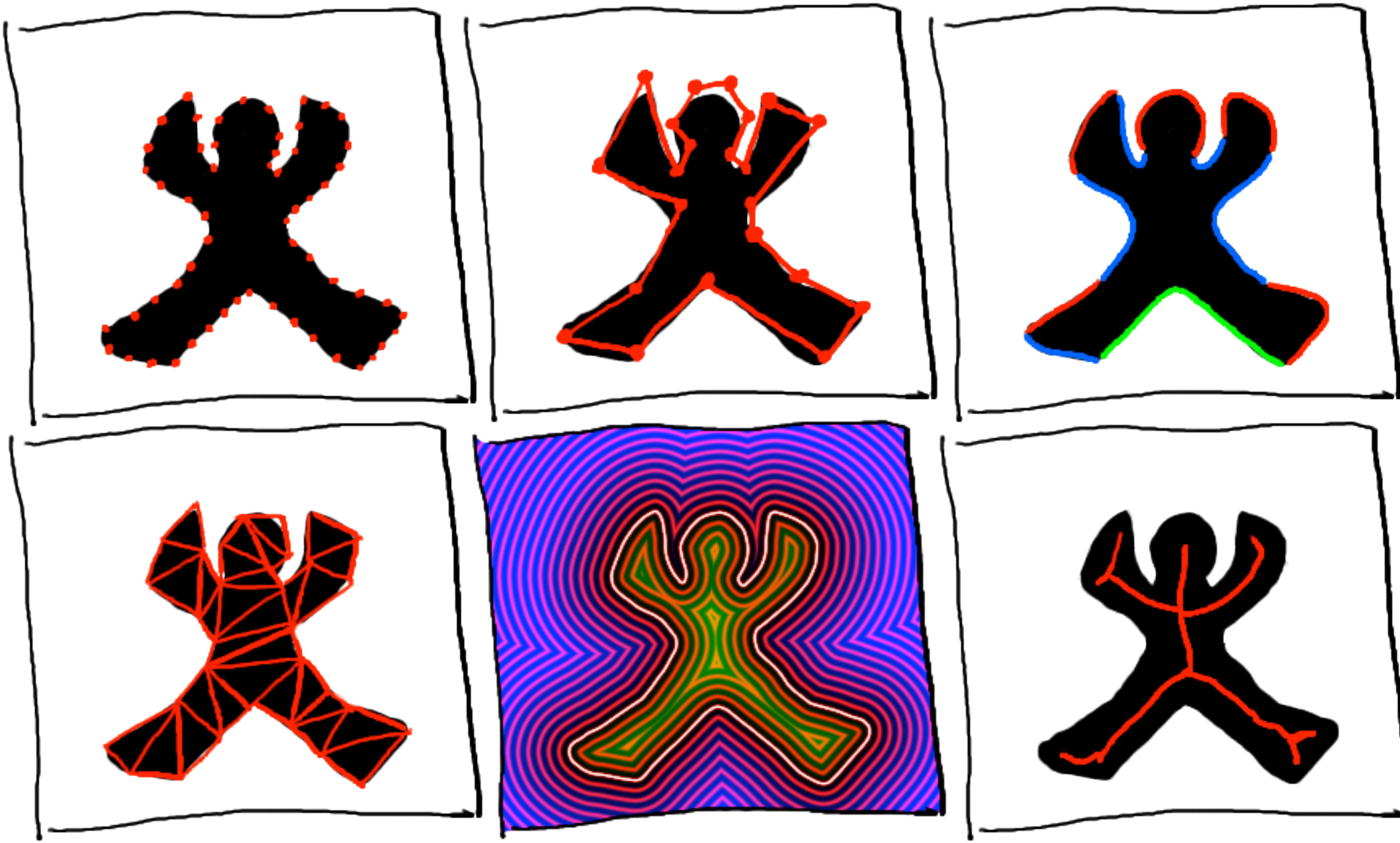
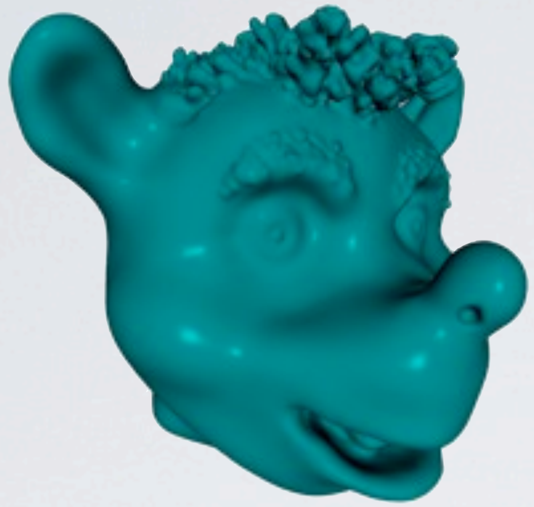
DEFORMABLE MODELS AND THEIR APPLICATIONS

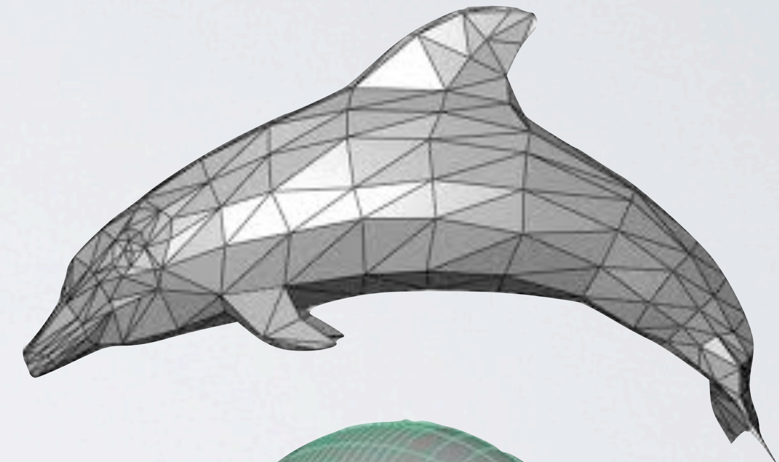
Andreas Bærentzen

Point Set | Piecewise linear | Parametric

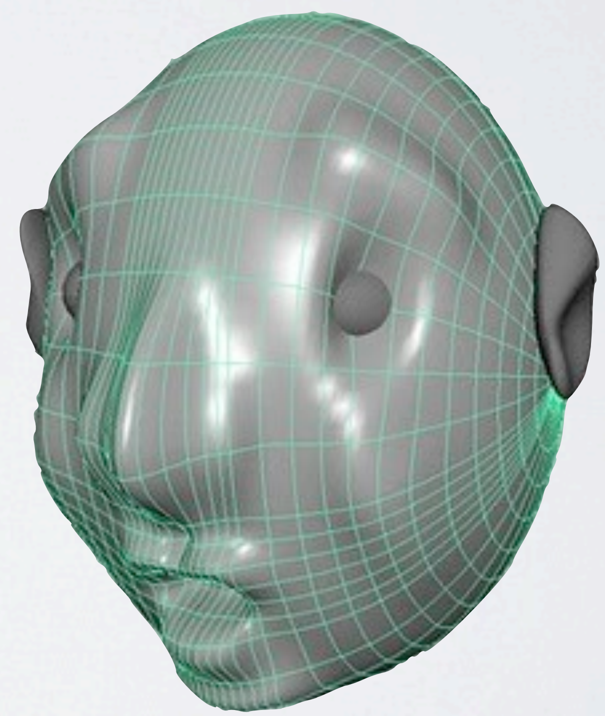


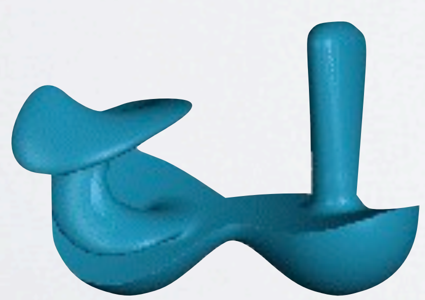
Simplicial Complex | Level Set | Skeleton generous

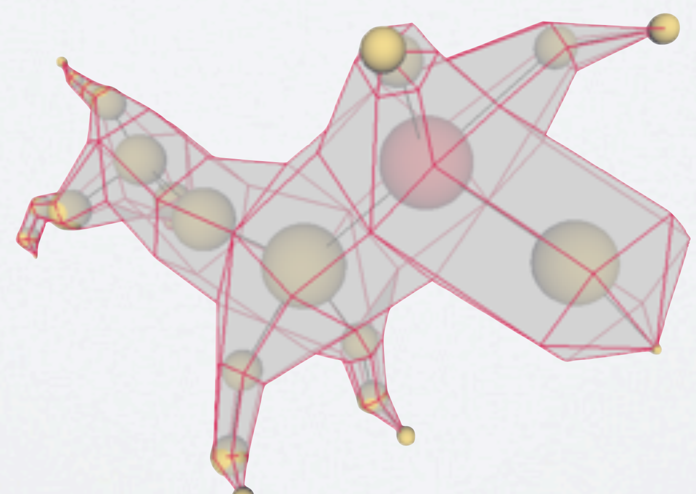




say you want to add a bump ...







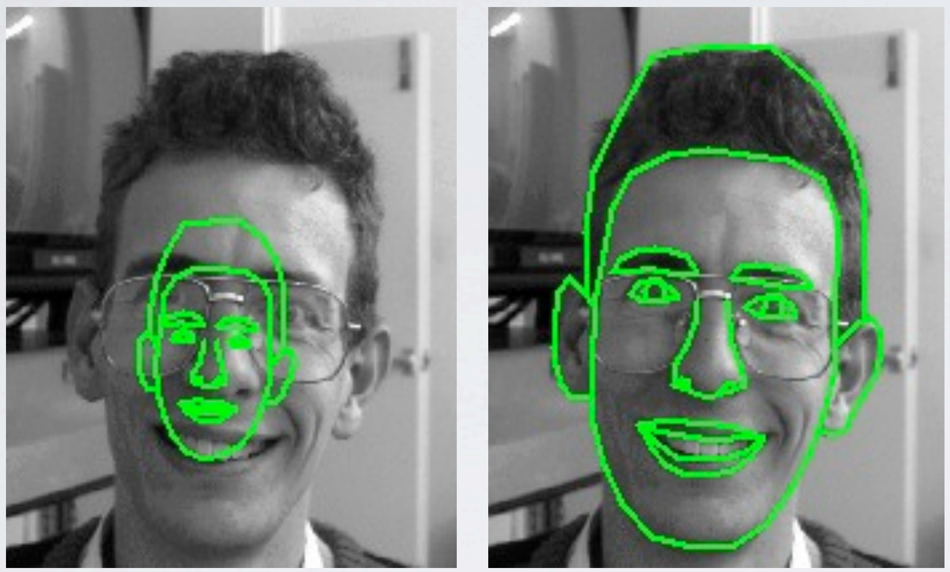
parsimonious

WHAT WAS THE POINT AGAIN?

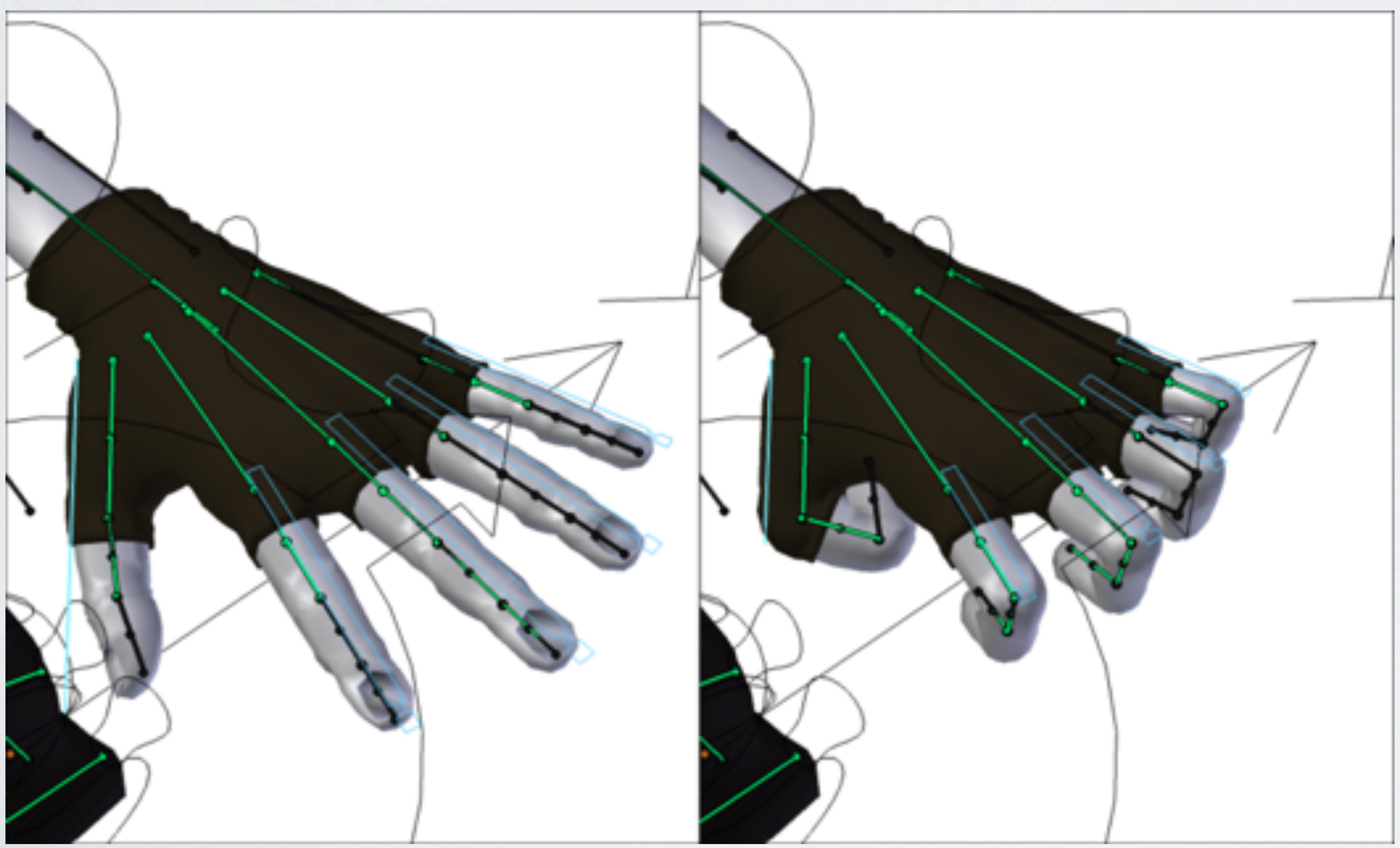
- Parsimonious shape reps let few parameters control much of the shape - very compact
- Generous shape reps have many parameters that control little
- Generous *tend* to be more amenable to arbitrary edits

TYPES OF DEFORMATION

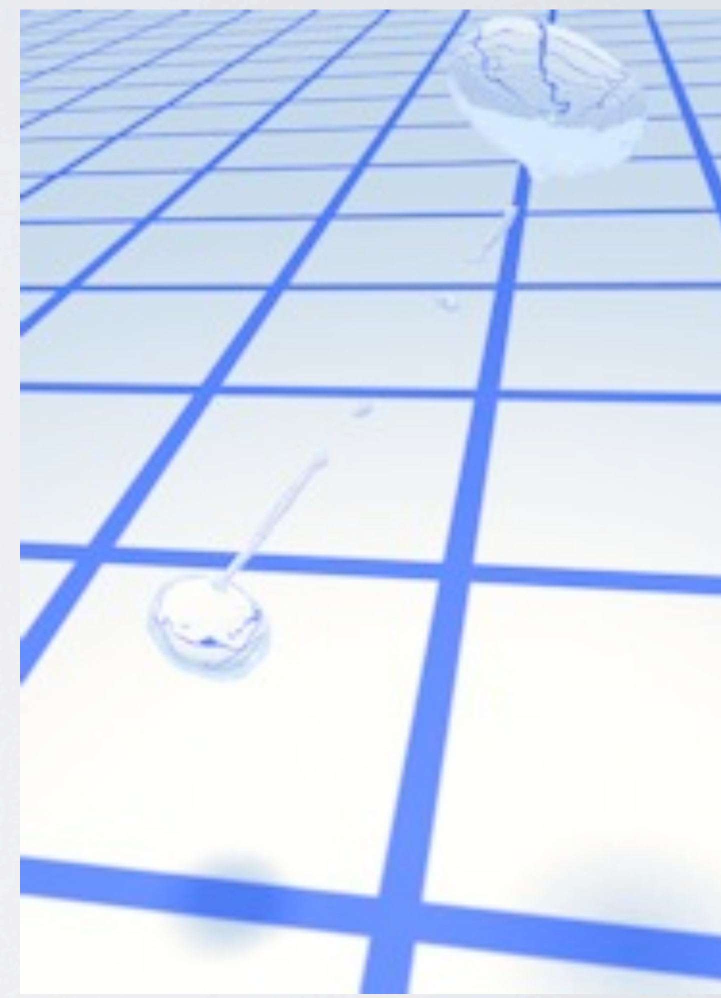
AAM
[Cootes]



SSD



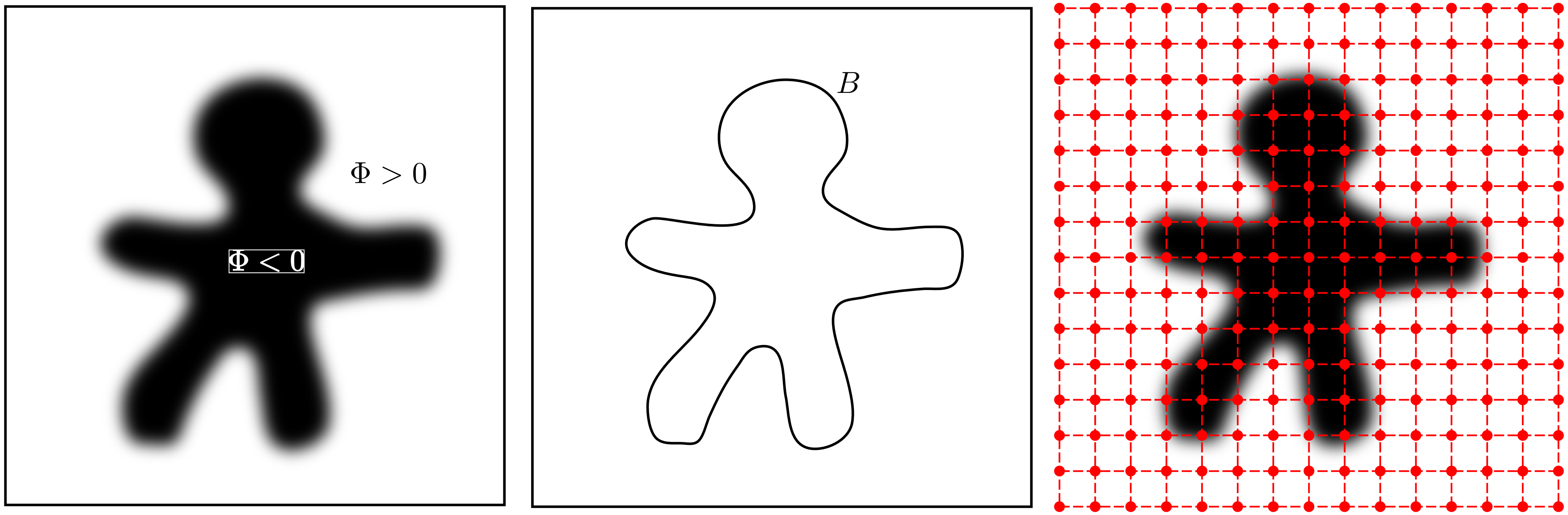
Fluid



TYPES OF DEFORMATION

- So what we learnt:
  - Parsimonious is good for deformable models if
    - the shape does not change its structure
    - fewer parameters to control.
    - But - often manual work creeps in.
  - Generous allows more arbitrary changes - in particular topology changes

The Level Set Method



- LSM is based on a distance field Φ

$$\Phi(B(t), t) = 0$$

$$\begin{aligned} d\Phi(B(t), t)/dt &= d\Phi(B^x(t), B^y(t), B^z(t), t)/dt \\ &= \frac{\partial \Phi}{\partial t} + \nabla \Phi \cdot \frac{dB}{dt} \end{aligned}$$

The Speed Function

- The interface B is moved by updating Φ
- We need a speed function F where

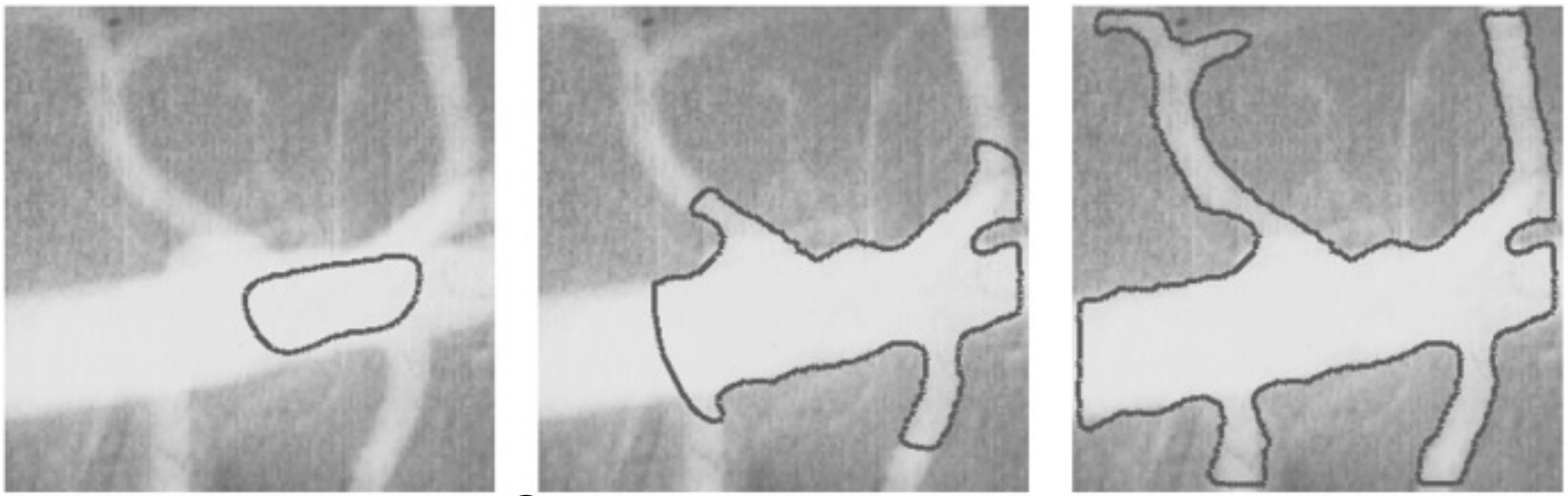
$$\frac{dB(t)}{dt} = F\frac{\nabla\Phi}{||\nabla\Phi||}$$

- Plugging back in: $$\frac{\partial\Phi}{\partial t} + F||\nabla\Phi|| = 0$$

Speed Functions *F*

- F is constant
- F = mean curvature
- F depends on data
- Combinations of the above

The Level Set Method

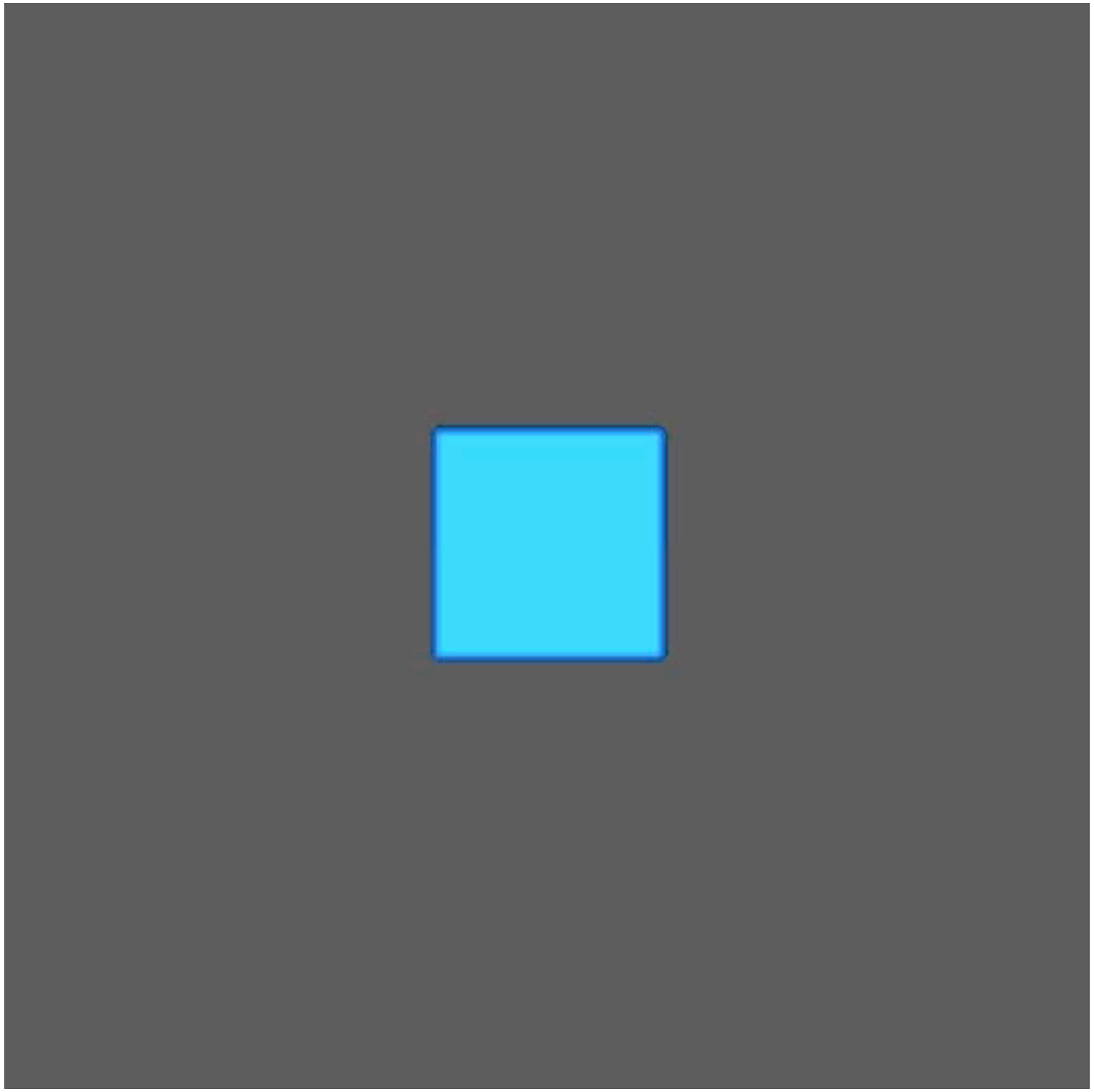


Segmentation

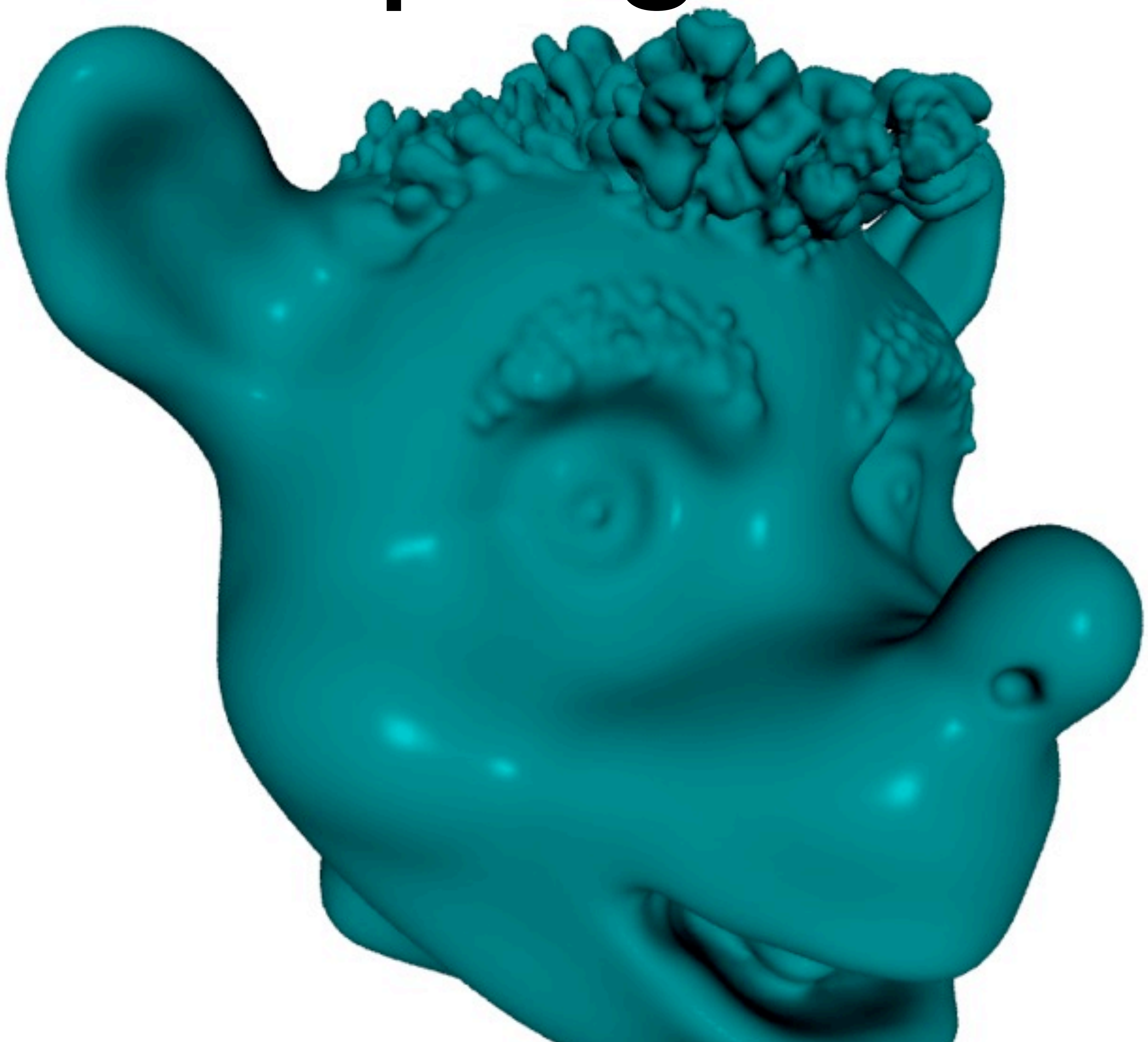
- For many applications we need deformable surfaces.
- In some of these, the surface must be able to change topology.

Sculpting

3D Sculpting with LSM



3D Sculpting with LSM



Implementation

- The LSM is a simple loop over all voxels where we update Φ
- One issue with the LSM is that Φ does not remain a distance field.
- Thus, we need to reinitialize every few iterations

Narrow Banding

- Problem: Grid is n^3 voxels but we are only interested in a surface occupying roughly

  k n^2 voxels

- Solution: Perform computations only near surface.

- Often the fast marching method (complex) is used to initialize, but reinitialization (next slide) is simpler.

The Reinitialization Equation

$$\Phi_t + s(\Phi_0)(\|\nabla\Phi\| - 1) = 0$$

where $$s(\Phi_0) = \frac{\Phi_0}{\sqrt{{\Phi_0}^2 + \epsilon^2}}$$

and $\nabla\Phi$ is computed in the upwind direction, i.e. one sided differences in direction of 0-level set.

Discrete version:

$\Phi^n = \Phi^{n-1}$+dt*s(Φ^0)*(1-len(grad(Φ^{n-1})))

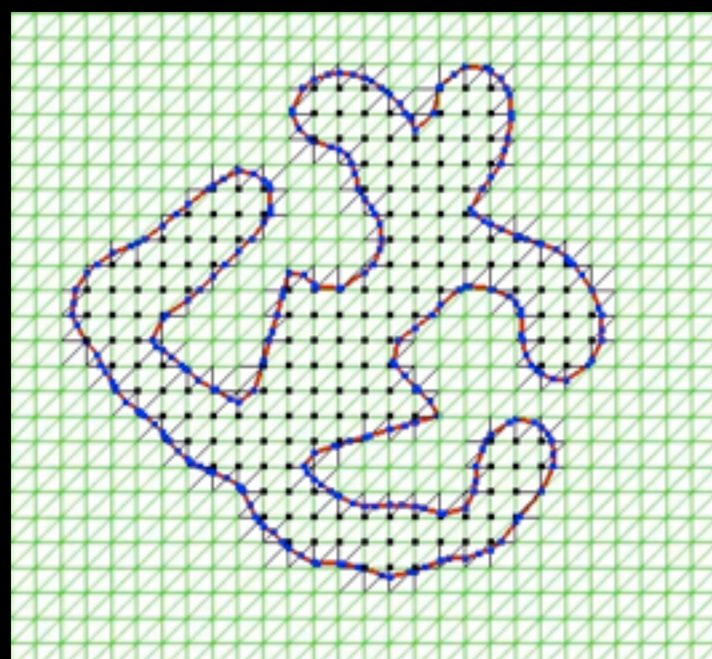
Example: Curvature dependent speed

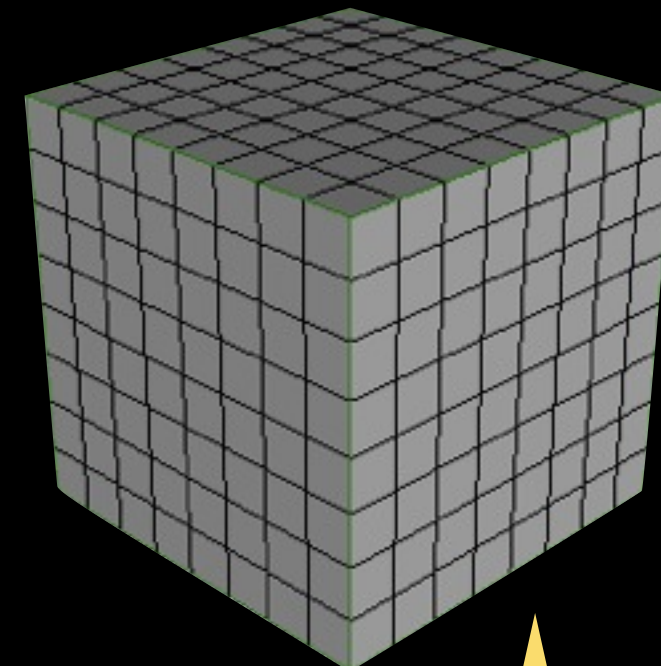
- We often use F = -*H*
- *H* is mean curvature
- $$\begin{aligned} H &= \nabla \cdot \nabla \Phi \\ &= \Delta \Phi \\ &= \Phi_{xx} + \Phi_{yy} + \Phi_{zz} \end{aligned}$$

Level Set Method Curvature Flow

Top 3 Problems with the LSM

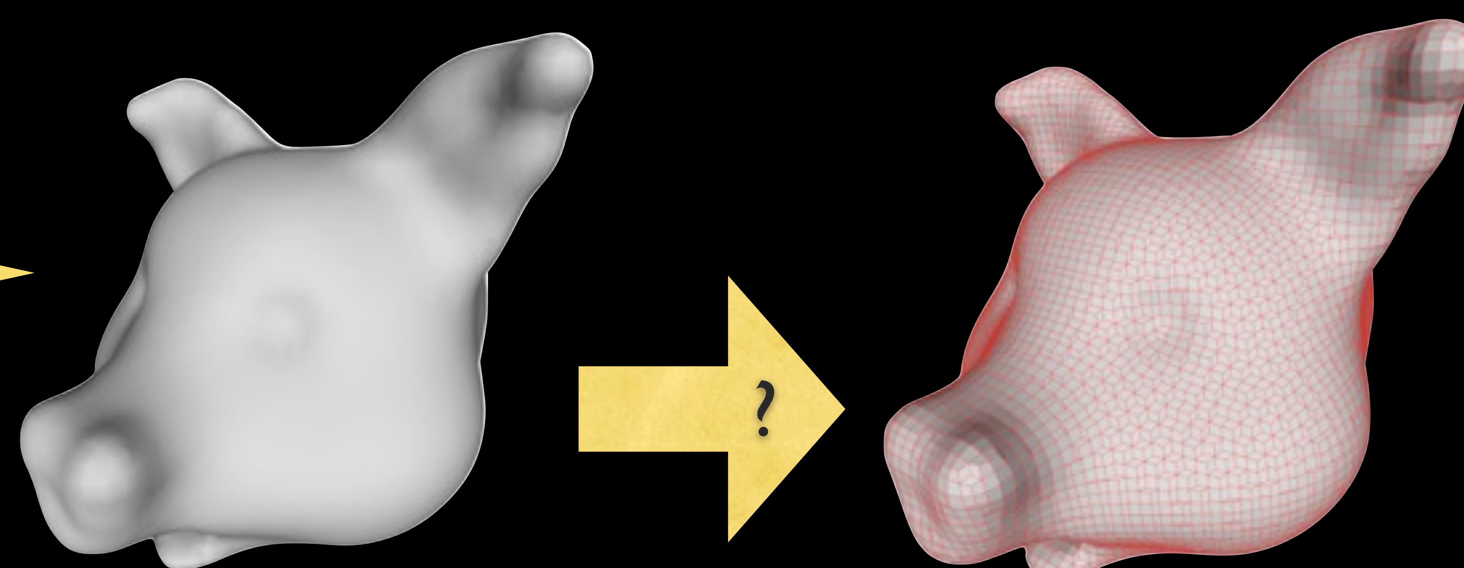
The LSM suffers from numerical diffusion





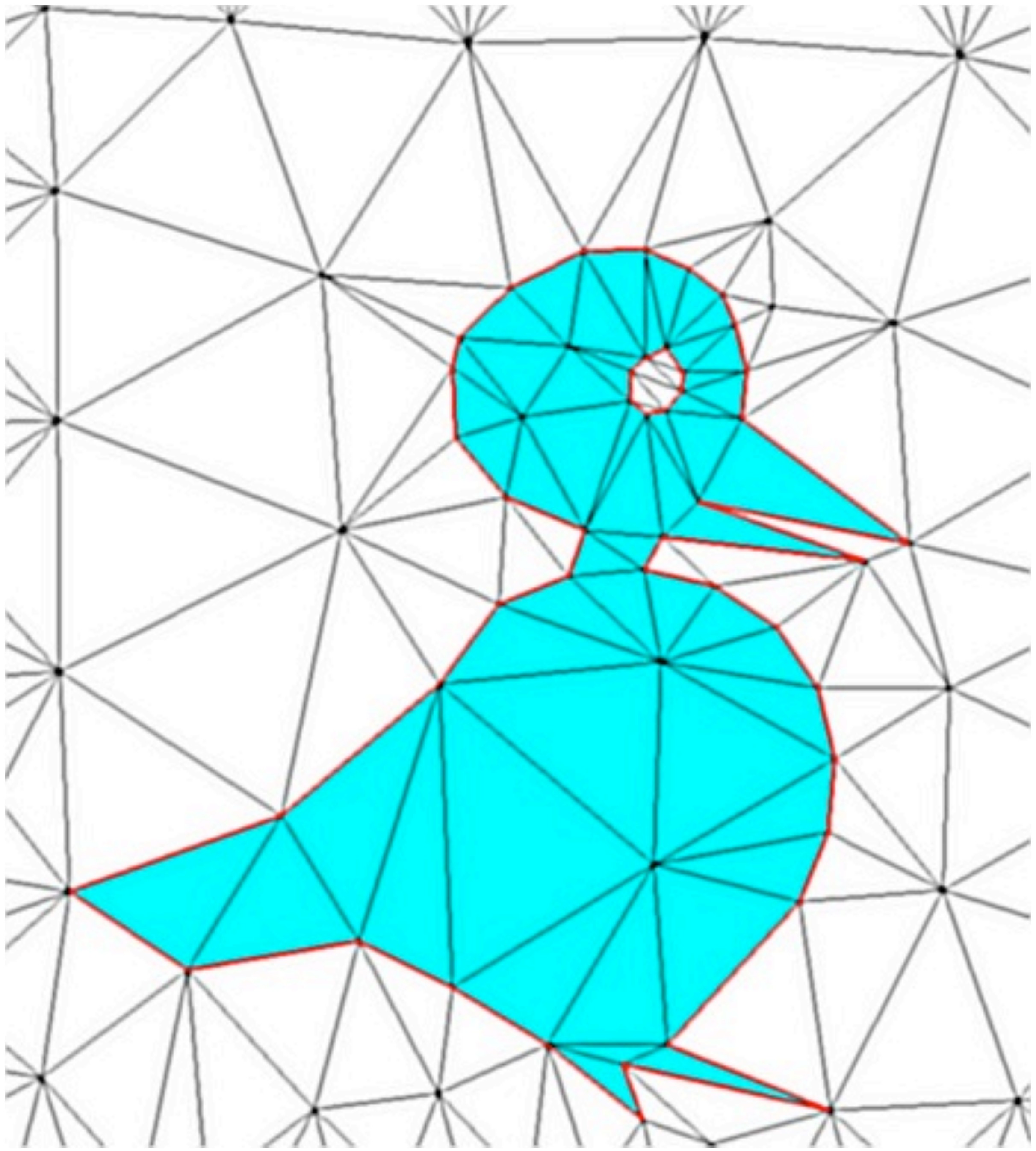
The LSM is bound to a scale interval dictated by the grid resolution

The LSM has no explicit interface representation



Deformable Simplicial Complexes

- Space is partitioned into triangles:
  - *interior* (cyan)
  - *exterior (white)*
- *Interface* is the set of segments dividing interior from exterior.
- Vertices on interface are moved.
- Triangle complex updated to accomodate

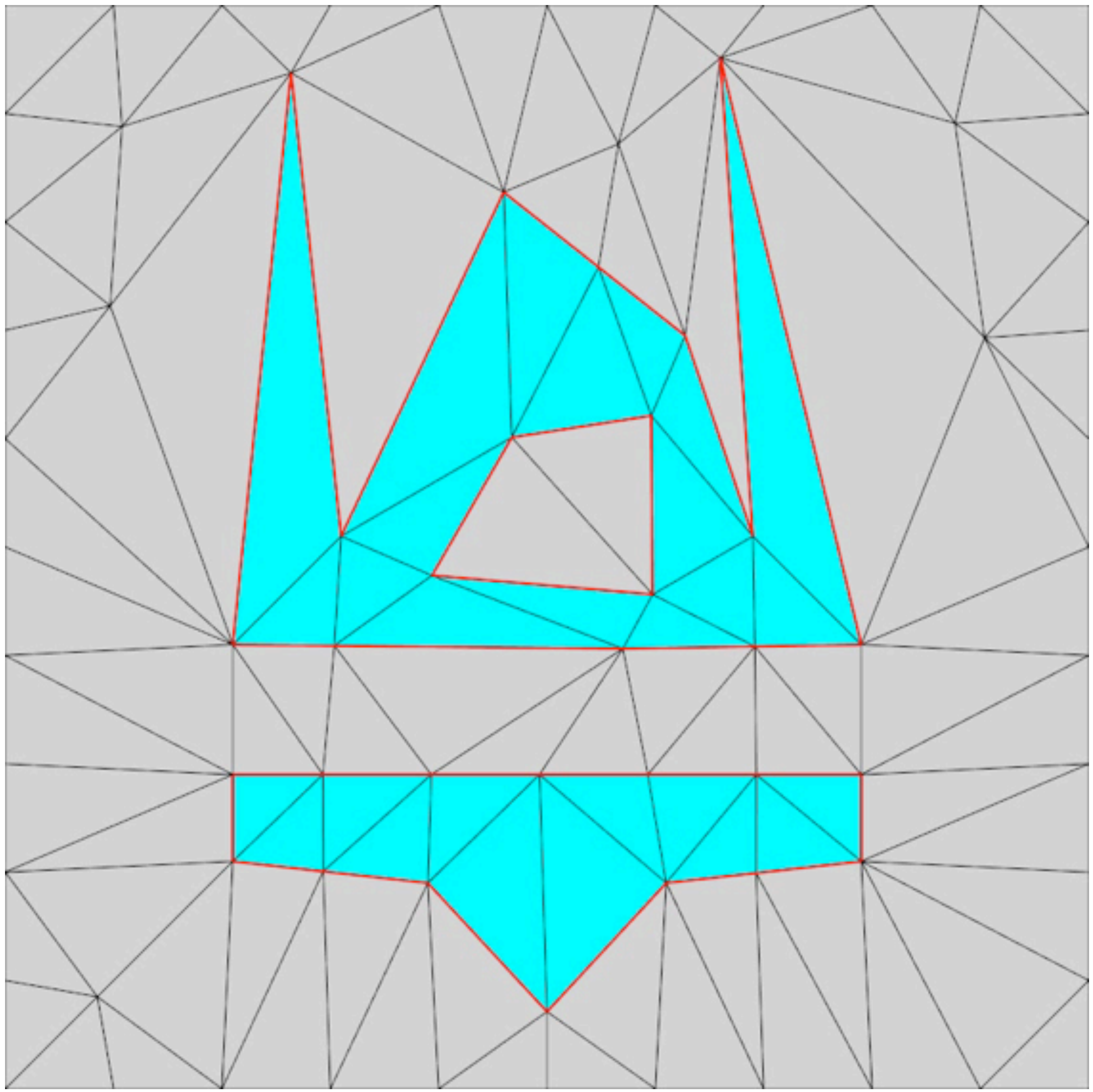


Advantages of Deformable Simplicial Complexes

- Inherently scale adaptive (due to irregular grid)
- Little diffusion
- No *gratuitous* re-parameterization
- Topologically adaptive
- Topology *control* is also possible
- Extensible to 3D (I hope :-)
  - Triangle → tetrahedron
  - Segment → triangle

DEMO

Constrained volume optimization



2D Implementation

1) [Move] each vertex to target or as far as possible without mesh degeneracies
2) [Smooth] Steiner vertices
3) [Thin] out Steiner vertices
4) [Flip caps] (allow interface topology to change)
5) [Split] needles (adapt to details)
6) Make [Delaunay] by flipping
7) [Remove degenerate] faces
8) Unless all vertices are at target goto 1

2D Implementation - explained

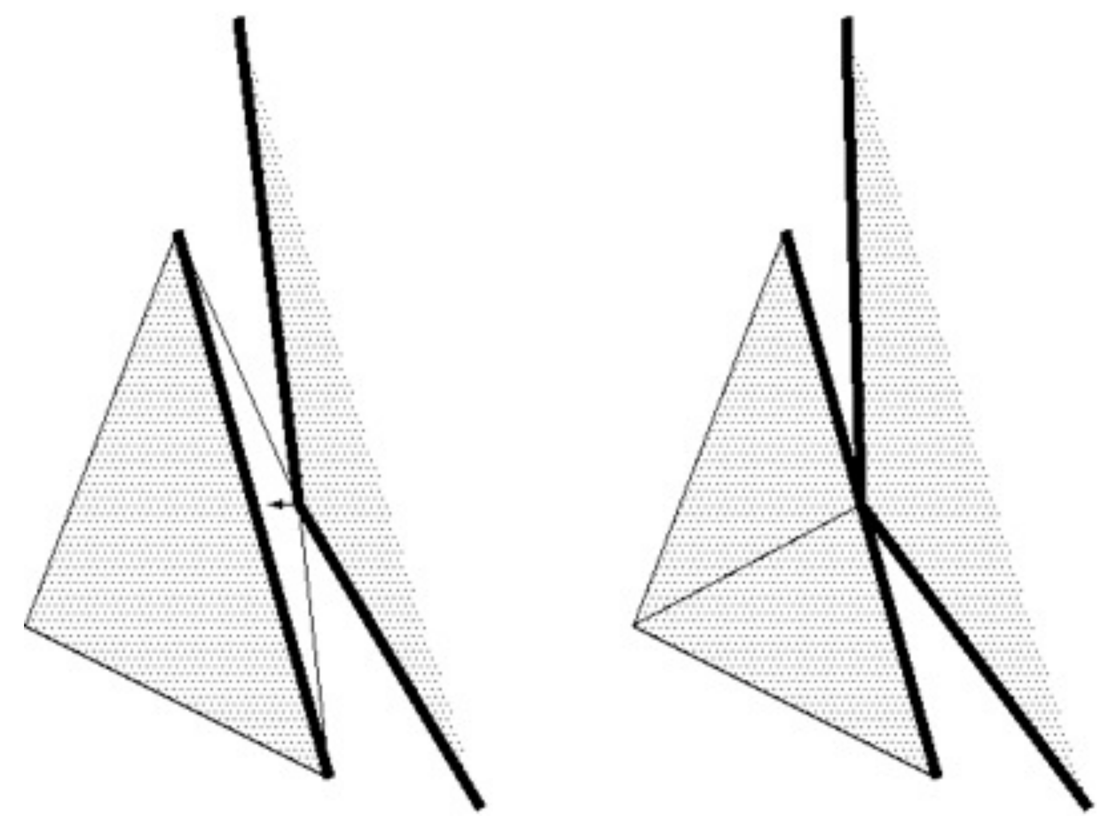
1) [Move] This step simply moves vertices towards the target as defined by some speed function

2) [Smooth] We need to avoid degenerate triangles in the mesh. Smoothing is very effective.

3) [Thin] We also need to reduce the amount of geometry we need to compute on. Thinning unneeded Steiner vertices is a great help.

2D Implementation - explained

4) [Flip Caps] This step handles the case where two interface (parts) collide. The flipping basically joins them at the point where a vertex has hit a interface edge.



5) [Split] introduces more details where needed.

6) [Delaunay] Keep the mesh nice by retriangulating.

7) [Remove degenerate] Remove triangles that are degenerate (e.g. became line segments or points).

2D Implementation - notes

- Robustness – not perfect but very good.
- All steps are needed, some may be reordered
- Most steps seem simple to implement in 3D except Delaunay-fication
- The described algorithm is essentially one time step of the evolution of the interface (surface).

Discussion

- The volume representation is very “generous”: Many voxels, each controls little.
- DSC is a variation where the grid is irregular and the representation is binary.
- 3D DSC was much harder than 2D DSC (which was not so easy)
- Sculpting will be done, but if you just want deformable surfaces you could use other methods.
- Real strength is when you care about tets.