Robust Inside-Outside Segmentation using Generalized Winding Numbers

Alec Jacobson — ETH Zurich

Ladislav Kavan — University of Pennsylvania

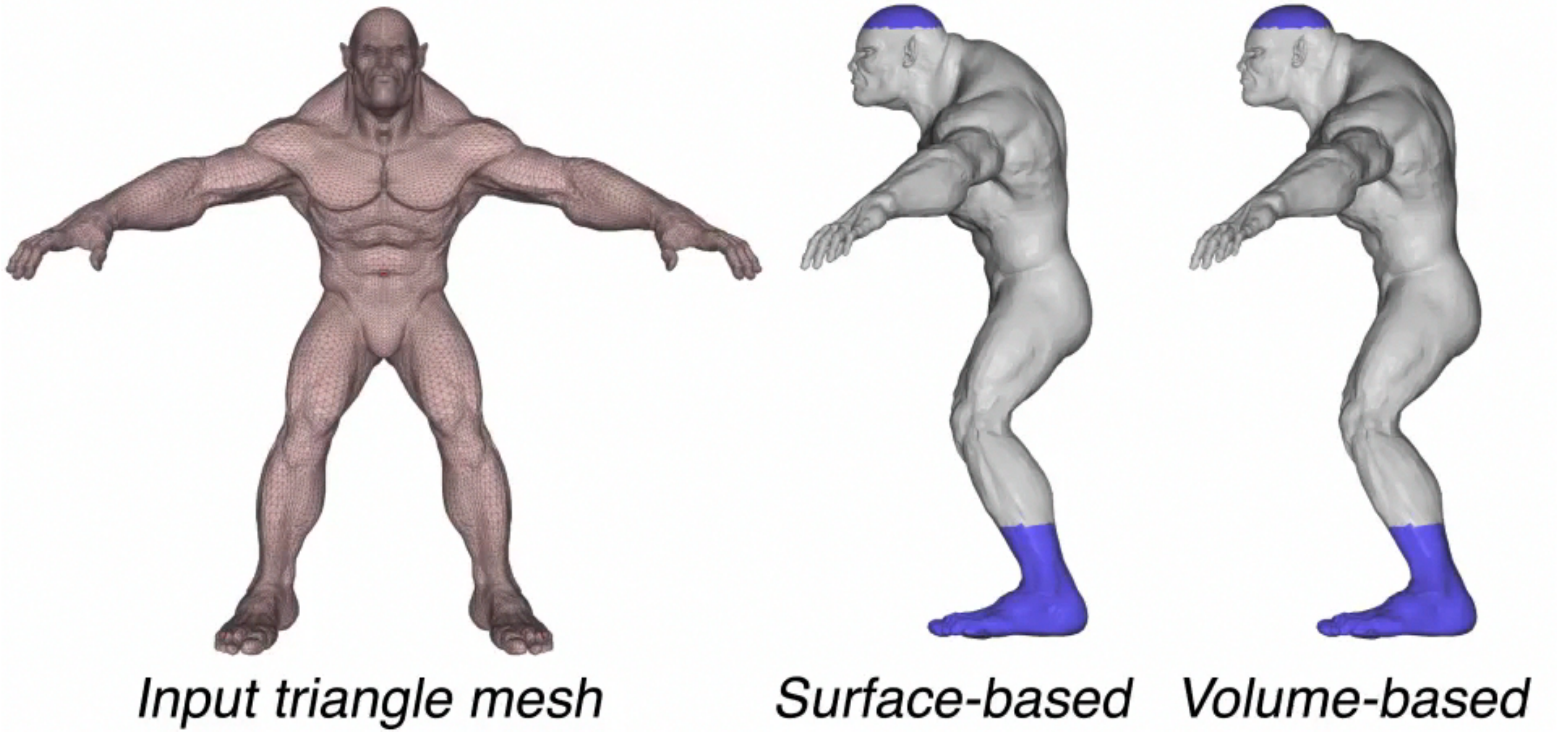
Olga Sorkine-Hornung — ETH Zurich

INTERACTIVE GEOMETRY LAB

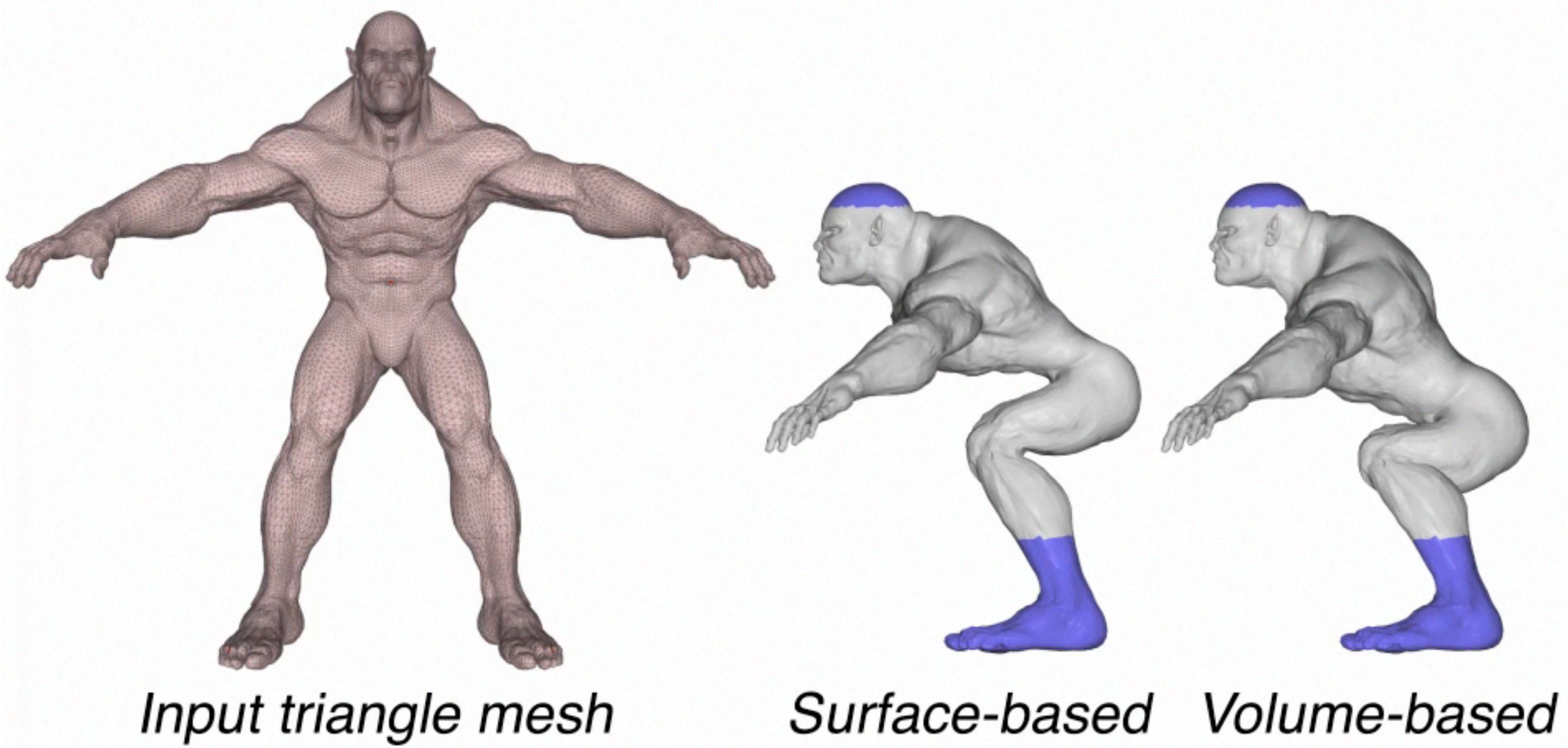
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

October 9, 2013

Processing solid shapes requires volumetric representation

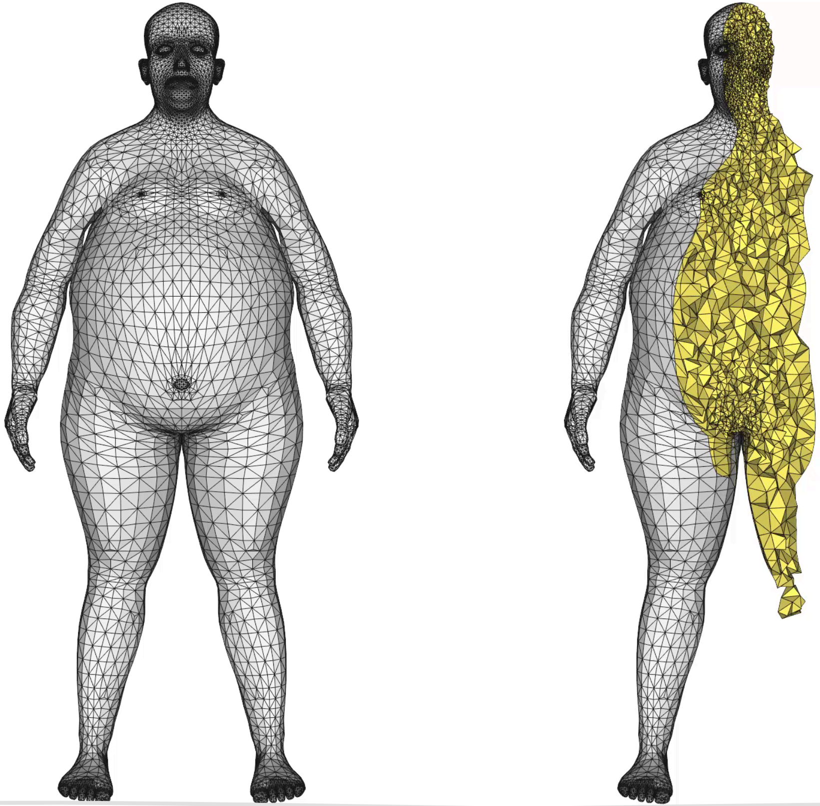


Processing solid shapes requires volumetric representation



Explicit representations are essential

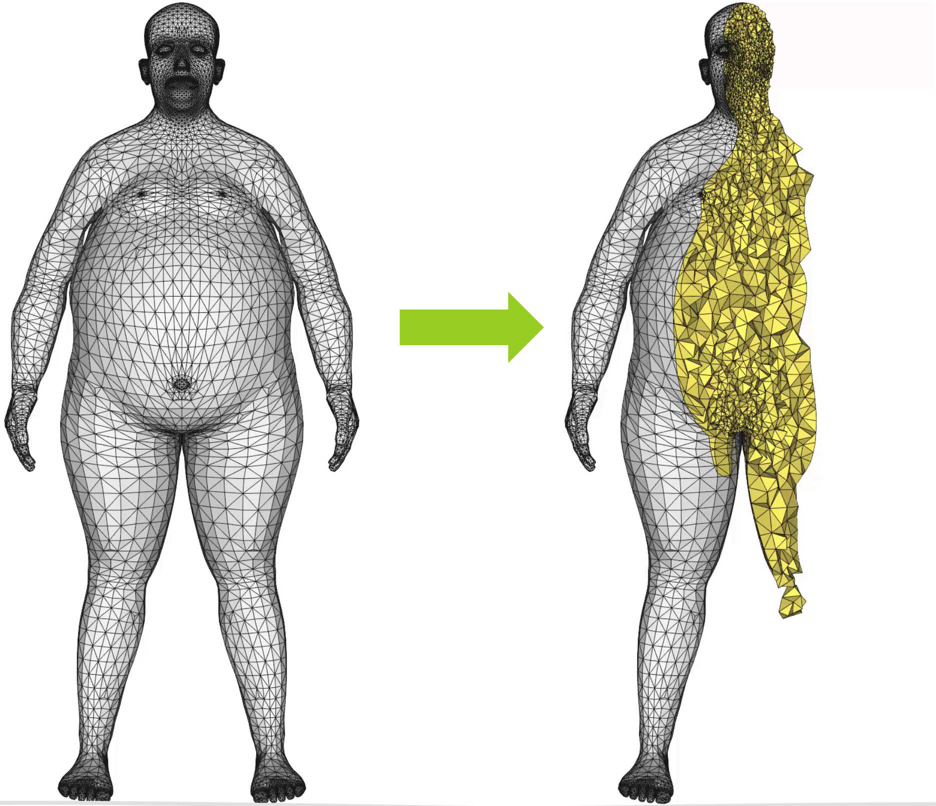
triangle mesh



tetrahedral mesh

Explicit representations are essential

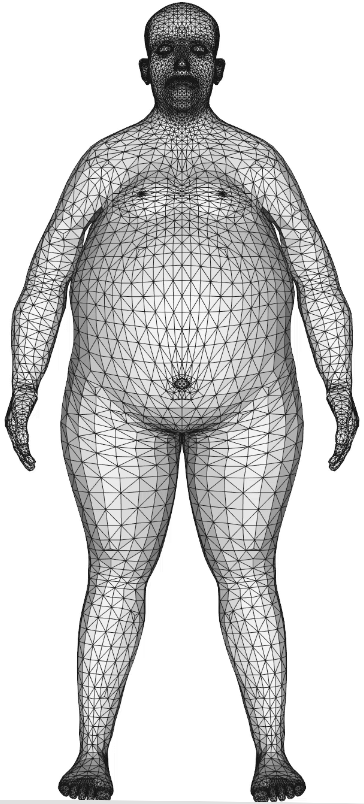
triangle mesh
watertight

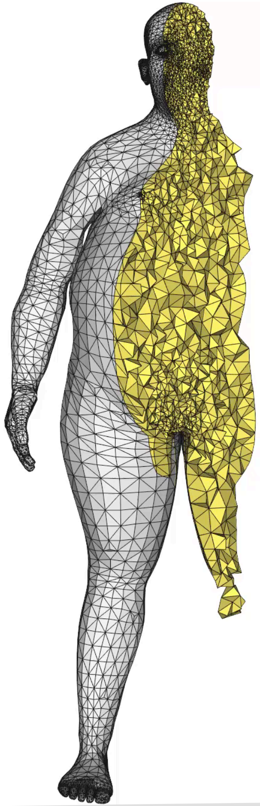


tetrahedral mesh
made by TetGen

Explicit representations are essential

triangle mesh
watertight

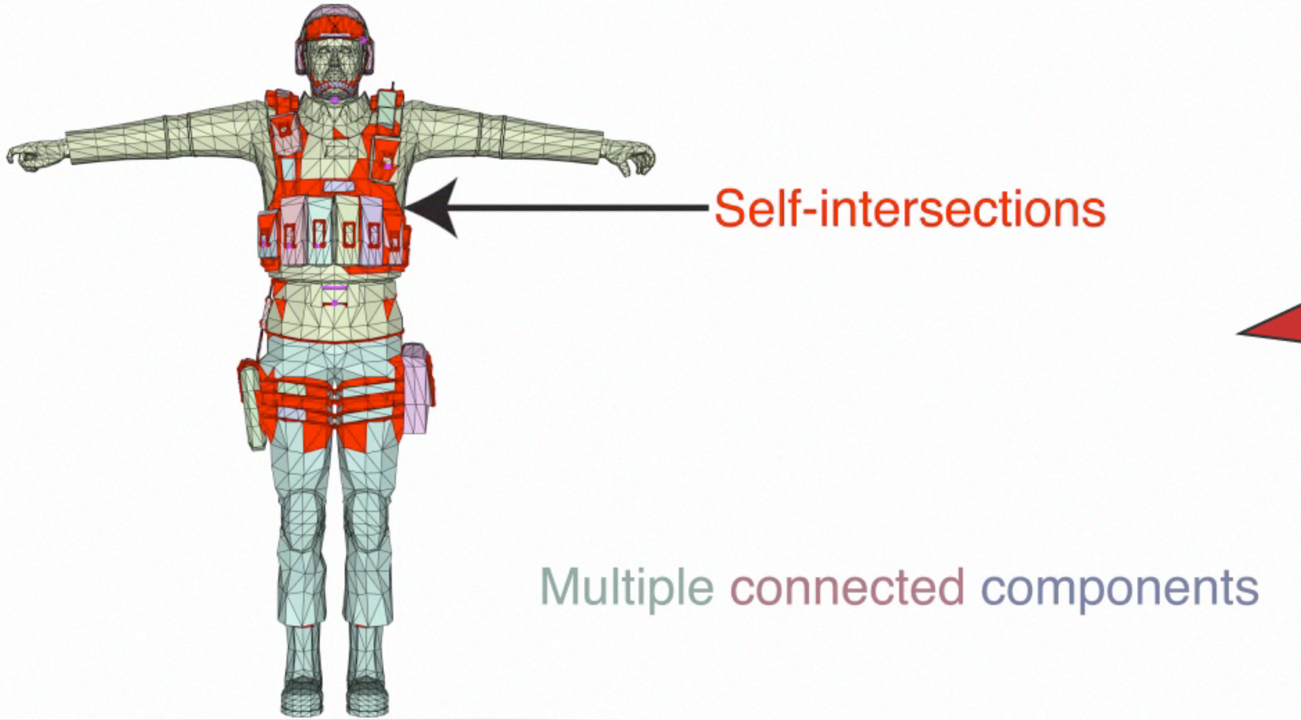


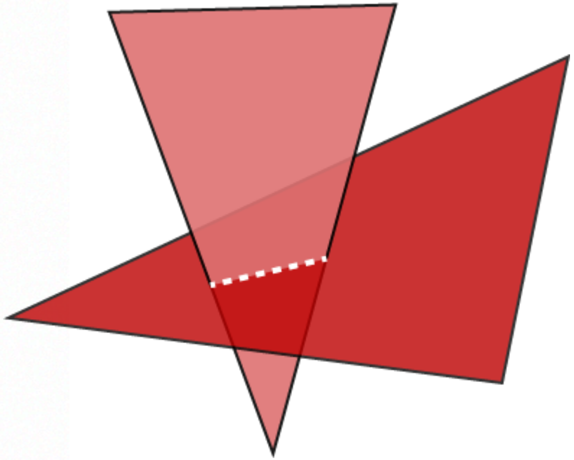


tetrahedral mesh
made by TetGen

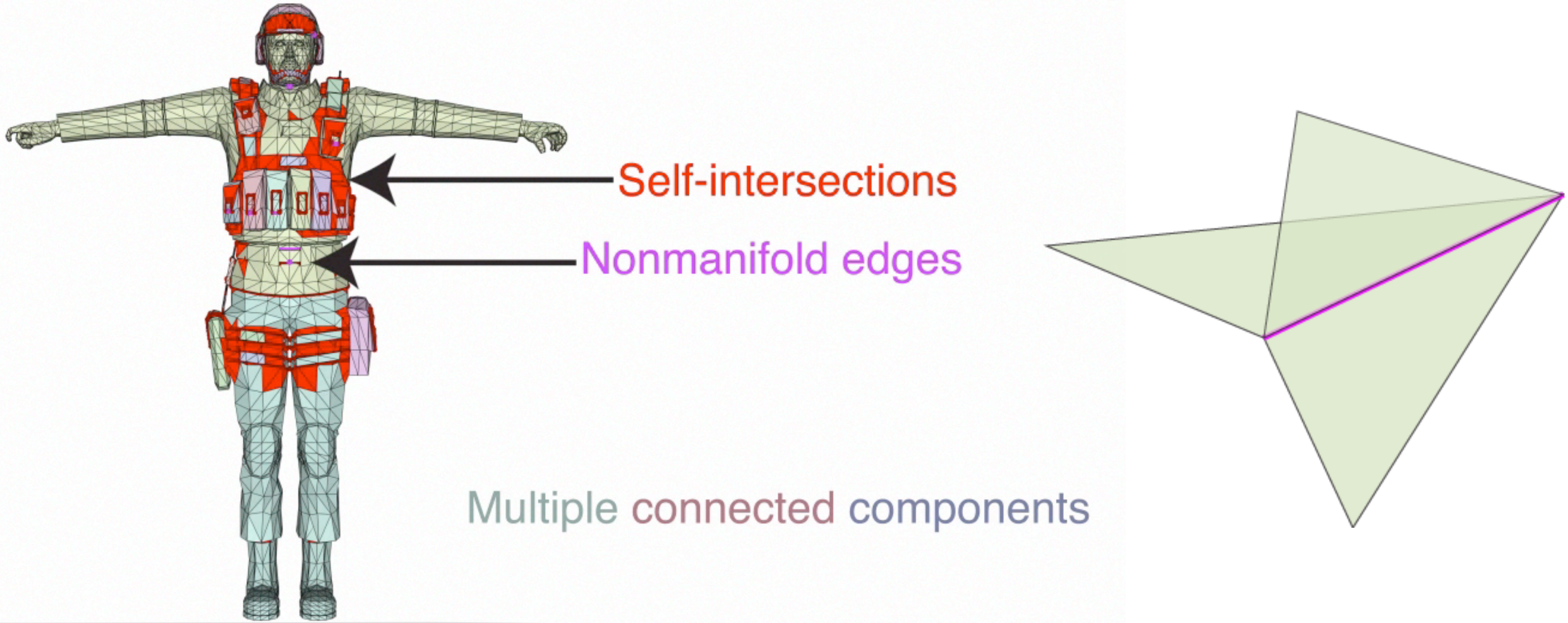
quality elements
varying density
conform to input

Apparent surface descriptions of solids are *unmeshable* with current tools

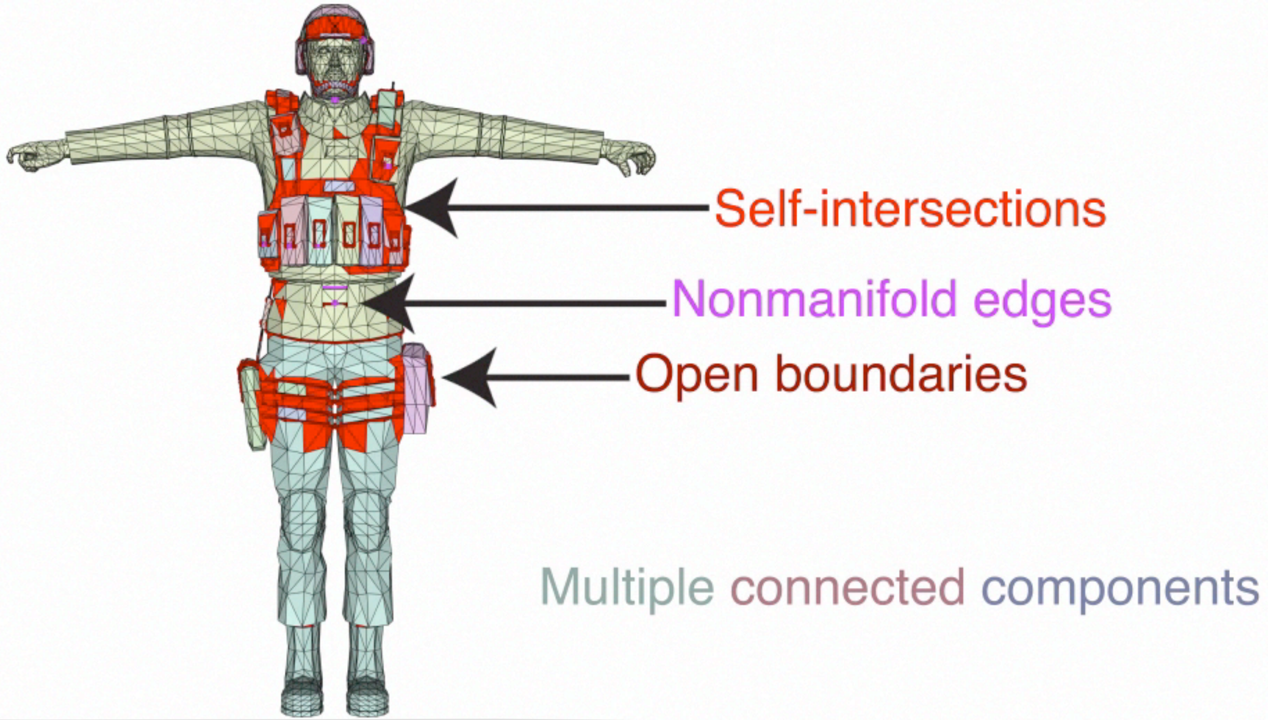


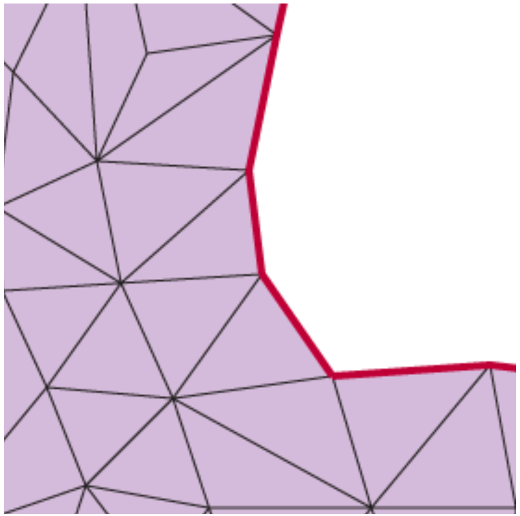


Apparent surface descriptions of solids are *unmeshable* with current tools



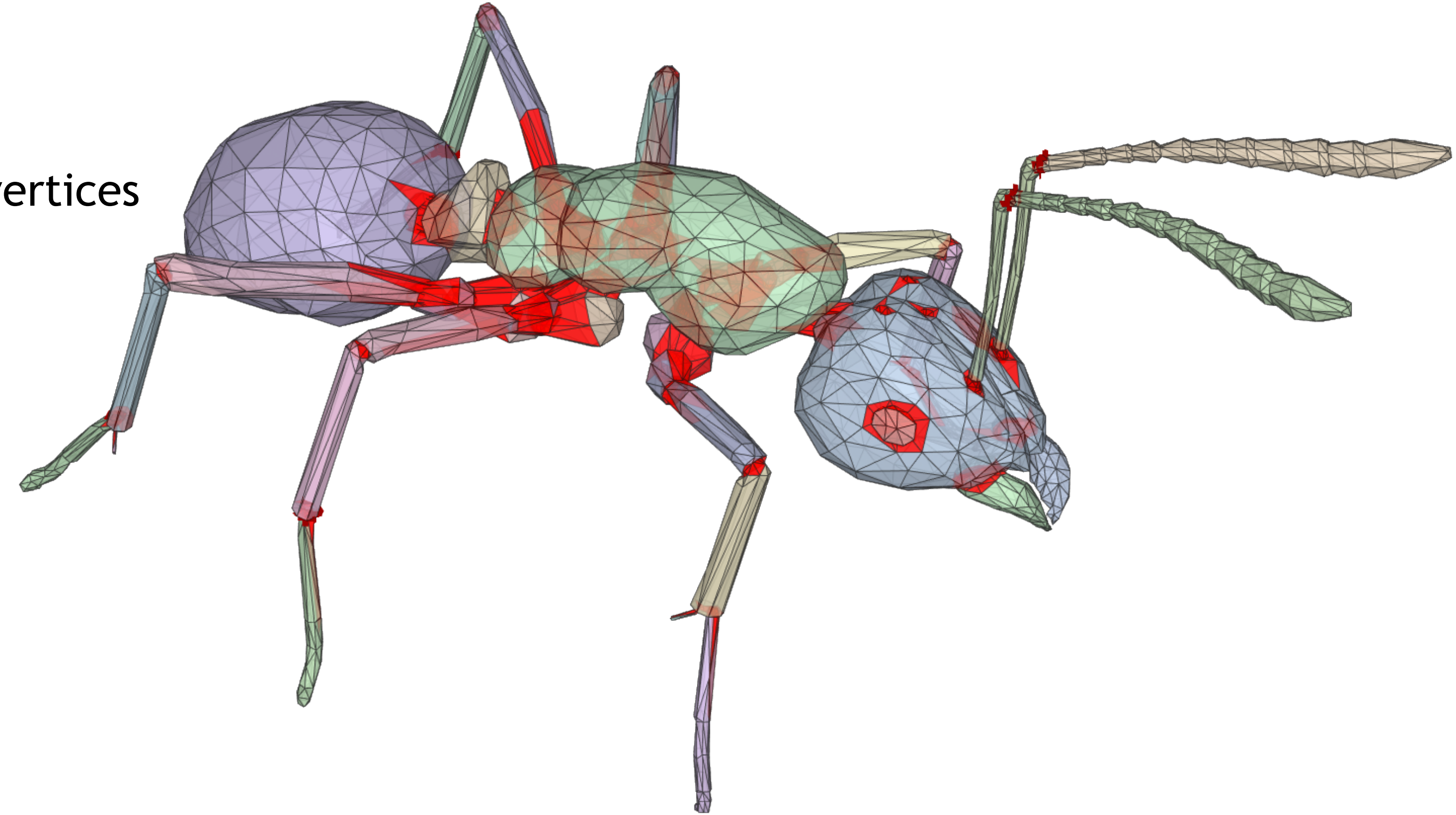
Apparent surface descriptions of solids are *unmeshable* with current tools



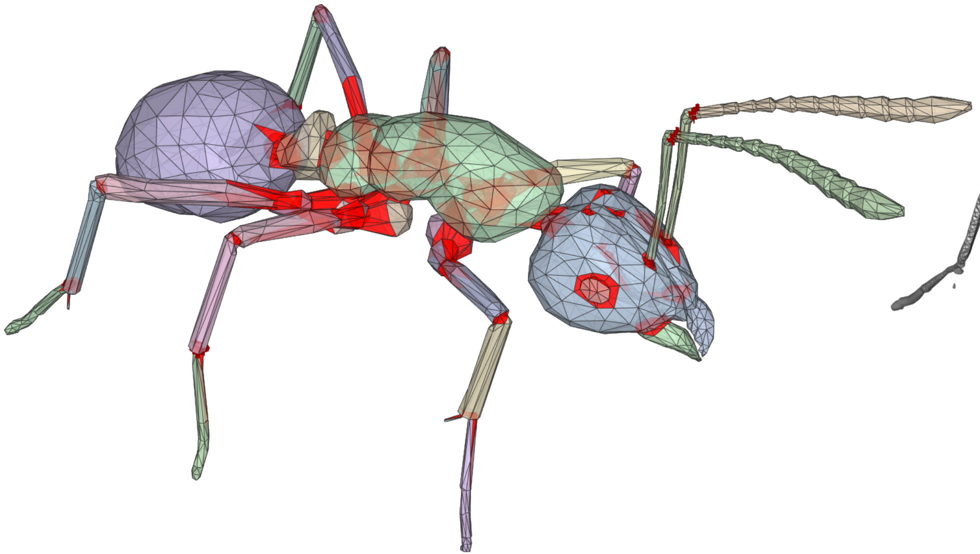


Meshes are often output of human creativity

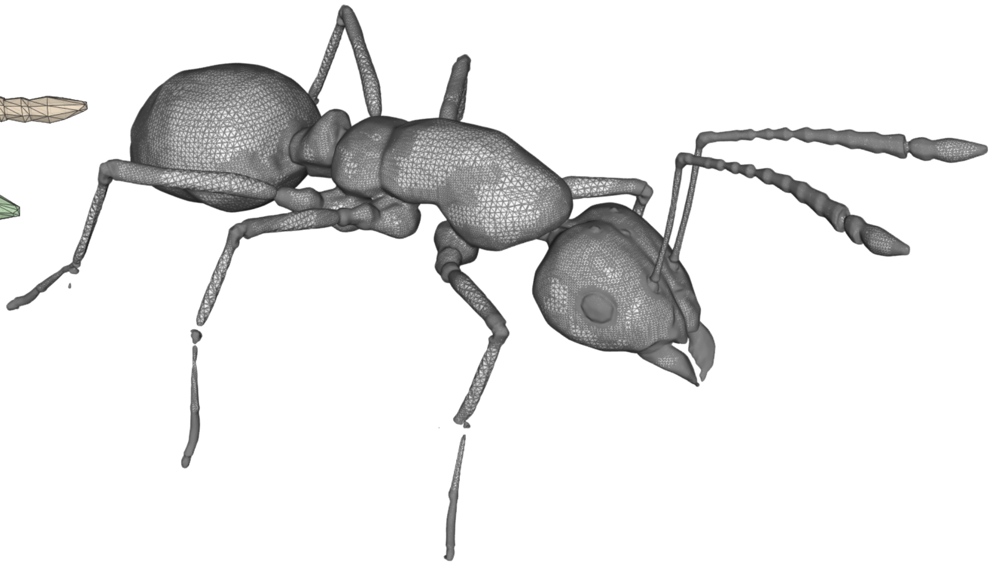
only 4000 vertices



Treating as scanned objects is inappropriate

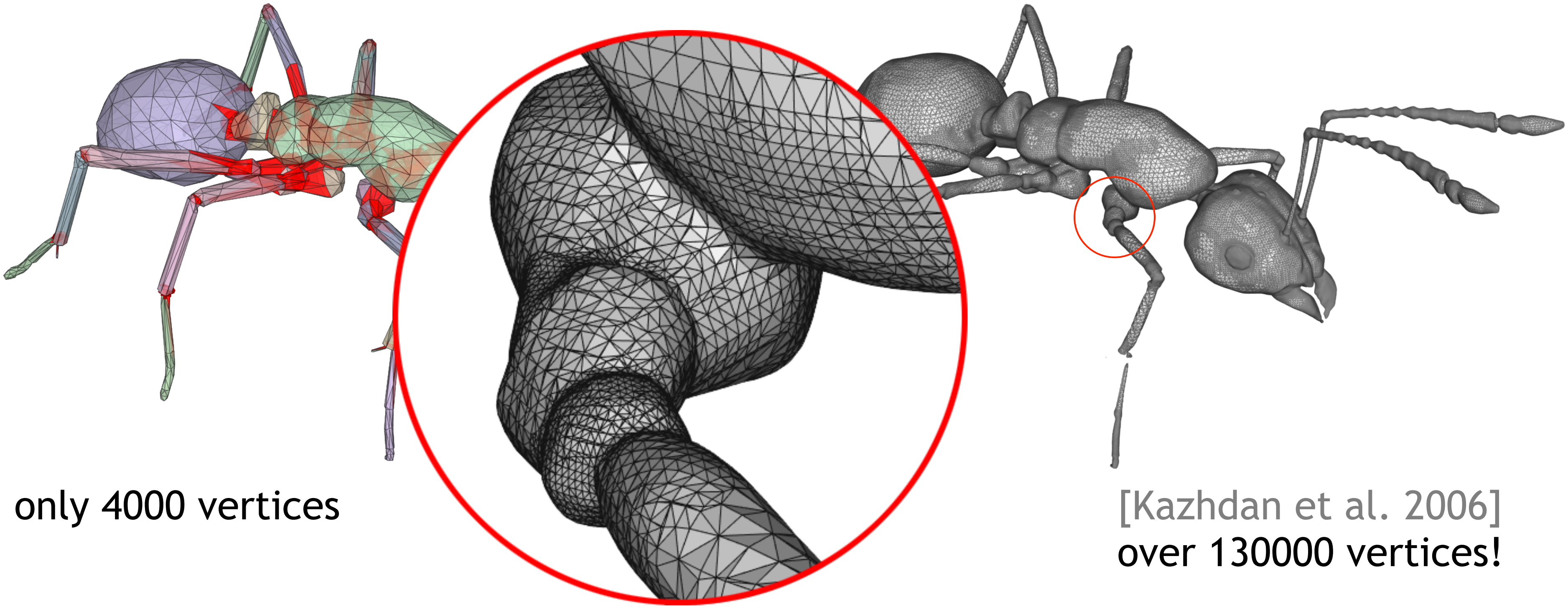


only 4000 vertices



[Kazhdan et al. 2006]
over 130000 vertices!

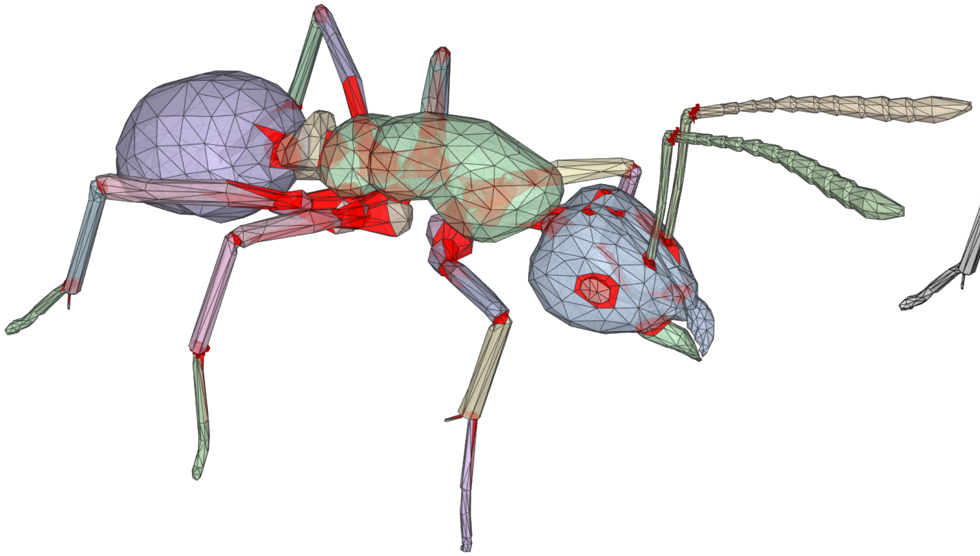
Treating as scanned objects is inappropriate



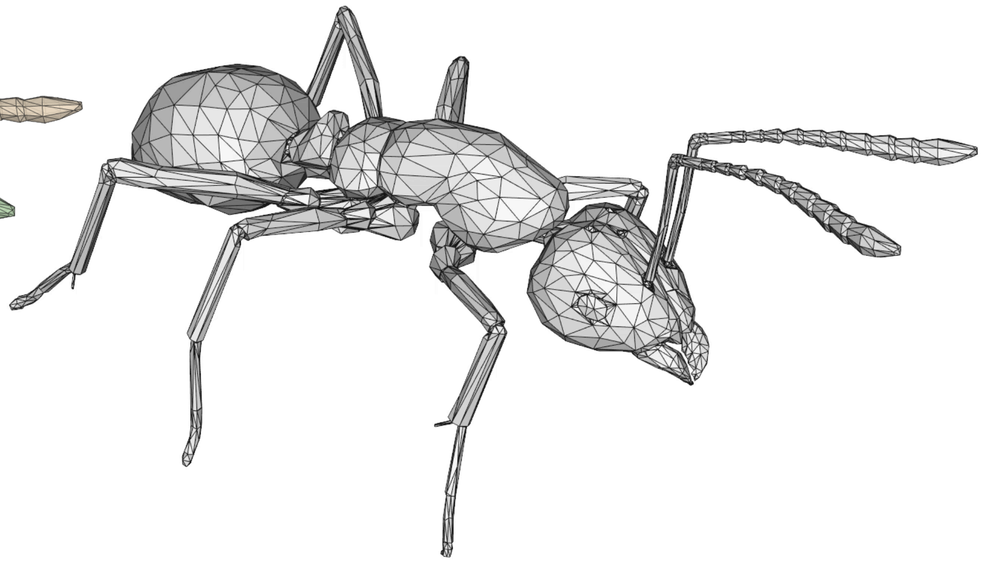
only 4000 vertices

[Kazhdan et al. 2006]
over 130000 vertices!

Volume mesh should conform to input

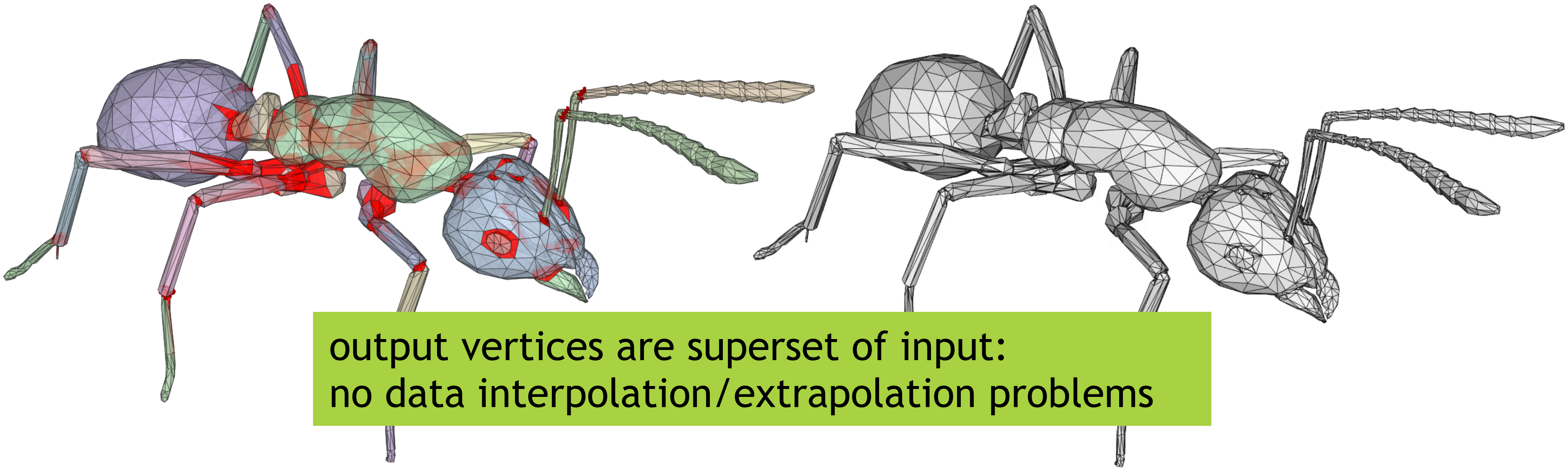


only 4000 vertices



our output tet mesh
only 4500 vertices

Volume mesh should conform to input

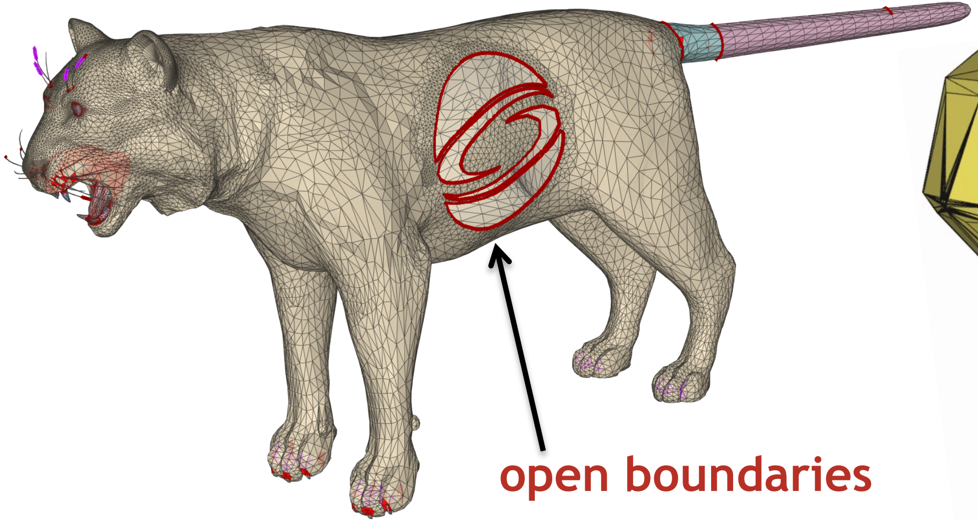


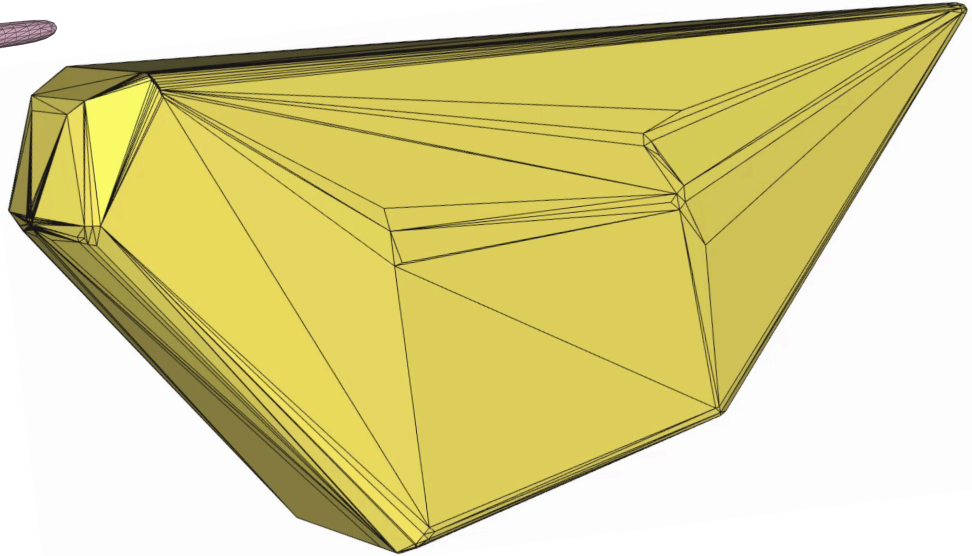
output vertices are superset of input:
no data interpolation/extrapolation problems

only 4000 vertices

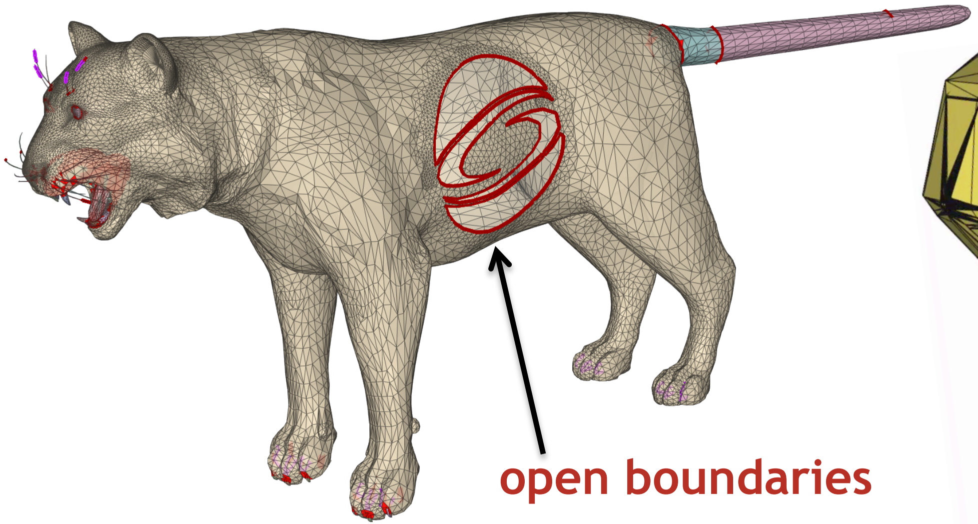
our output tet mesh
only 4500 vertices

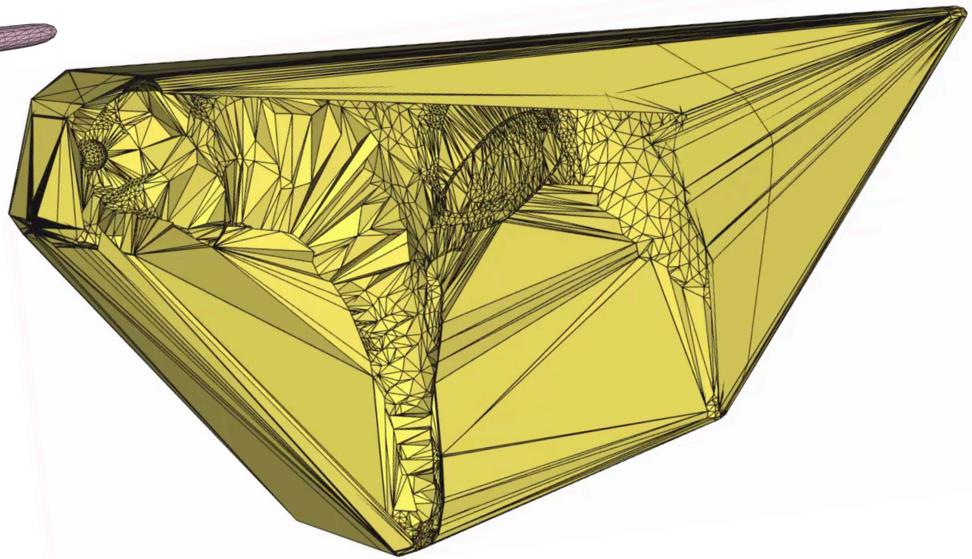
Can mesh the entire convex hull, but what's inside? What's outside?



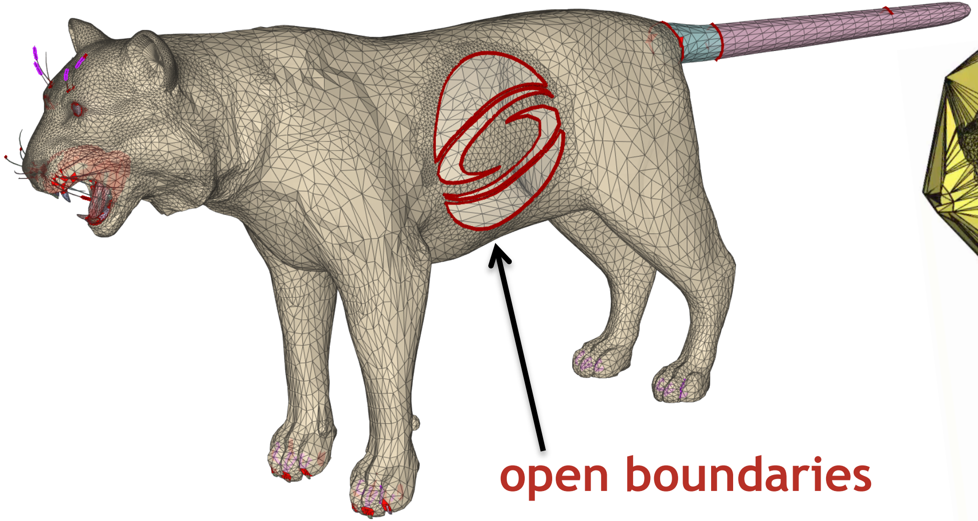


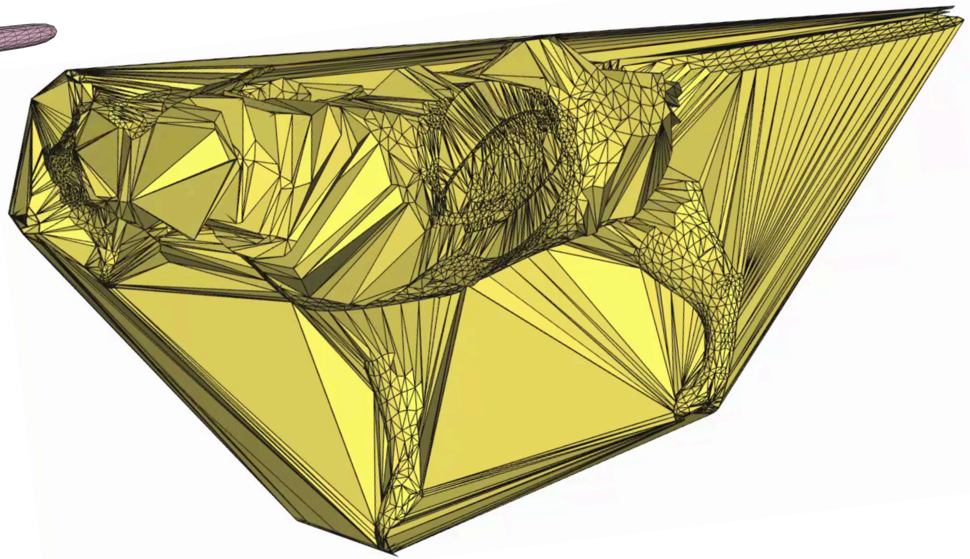
Can mesh the entire convex hull, but what's inside? What's outside?



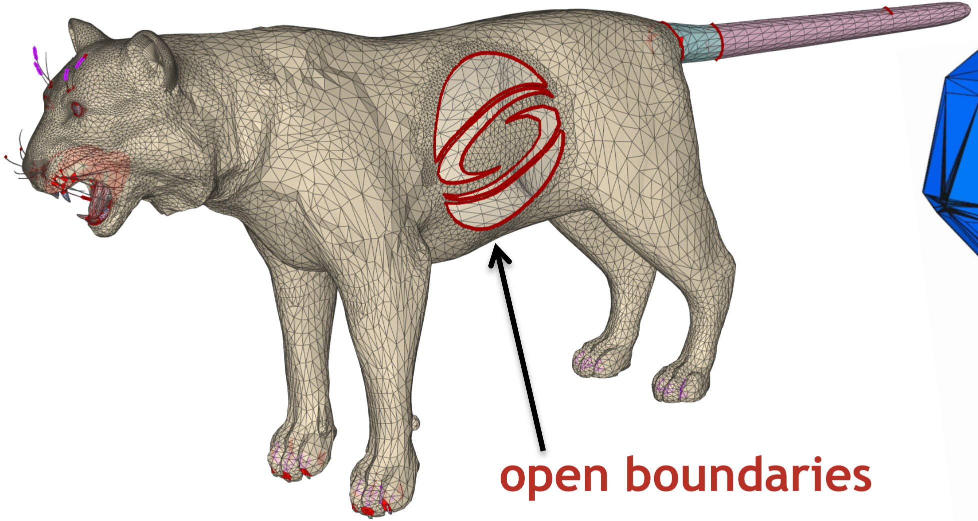


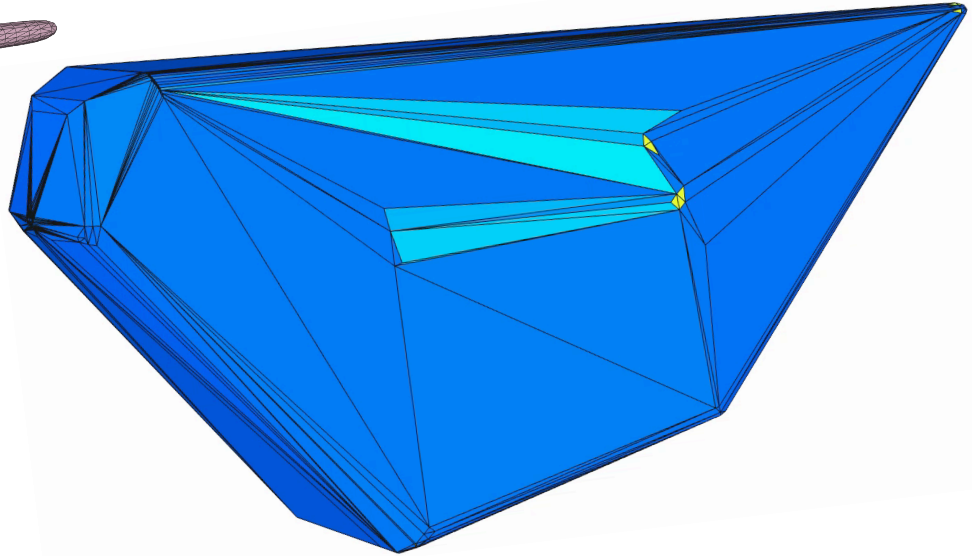
Can mesh the entire convex hull, but what's inside? What's outside?



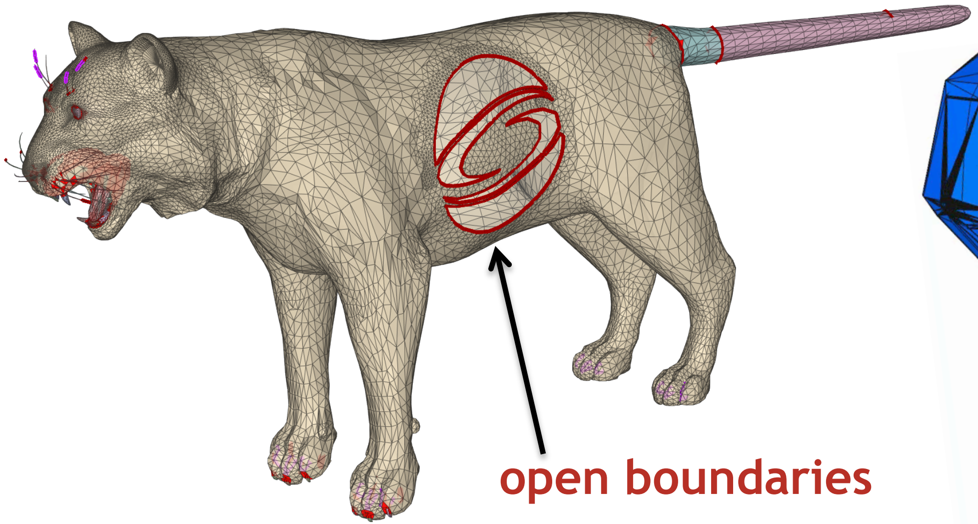


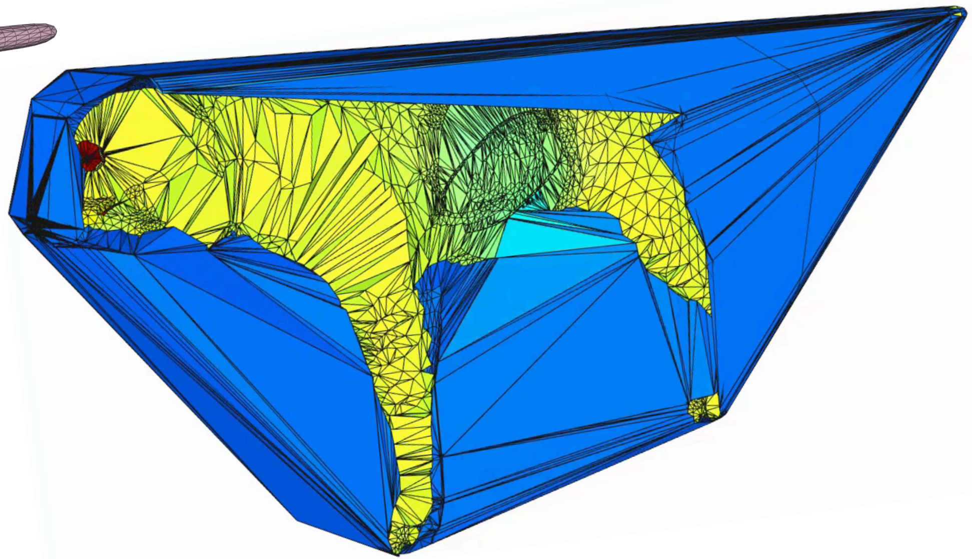
Generalized function indicates *insideness*



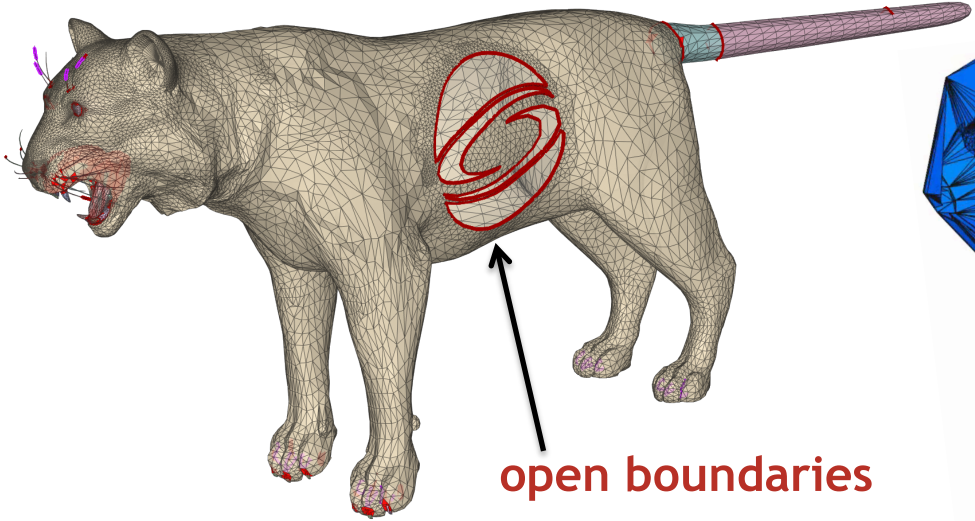


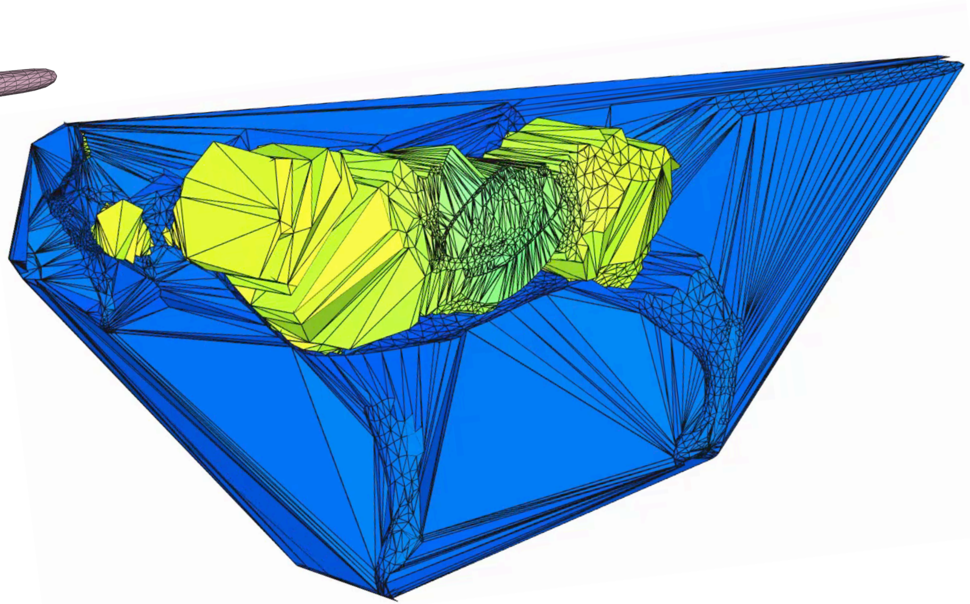
Generalized function indicates *insideness*



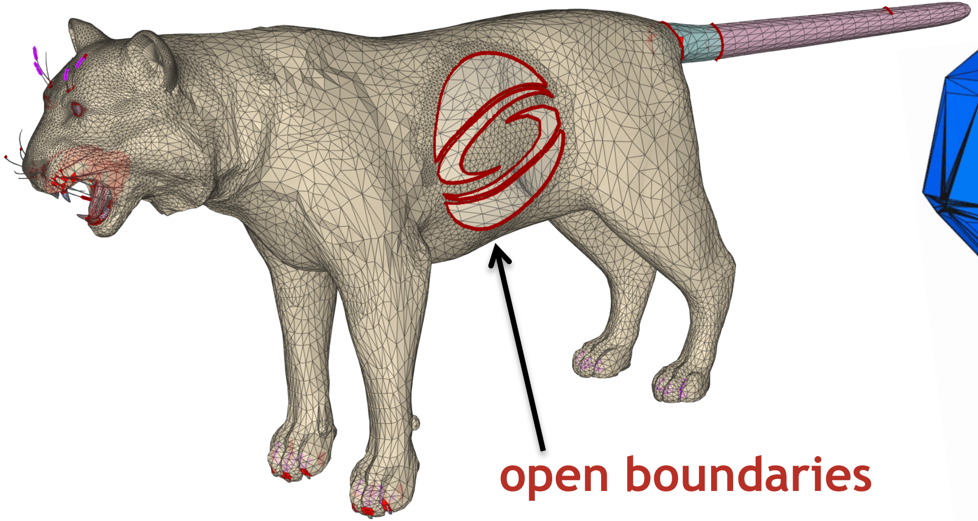


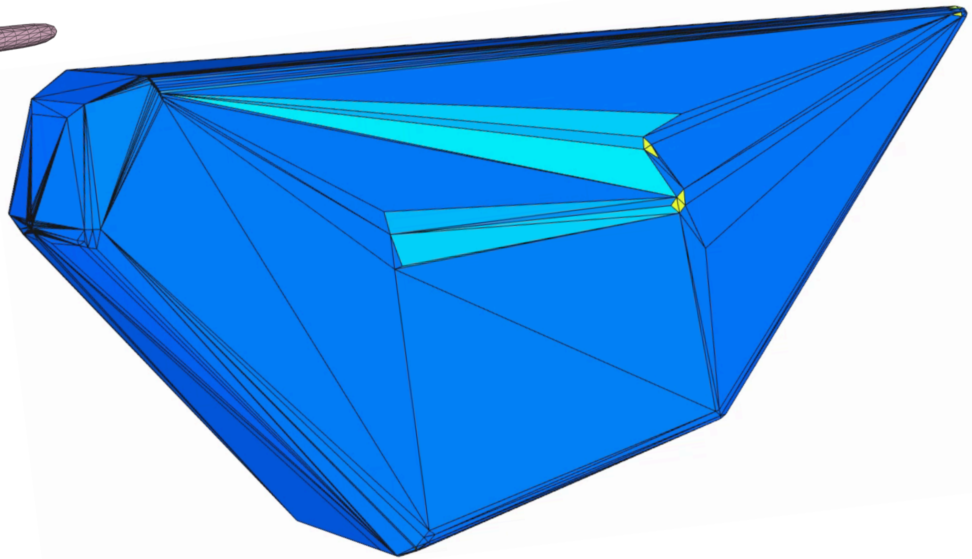
Generalized function indicates *insideness*



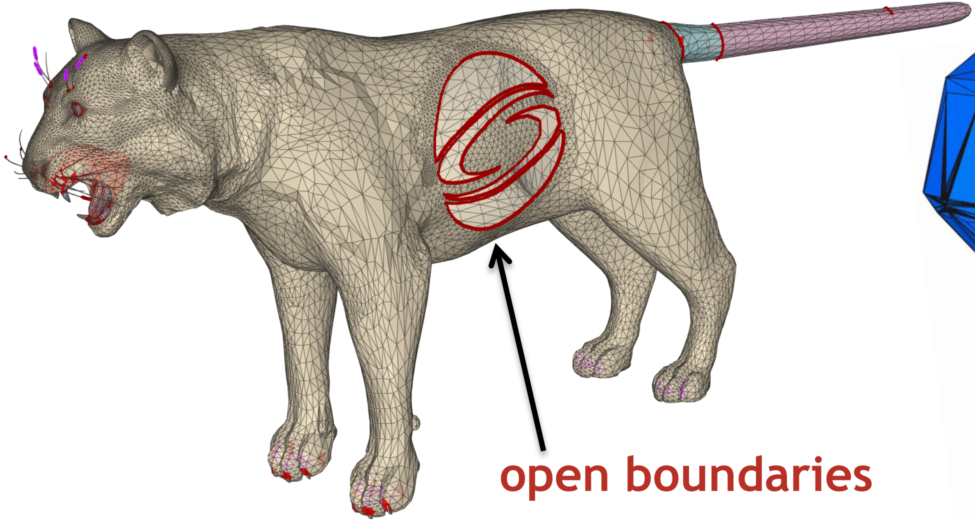


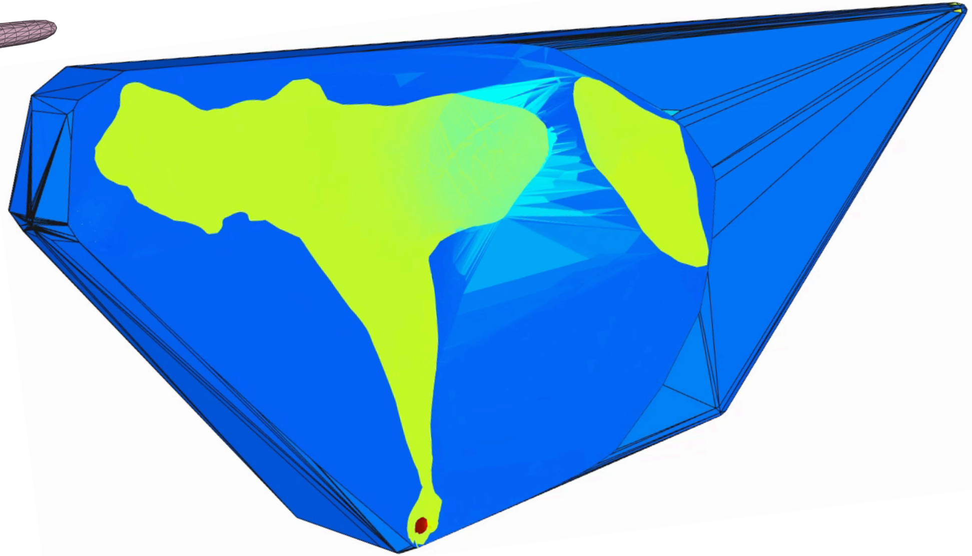
Generalized function indicates *insideness*



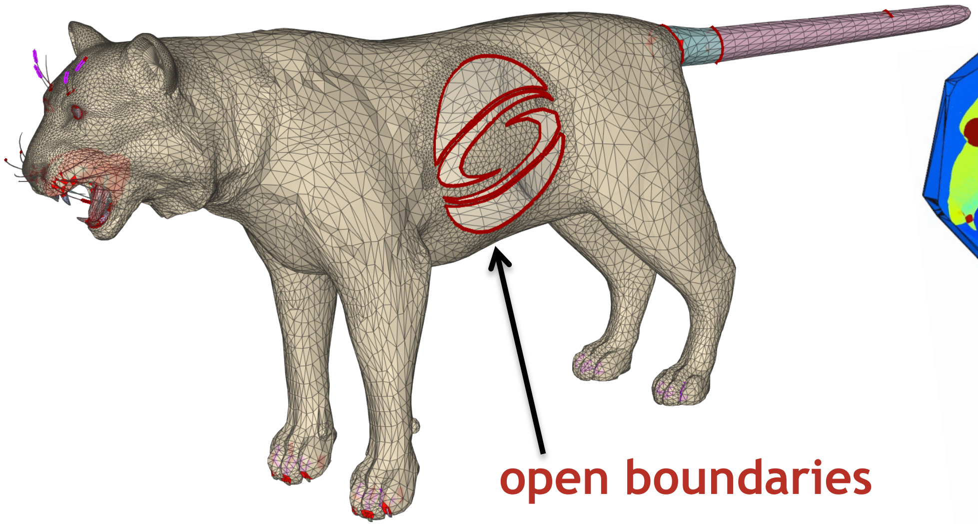


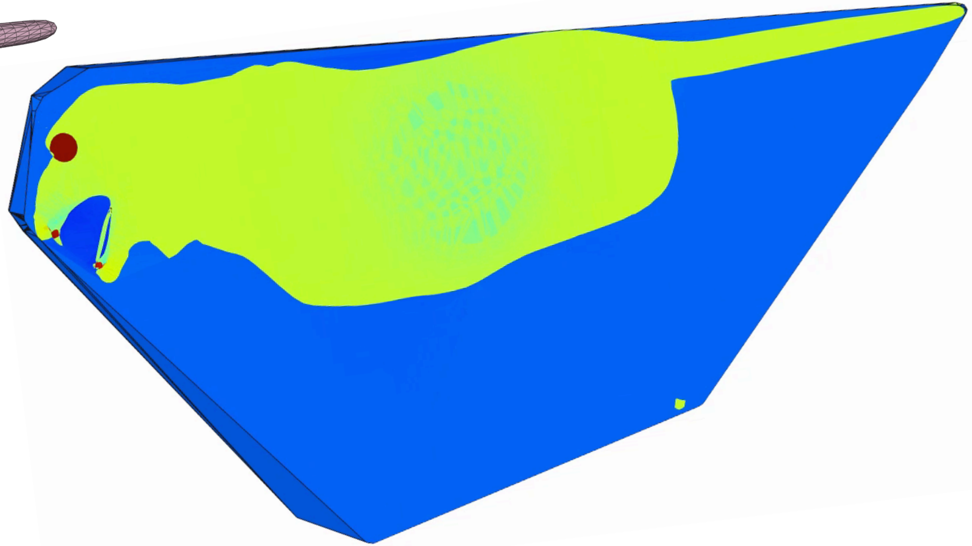
Generalized function indicates *insideness*



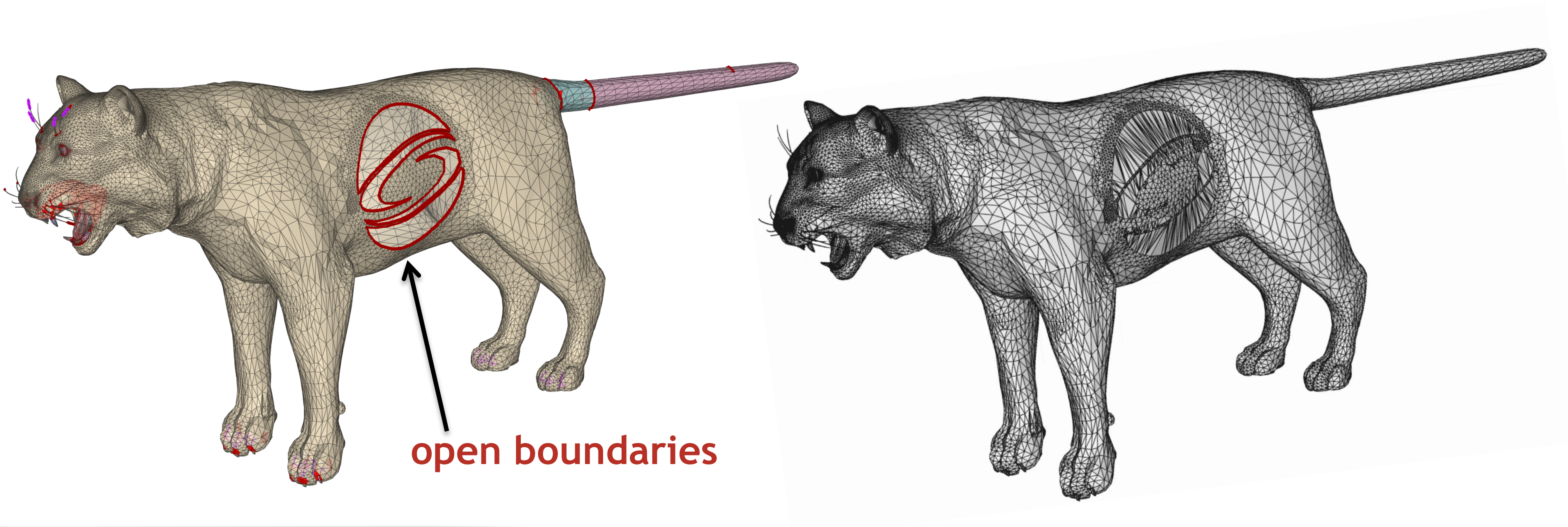


Generalized function indicates *insideness*

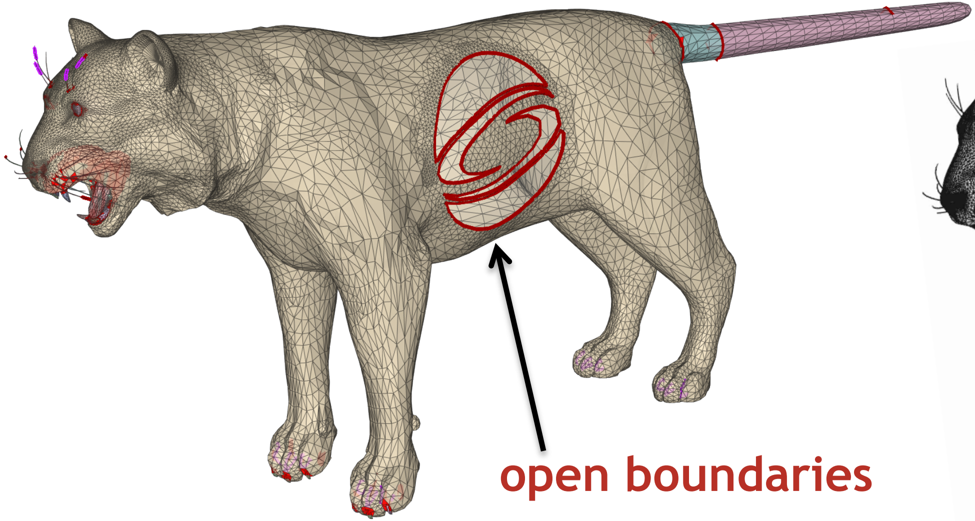


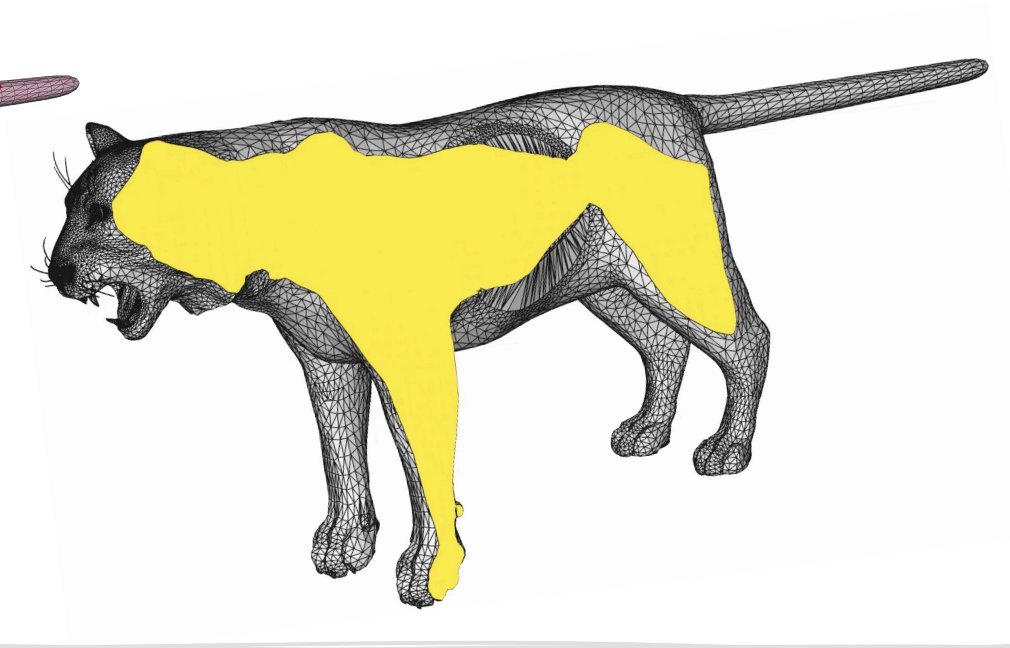


Function guides a crisp segmentation

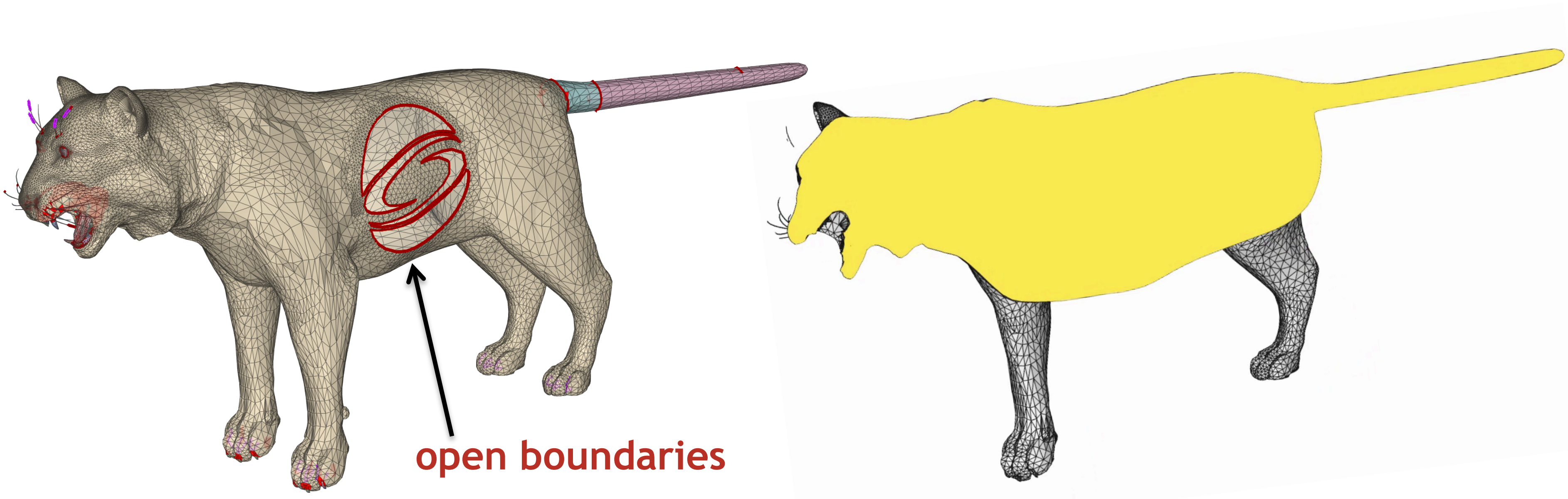


Function guides a crisp segmentation



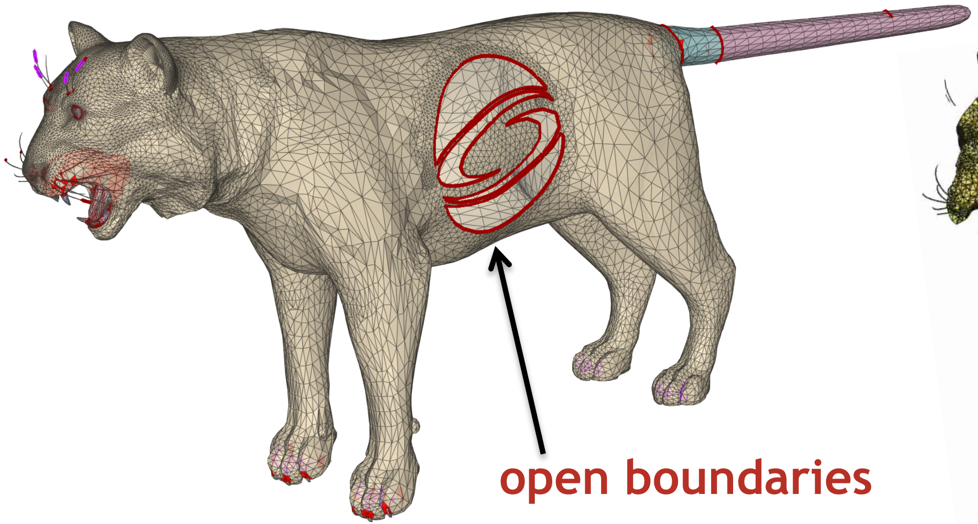


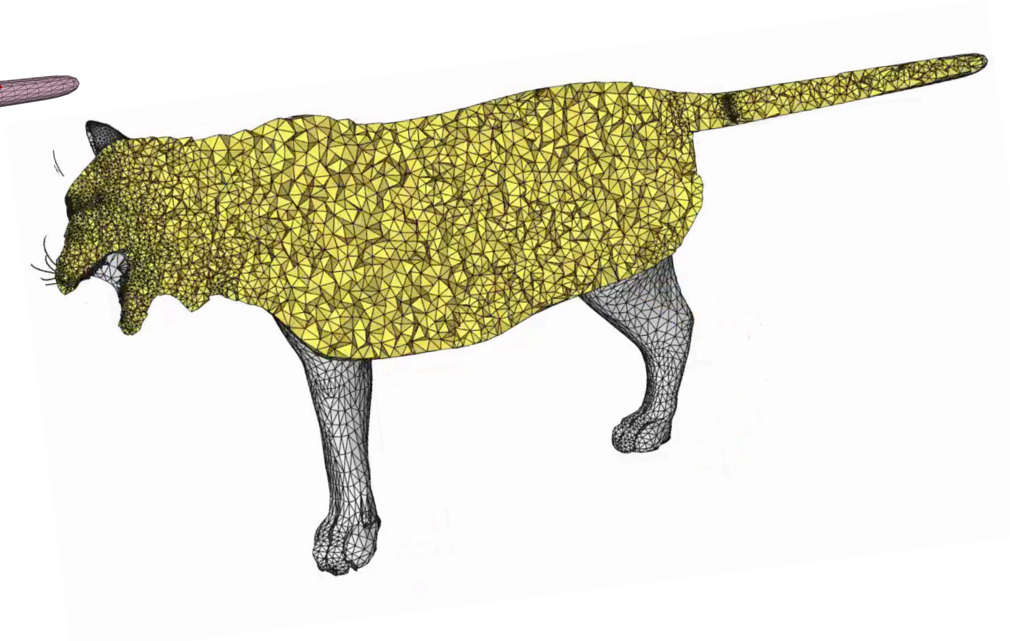
Function guides a crisp segmentation



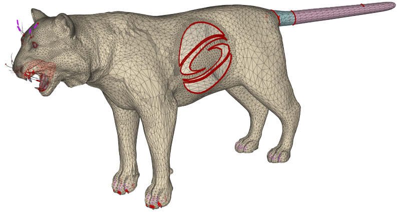
Output is minimal, ripe for post-processing

Refined mesh using TETGEN, STELLAR, etc.



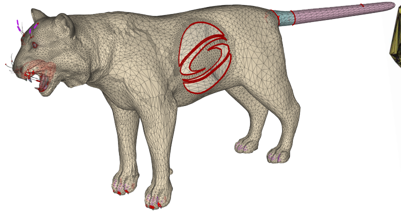


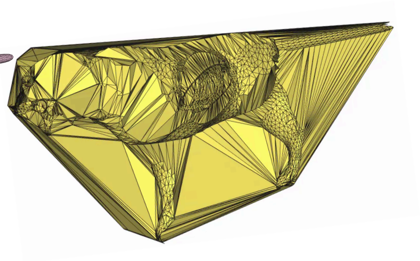
Idea: mesh entire convex hull, segment inside tets from outside ones



input triangles

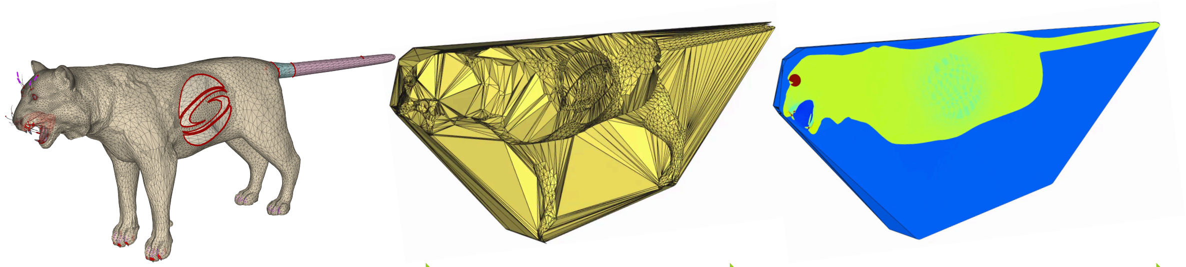
Idea: mesh entire convex hull, segment inside tets from outside ones





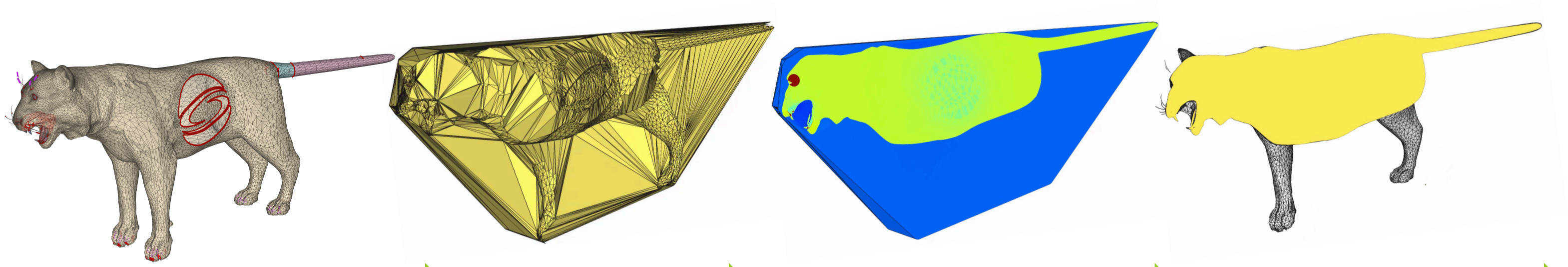
input triangles → CDT →

Idea: mesh entire convex hull, segment inside tets from outside ones



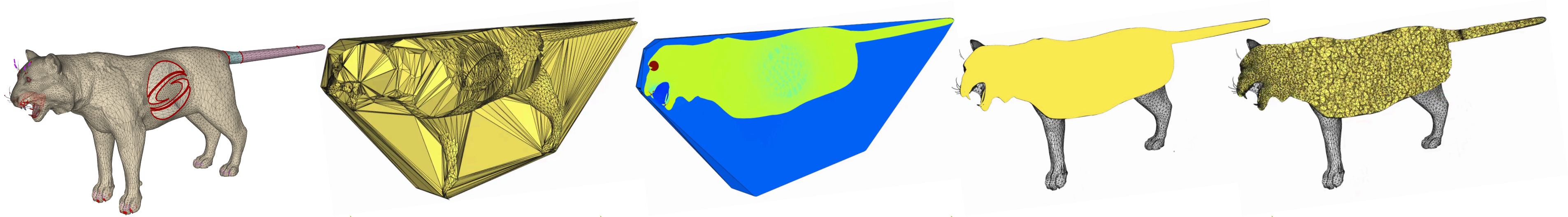
input triangles → CDT → measure of insideness →

Idea: mesh entire convex hull, segment inside tets from outside ones



input triangles → CDT → measure of insideness → segment CDT →

Idea: mesh entire convex hull, segment inside tets from outside ones

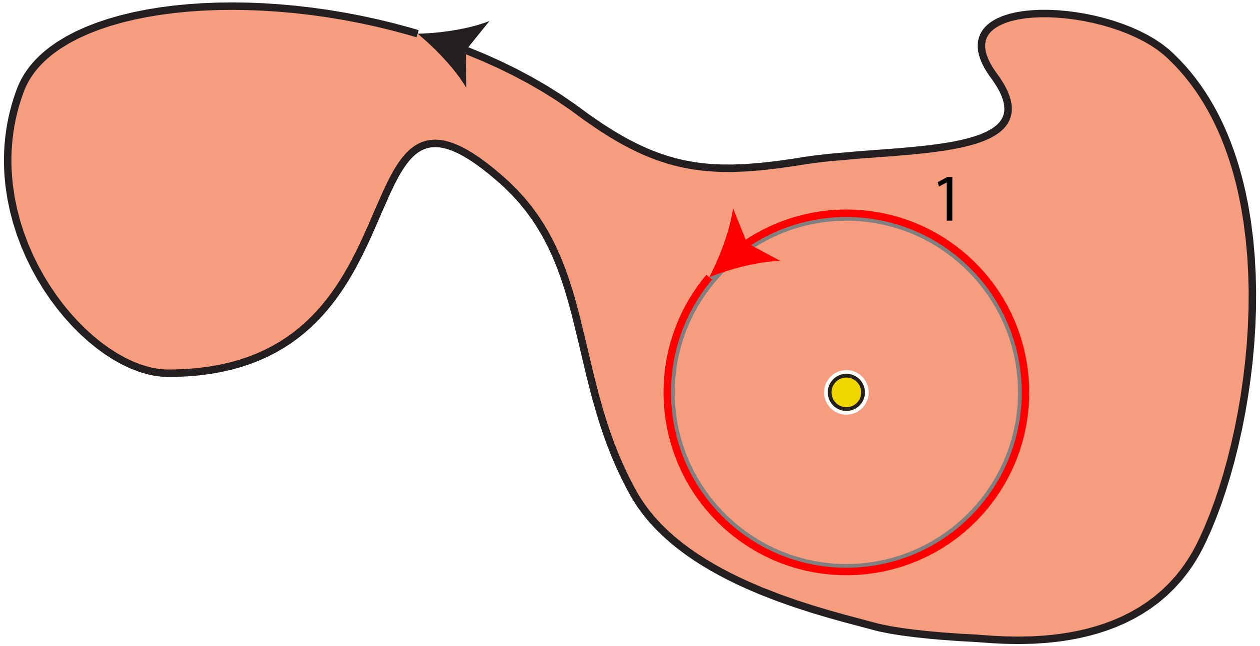


input triangles → CDT → measure of insideness → segment CDT → refine, etc.

Idea: mesh entire convex hull, segment inside tets from outside ones

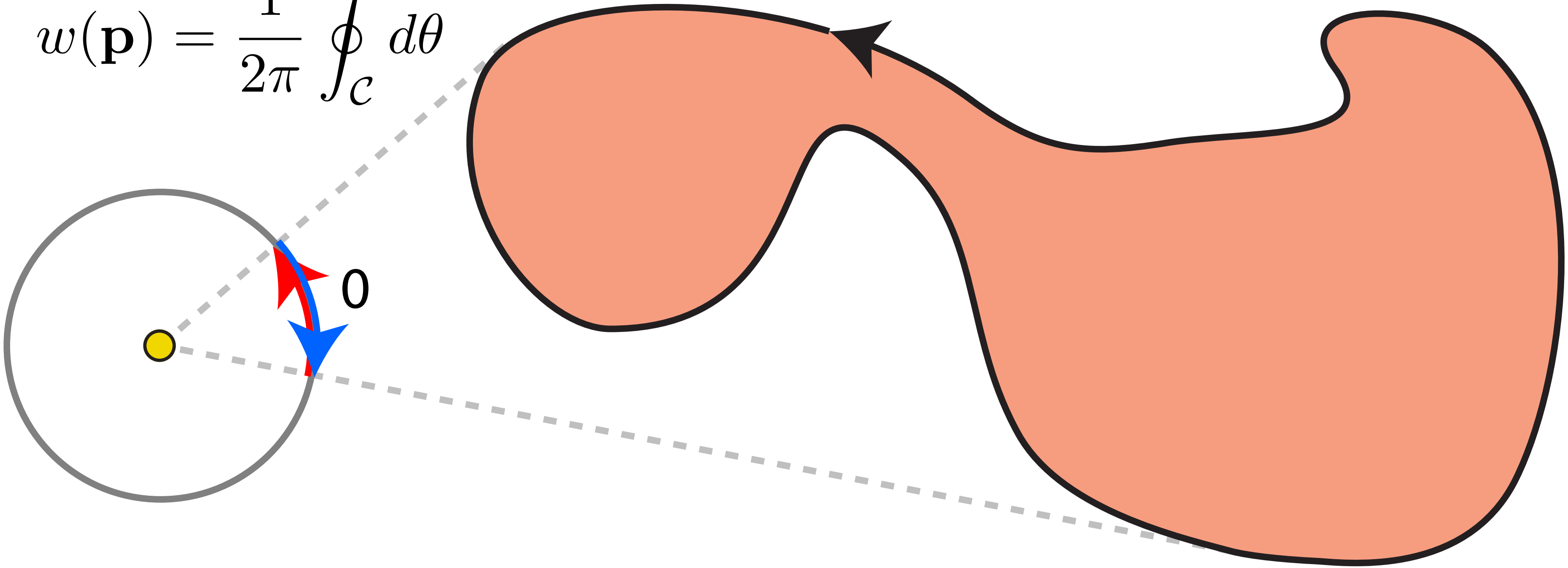
If shape is watertight, winding number is perfect measure of inside

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



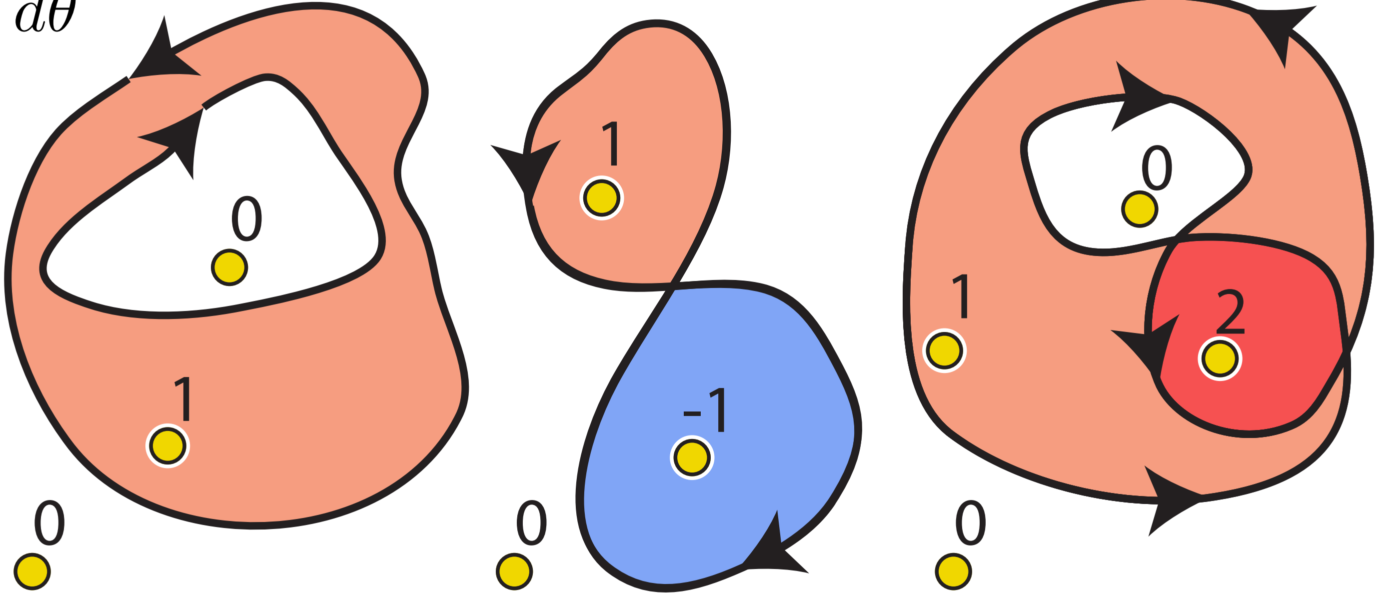
If shape is watertight, winding number is perfect measure of inside

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



Winding number uses orientation to treat insideness as *signed integer*

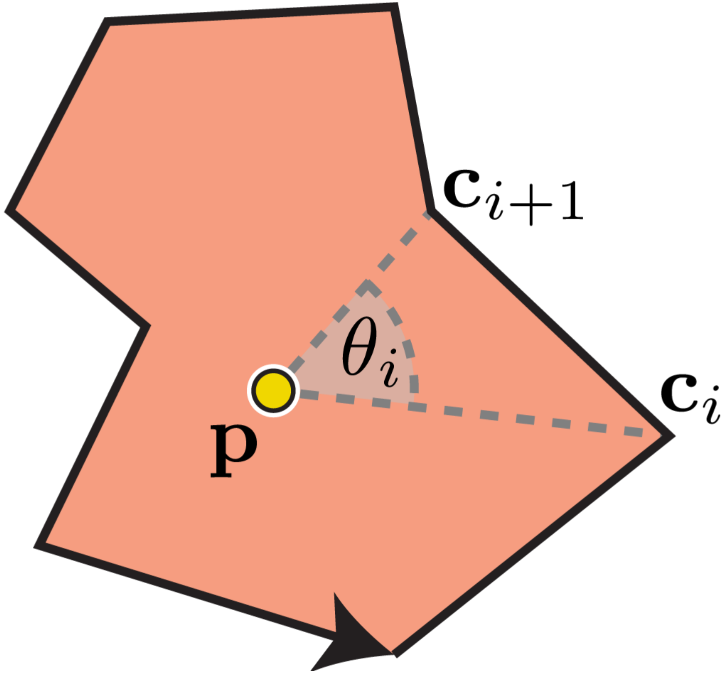
$$w(\mathbf{p}) = \frac{1}{2\pi}\oint_C d\theta$$



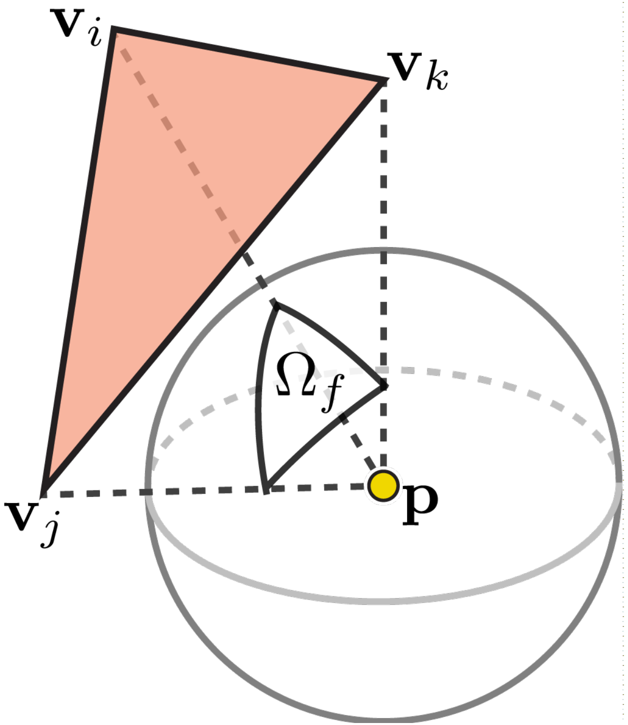
Naive discretization is simple and exact

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$

$$w(\mathbf{p}) = \frac{1}{2\pi} \sum_{i=1}^{n} \theta_i$$



Generalizes elegantly to 3D via solid angle

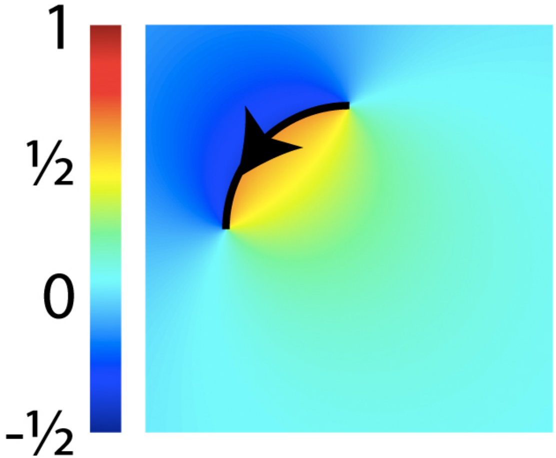


$$w(\mathbf{p}) = \frac{1}{4\pi}\iint_S \sin(\phi)\,d\theta\,d\phi$$

$$w(\mathbf{p}) = \frac{1}{4\pi}\sum_{f=1}^{m}\Omega_f$$

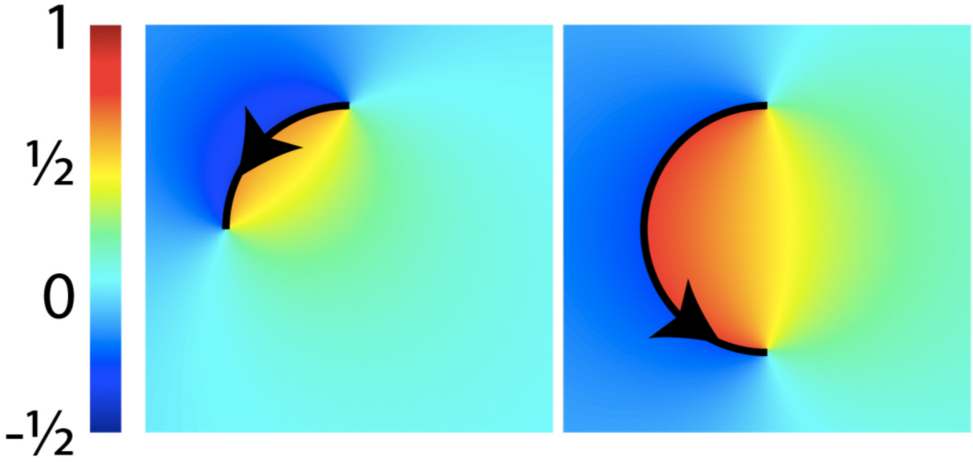
What happens if the shape is open?

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



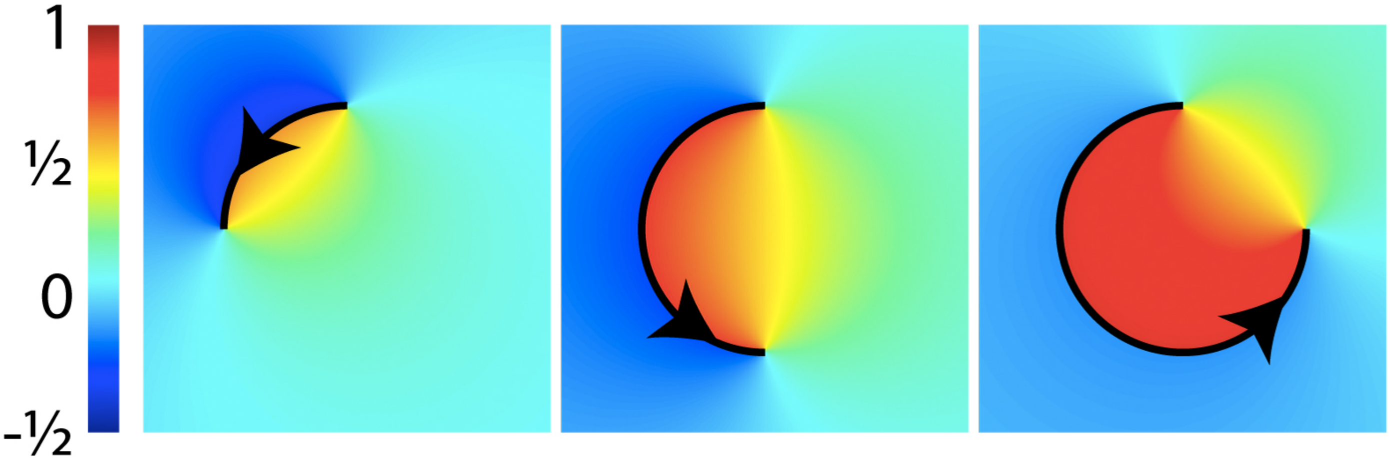
What happens if the shape is open?

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



What happens if the shape is open?

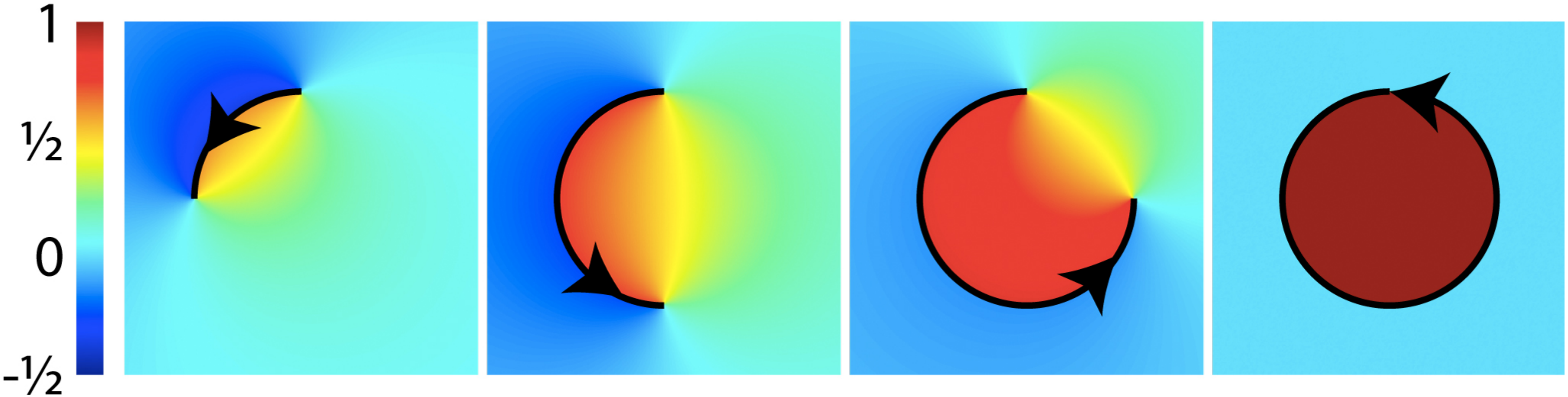
$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

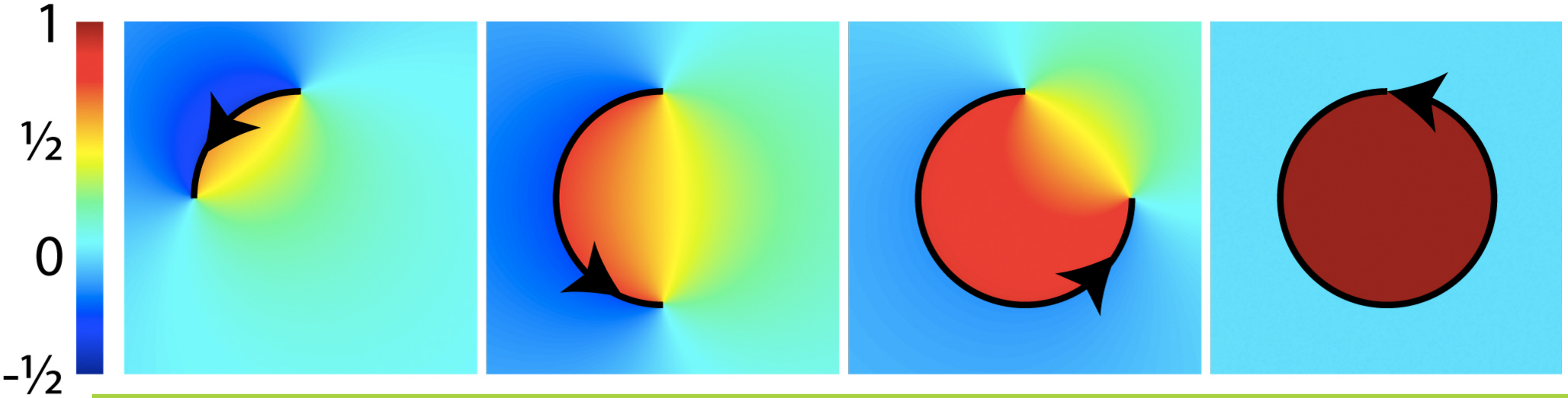
What happens if the shape is open?

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



What happens if the shape is open?

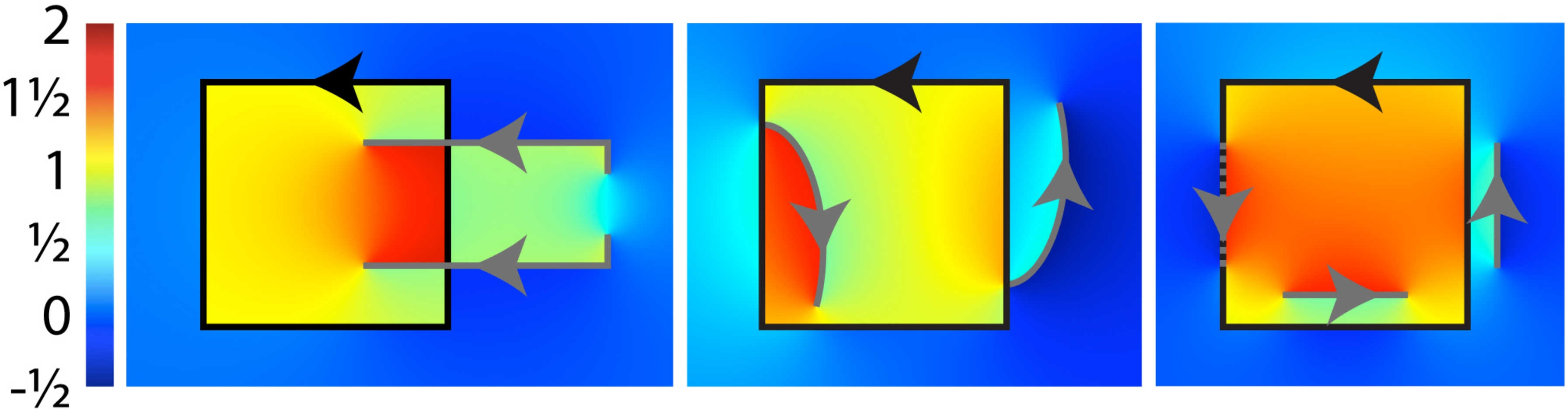
$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$



Gracefully tends toward perfect indicator as shape tends towards watertight

What if shape is self-intersecting? Non-manifold?

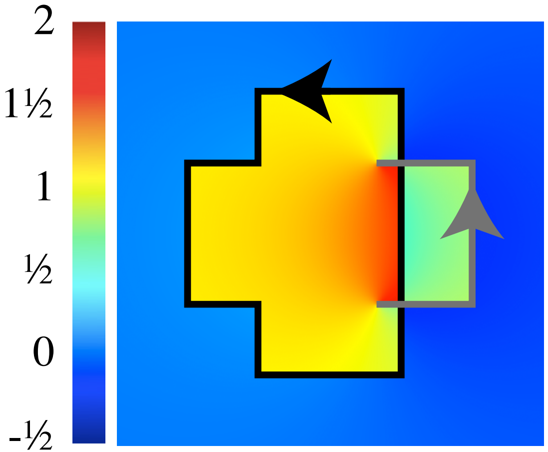
$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$

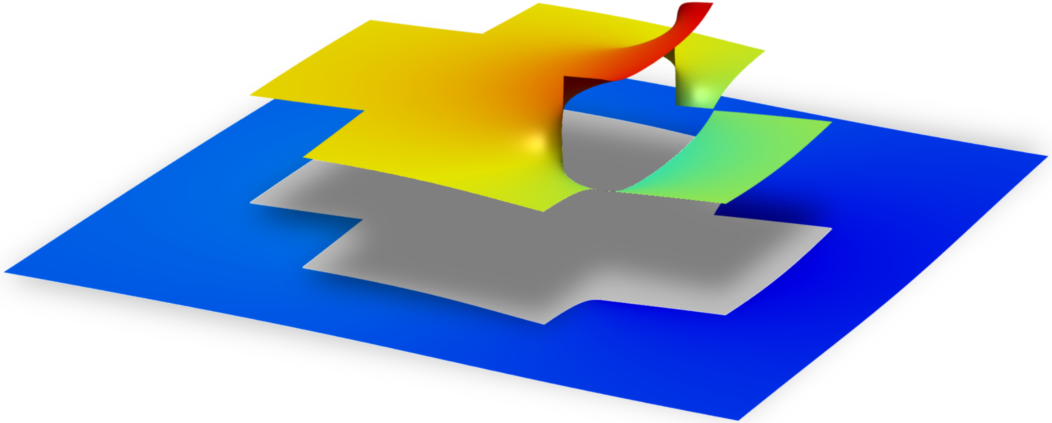


Jumps by ±1 across input facets

Winding number jumps across boundaries, otherwise harmonic!

$$w(\mathbf{p}) = \frac{1}{2\pi} \oint_C d\theta$$

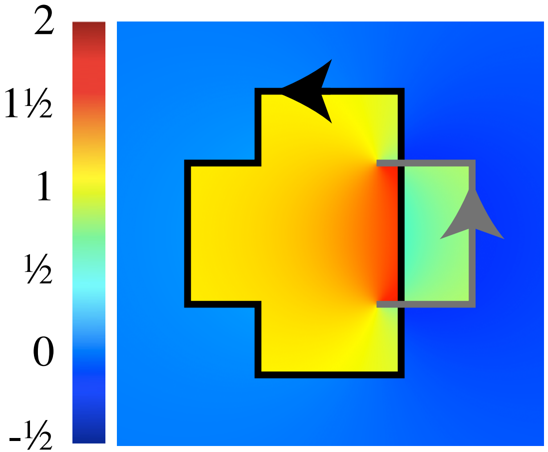


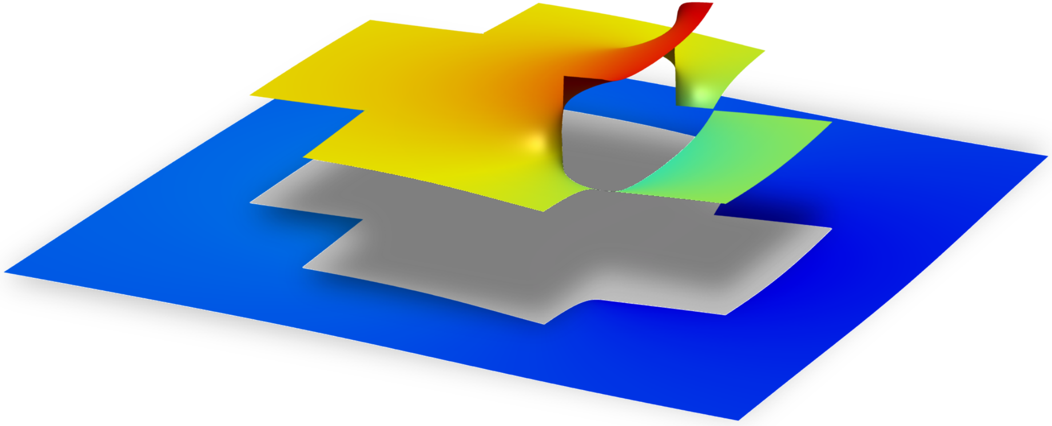


Winding number jumps across boundaries, otherwise harmonic!

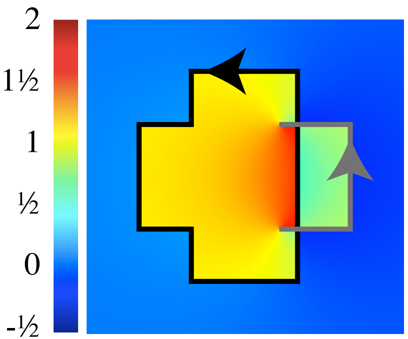
$$w(\mathbf{p}) = \frac{1}{2\pi}\oint_{\mathcal{C}} d\theta$$

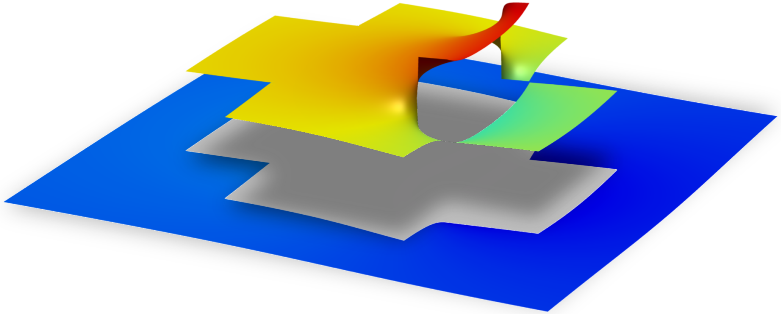
See MAPLE proof in paper or Rahul Narain's recent proof http://goo.gl/5LJWf

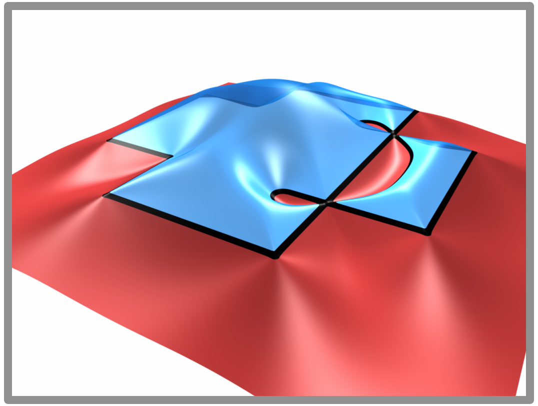




Other interpolating implicit functions are confused by overlap…

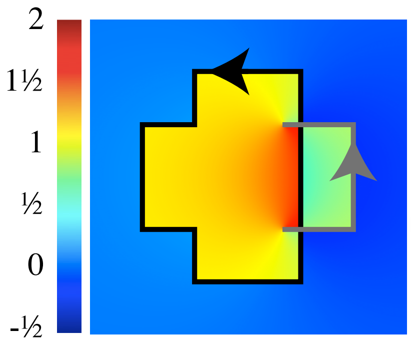


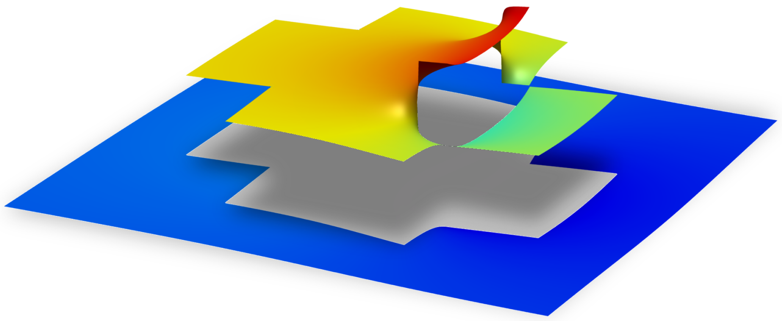


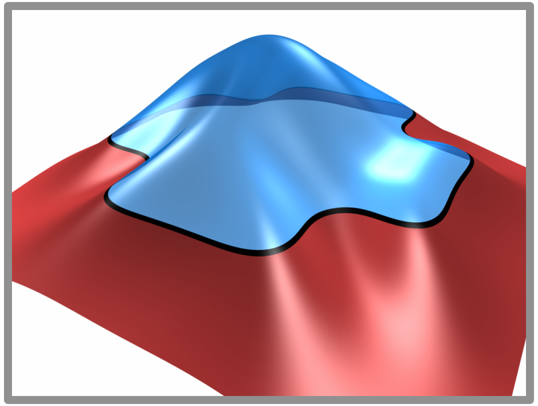


[Shen et al. 2004]

…or resort to approximation

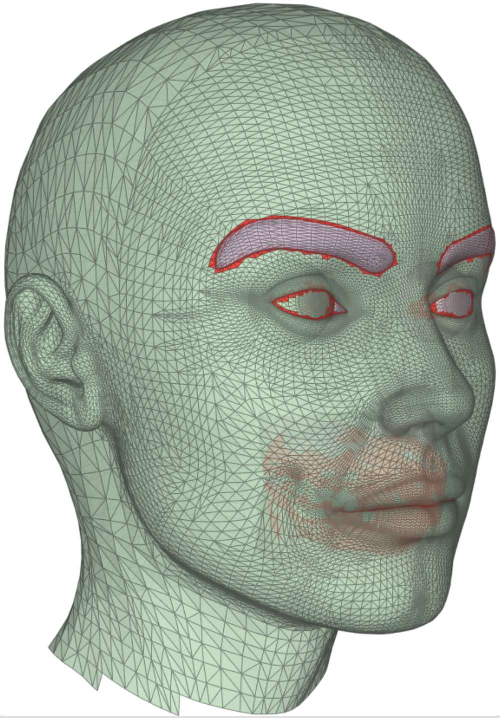


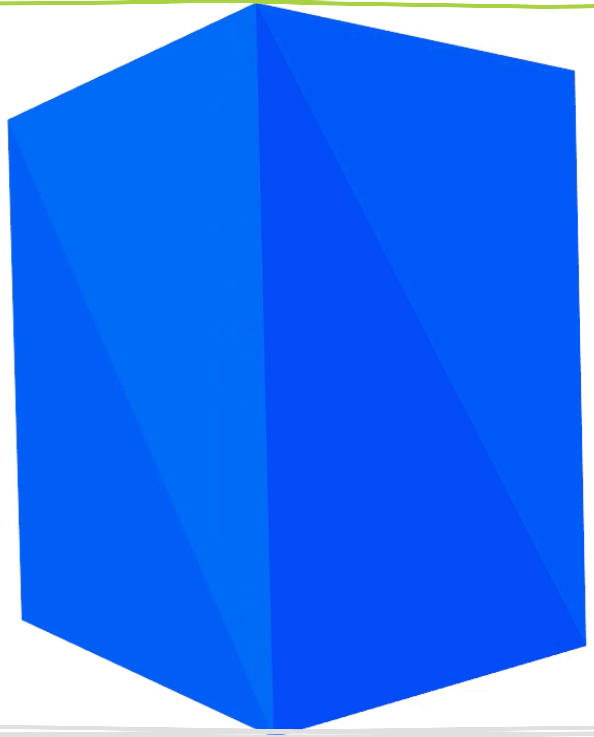




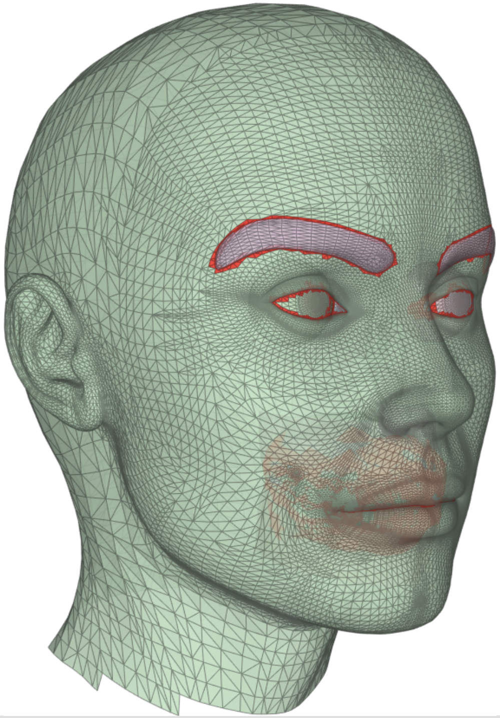
[Shen et al. 2004]

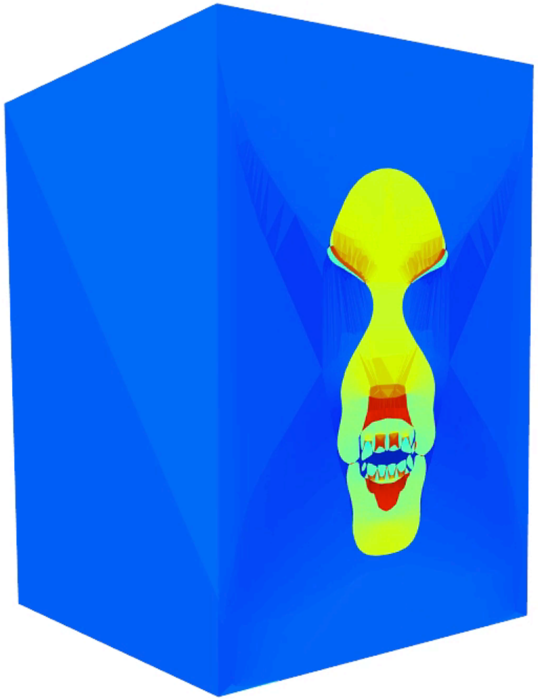
Sharp discontinuity across input eases precise, *conformal* segmentation



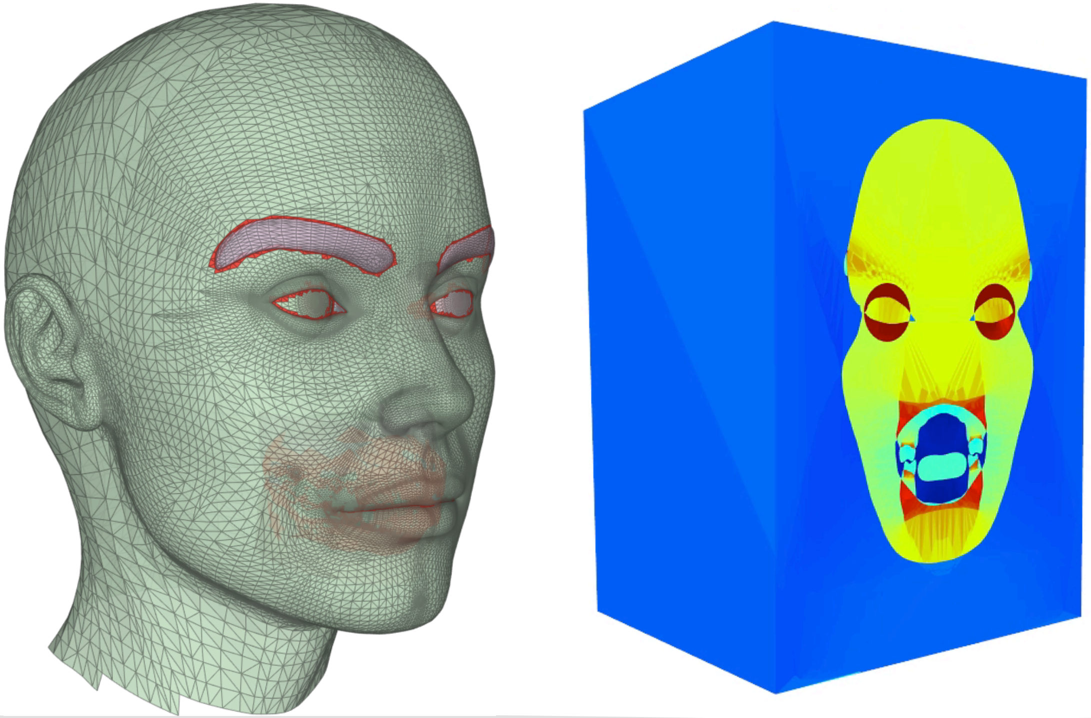


Sharp discontinuity across input eases precise, *conformal* segmentation

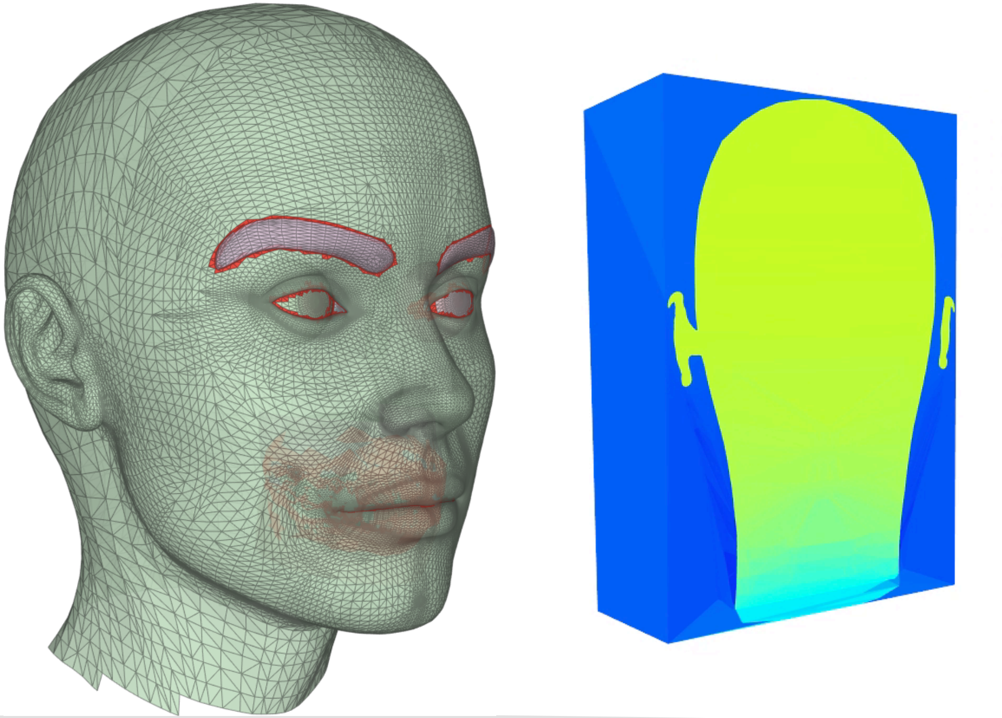




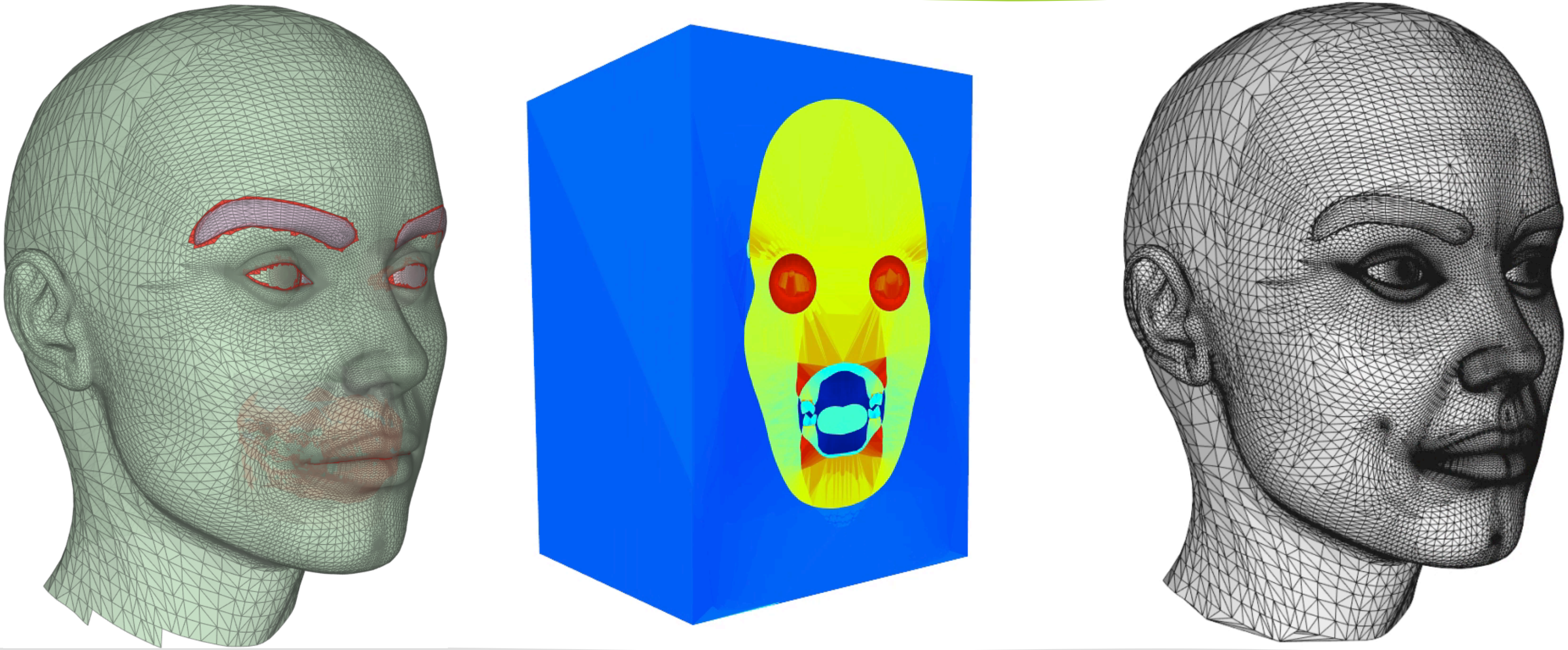
Sharp discontinuity across input eases precise, *conformal* segmentation



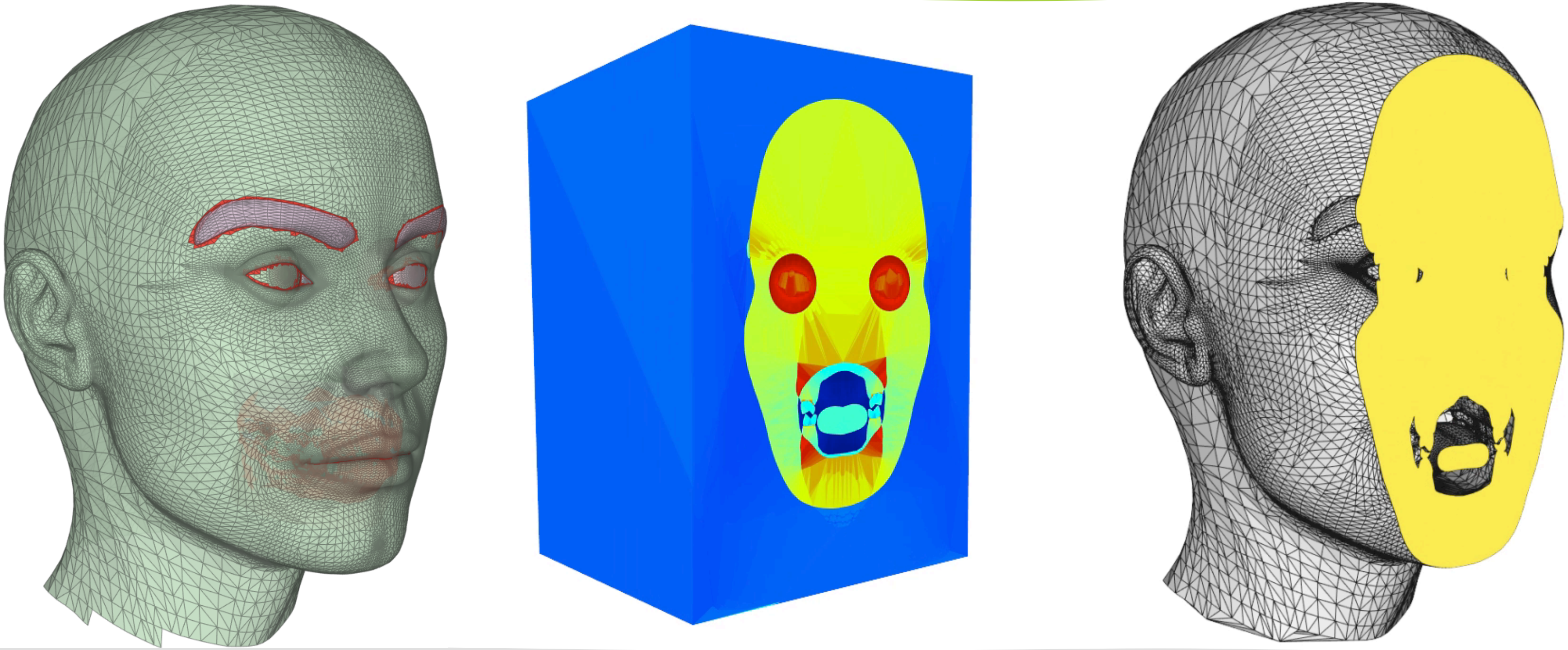
Sharp discontinuity across input eases precise, *conformal* segmentation



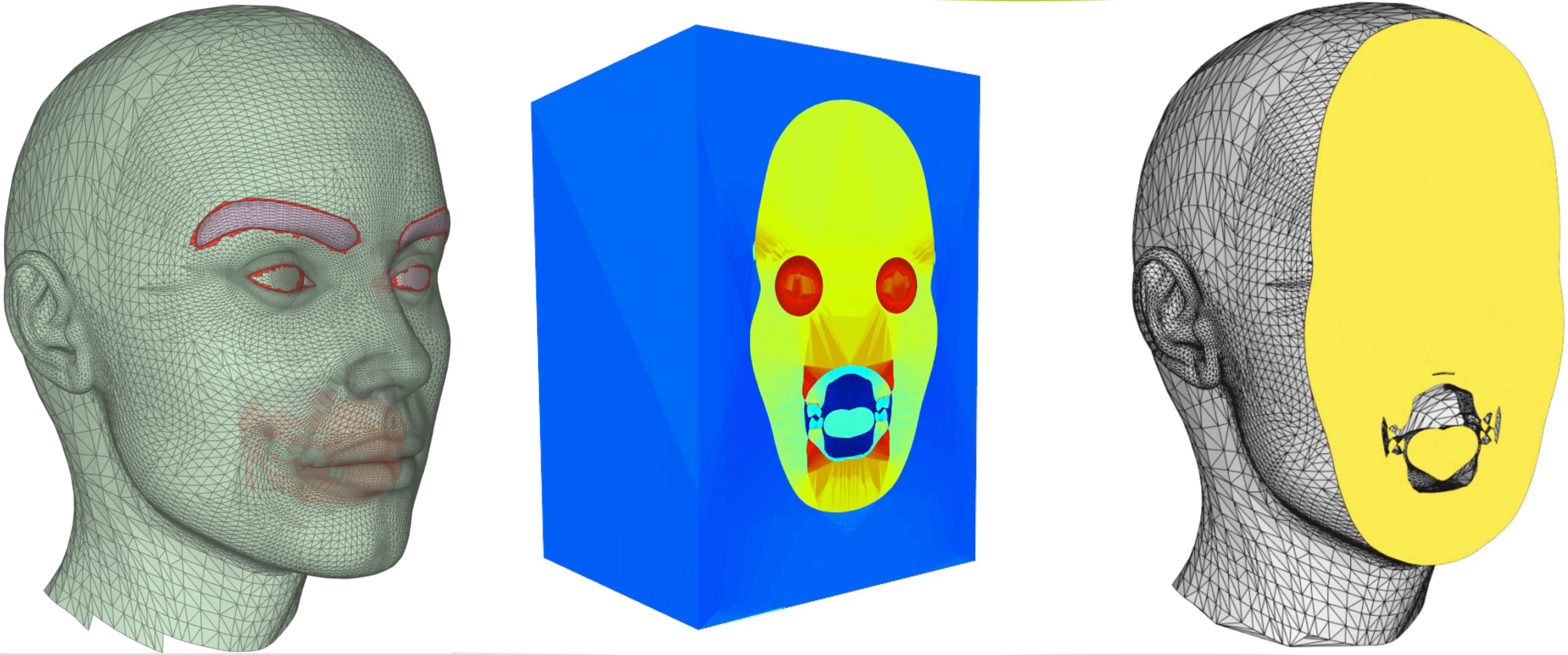
Sharp discontinuity across input eases precise, *conformal* segmentation



Sharp discontinuity across input eases precise, *conformal* segmentation

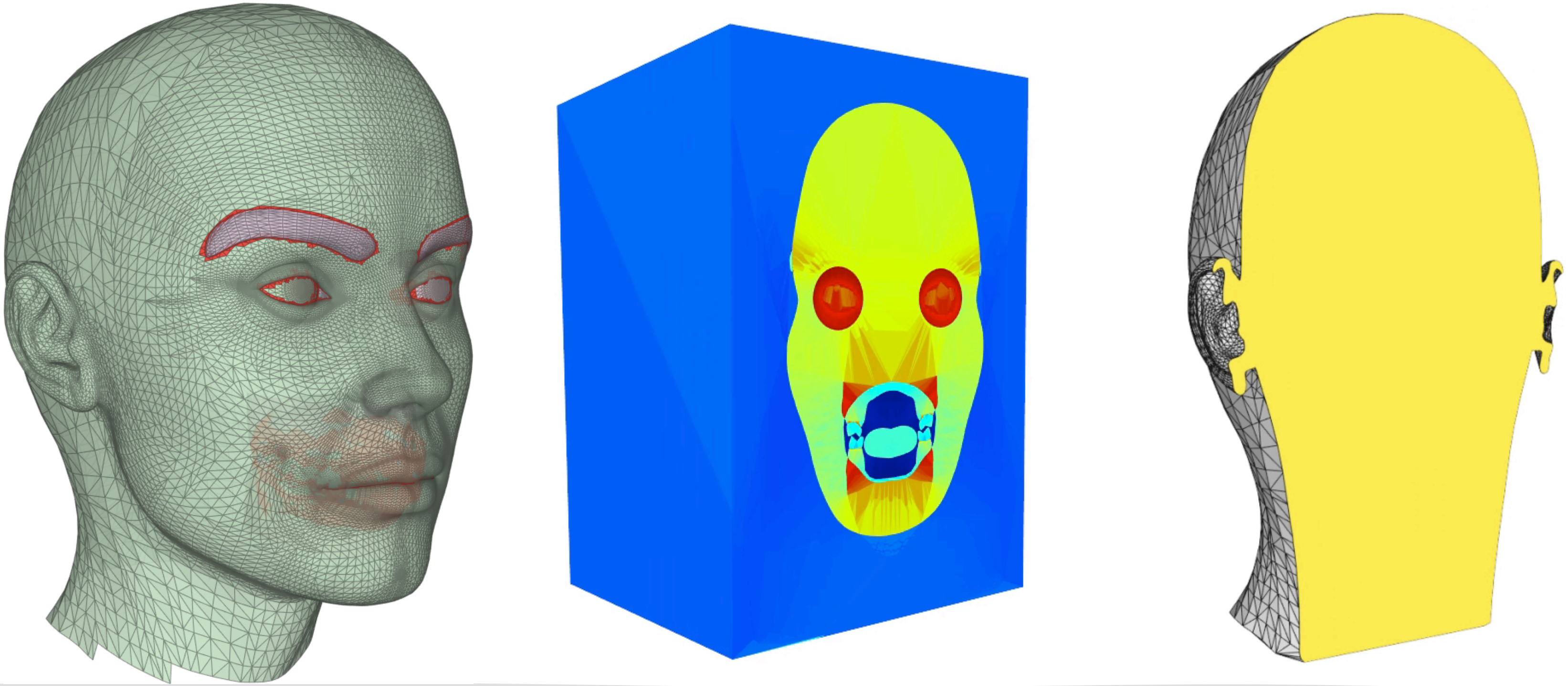


Sharp discontinuity across input eases precise, *conformal* segmentation



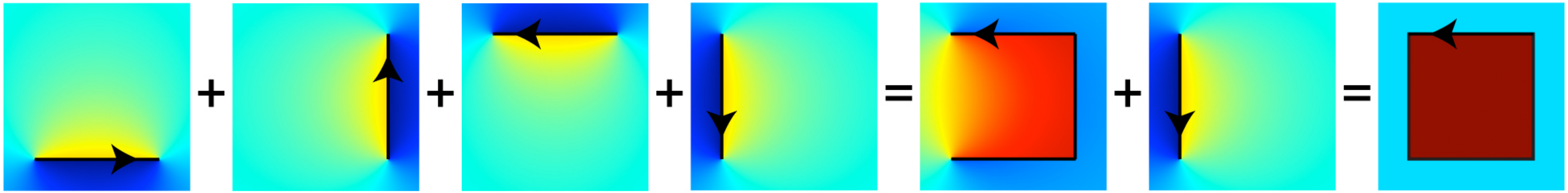
ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

Sharp discontinuity across input eases precise, *conformal* segmentation



Naive implementation is too expensive

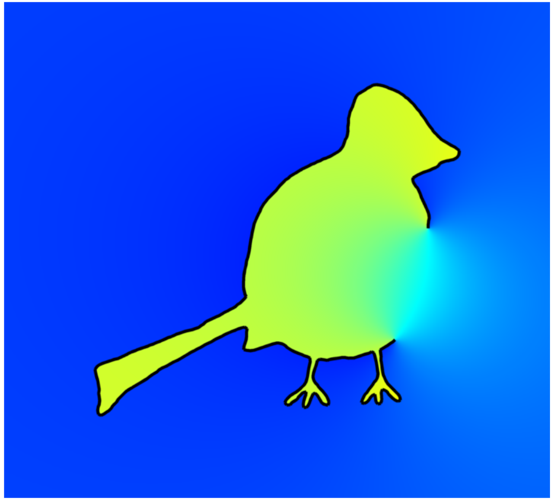
$$w(\mathbf{p}) = \frac{1}{2\pi}\sum_{i=1}^{n}\theta_i$$



Winding number is sum of winding numbers: $O(m)$

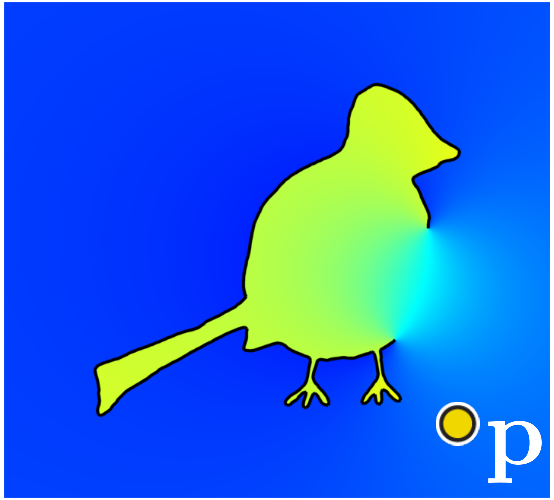
Interesting fact reveals asymptotic speedup

$\mathcal{C}$



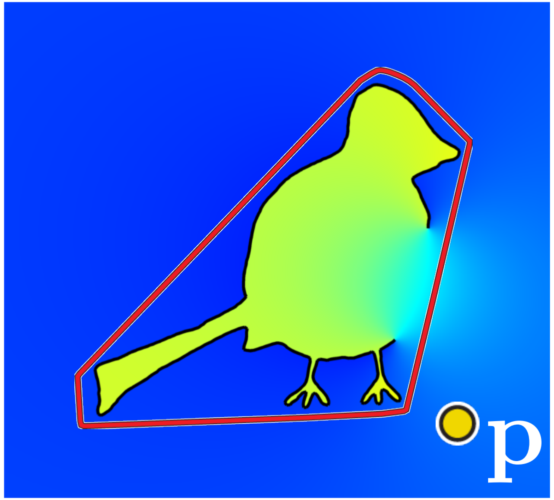
Interesting fact reveals asymptotic speedup

$\mathcal{C}$

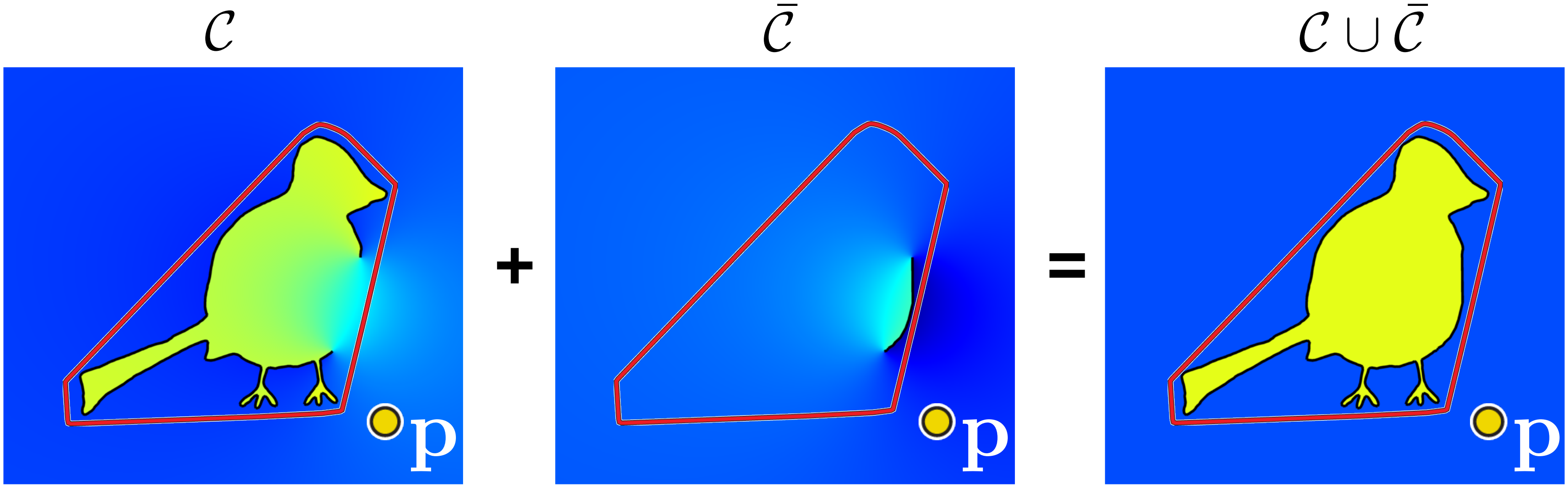


Interesting fact reveals asymptotic speedup

$\mathcal{C}$

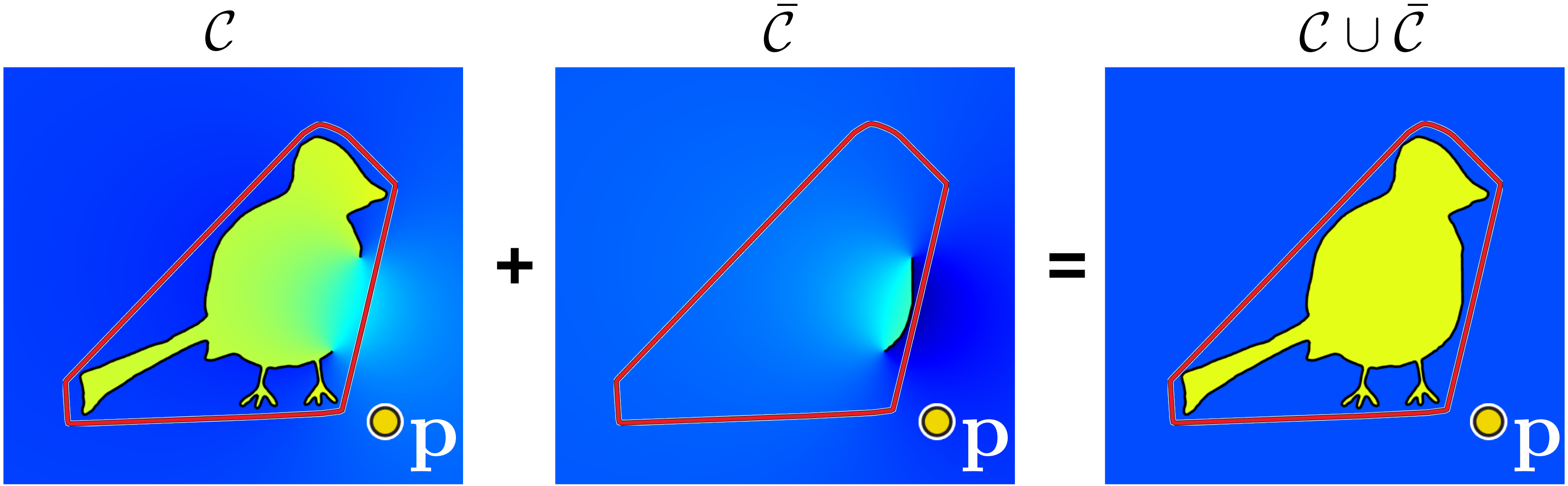


Interesting fact reveals asymptotic speedup



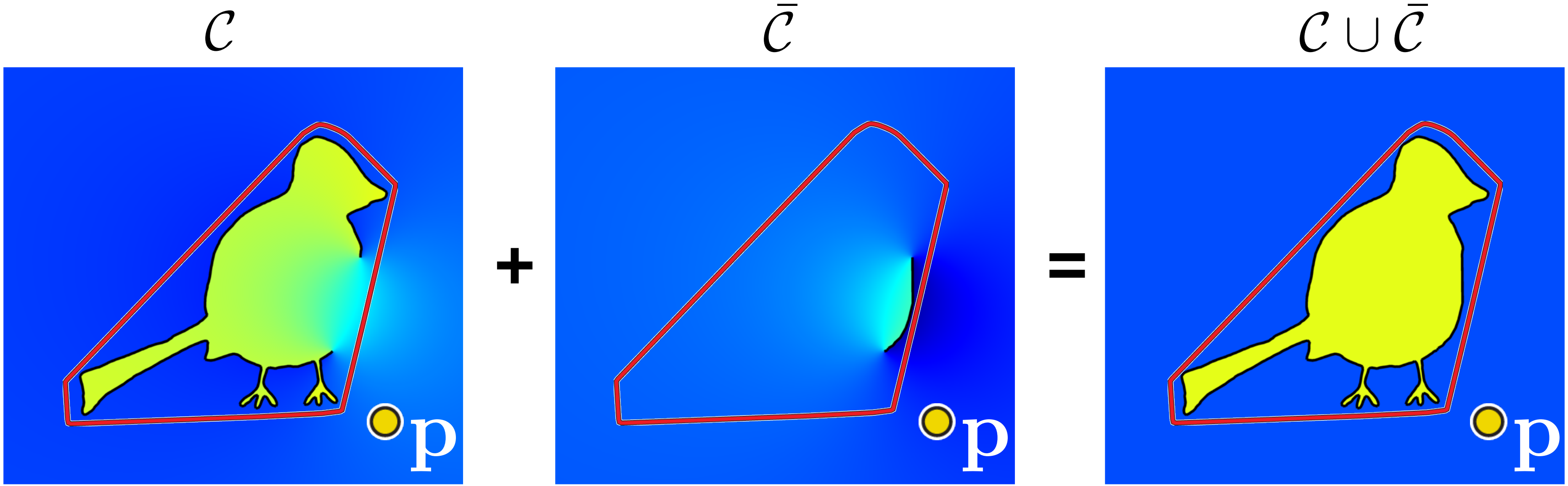
$$w_{\mathcal{C} \cup \bar{\mathcal{C}}}(\mathbf{p}) = 0$$

Interesting fact reveals asymptotic speedup



$$w_{\mathcal{C}}(\mathbf{p}) + w_{\bar{\mathcal{C}}}(\mathbf{p}) = w_{\mathcal{C}\cup\bar{\mathcal{C}}}(\mathbf{p}) = 0$$

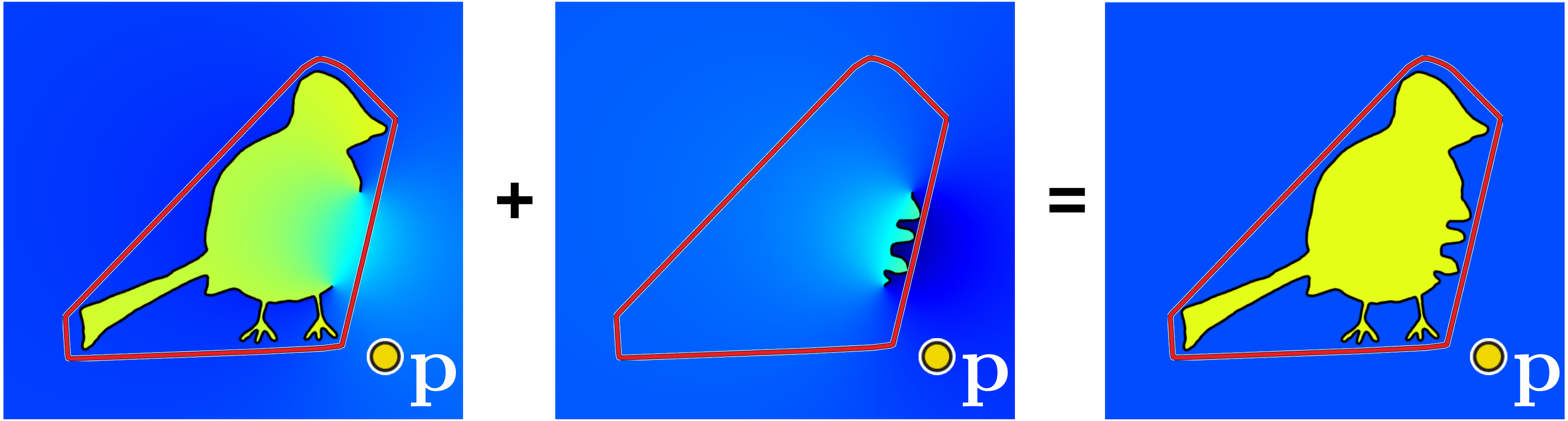
Interesting fact reveals asymptotic speedup



$$w_{\mathcal{C}}(\mathbf{p}) = -w_{\bar{\mathcal{C}}}(\mathbf{p})$$

Interesting fact reveals asymptotic speedup

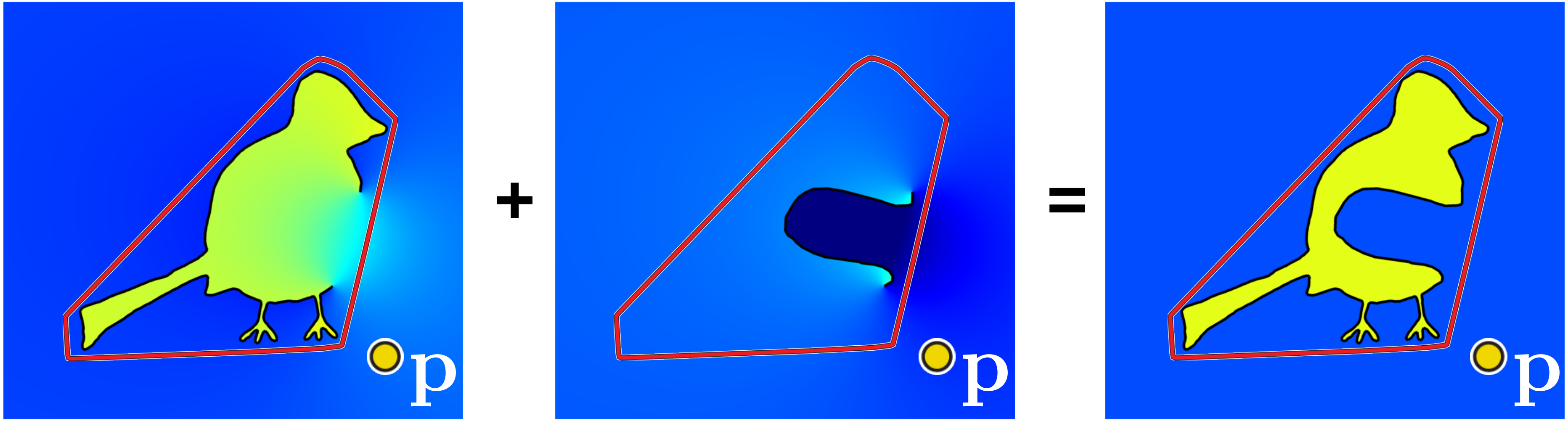
$\mathcal{C}$ + $\bar{\mathcal{C}}$ = $\mathcal{C} \cup \bar{\mathcal{C}}$



$$w_{\mathcal{C}}(\mathbf{p}) = -w_{\bar{\mathcal{C}}}(\mathbf{p})$$

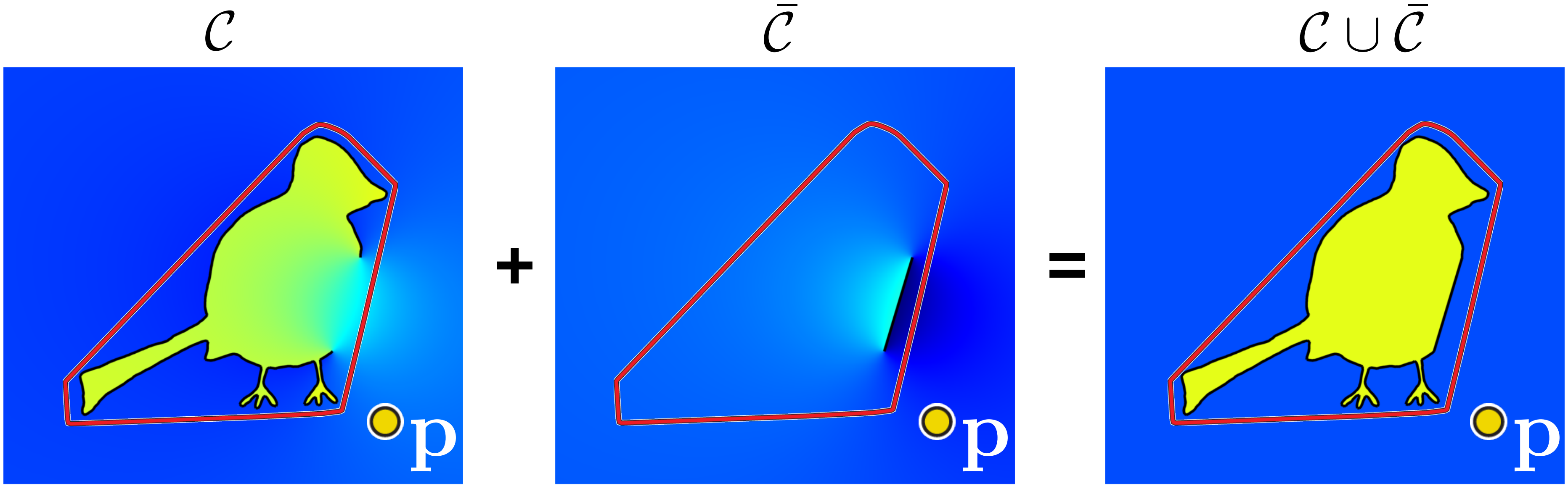
Interesting fact reveals asymptotic speedup

$\mathcal{C}$ $\quad\quad$ $\bar{\mathcal{C}}$ $\quad\quad$ $\mathcal{C} \cup \bar{\mathcal{C}}$



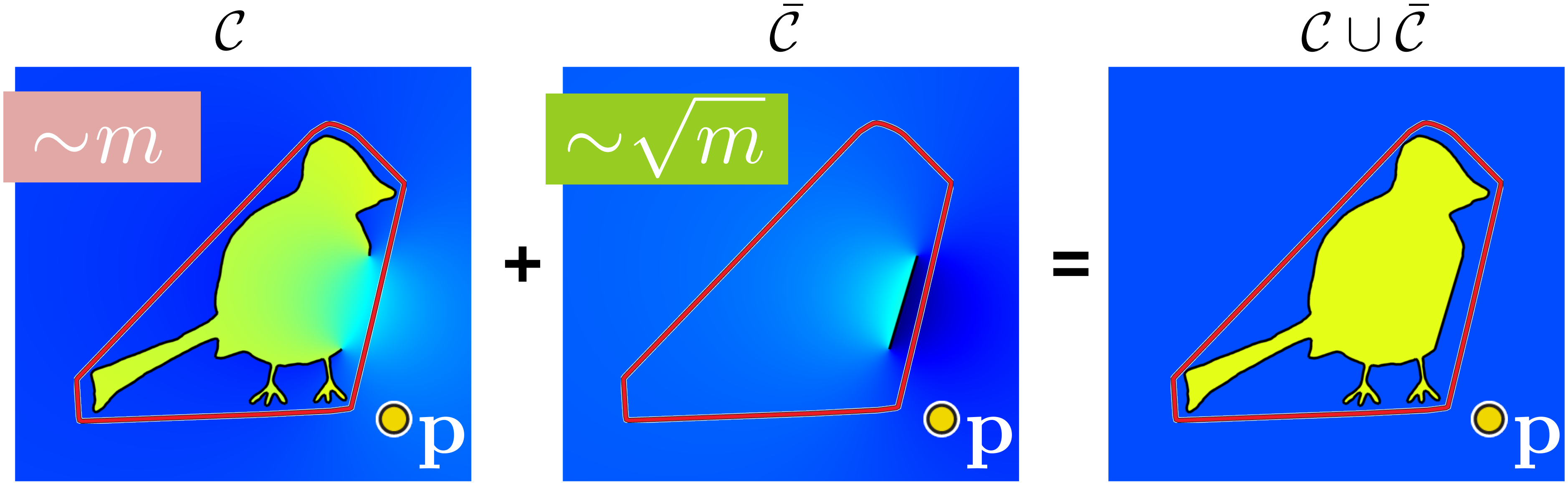
$$w_{\mathcal{C}}(\mathbf{p}) = -w_{\bar{\mathcal{C}}}(\mathbf{p})$$

Interesting fact reveals asymptotic speedup



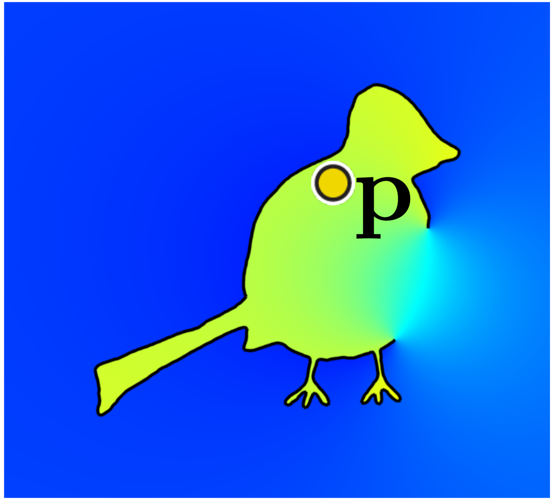
$$w_{\mathcal{C}}(\mathbf{p}) = -w_{\bar{\mathcal{C}}}(\mathbf{p})$$

Interesting fact reveals asymptotic speedup

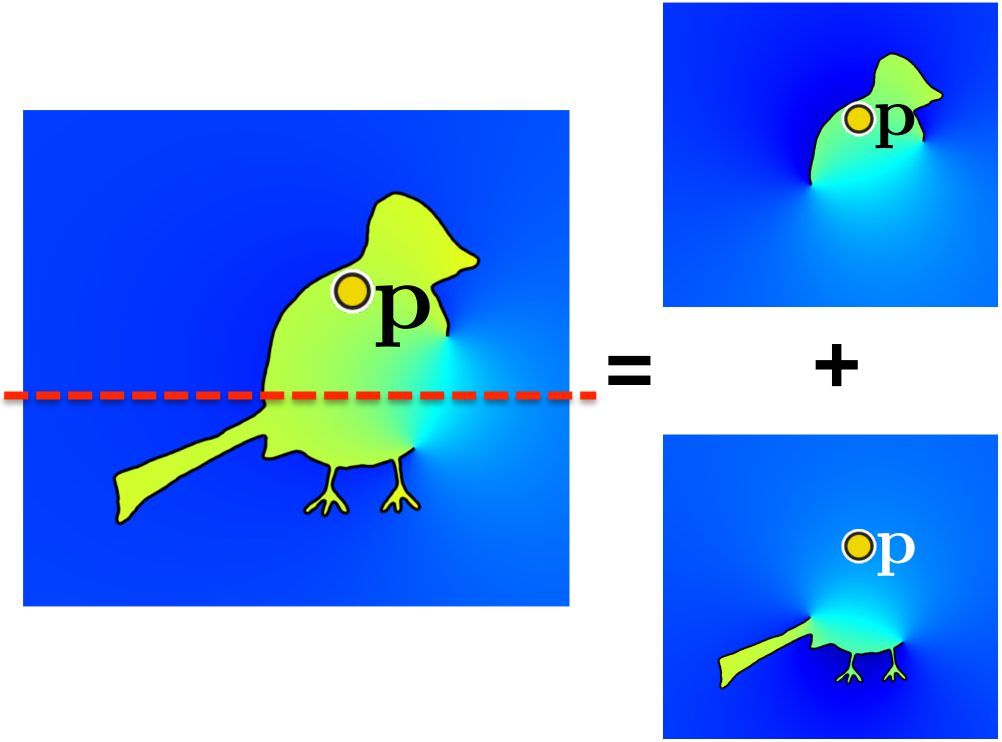


$$w_{\mathcal{C}}(\mathbf{p}) = -w_{\bar{\mathcal{C}}}(\mathbf{p})$$

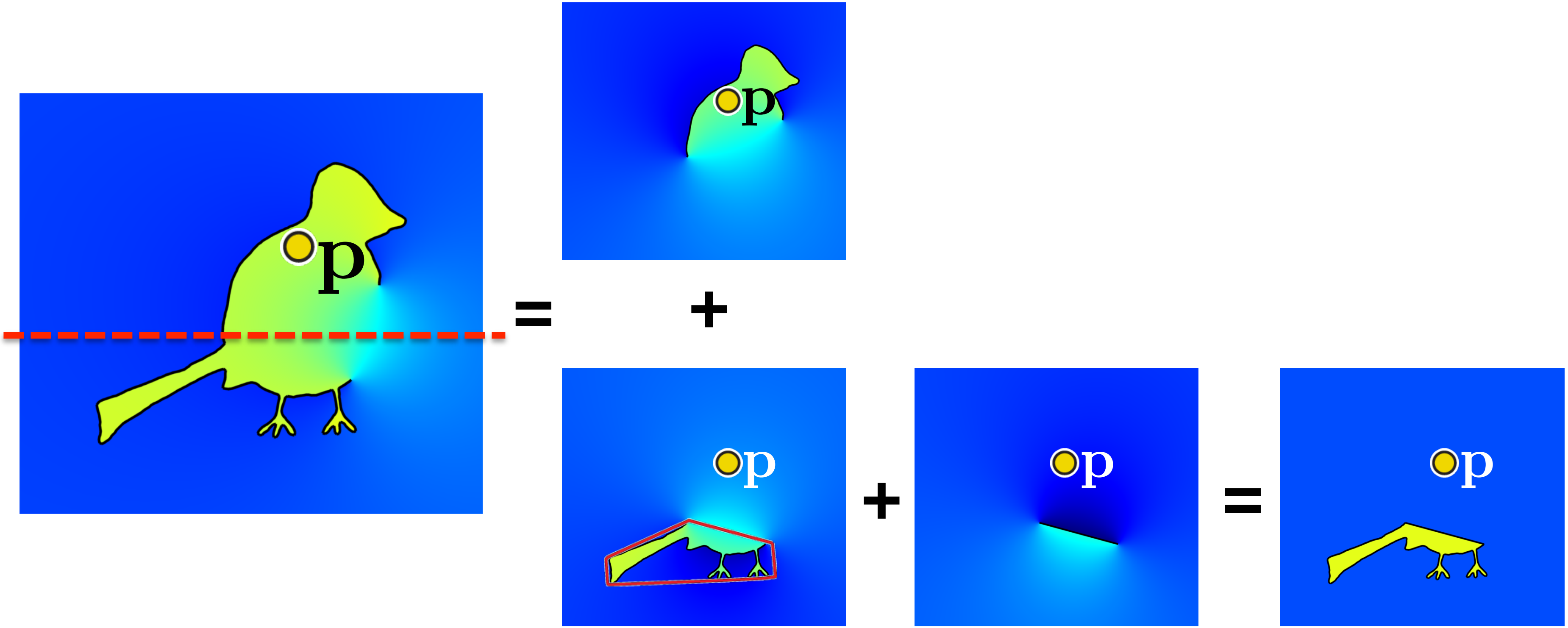
Divide and conquer!



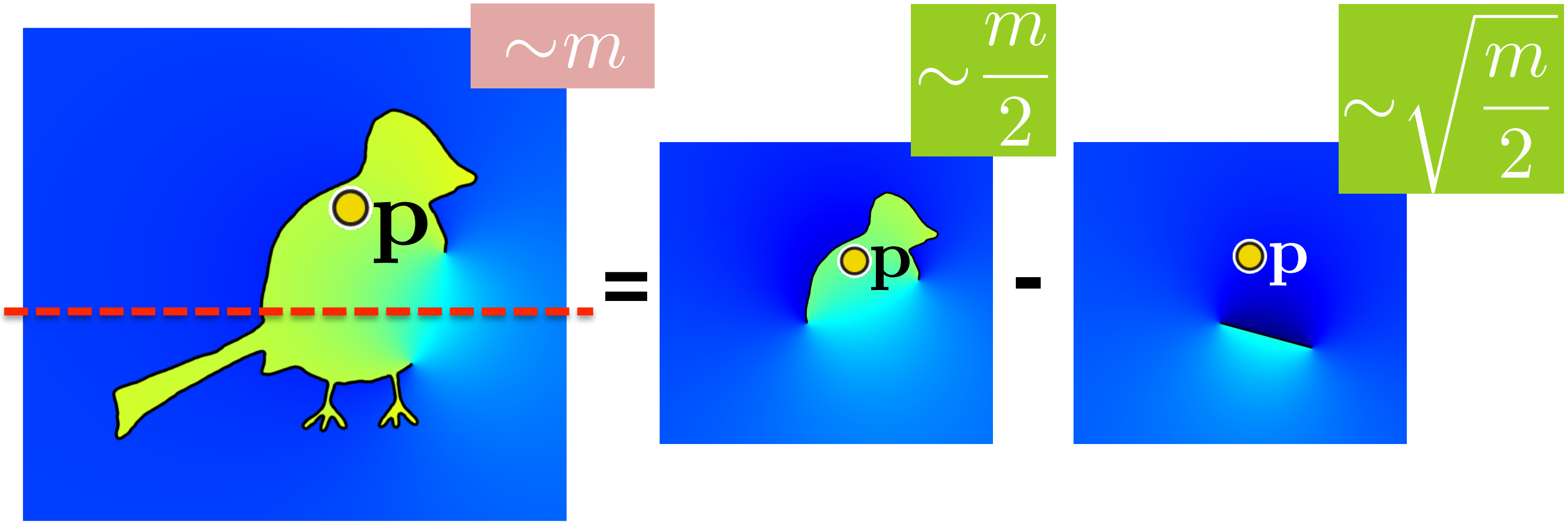
Divide and conquer!



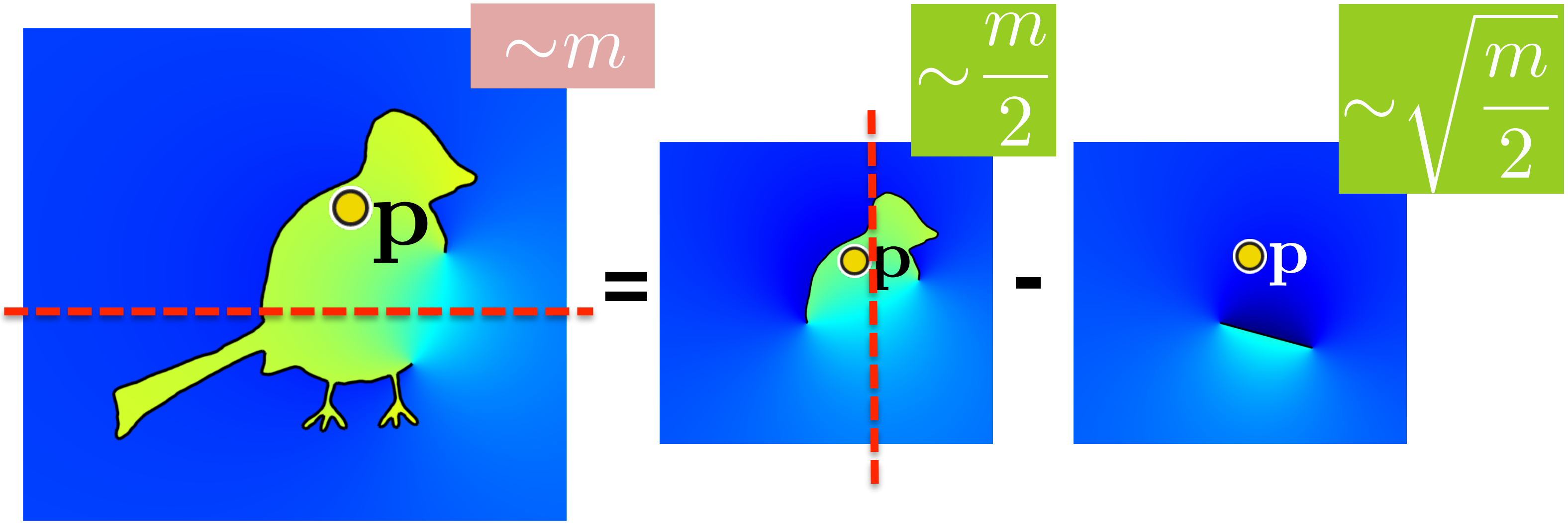
Divide and conquer!



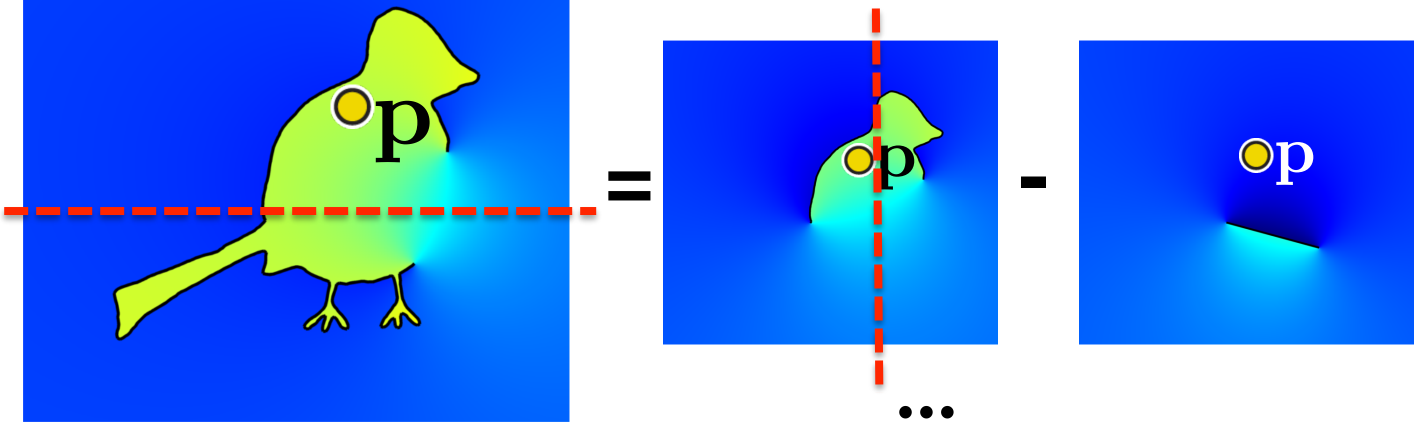
Divide and conquer!



Divide and conquer!



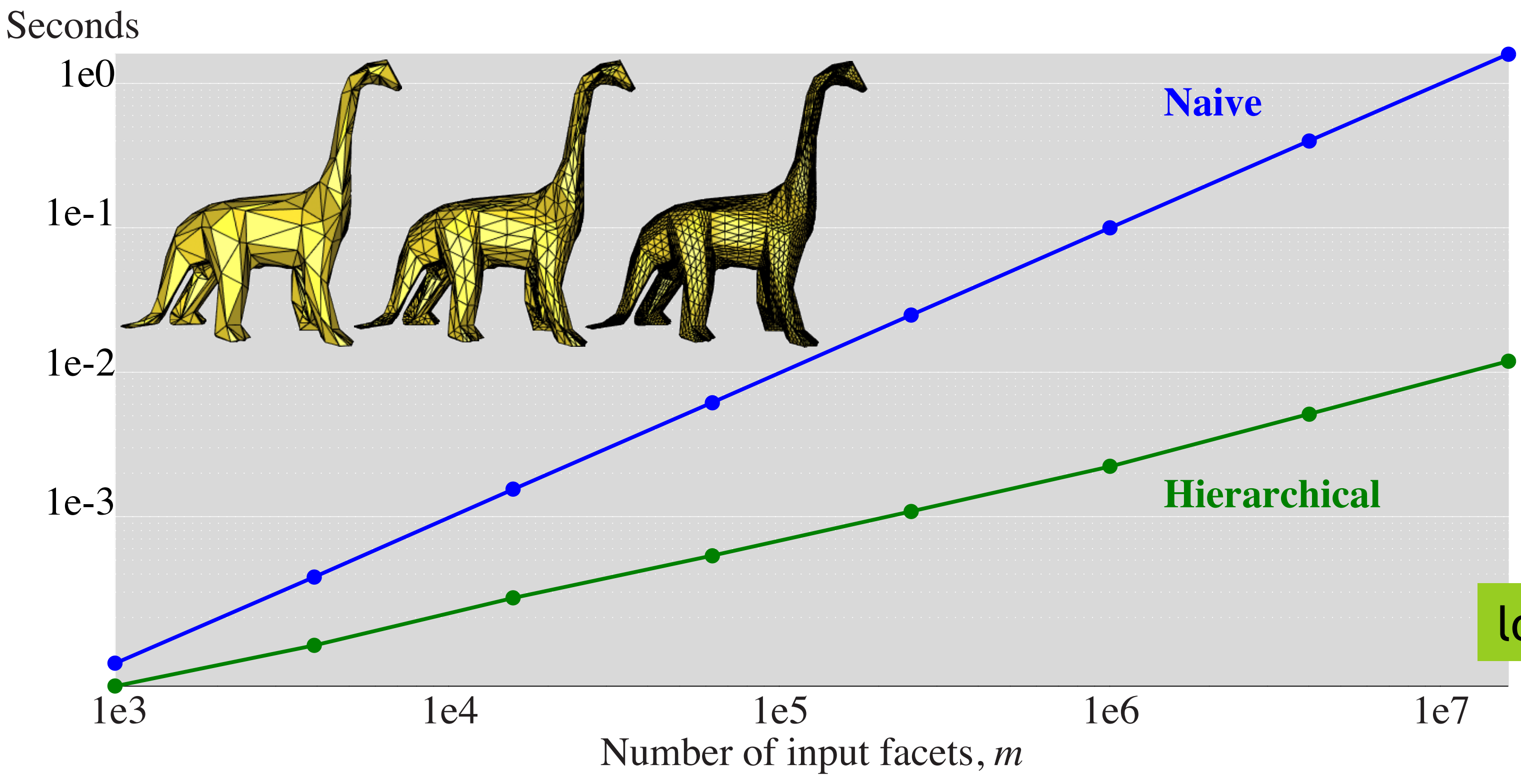
Divide and conquer!



$\sim m$

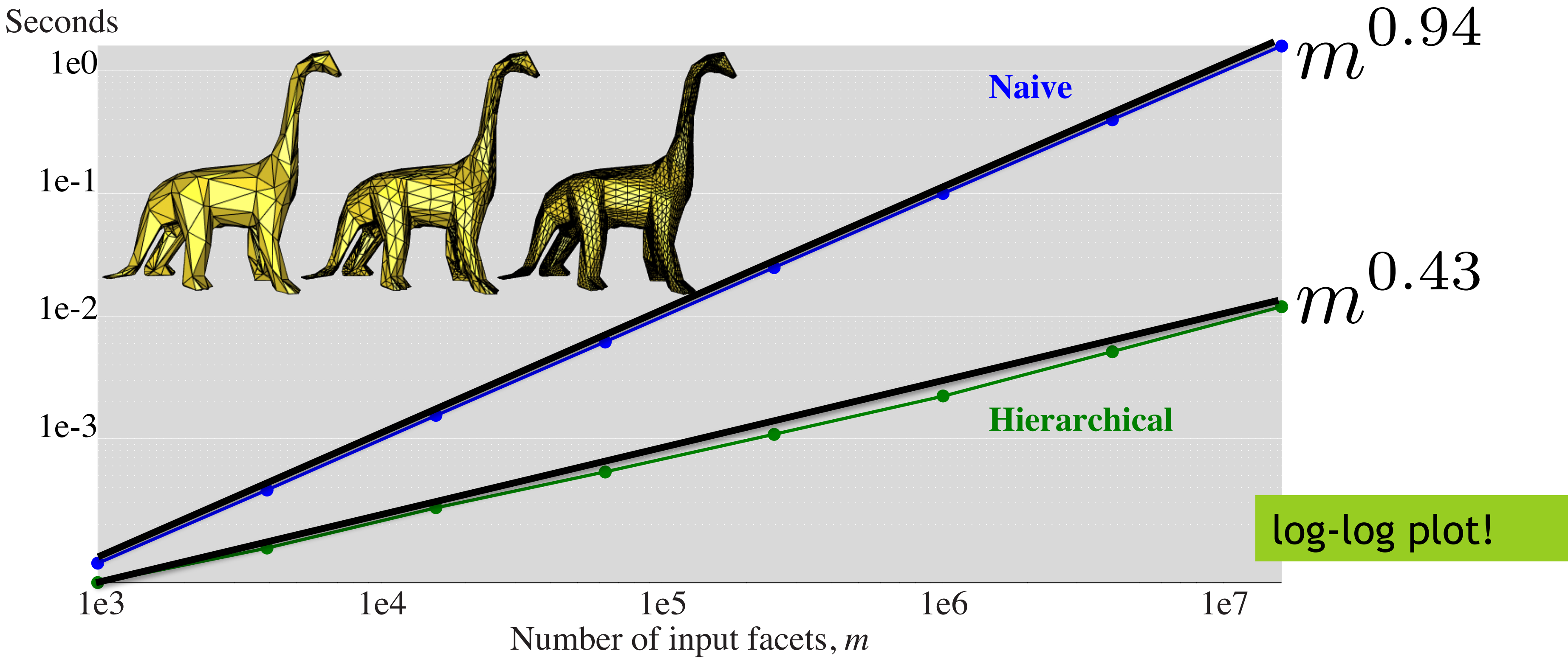
$\sim\sqrt{m}$

Divide-and-conquer evaluation performs asymptotically better

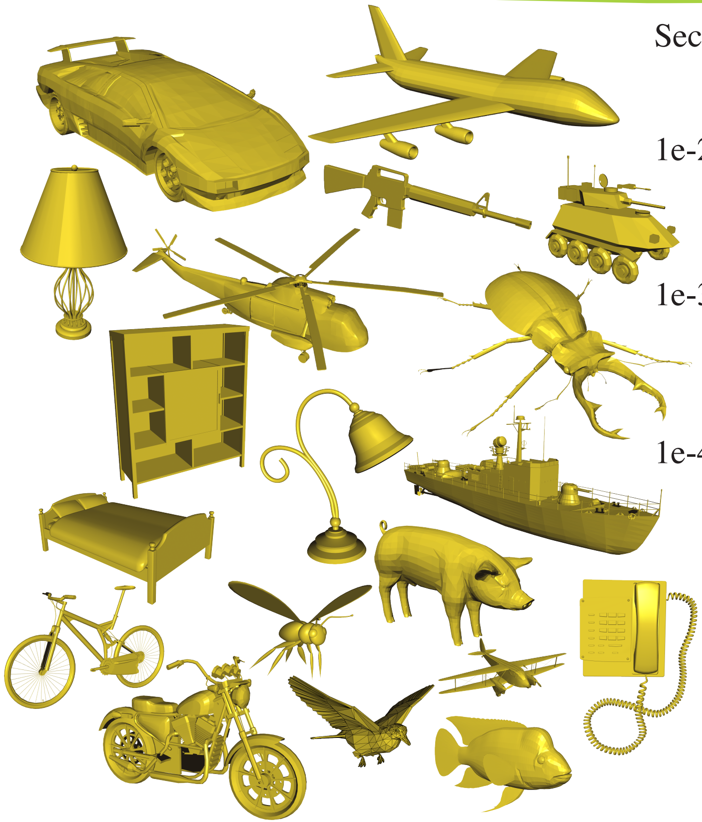


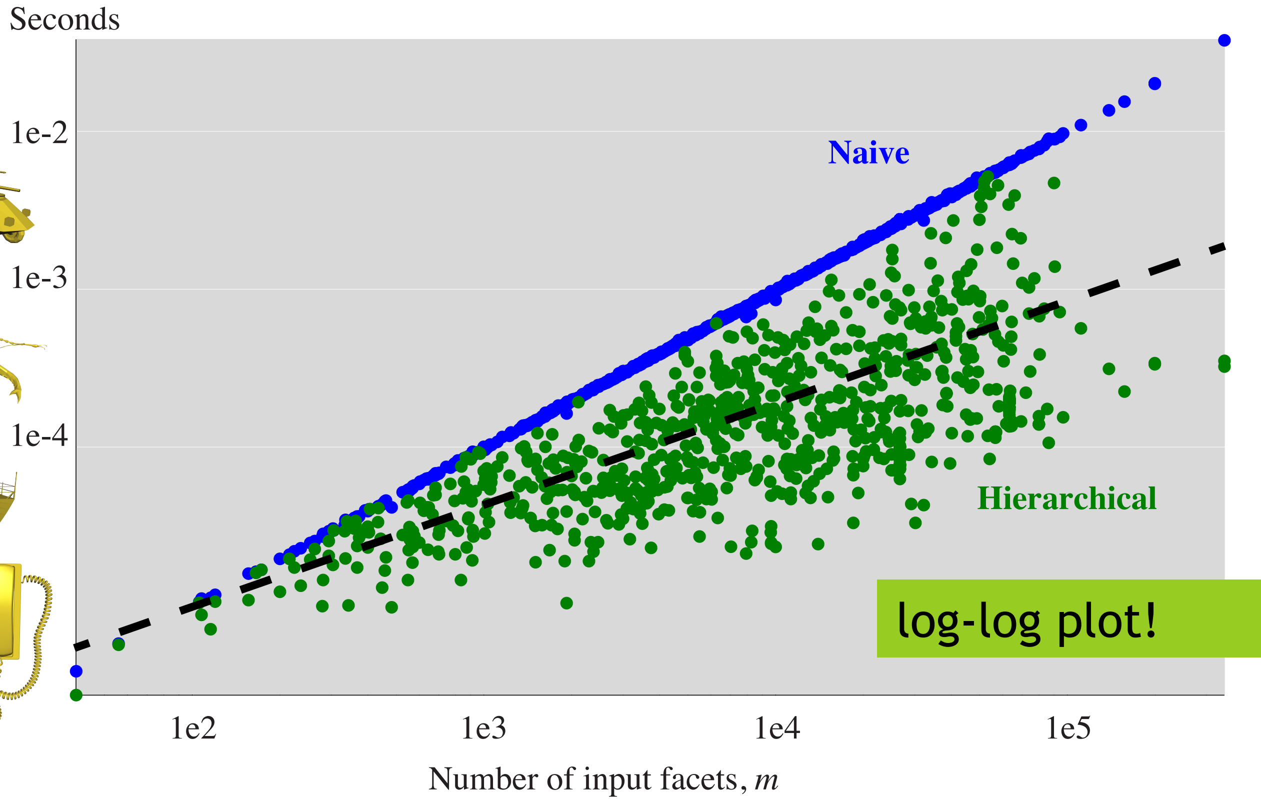
log-log plot!

Divide-and-conquer evaluation performs asymptotically better



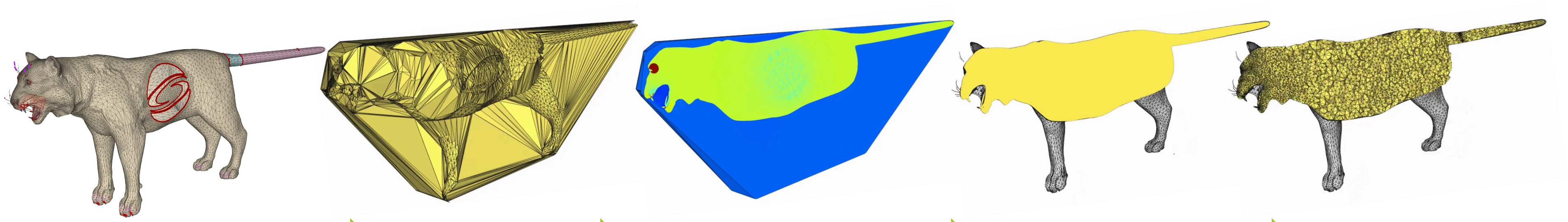
Divide-and-conquer evaluation performs asymptotically better





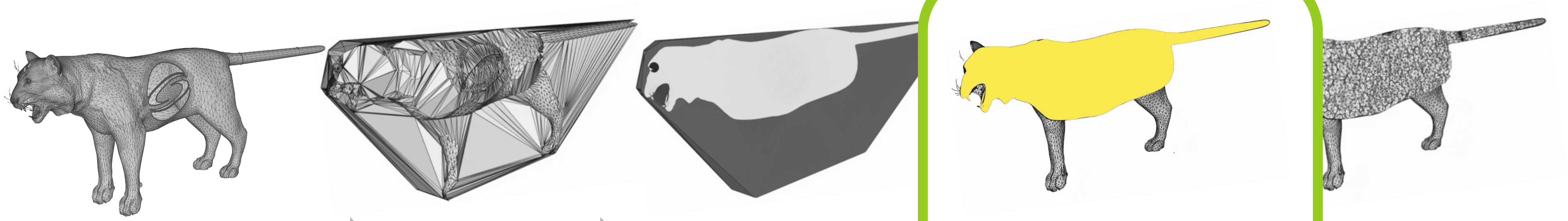
log-log plot!

Idea: mesh entire convex hull, segment inside tets from outside ones



input triangles → CDT → measure of insideness → segment CDT → refine, etc.

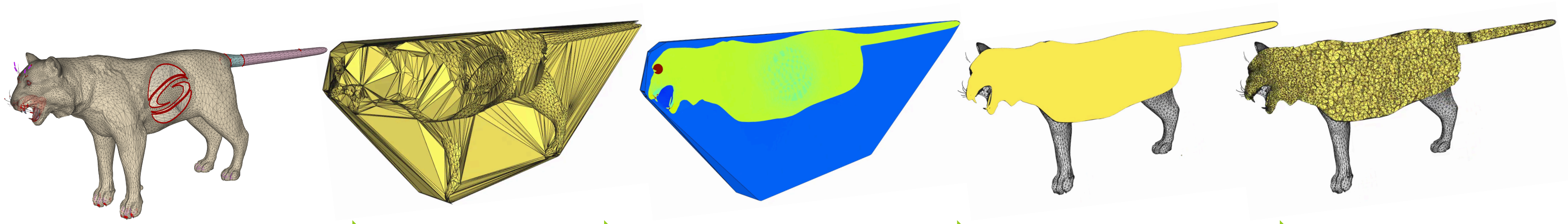
Segmentation is a labeling problem, labels should agree with w.n.



input triangles → CDT → measure of inside/outside → **segment CDT** → efine, etc.

graphcut energy optimization with nonlinear coherency term
+ optional facet or surface-manifoldness constraints

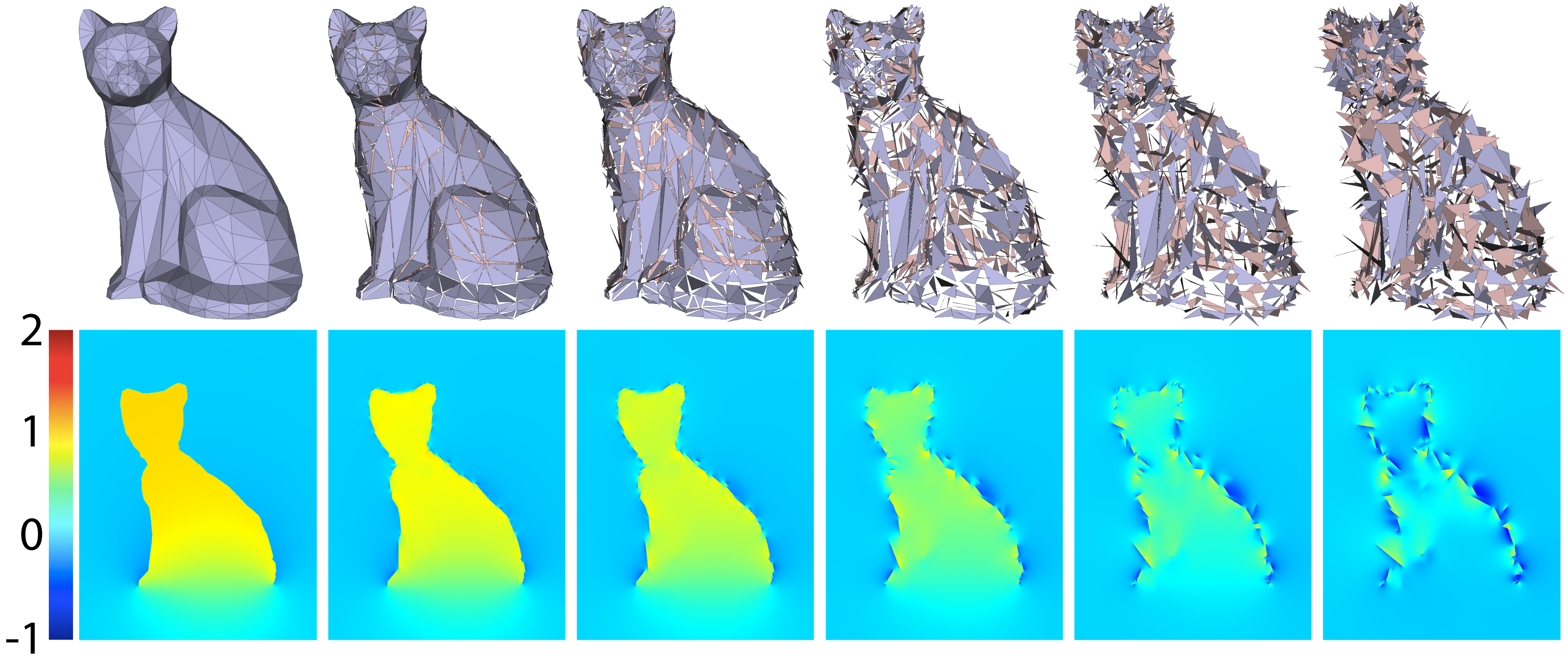
Preprocessing and meshing convex hull dominates runtime



input triangles → CDT → measure of insideness → segment CDT → refine, etc.

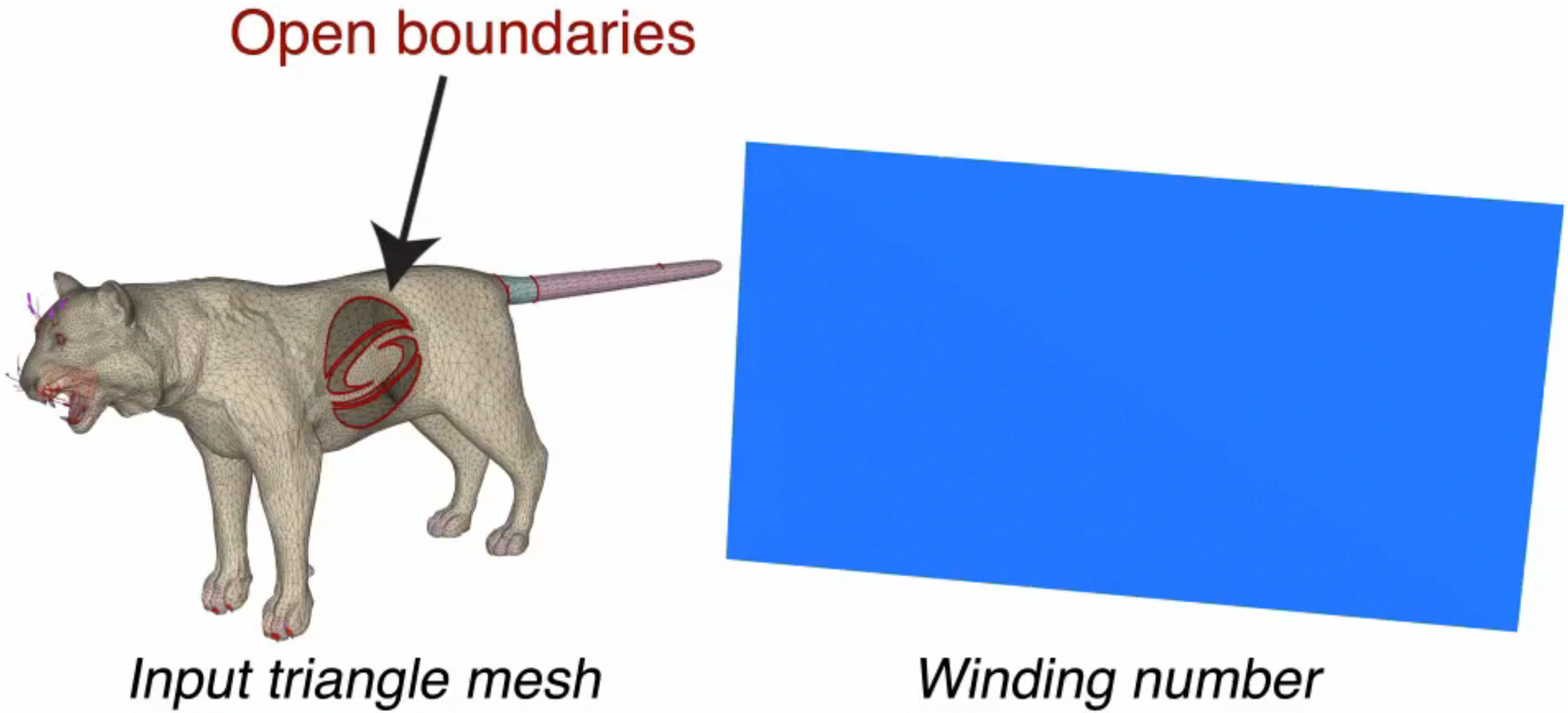
60K facet mesh	~22 secs	~10 secs	~0.1 secs

Winding number degrades gracefully

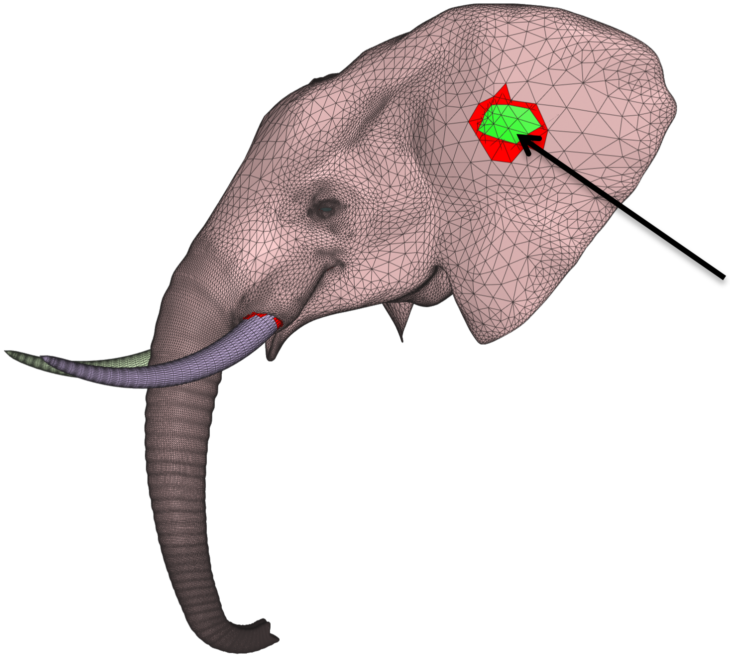


ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich

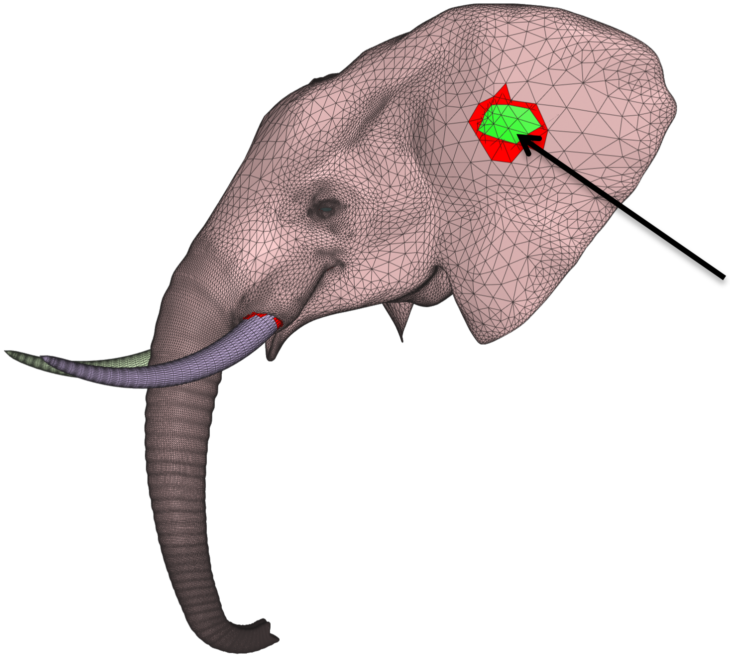
CDT maintains small features



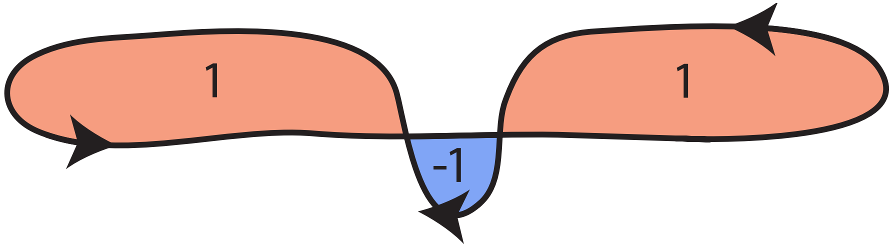
We rely heavily on orientation



We rely heavily on orientation

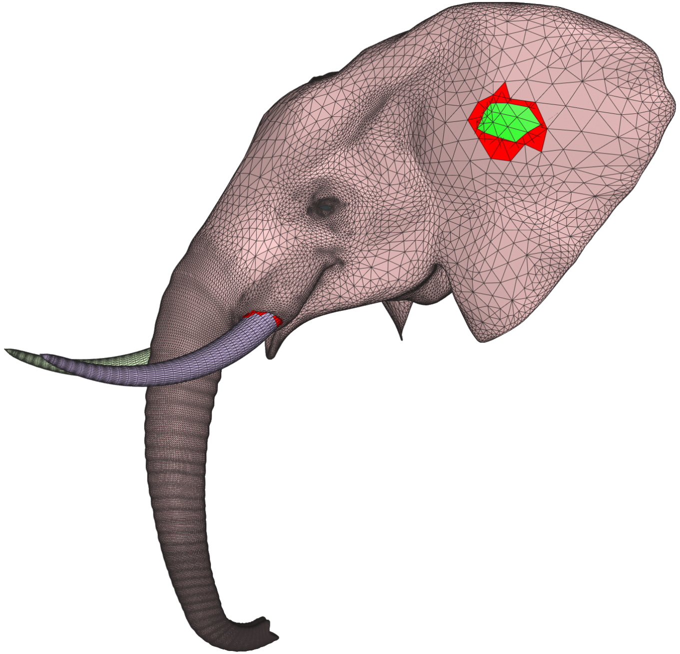


input mesh

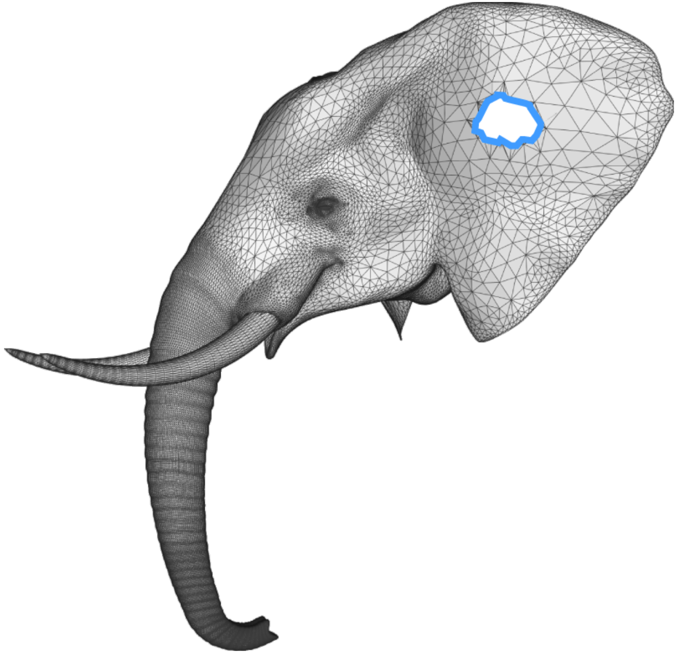


backside of ear penetrates front
(inside-out region)

We rely heavily on orientation



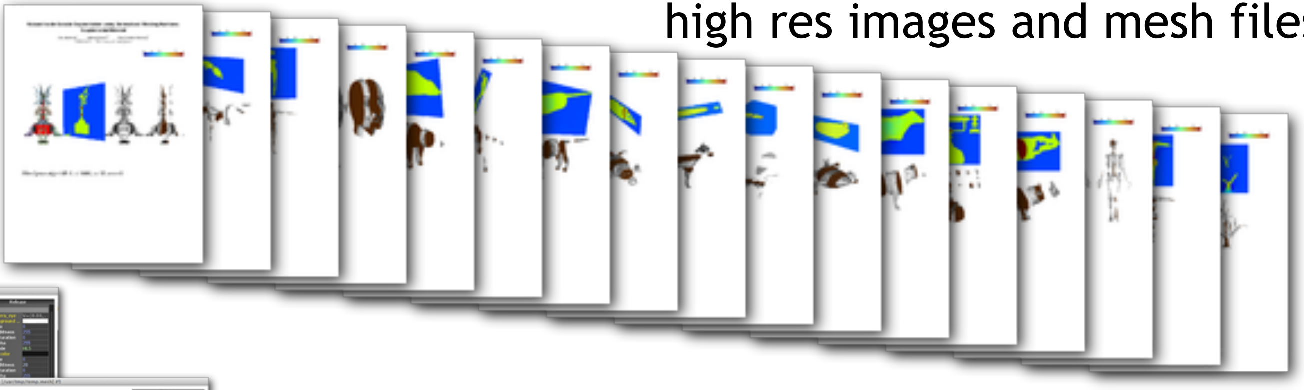
input mesh



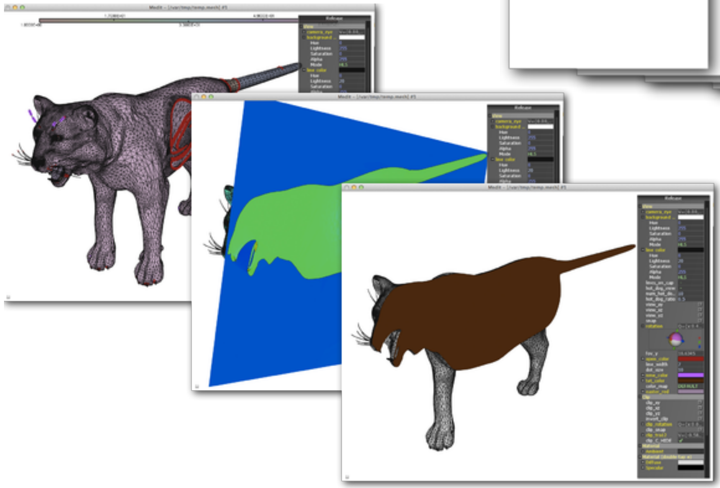
our output

Brings a new level of robustness to volume meshing for a variety of shapes

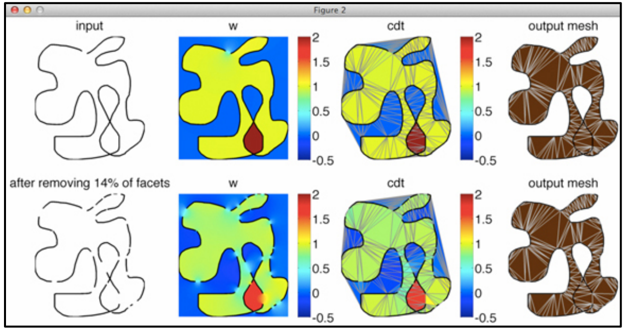
high res images and mesh files



3D demo



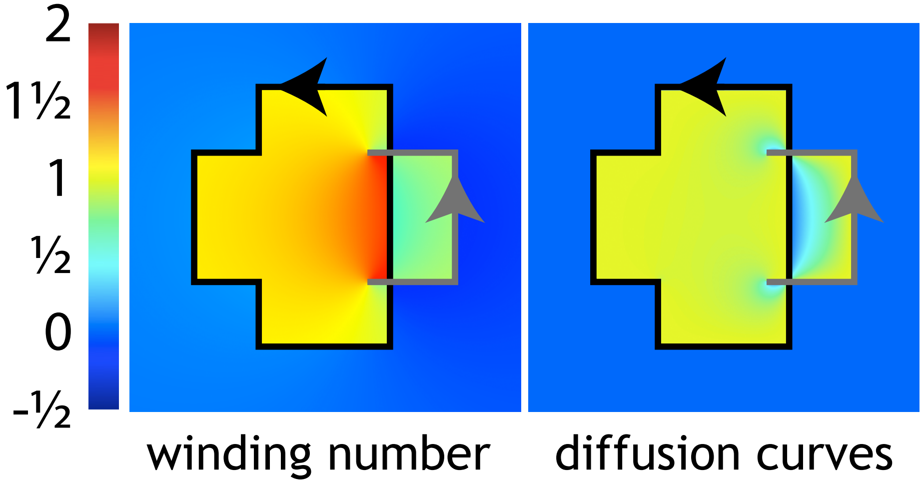
2D demo



http://goo.gl/m0oL9

Future work

- Even faster approximation
- Relationship to:
  diffusion curves,
  Mean Value Coordinates,
  etc.



Acknowledgements

Pierre Alliez, Ilya Baran, Leo Guibas, Fabian Hahn, James O'Brien, Daniele Panozzo, Leonardo Koller Sacht, Alexander Sorkine-Hornung, Josef Pelikan, Kenshi Takayama, Kaan Yücer

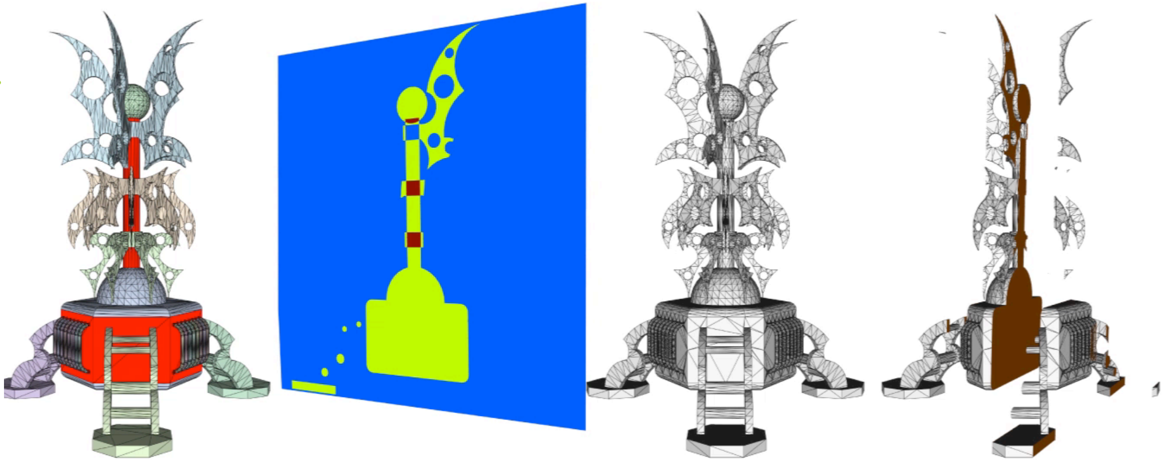
Marco Attene for MESHFIX

Hang Si for TETGEN

This work was supported in part by the ERC grant iModel (StG-2012-306877), by an SNF award 200021 137879 and the Intel Doctoral Fellowship.

Robust Inside-Outside Segmentation using Generalized Winding Numbers

http://igl.ethz.ch/projects/winding-number/
(paper, code, video)



Alec Jacobson
jacobson@inf.ethz.ch
Ladislav Kavan
Olga Sorkine-Hornung

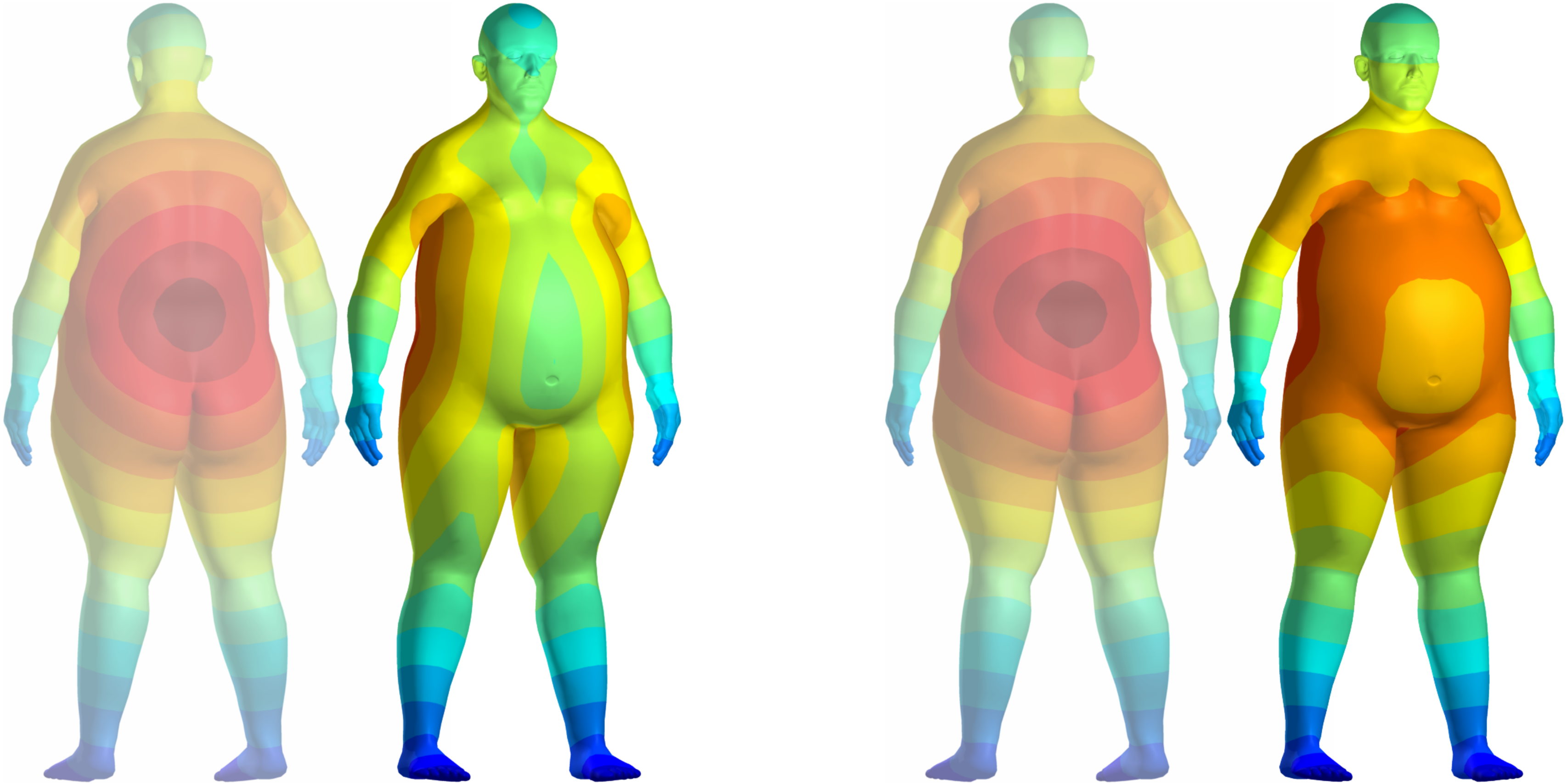
INTERACTIVE GEOMETRY LAB

October 9, 2013

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Additional material

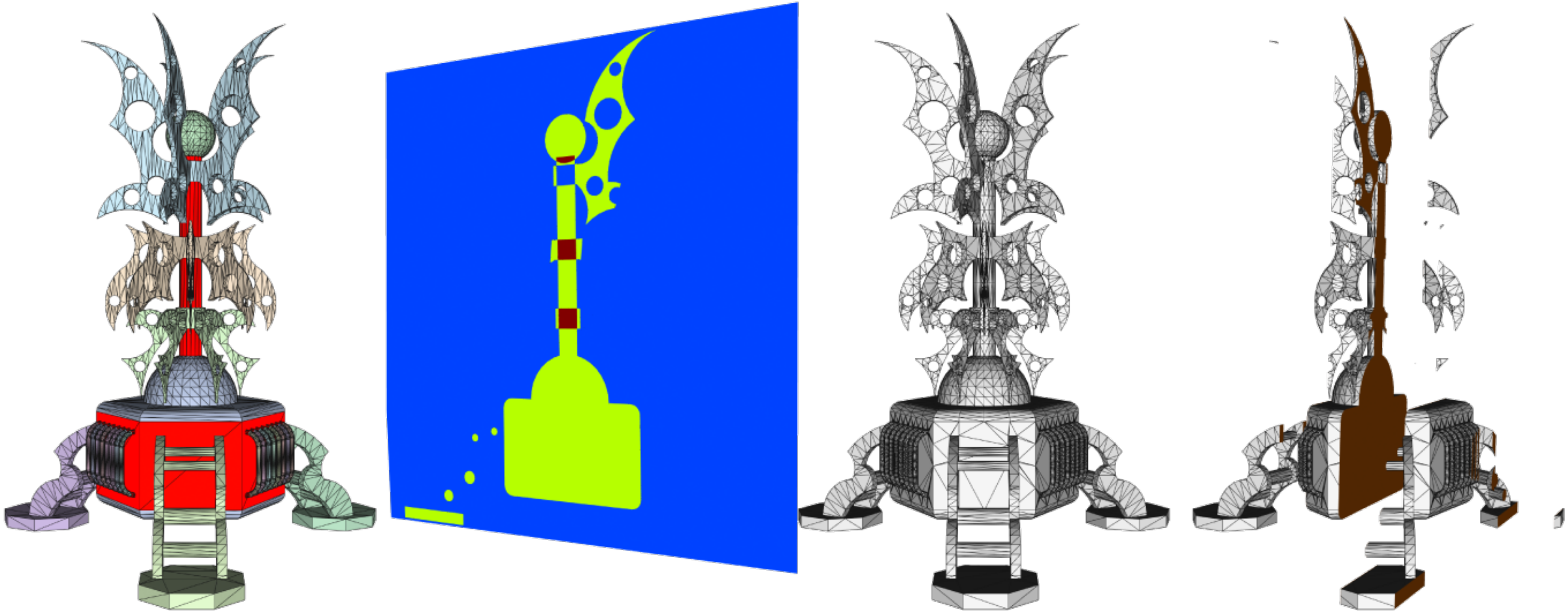
Surface processing is distinct from volumetric



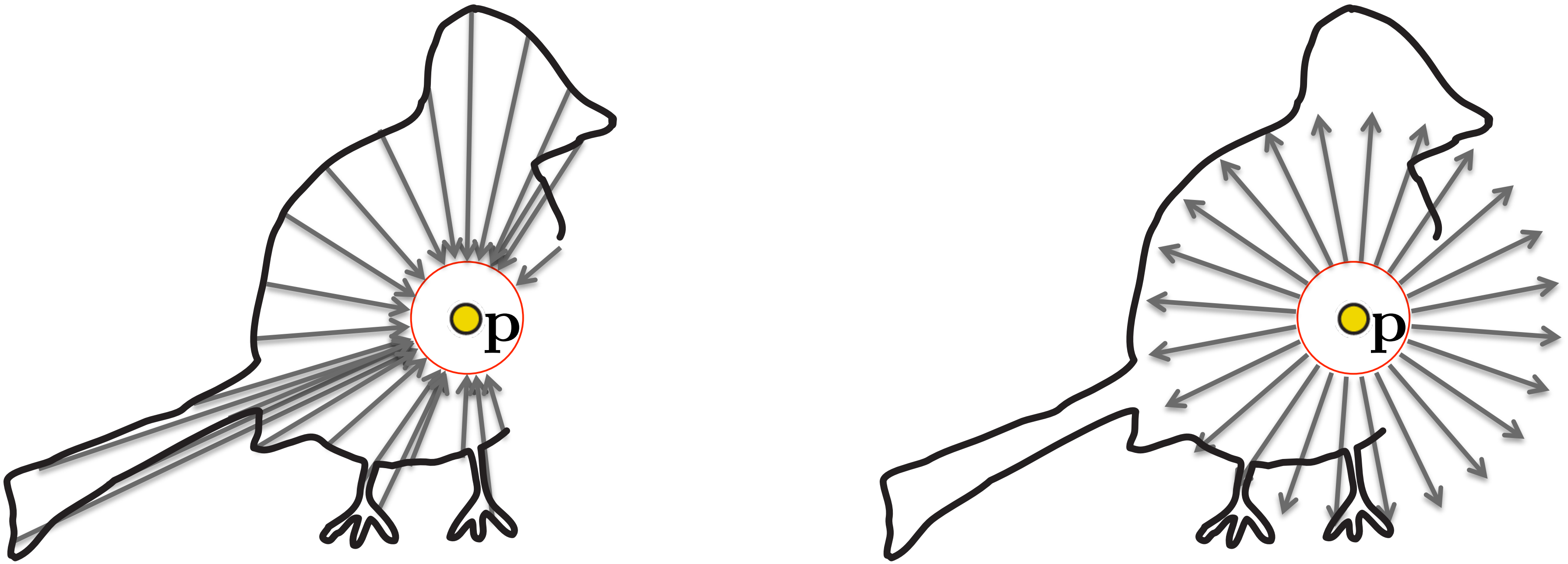
surface distance

volumetric distance

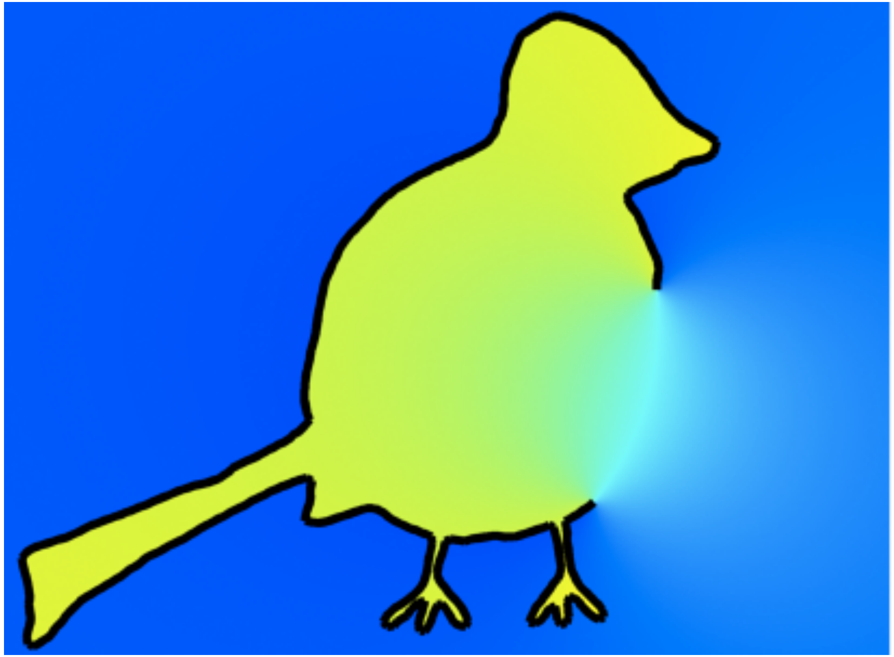
Brings a new level of robustness to volume meshing for a variety of shapes

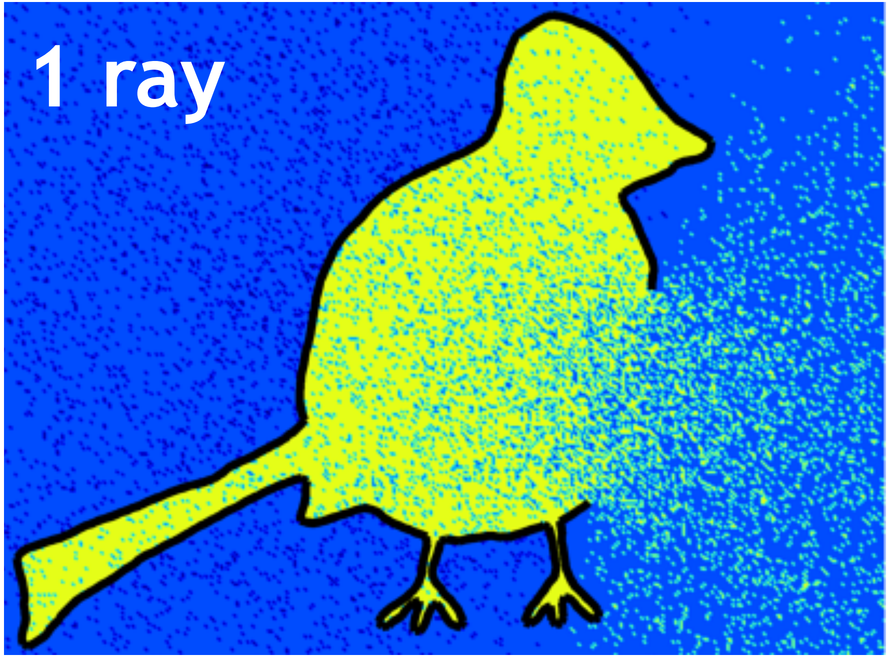


We rasterize the winding number, rather than ray cast

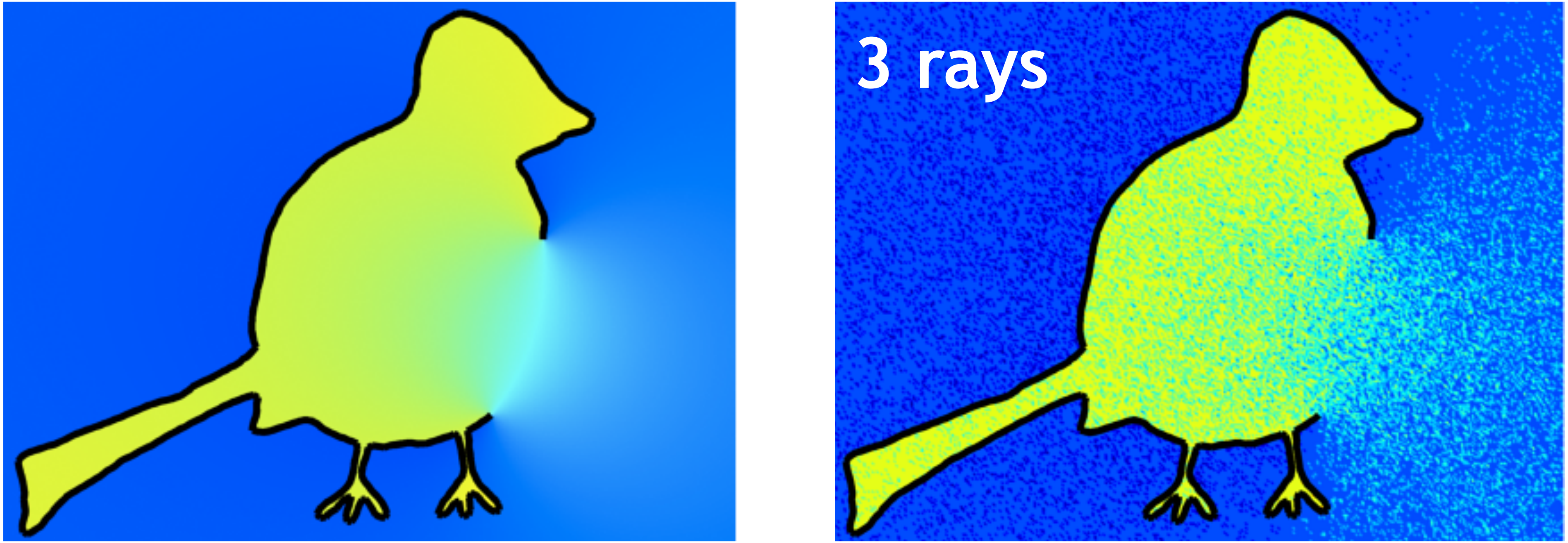


We rasterize the winding number, rather than ray cast

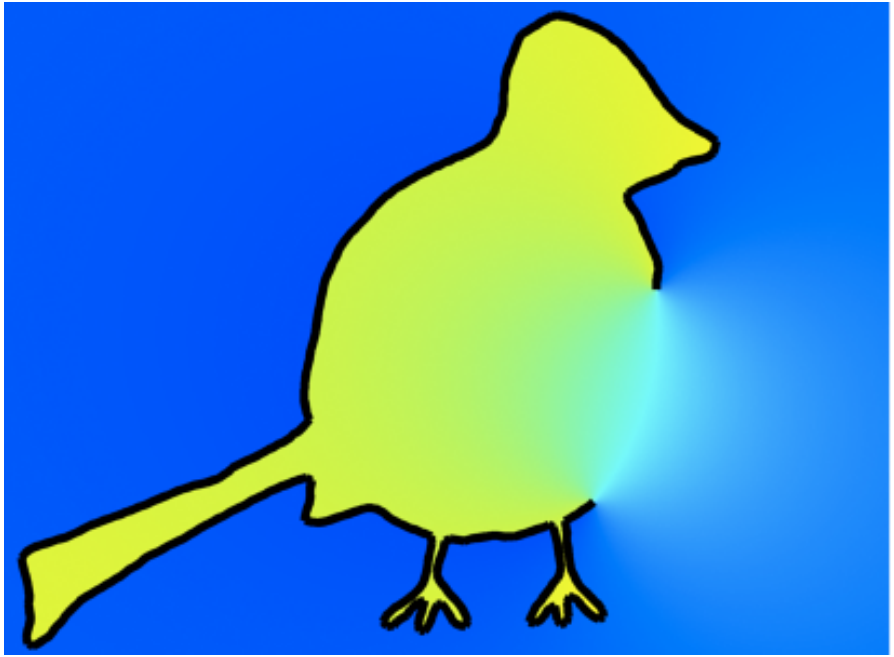


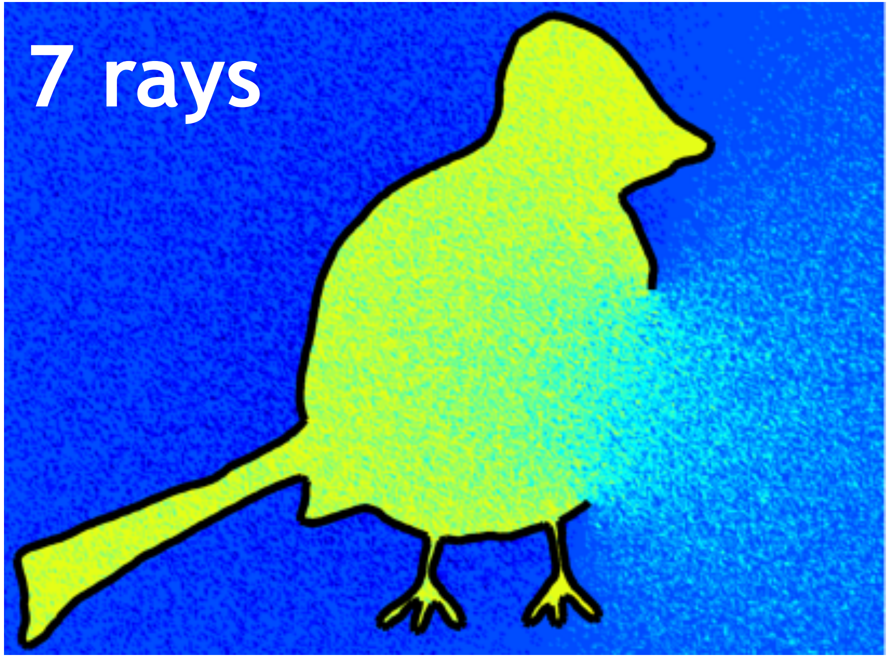


We rasterize the winding number, rather than ray cast

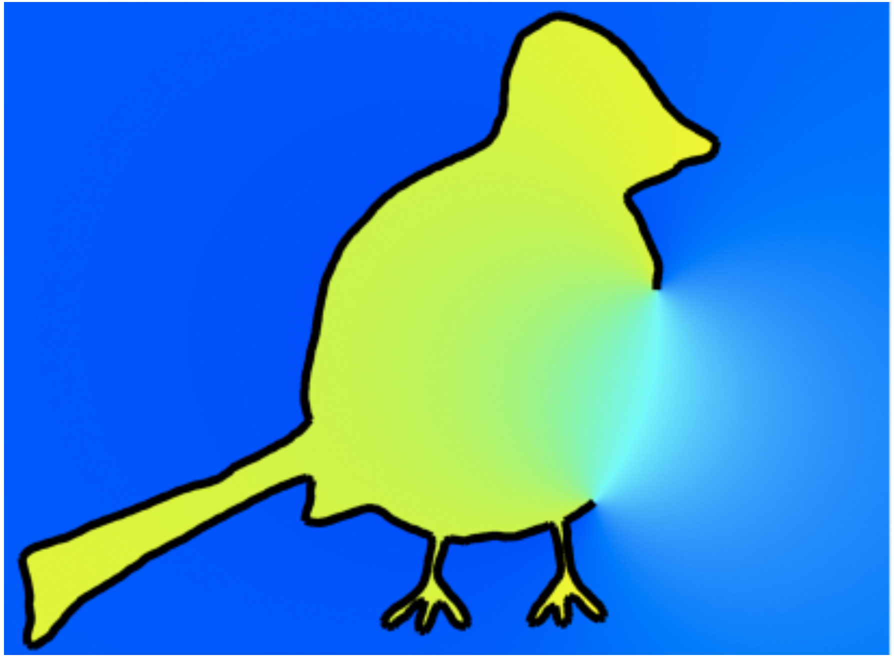


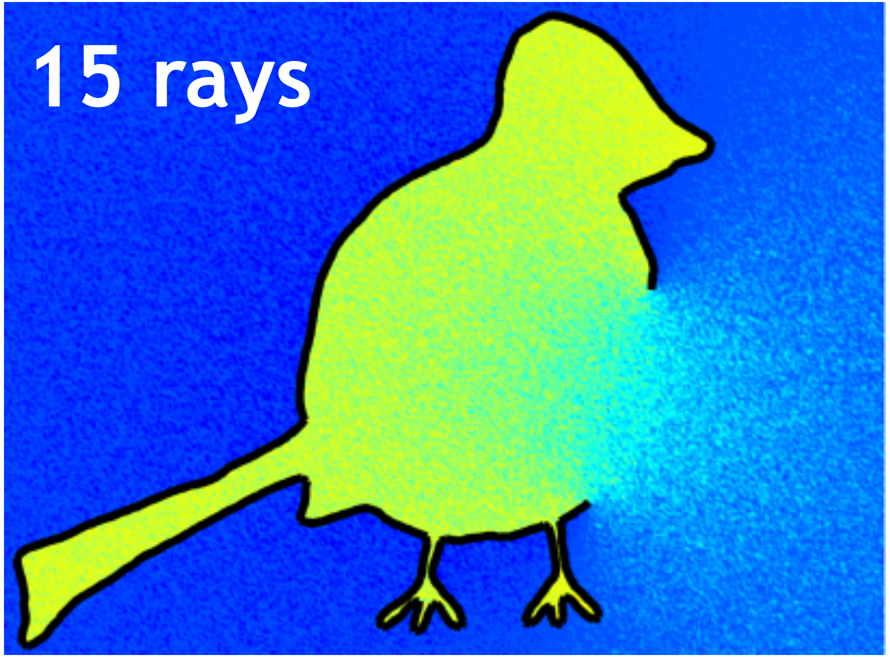
We rasterize the winding number, rather than ray cast



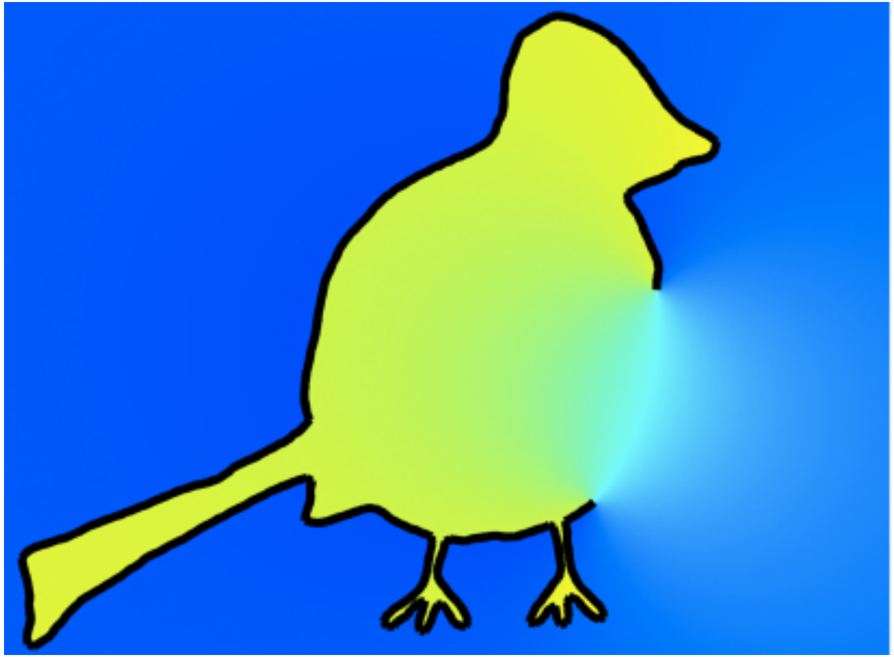


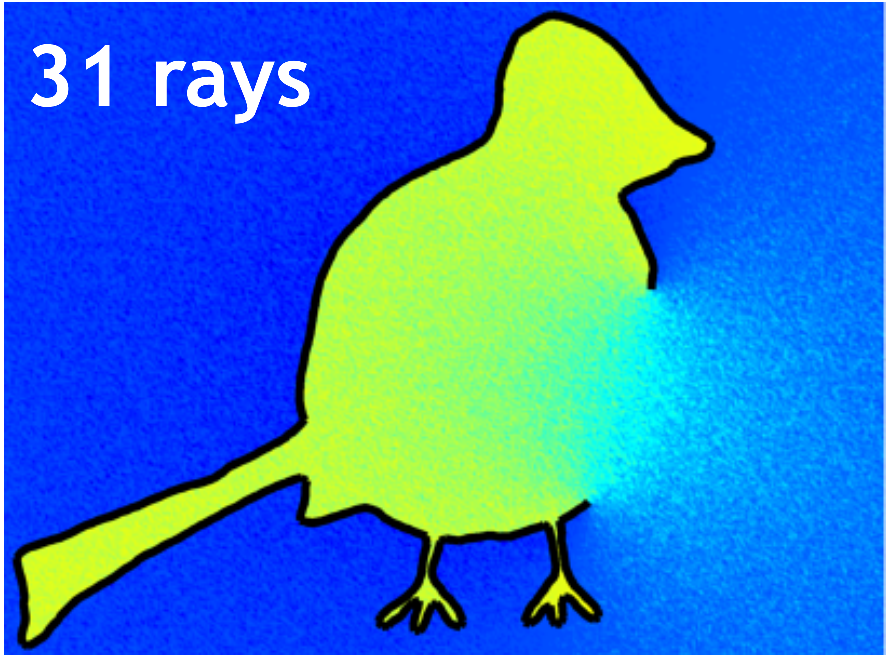
We rasterize the winding number, rather than ray cast



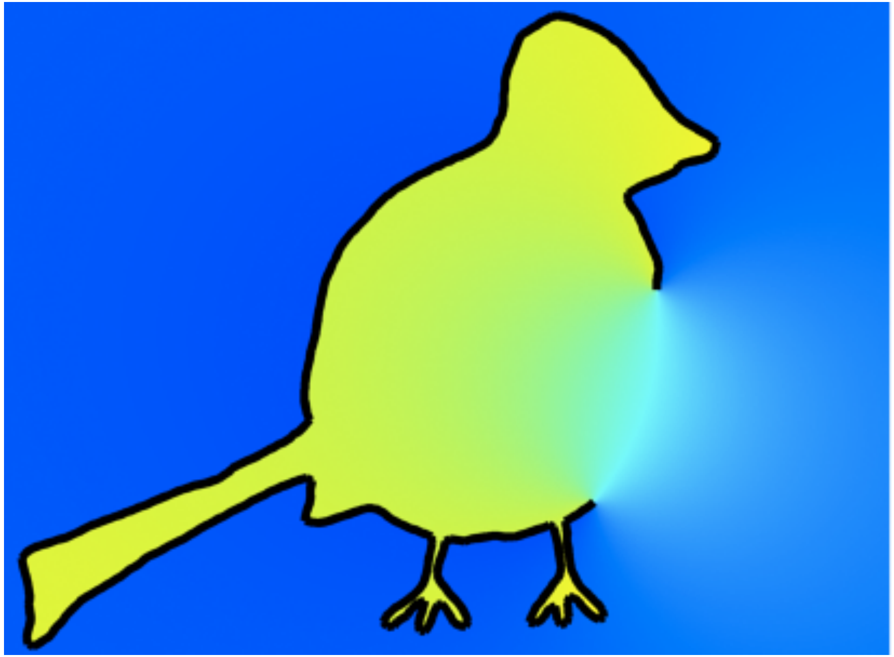


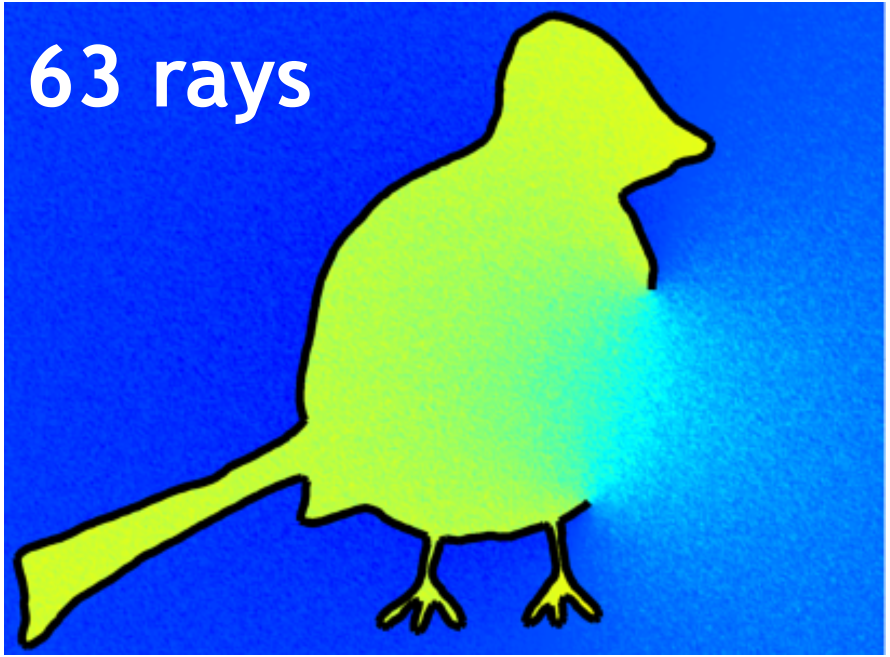
We rasterize the winding number, rather than ray cast



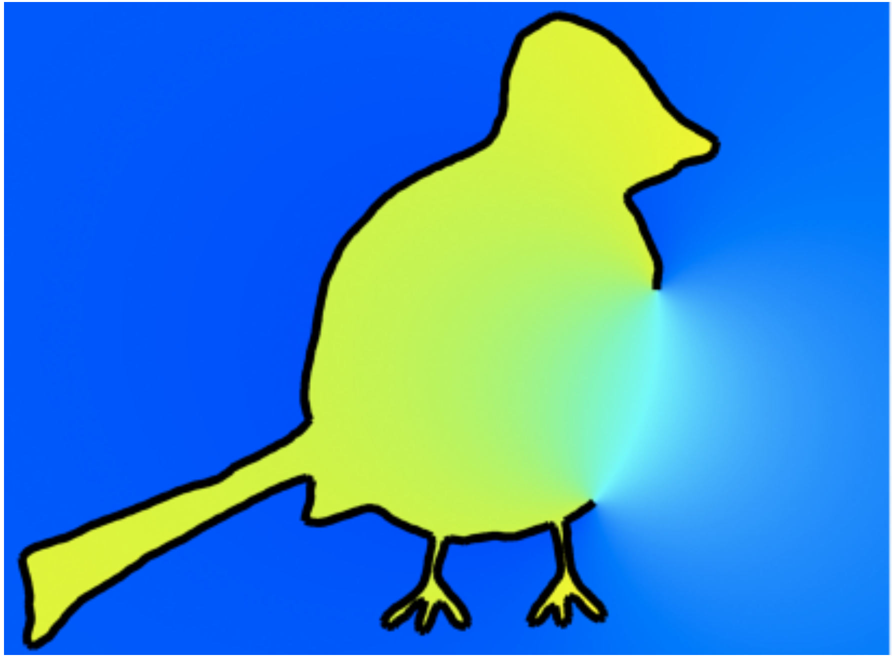


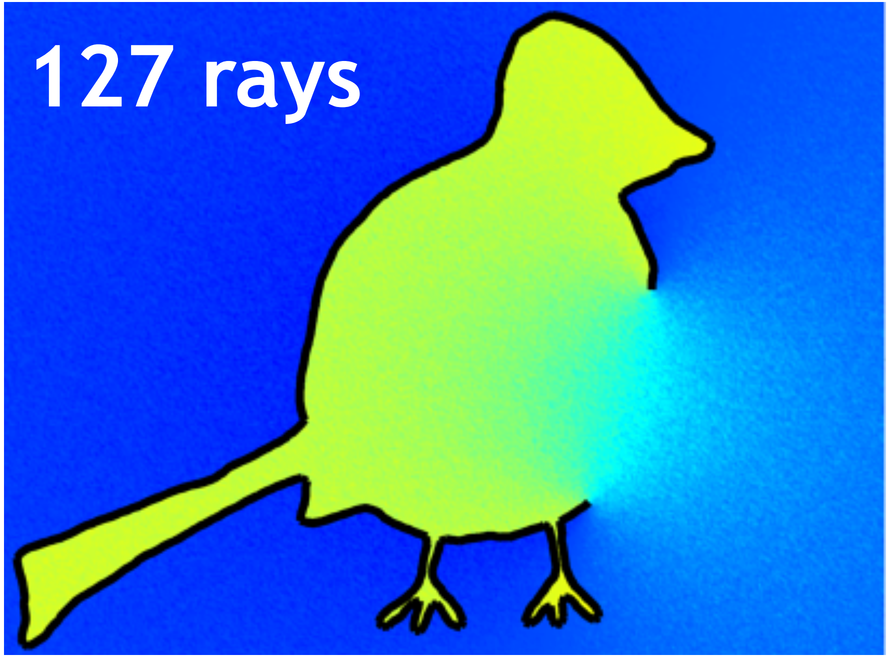
We rasterize the winding number, rather than ray cast



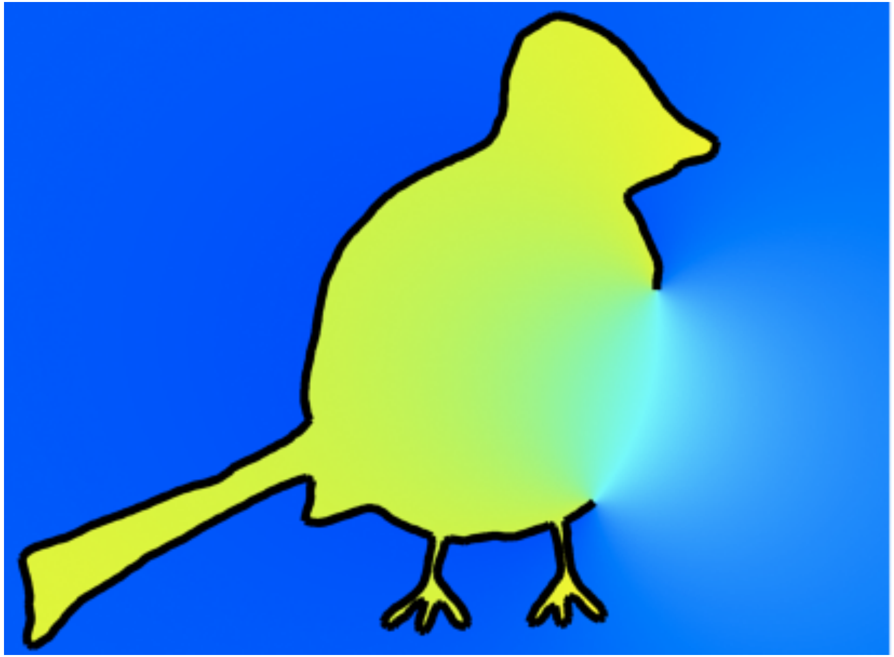


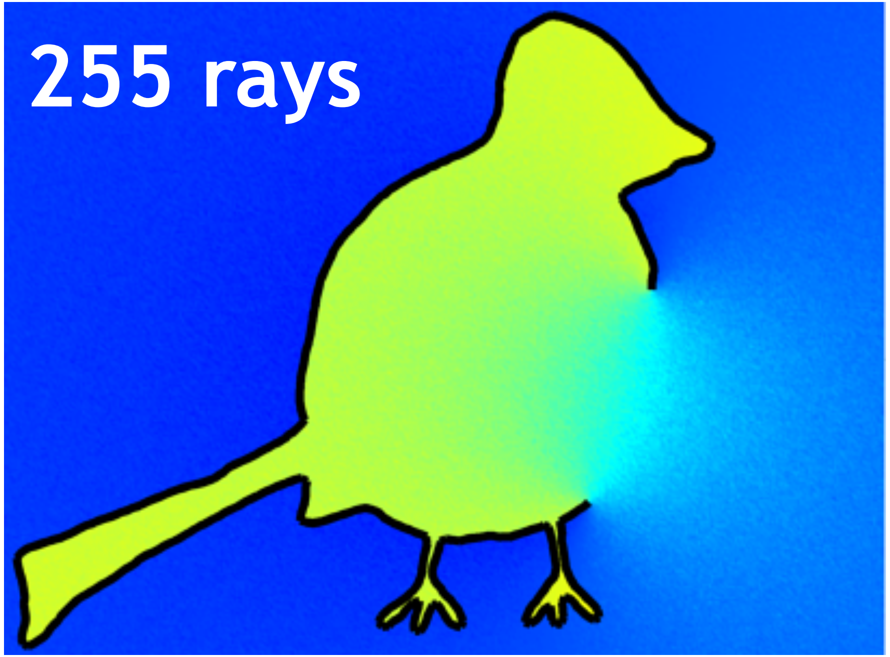
We rasterize the winding number, rather than ray cast



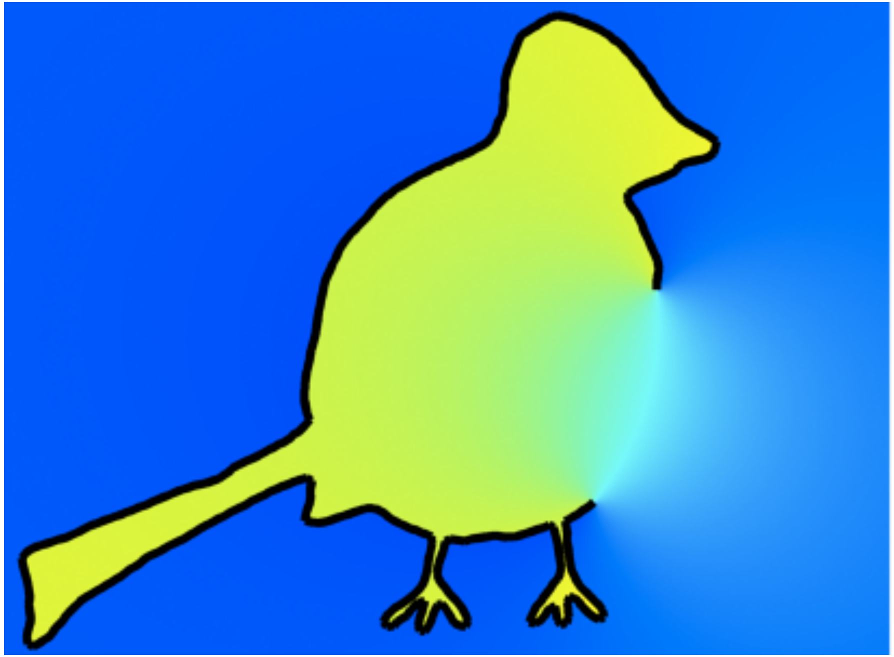


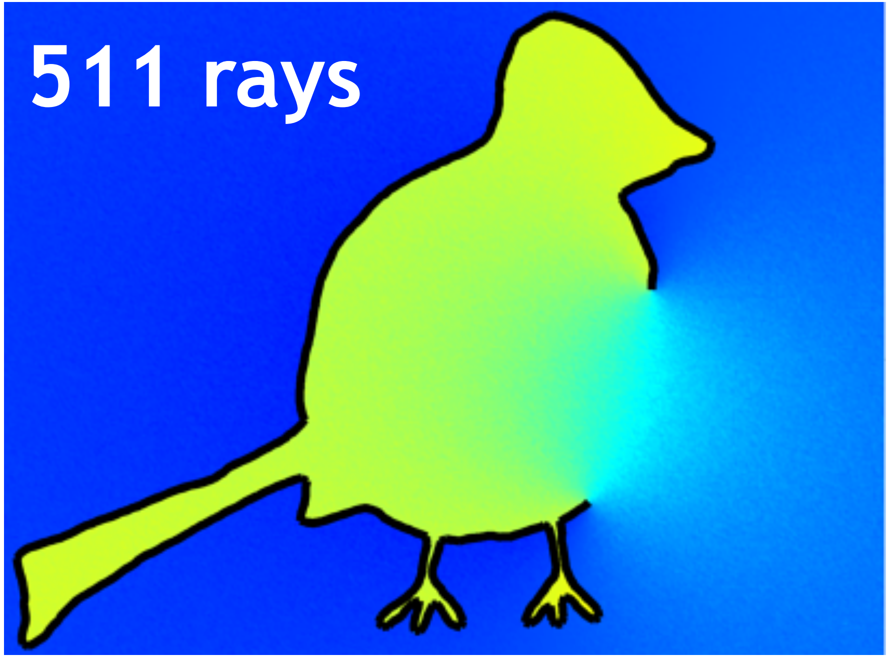
We rasterize the winding number, rather than ray cast



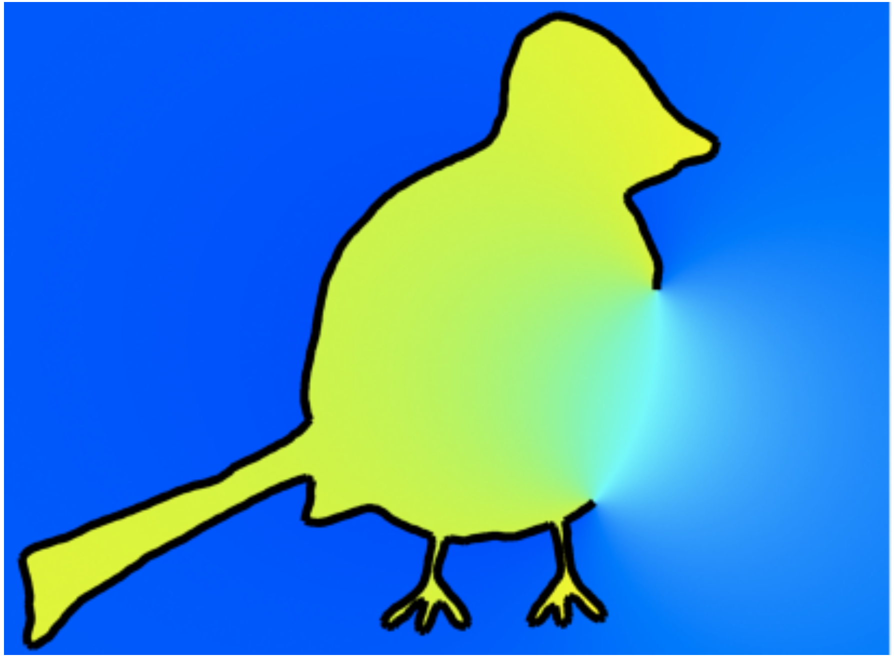


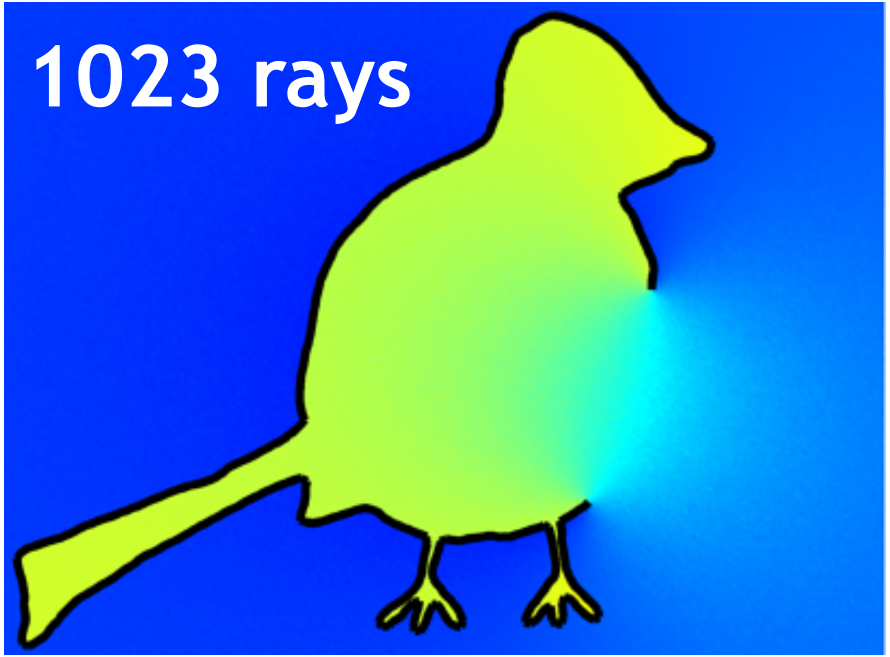
We rasterize the winding number, rather than ray cast



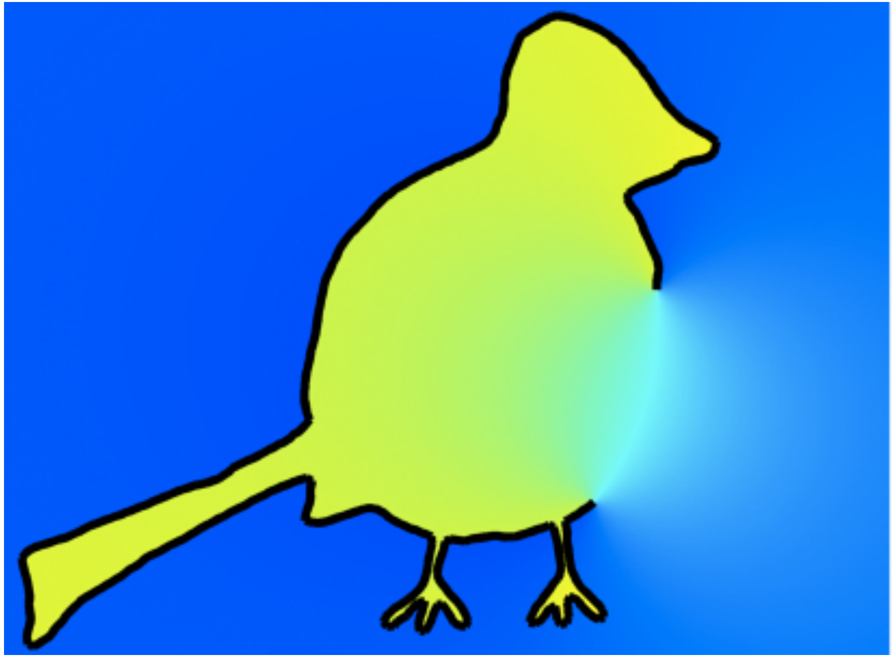


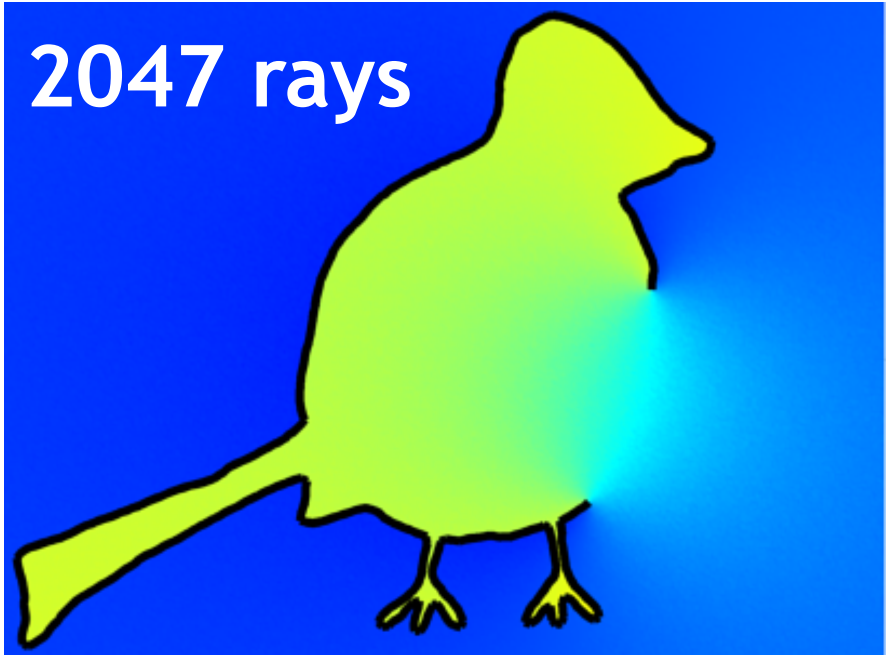
We rasterize the winding number, rather than ray cast



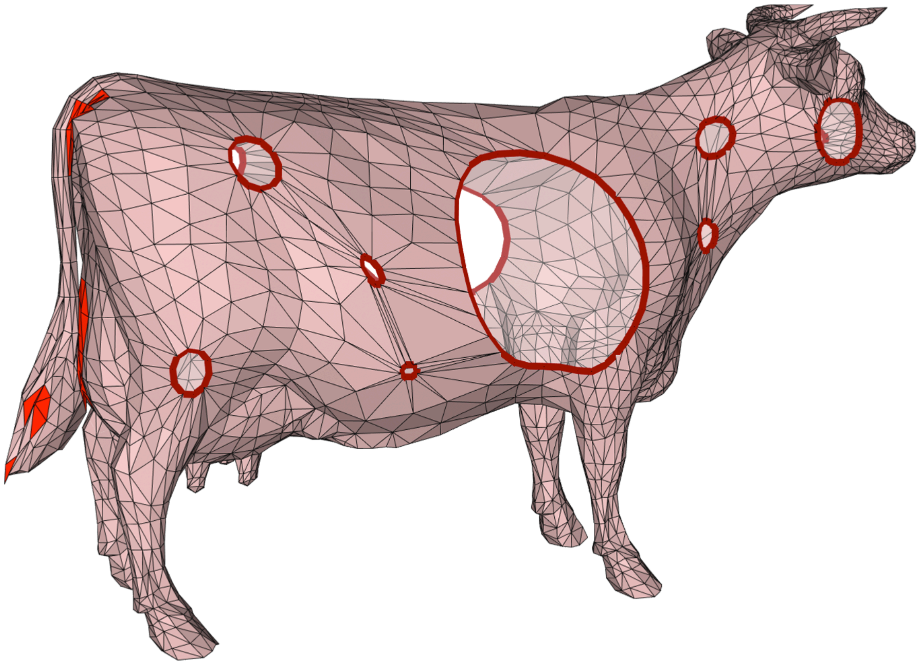


We rasterize the winding number, rather than ray cast

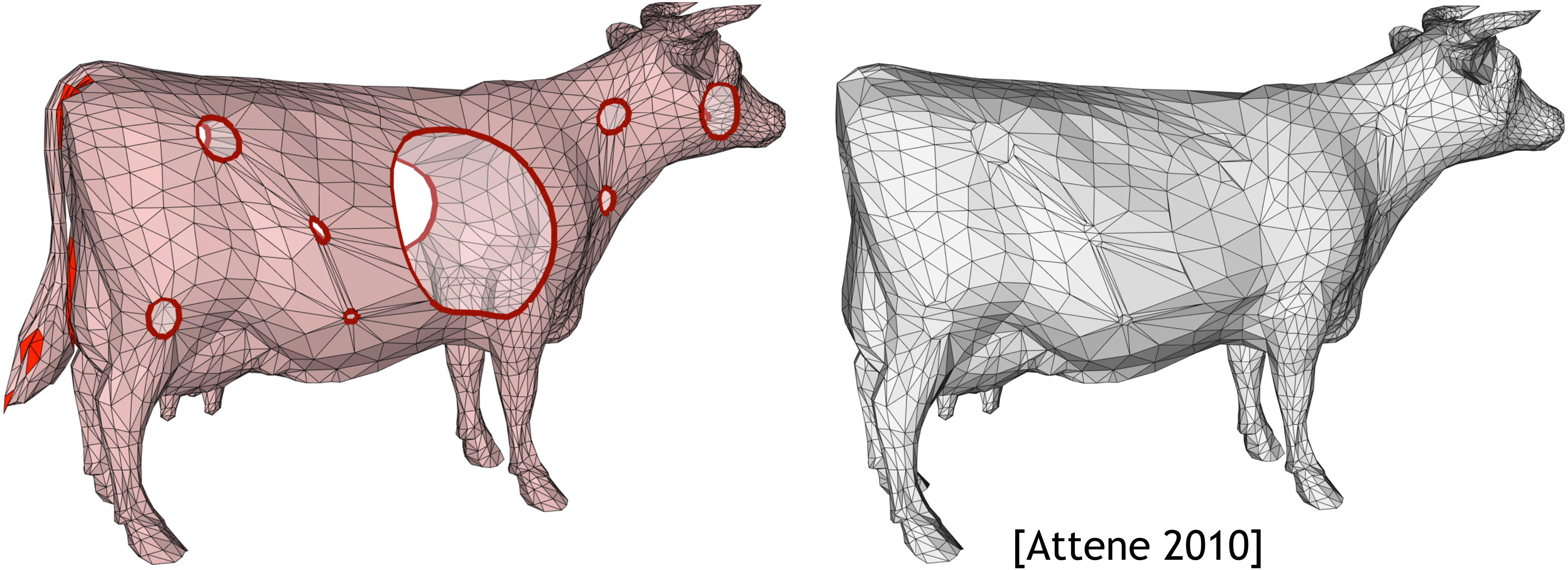




Surface cleanup methods modify the input too much

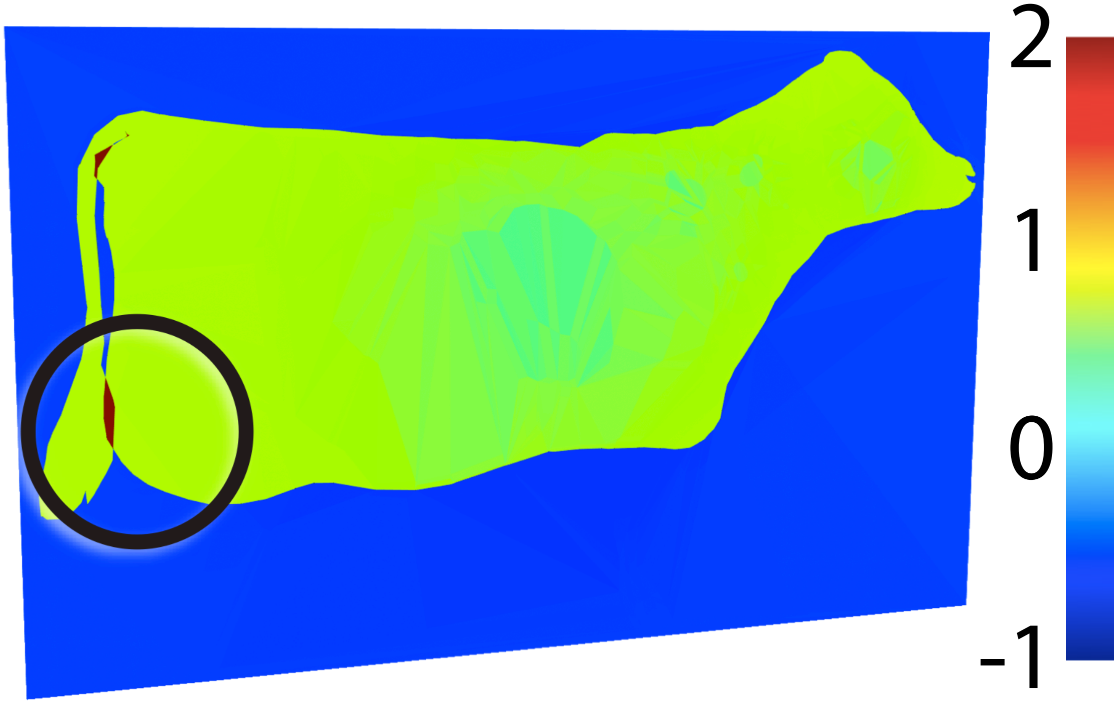


Surface cleanup methods modify the input too much

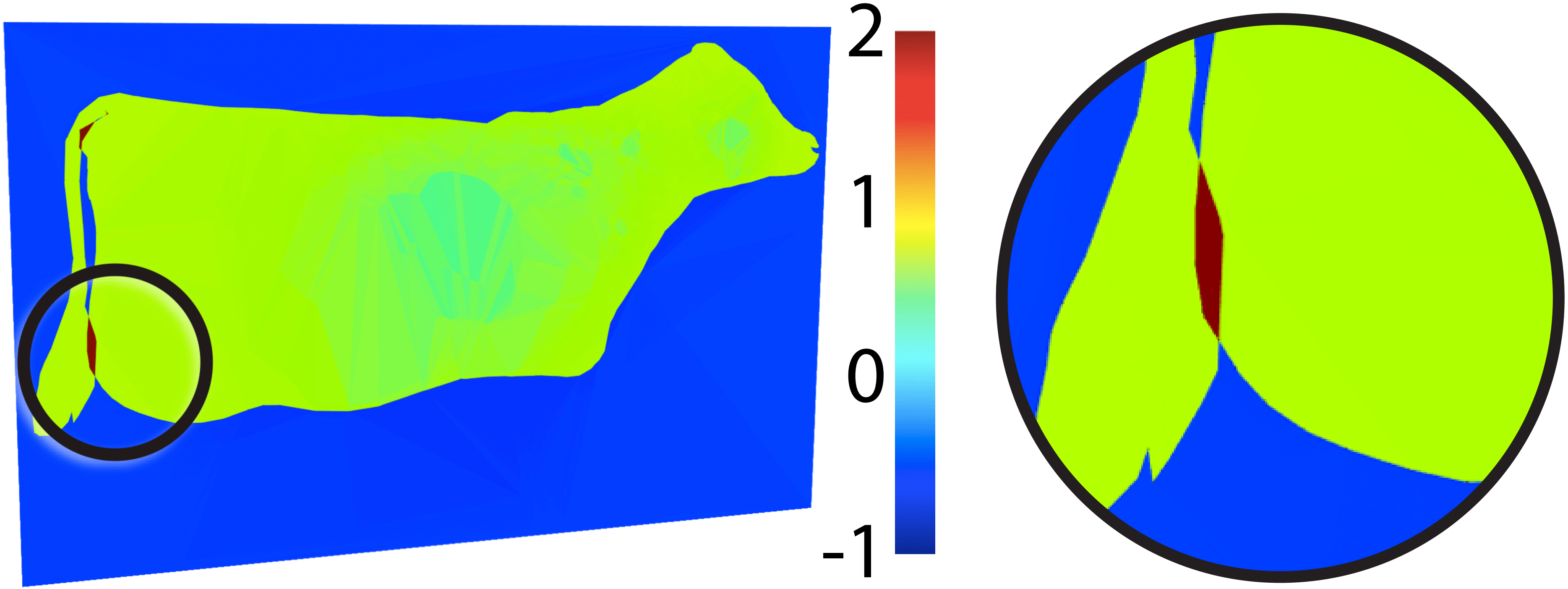


[Attene 2010]

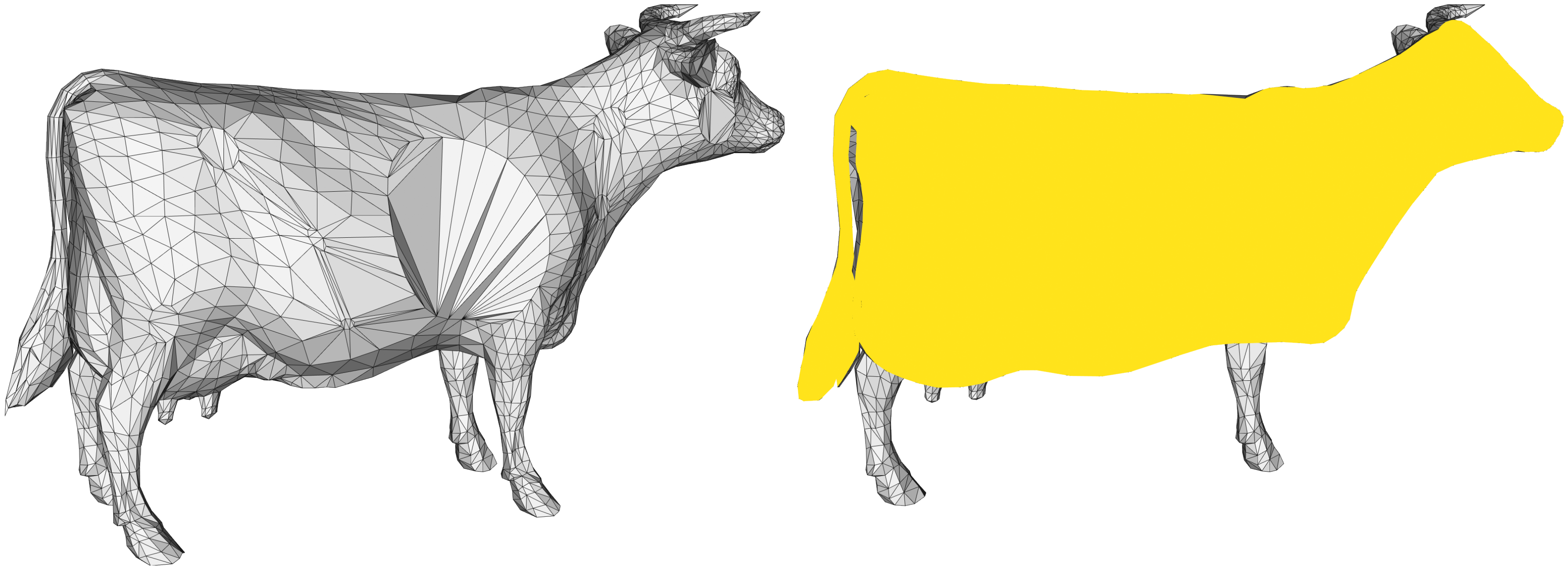
Winding number tells more than just inside: *how many times inside*



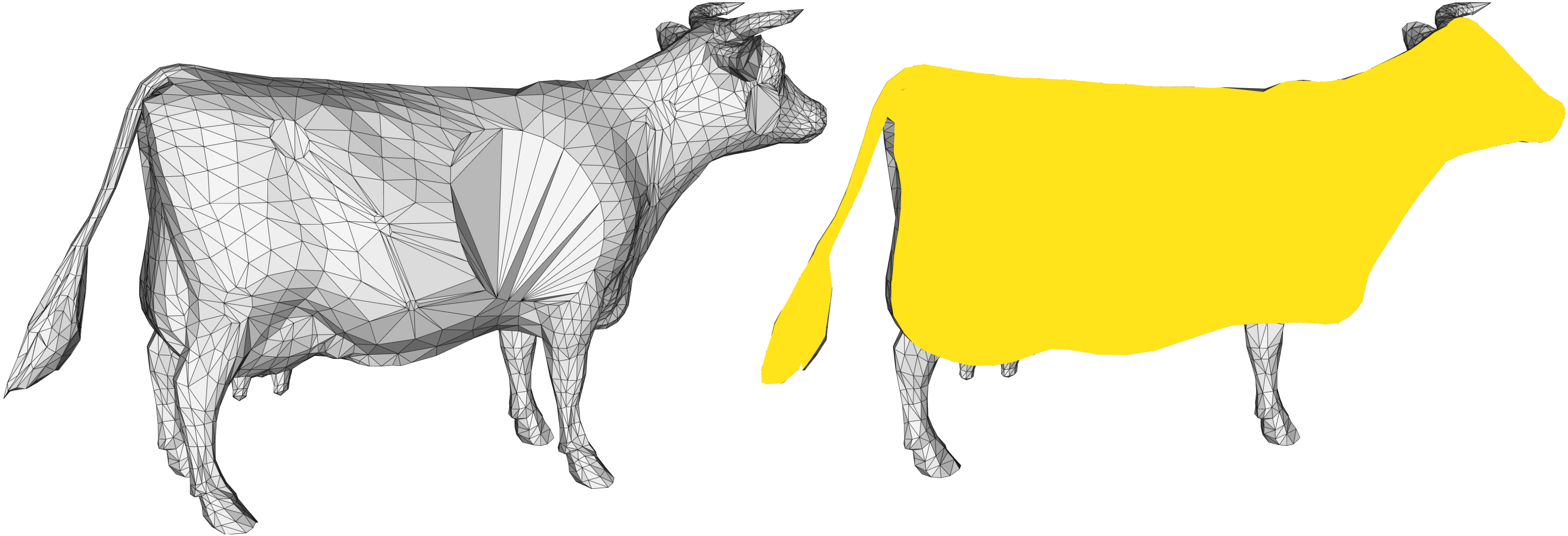
Winding number tells more than just inside: *how many times inside*



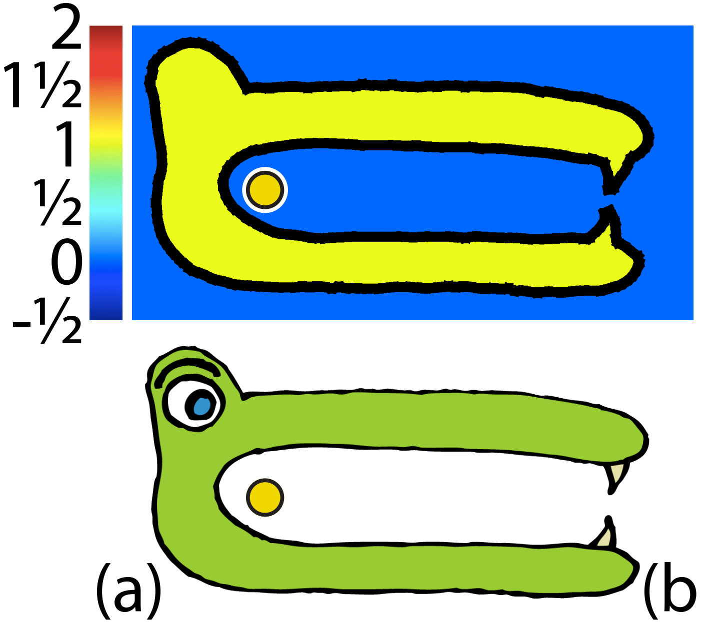
Duplicate any multiply inside parts: consistently overlapping tet mesh



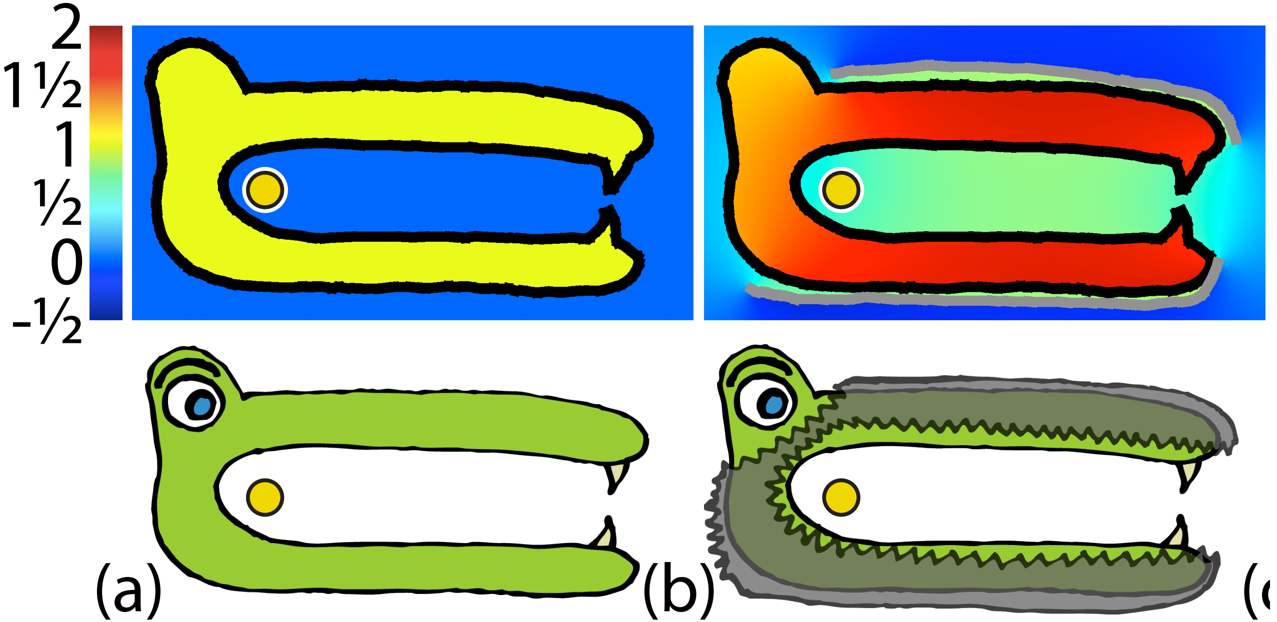
Duplicate any multiply inside parts: consistently overlapping tet mesh



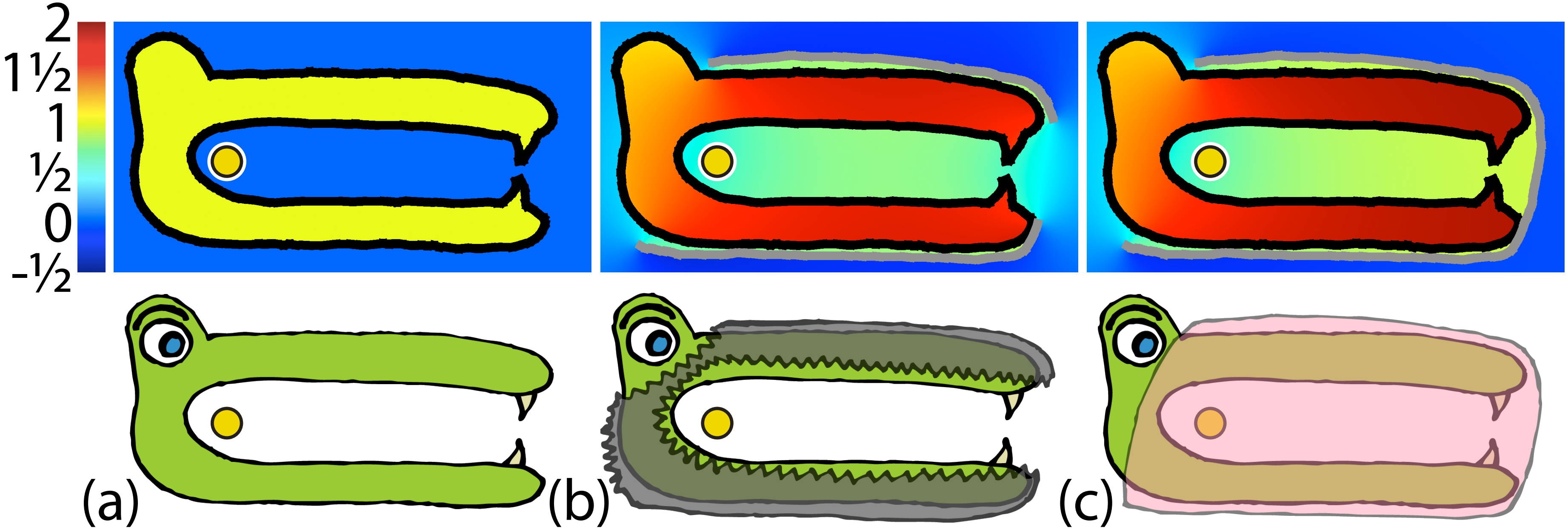
Some ambiguities are just semantics



Some ambiguities are just semantics



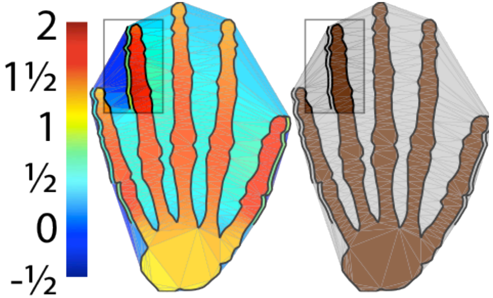
Some ambiguities are just semantics

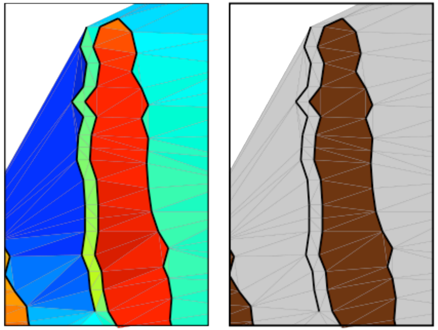


Simple thresholding is not enough

$$\text{is\_outside}(e_i) = \begin{cases} \text{true} & \text{if } w(e_i) < 0.5 \\ \text{false} & \text{otherwise} \end{cases}$$

Each element in CDT





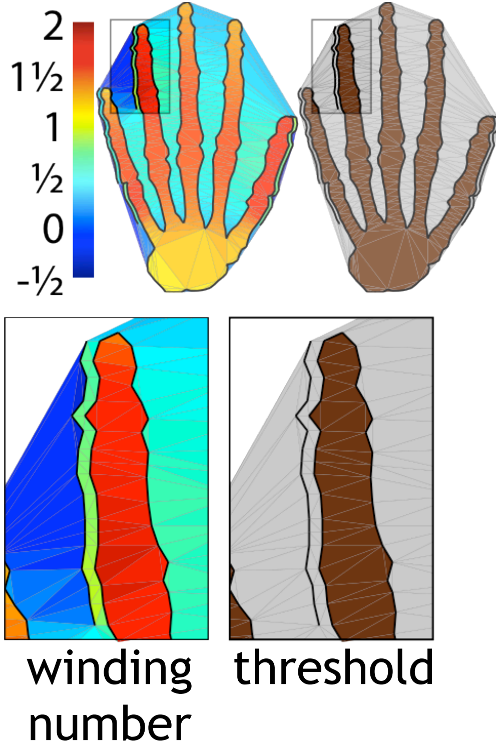
winding number

threshold

Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

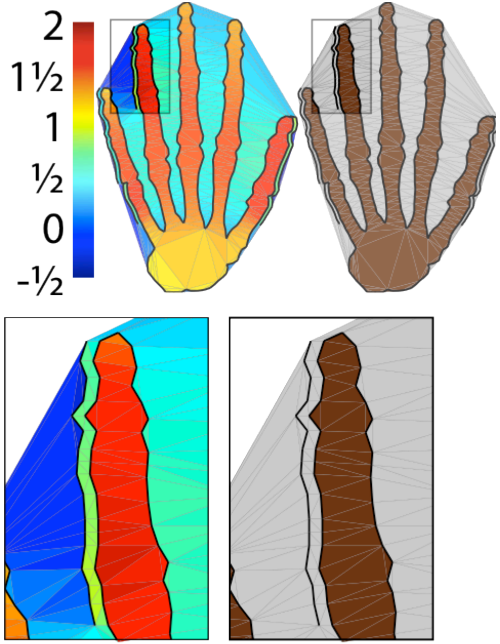
data coherency



Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

$$u(x_i) = \begin{cases} \max(w(e_i) - 0, 0) & \text{if } x_i = \texttt{outside} \\ \max(1 - w(e_i), 0) & \text{otherwise} \end{cases}$$

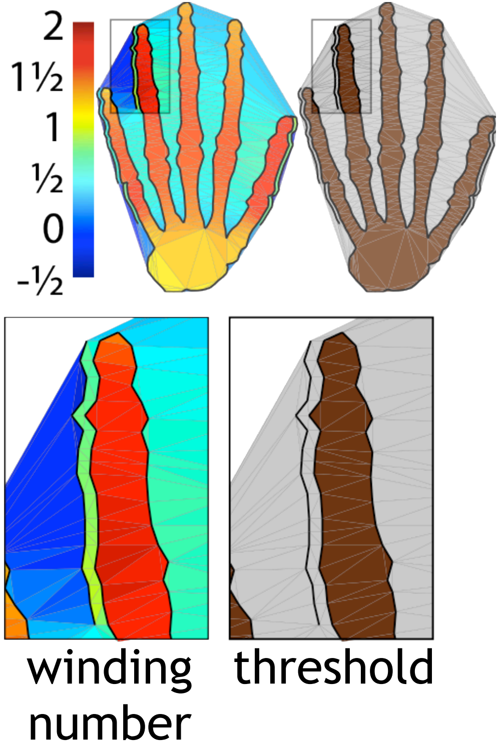


winding number threshold

Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

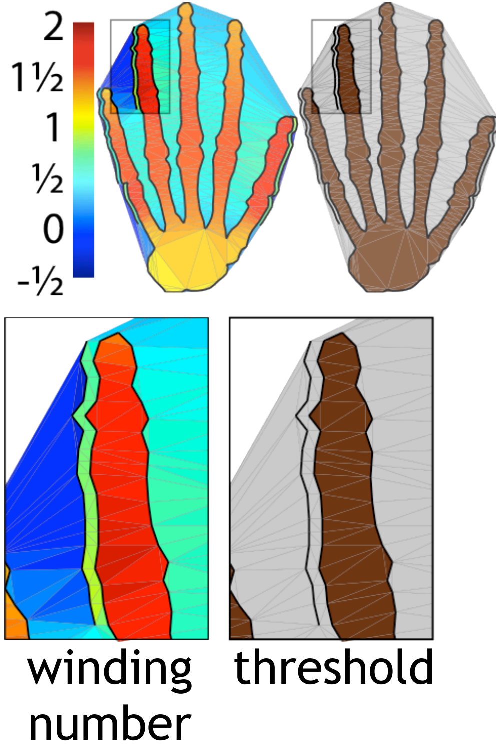
$$v(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j \\ \frac{a_{ij} \exp\left(|w(e_i) - w(e_j)|^2\right)}{2\sigma^2} & \text{otherwise} \end{cases}$$



Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

$$\underset{\mathbf{x} | x_i \in [0,1]}{\text{argmin}} \; E(\mathbf{x})$$ use graphcut (maxflow)

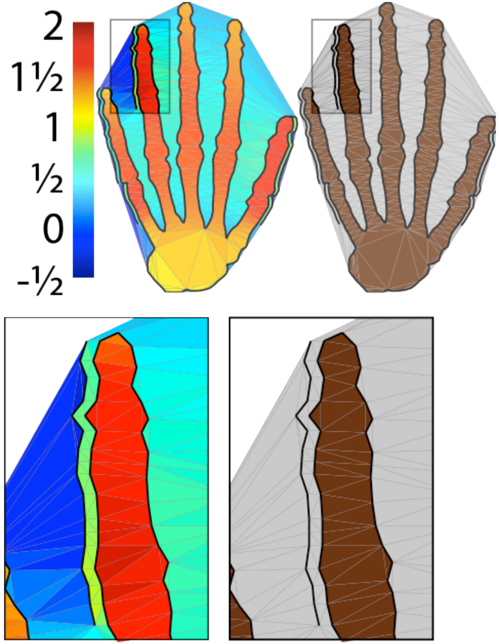


Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

$$\underset{\mathbf{x} | x_i \in [0,1]}{\operatorname{argmin}} \; E(\mathbf{x})$$ use graphcut (maxflow)

subject to hard *facet constraints*



winding number threshold

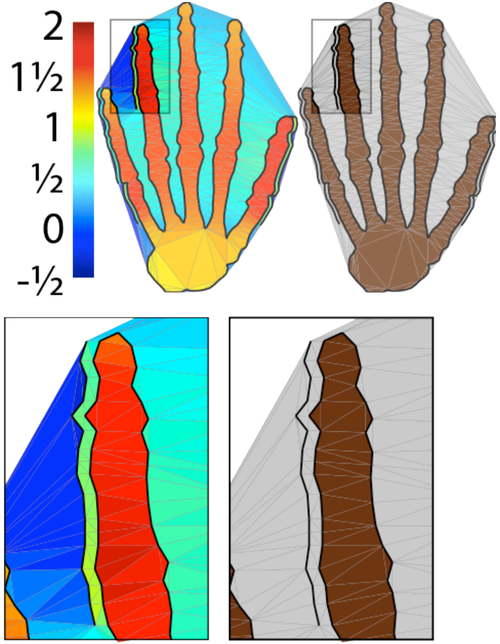
Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

$$\underset{\mathbf{x} | x_i \in [0,1]}{\operatorname{argmin}} \; E(\mathbf{x})$$ use graphcut (maxflow)

subject to hard *facet constraints*

"nonregular"
[Kolmogorov & Zabin 2004]

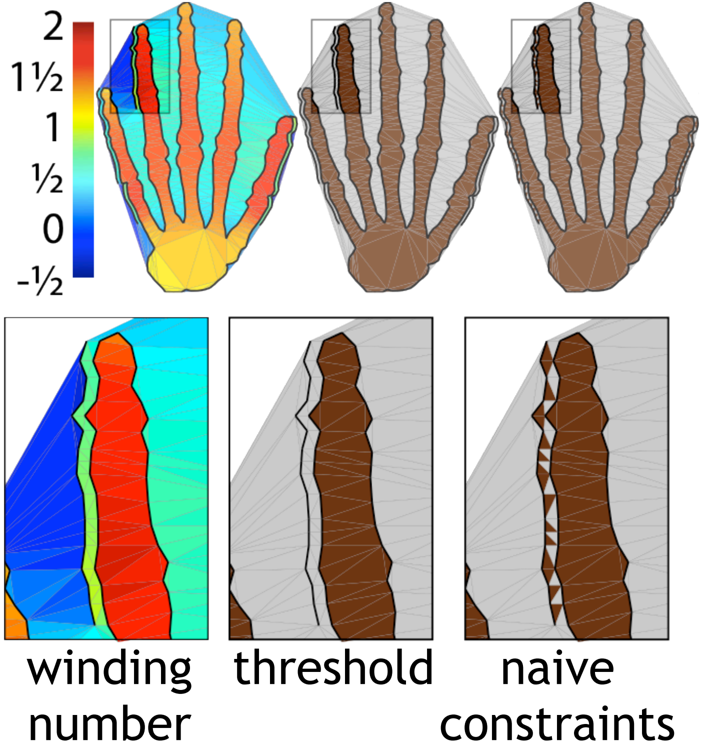


Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

$$\underset{\mathbf{x} | x_i \in [0,1]}{\operatorname{argmin}} \; E(\mathbf{x})$$ use graphcut (maxflow)

subject to hard *facet constraints*



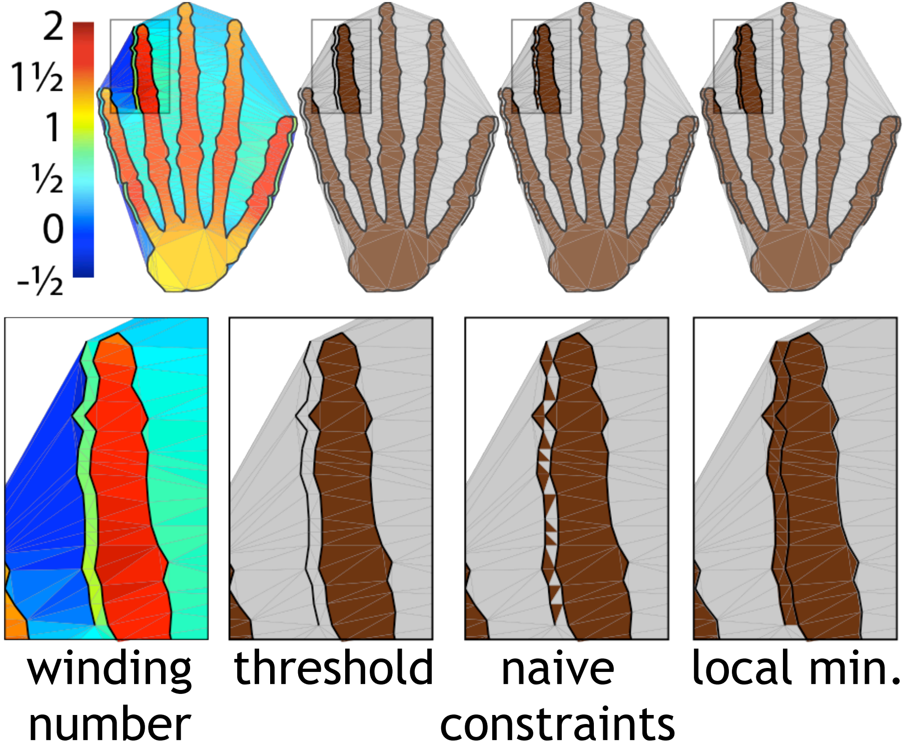
Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

$$\underset{\mathbf{x} | x_i \in [0,1]}{\operatorname{argmin}} \; E(\mathbf{x})$$ use graphcut (maxflow)

subject to hard *facet constraints*

use heuristic → local min.



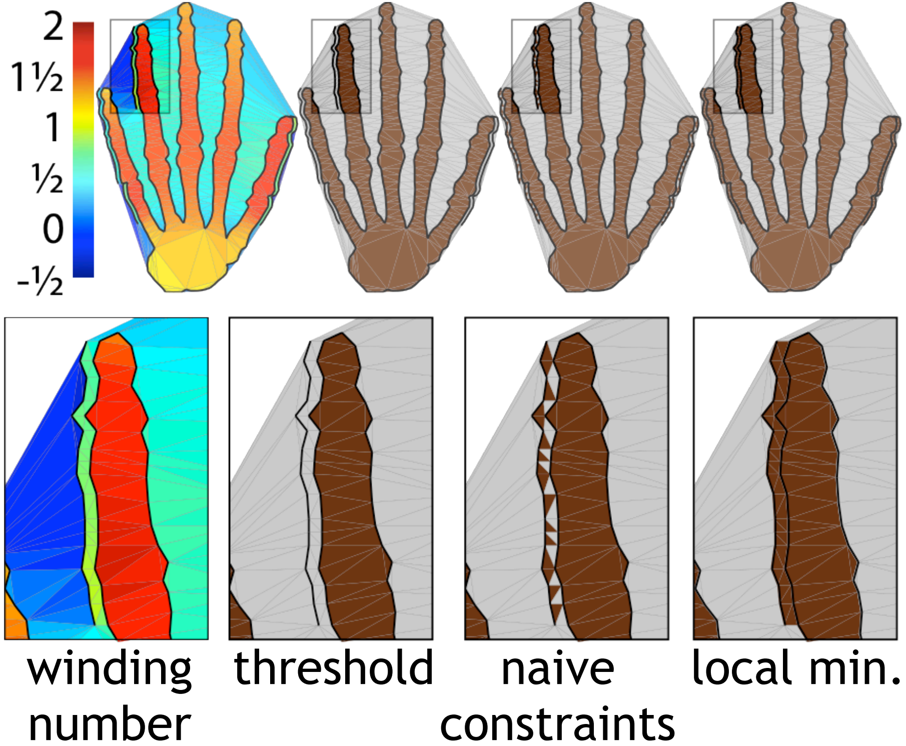
Graphcut encourages coherency

$$E = \sum_{i=1}^{m} \left[u(x_i) + \gamma \frac{1}{2} \sum_{j \in N(i)} v(x_i, x_j) \right]$$

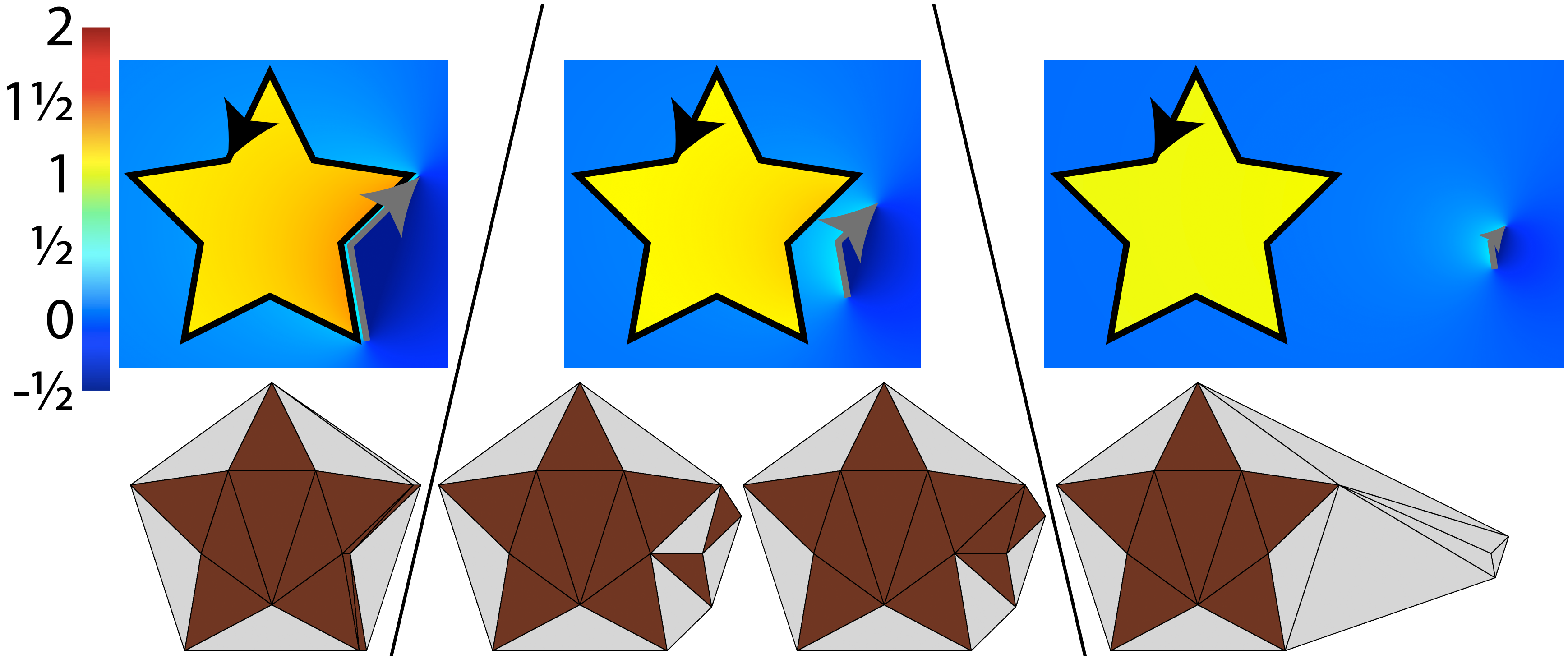
$$\underset{\mathbf{x} | x_i \in [0,1]}{\operatorname{argmin}} \; E(\mathbf{x})$$ use graphcut (maxflow)

subject to hard *facet constraints*

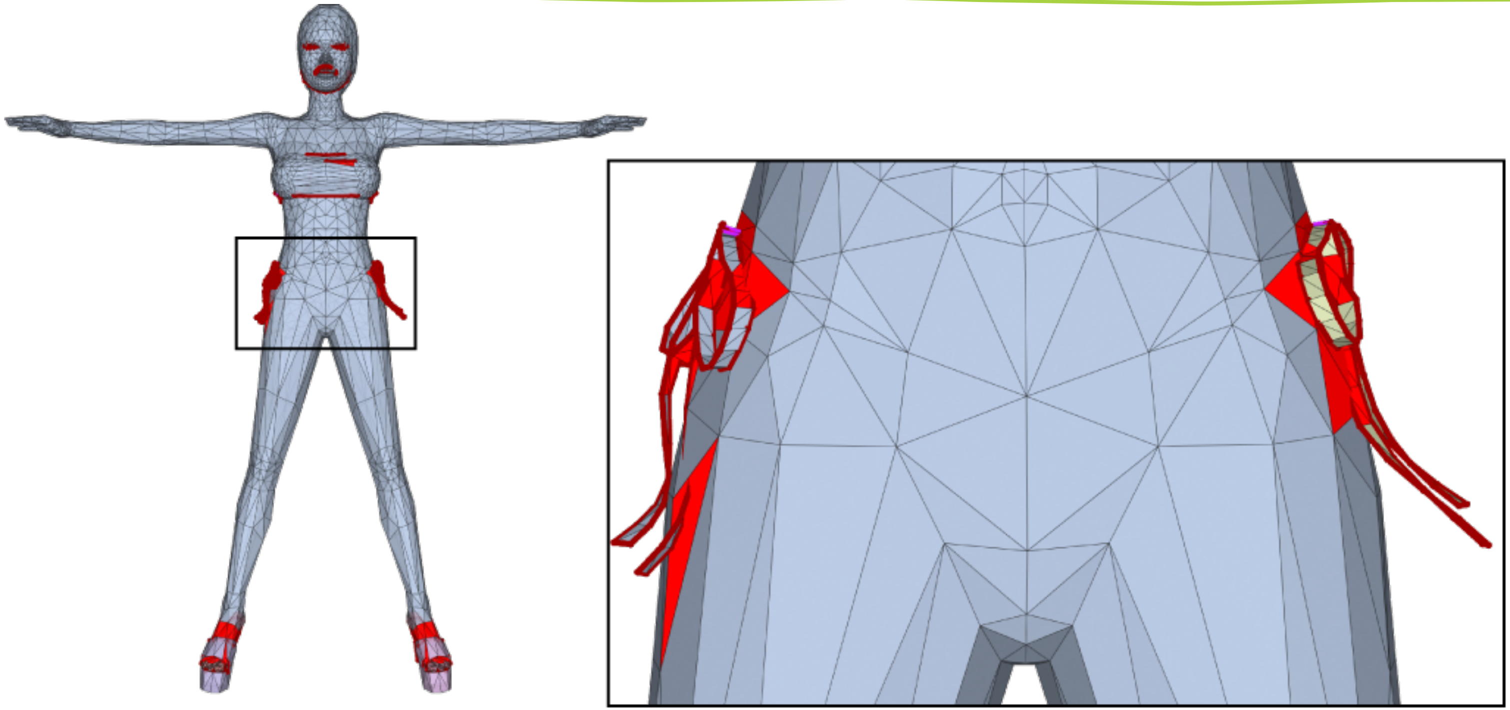
+subject to hard *manifoldness constraints*



Hard constraints are optional: outliers

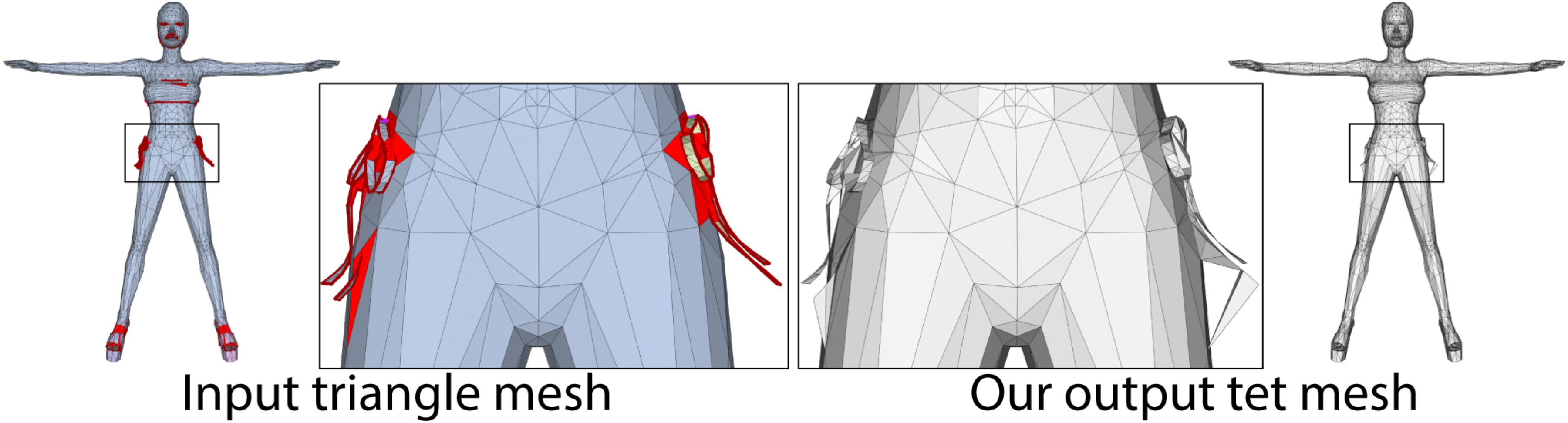


Even failure to create beautiful *surface*, can be success as volume representation

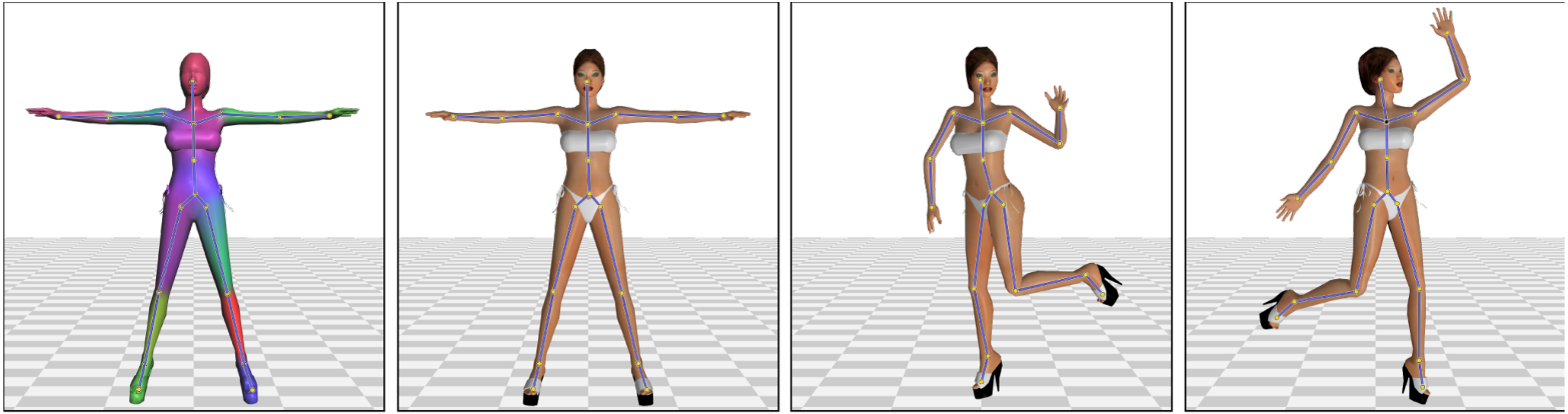


Input triangle mesh

Even failure to create beautiful *surface*, can be success as volume representation



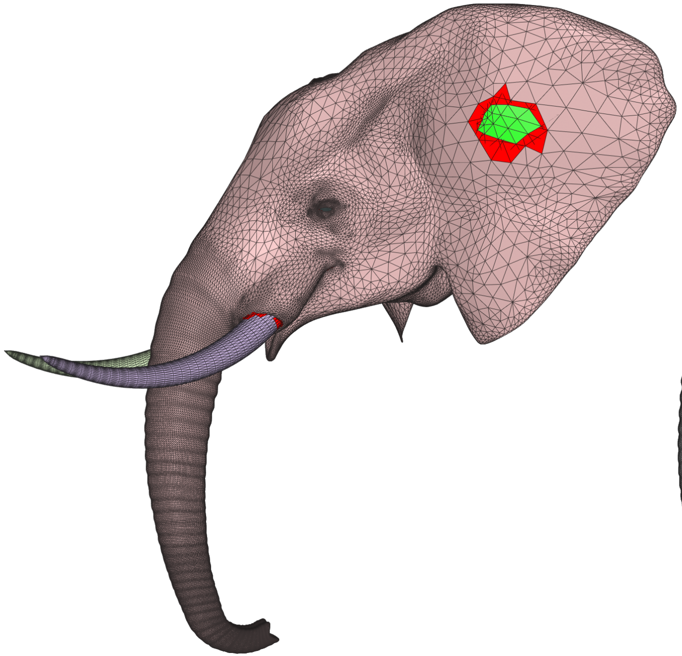
Even failure to create beautiful *surface*, can be success as volume representation



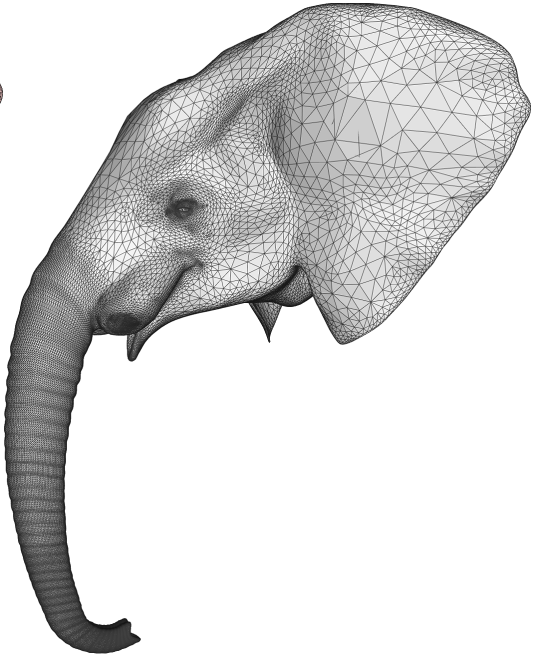
Auto. weights

Novel poses of textured input mesh

Cleanup methods modify input too much, …



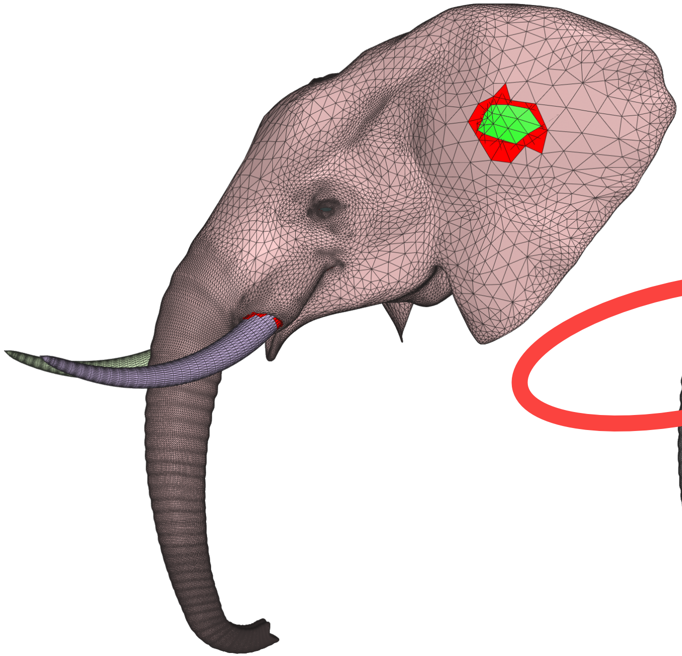
input mesh



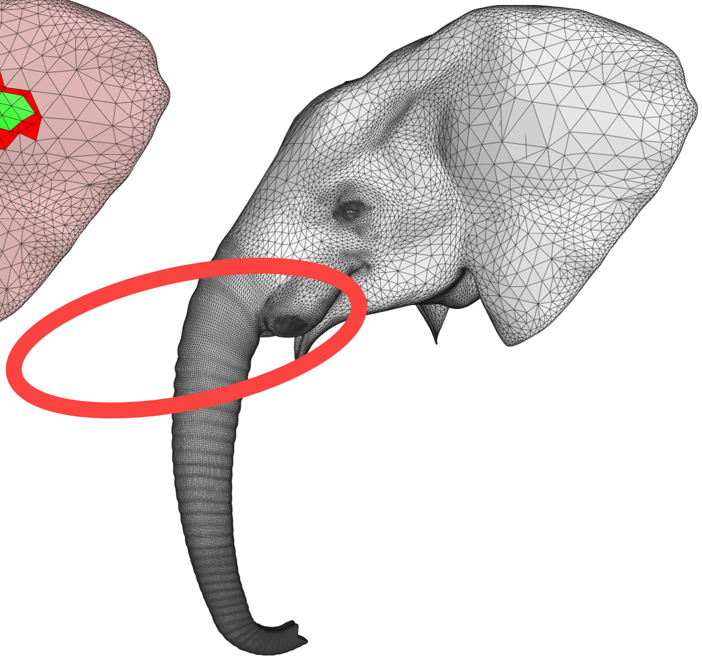
[Attene 2010]

ETH Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Cleanup methods modify input too much, …



input mesh



[Attene 2010]

… but we rely heavily on orientation