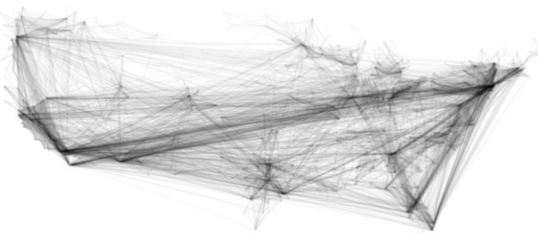
Results

France airlines graph: 34550 nodes, 17272 edges, 207 clusters, 27 sec.

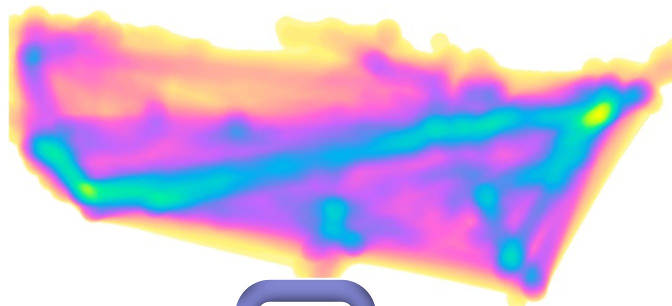
Kernel density edge bundling (KDEEB)

[Hurter *et al.*, CGF'12]

If bundling sharpens the edge density, then sharpening the edge density should bundle

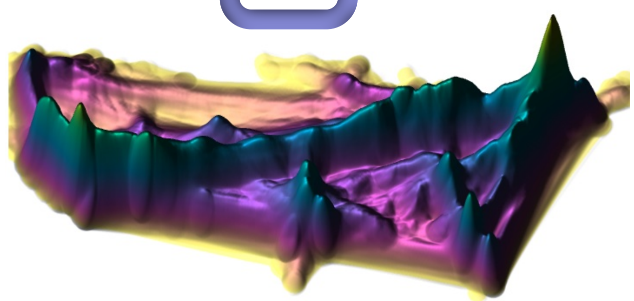


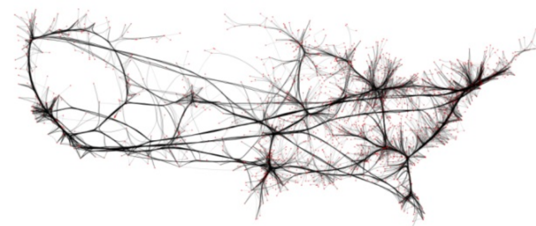
unbundled graph



sharpening

edge density signal

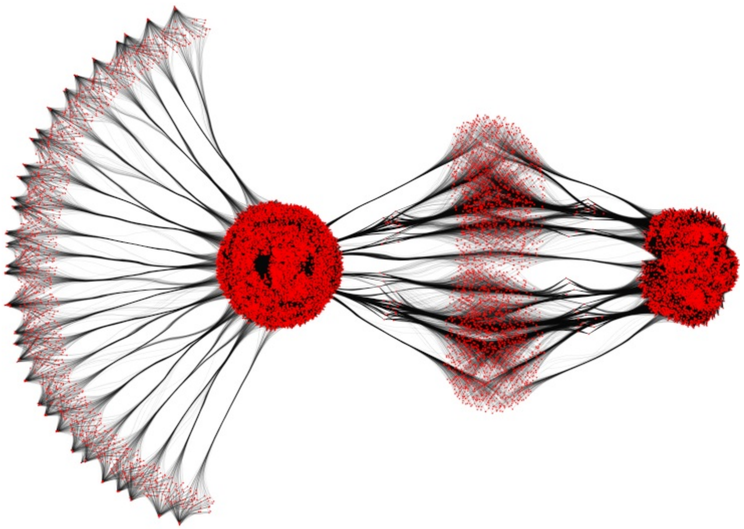




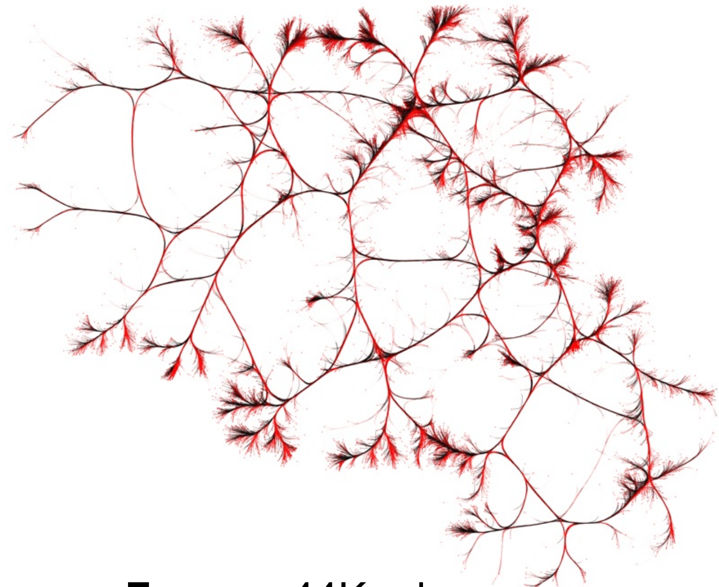
bundled graph

This is nothing but mean shift [Comaniciu & Meer '02] on the edge-space!

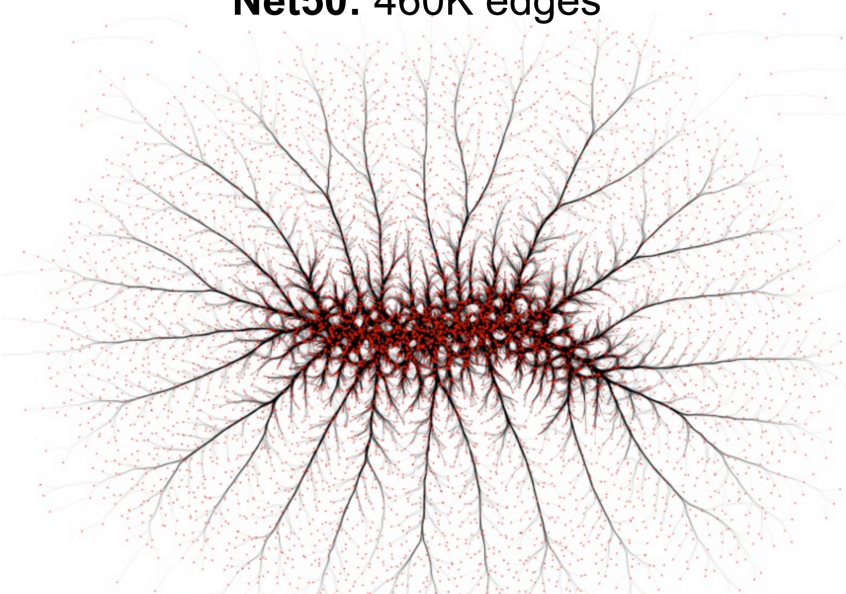
Results



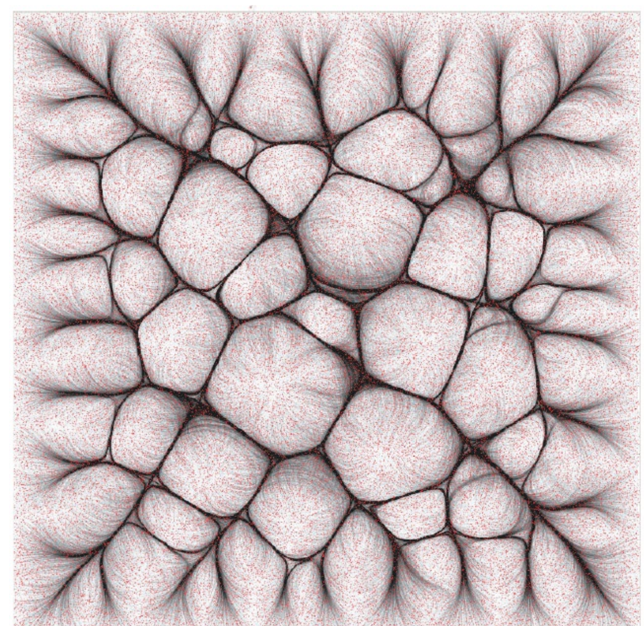
Net50: 460K edges



France: 44K edges



Wiki: 200K edges

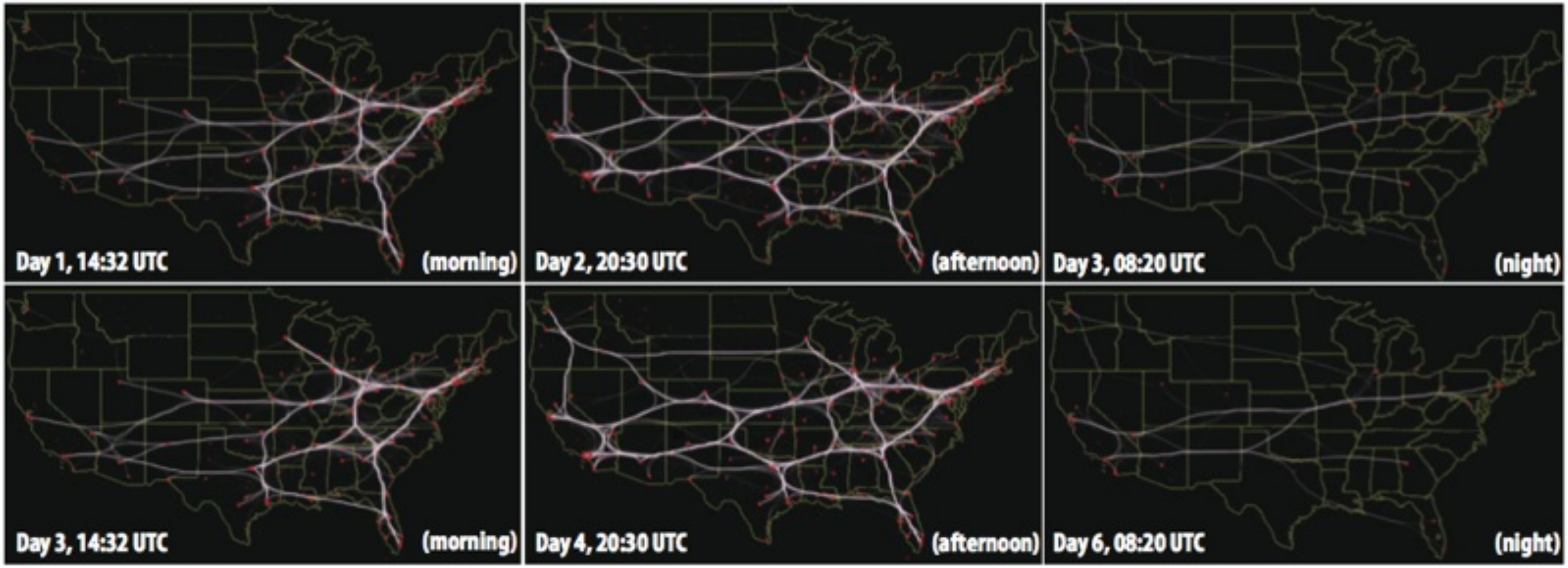


Random: 200K edges

Bundling dynamic graphs

Time-dependent graphs

- streaming data (**millions** of edges, arriving in real time)
- solution: time-dependent mean shift – **real-time** bundling on the GPU!

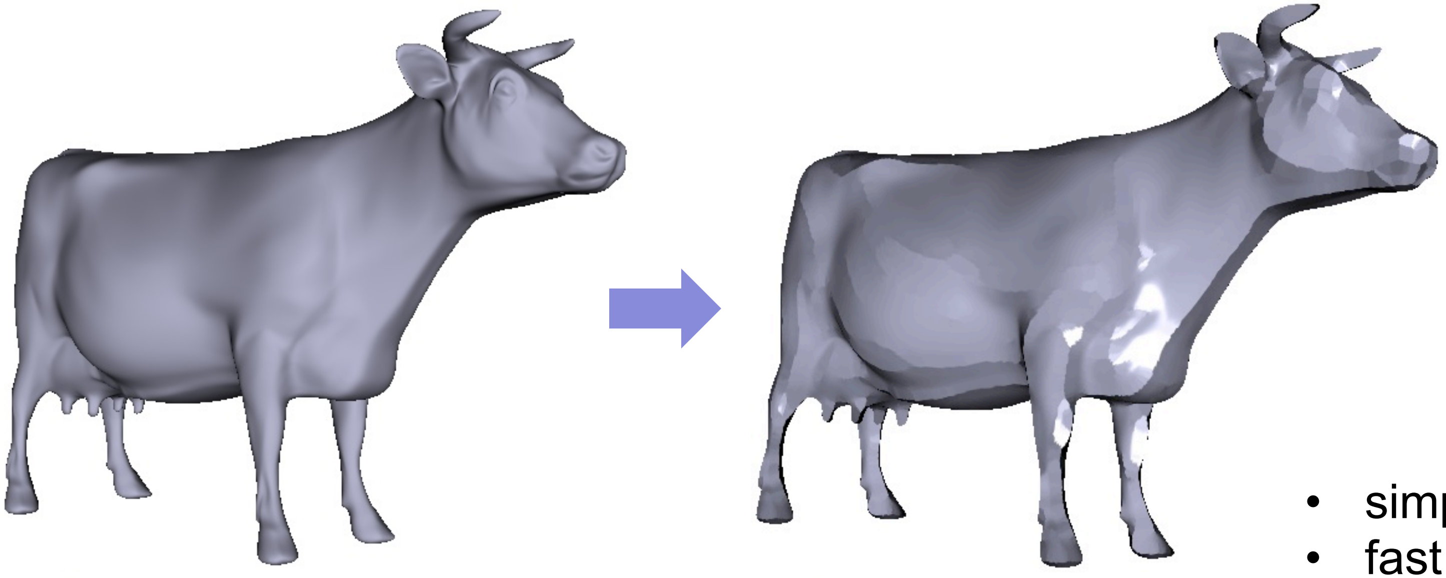


[Hurter *et al.* PacificVis'13]

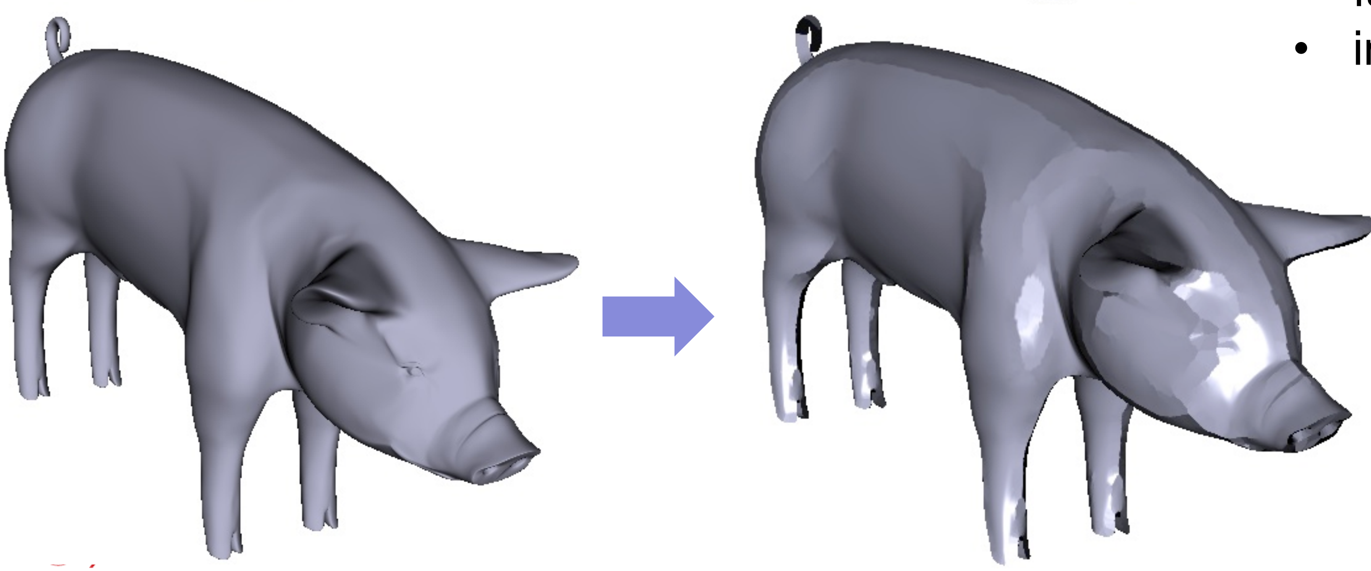
Ongoing work

NPR sculpting

- reduce 3D shape to smooth surfaces bounded by **pixel-sharp feature edges**
- solution: process surface normal using 3D surface skeleton



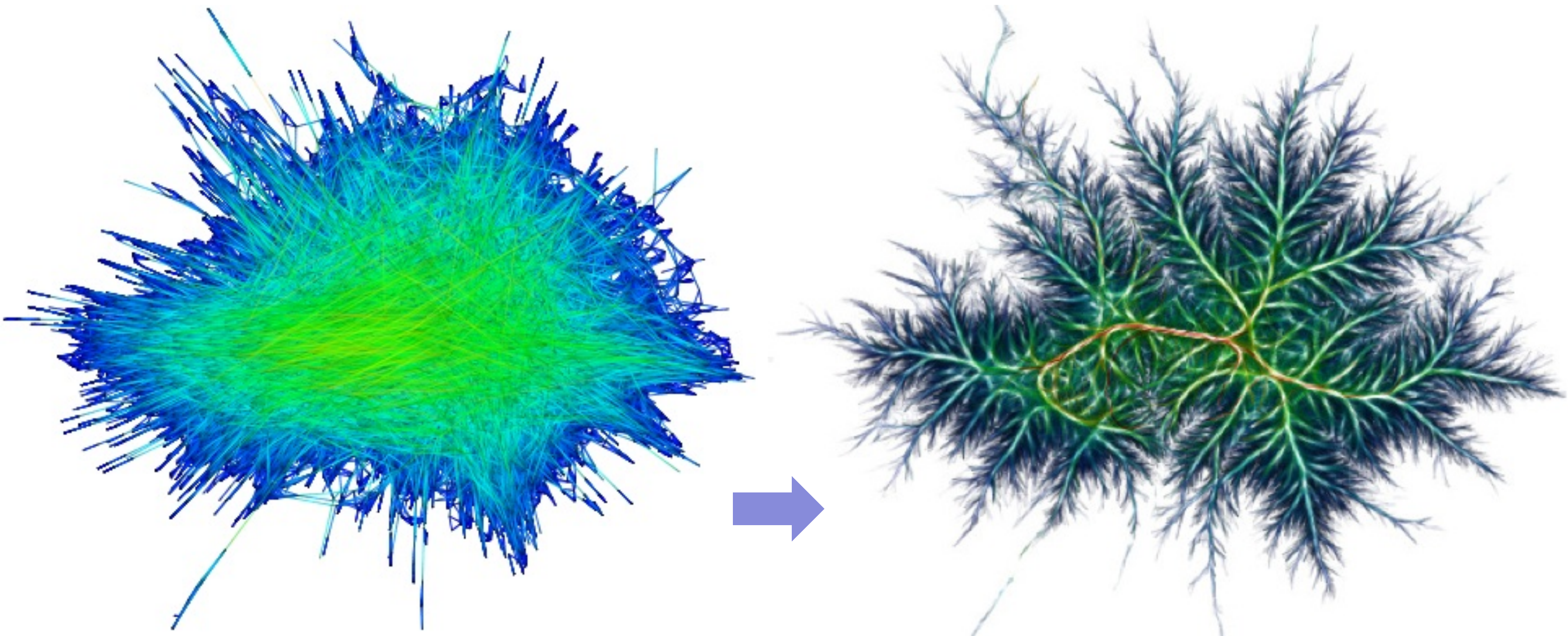
- simple (~20 lines C++)
- fast (real-time)
- intriguing…



Ongoing work

Large graph visualization

- reduce huge **graphs** to **shapes**
- encode data in shading/color

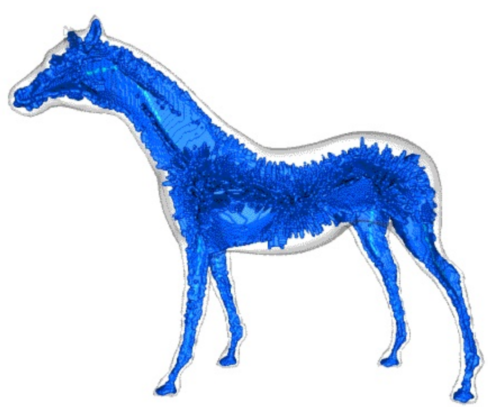


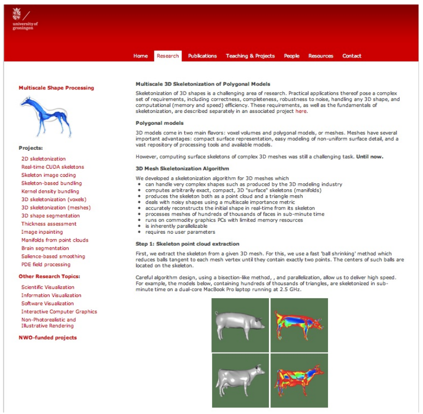
amazon graph (~1M edges): image generated in real-time (GTX 680)

Conclusions

Revisited a few ‘myths’: Skeletons are

- fundamentally **stable** and **robust** shape descriptors
- computable **accurately** in **real-time** for large shapes
- admitting a single **unified definition** in nD
- **useful** for much more than shape matching





www.cs.rug.nl/svcg/Shapes

- examples, applications
- code
- papers