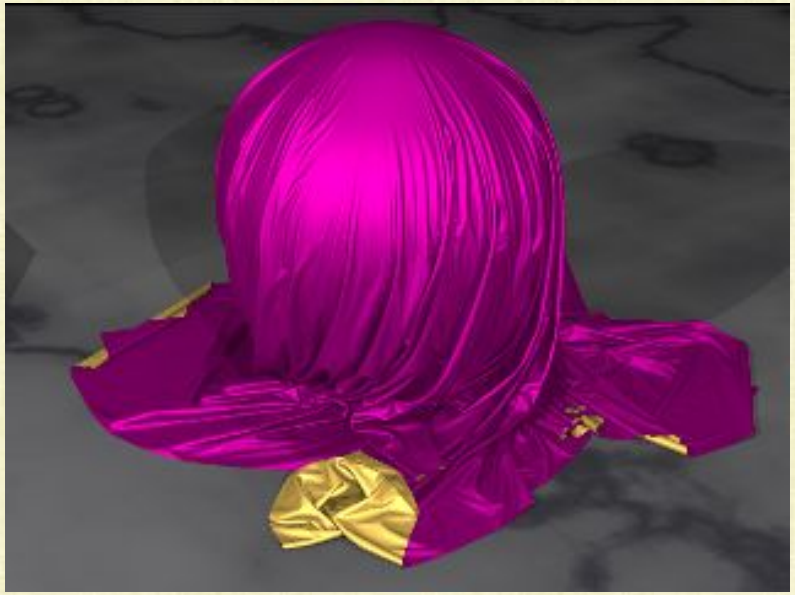
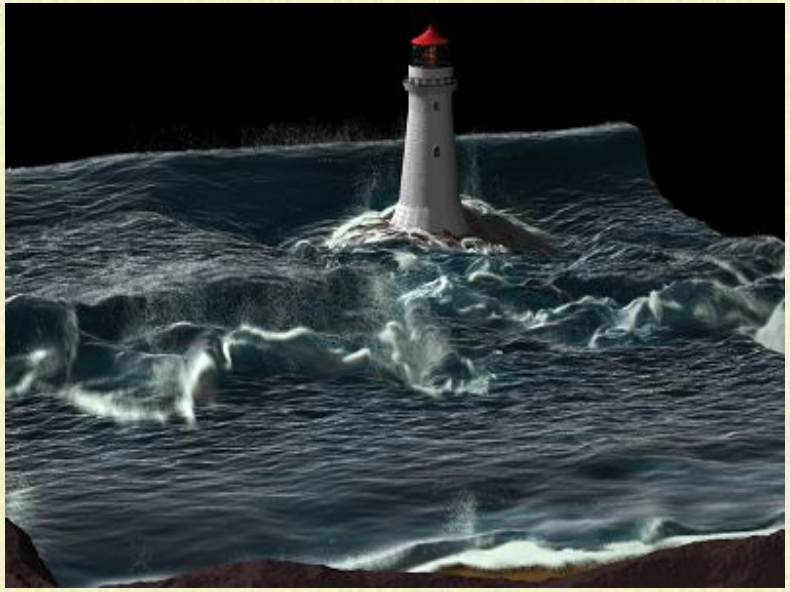
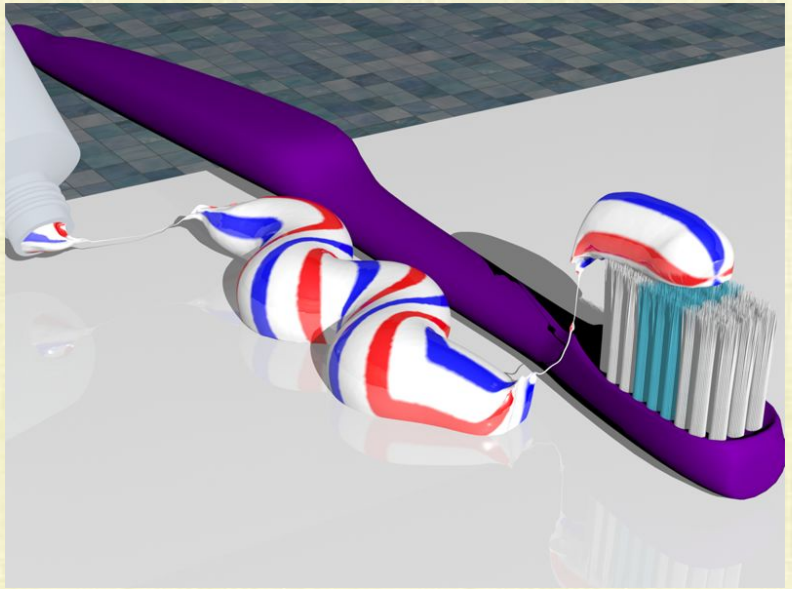
More Geometric Modeling



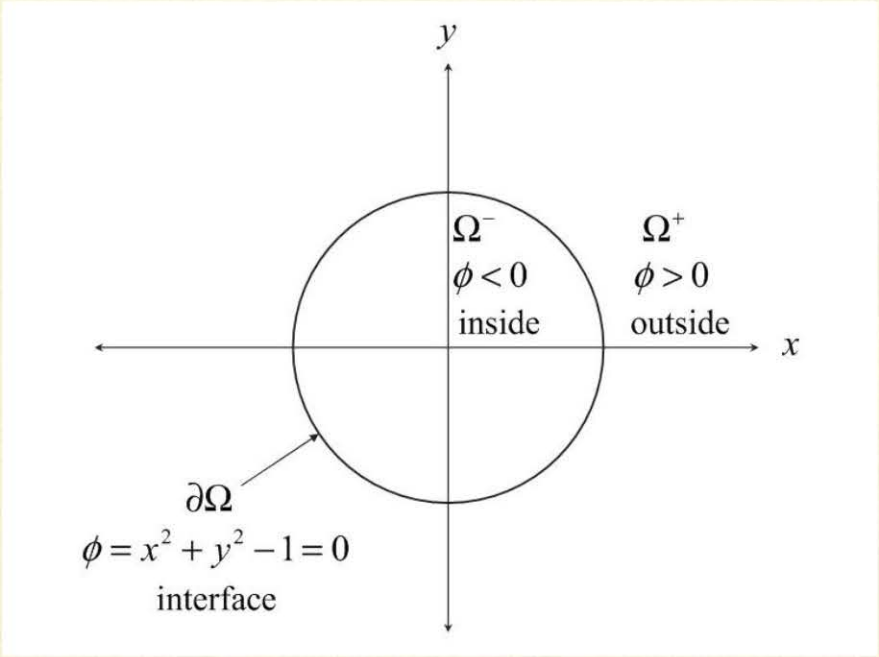


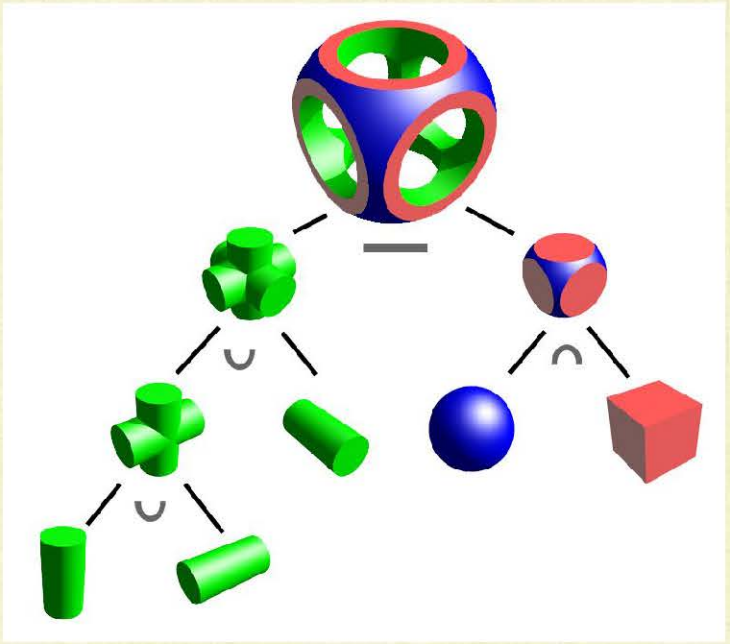


Implicit Surfaces

Implicit Surfaces

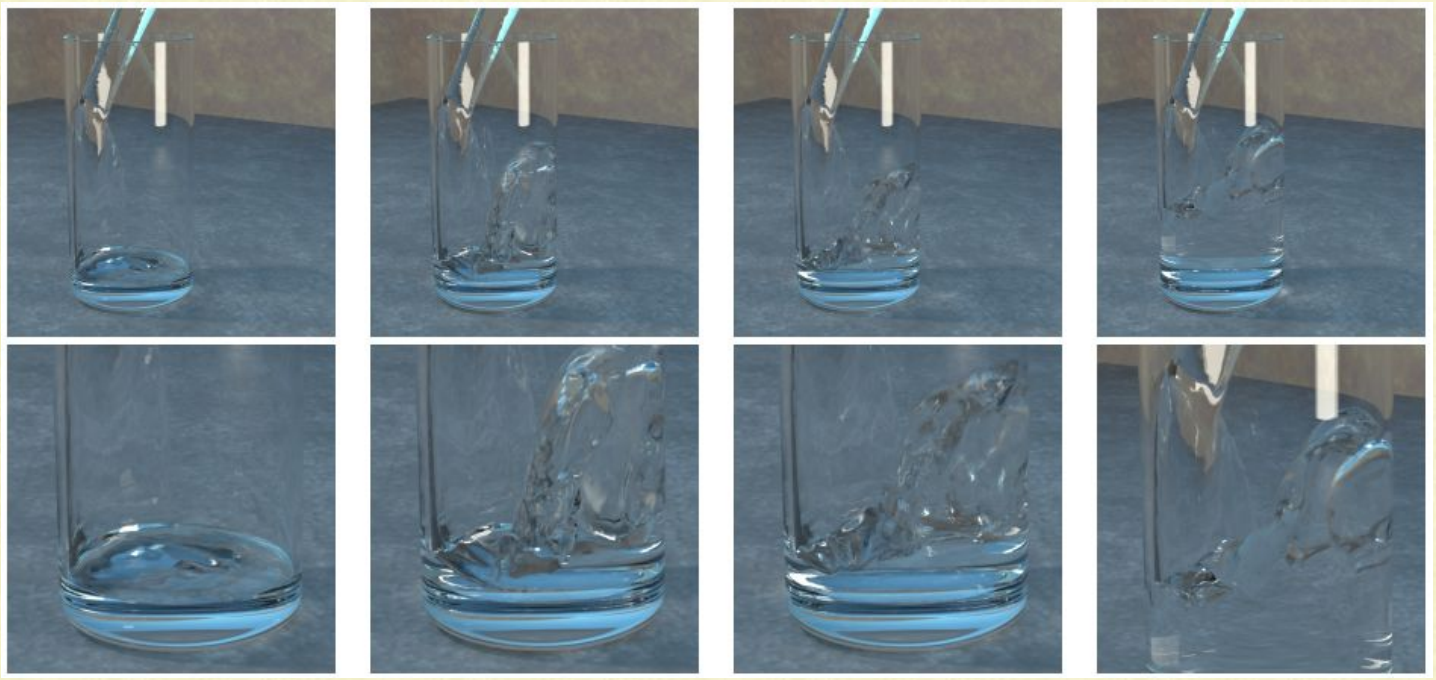
- Define a function $\phi(x)$ over $\forall x \in R^3$
- Inside region Ω^- defined by $\phi(x) < 0$, outside region Ω^+ defined by $\phi(x) > 0$
- The surface is defined as $\partial\Omega$ where $\phi(x) = 0$
  - We have already seen planes/spheres (and lines/rays/circles in 2D) defined by implicit surfaces
- Easy to check if a point x is inside/outside: just evaluate $\phi(x)$
- Constructive Solid Geometry (CSG) operations: union, difference, intersection, etc.
- Easy to ray trace implicit objects

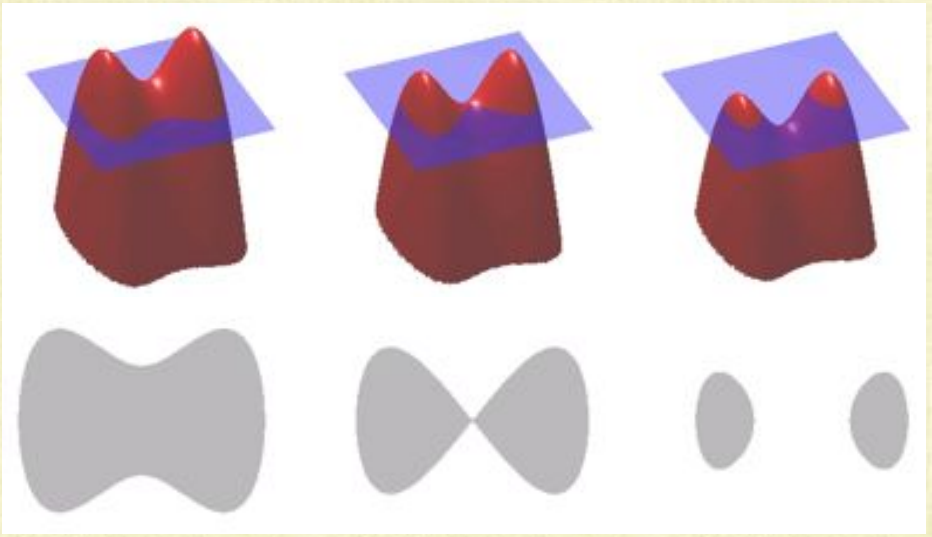


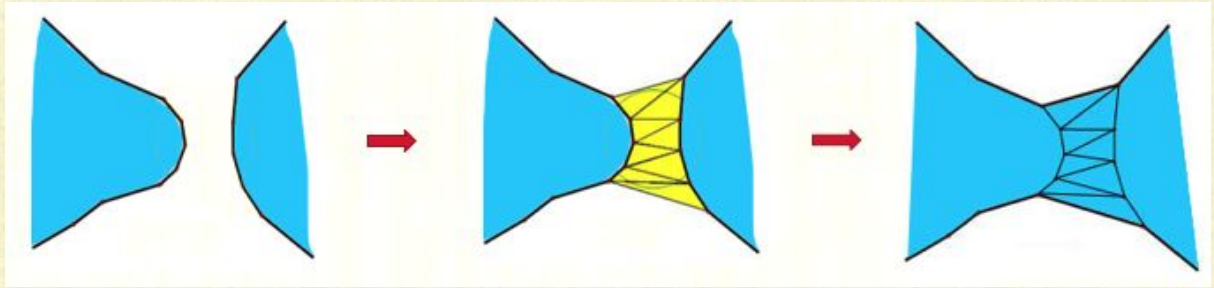


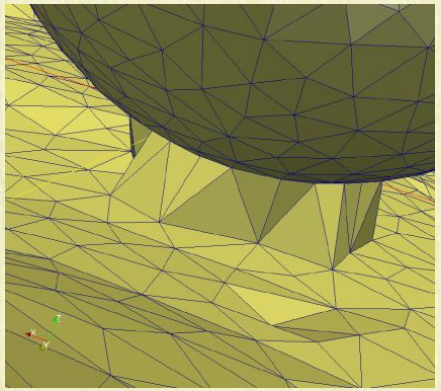
Topological Change

- Greatly superior to triangle meshes for topological change!



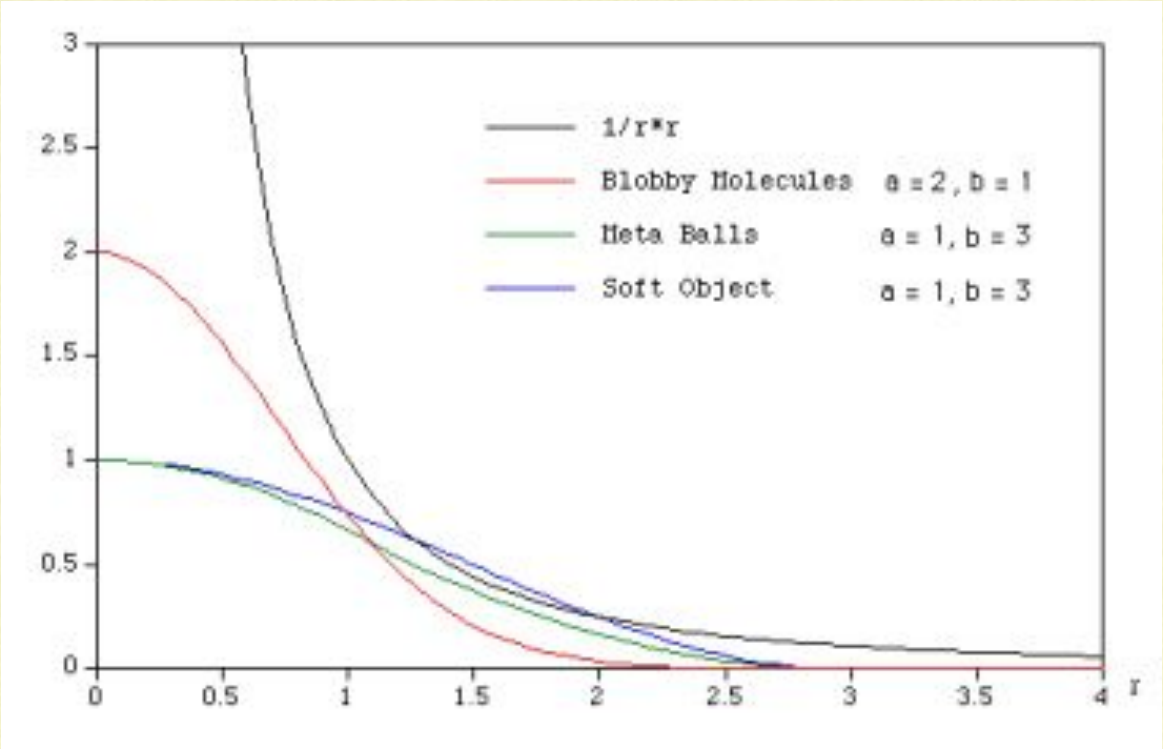




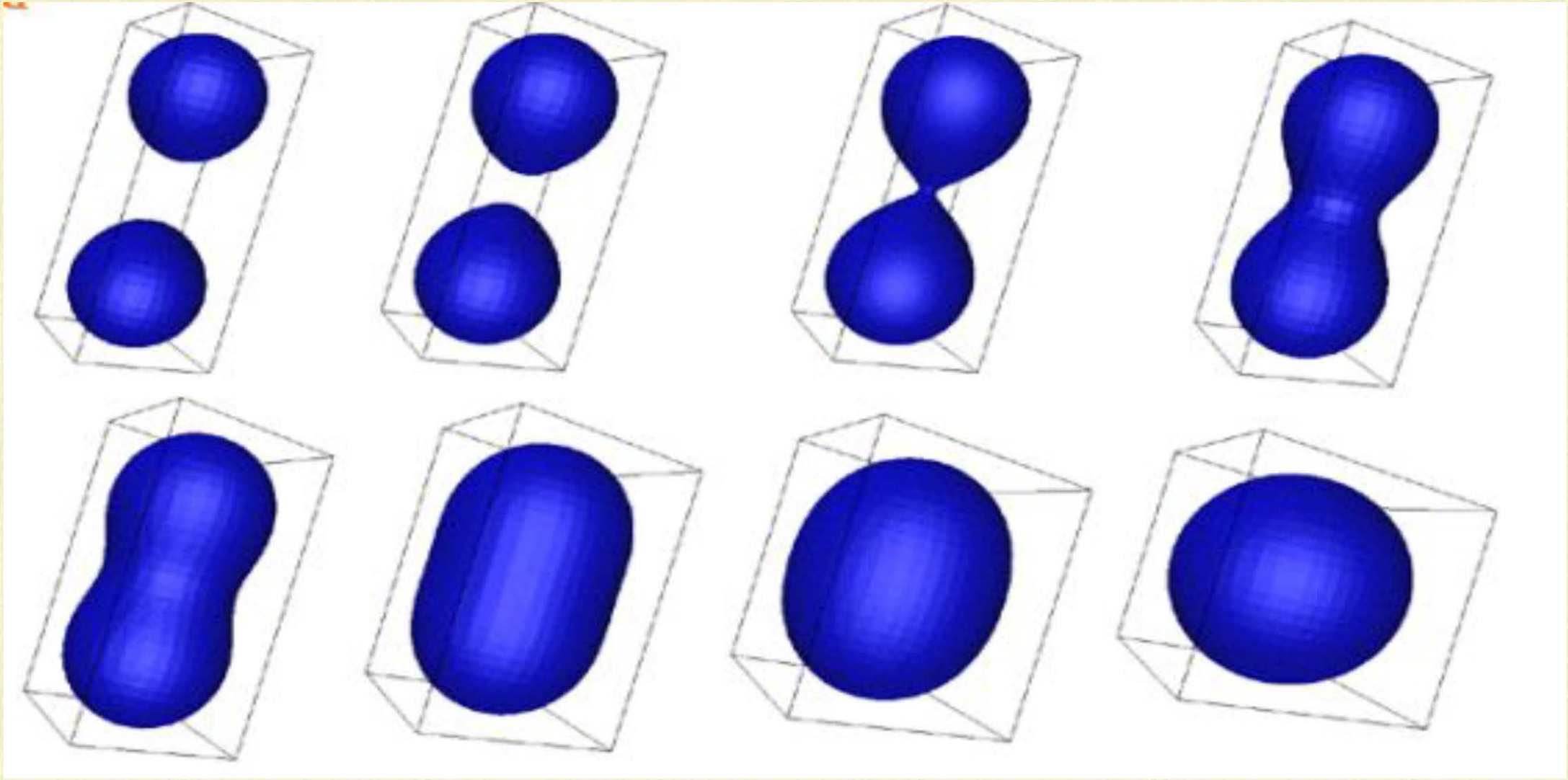


Blobbies

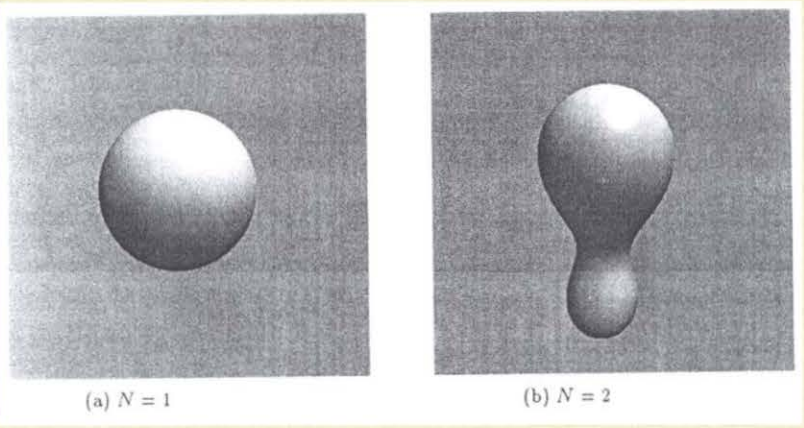
- Each blob is defined as a density function around a particle
- Blob kernels can be: 2D ellipses, 2D diamonds, 3D spheres, etc.
- For each pixel, the aggregate density is summed from all overlapping blobs
- Also known as:
  - Metaballs (in Japan), Soft objects (in Canada and New Zealand)
  - Slightly different density kernel functions



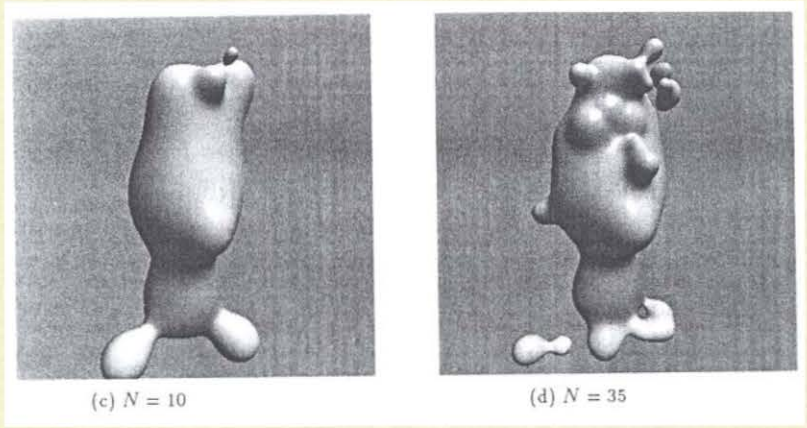
Topological Change



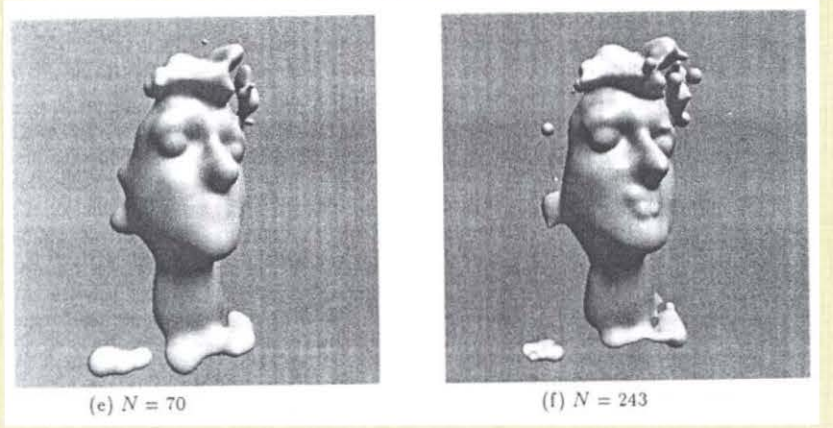
Blobby Modeling



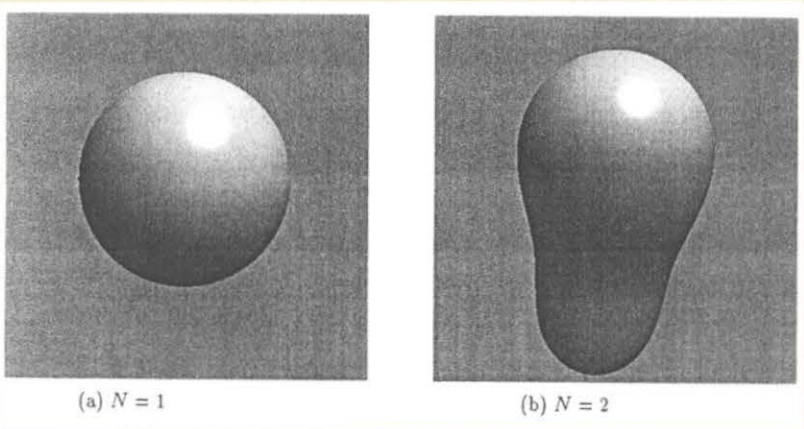
(a) $N = 1$ (b) $N = 2$



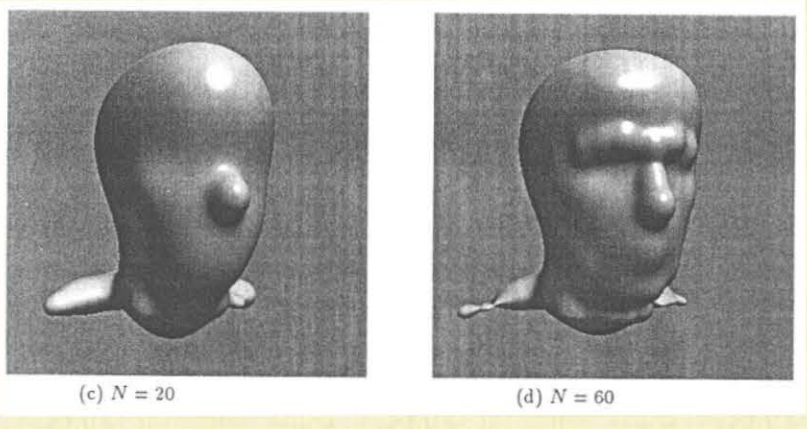
(c) $N = 10$ (d) $N = 35$



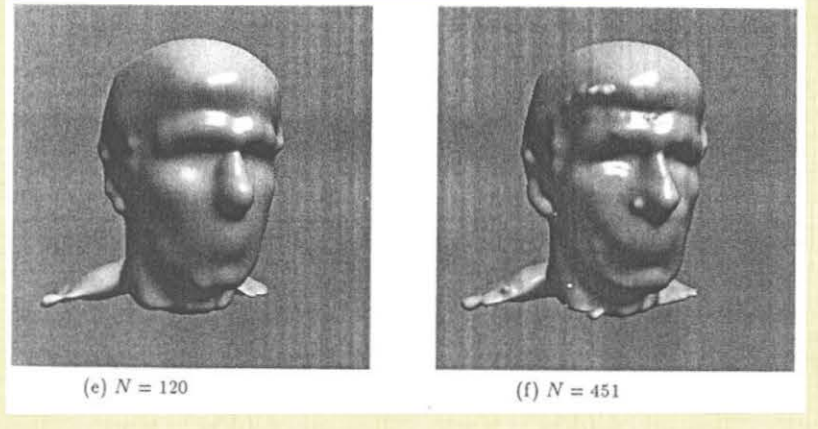
(e) $N = 70$ (f) $N = 243$



(a) $N = 1$ (b) $N = 2$



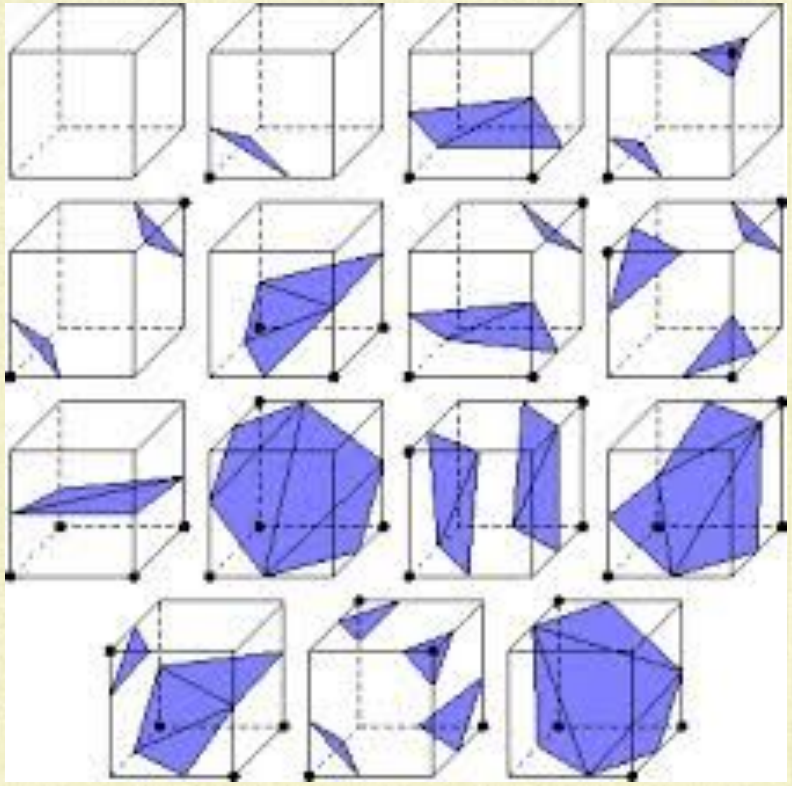
(c) $N = 20$ (d) $N = 60$

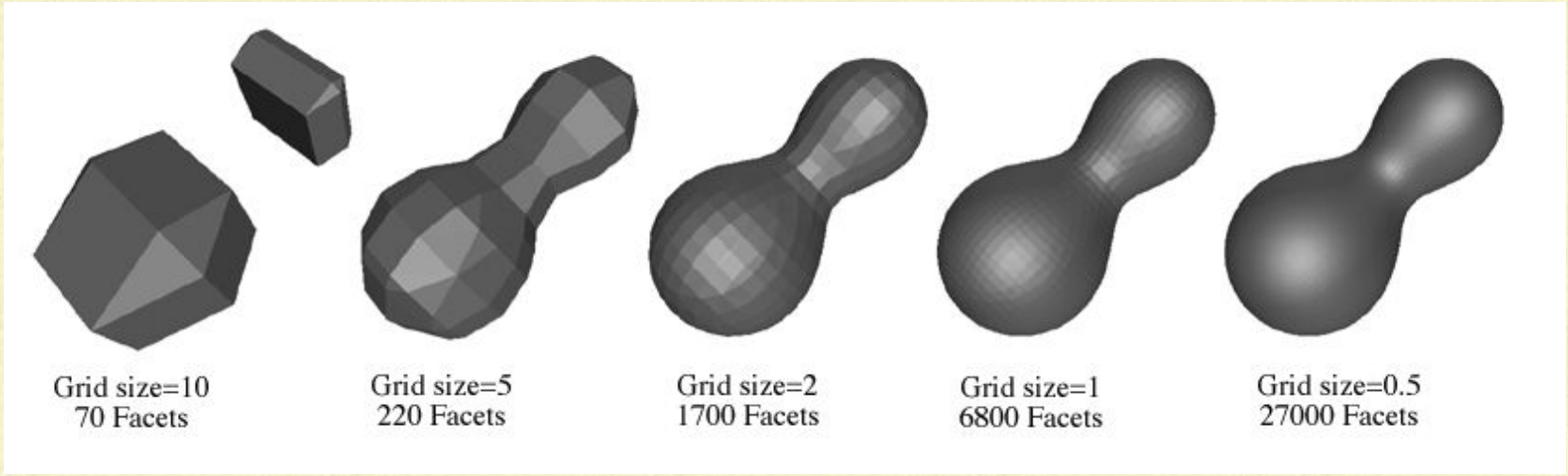


(e) $N = 120$ (f) $N = 451$

Marching Cubes (or Tetrahedra)

- Turns an implicit surface into triangles
  - Define the implicit surface on a 3D grid
  - For each grid cell, use the topology of the volume to construct surface triangles

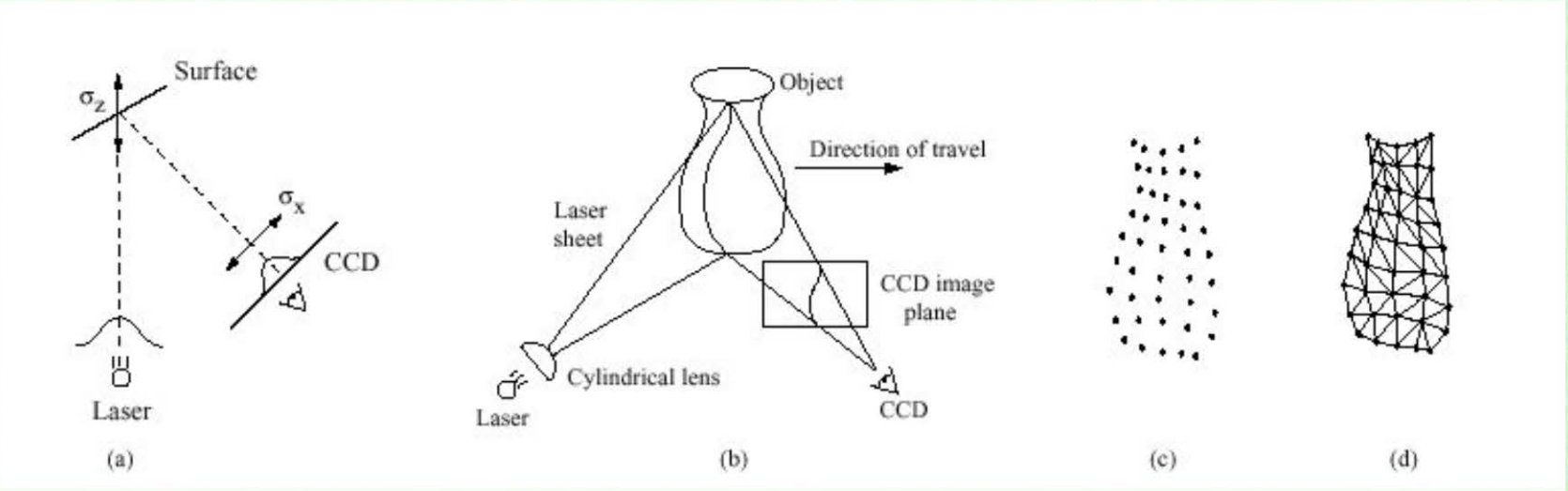


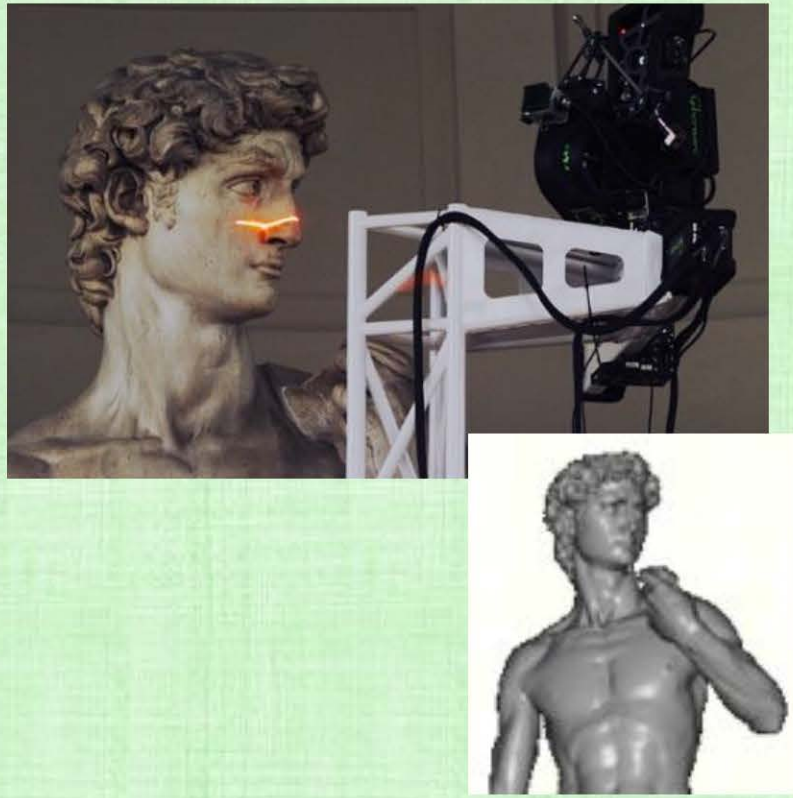


Computer Vision

Range Scanners

- Senses 3D positions on an object's surface, and returns a range image:
  - $m \times n$ grid of distances (m points per laser sheet, n laser sheets)
- Multiple range images are aligned with transformations
  - Transformations determined via Iterative Closest Point (ICP) and related/similar methods
- Aligned range images are combined using a zippering algorithm





Range Scanners

- Each sample point in the $m \times n$ range image is a potential vertex in a triangle mesh
- Special care is required to avoid joining together vertices separated by depth discontinuities

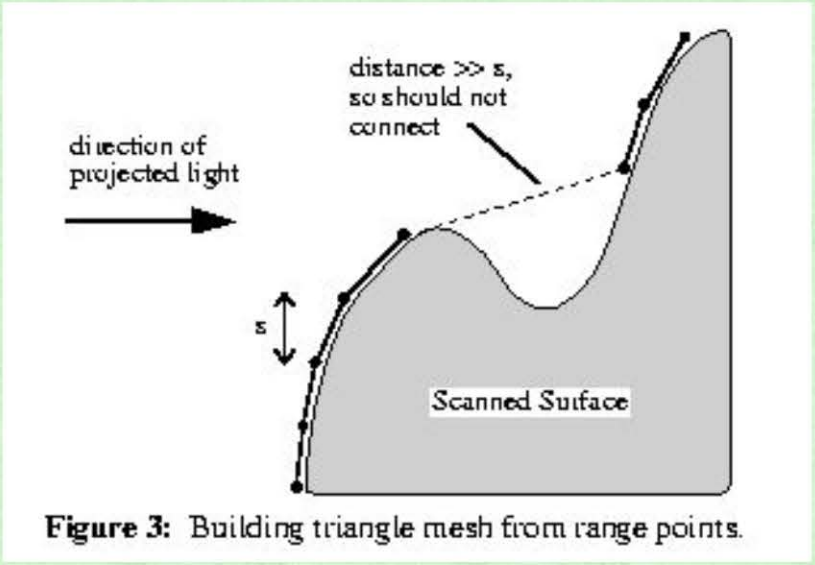
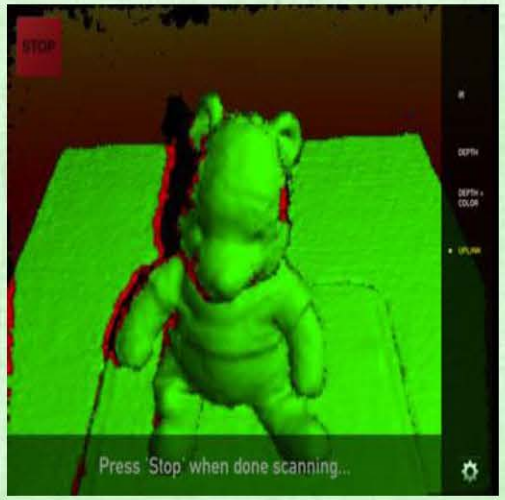


Figure 3: Building triangle mesh from range points.

Scanning w/Mobile Devices



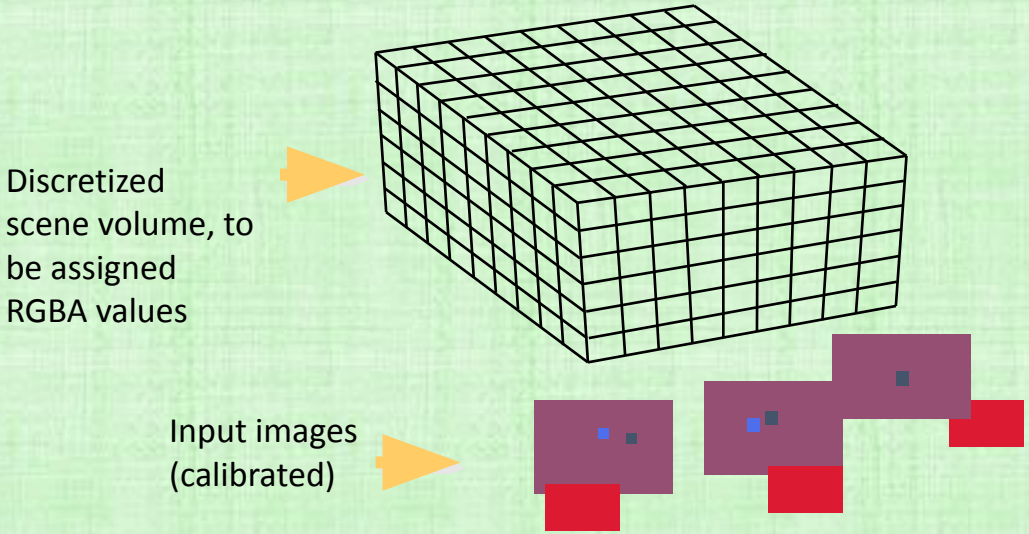
Structure Sensor for iPad

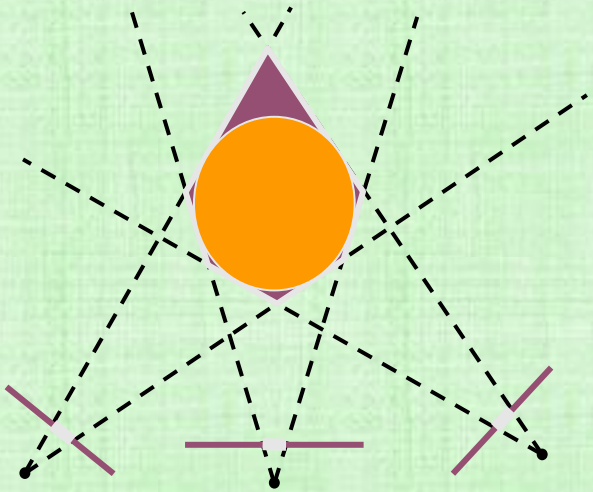
AUTODESK® 123D®

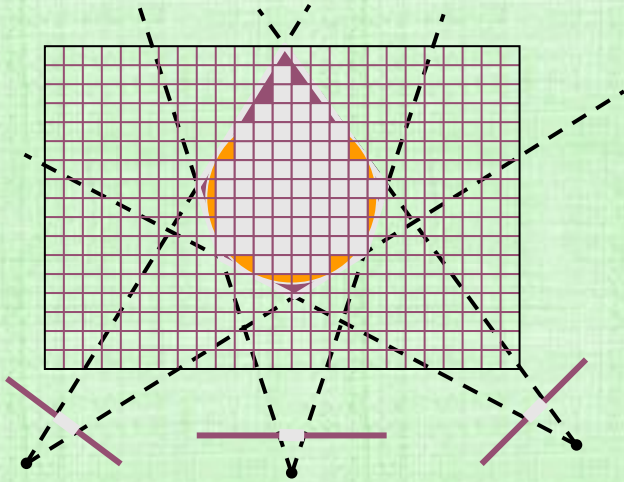
Autodesk 123D Catch

Voxel Carving

- Requires multiple images (from calibrated cameras) taken from different directions
- Construct a voxelized 3D model:
  - A silhouette is computed for each image
  - The silhouette (for each image) is back projected onto the grid
  - The voxels outside the back-projection of an image are carved away
- Colors are also back projected



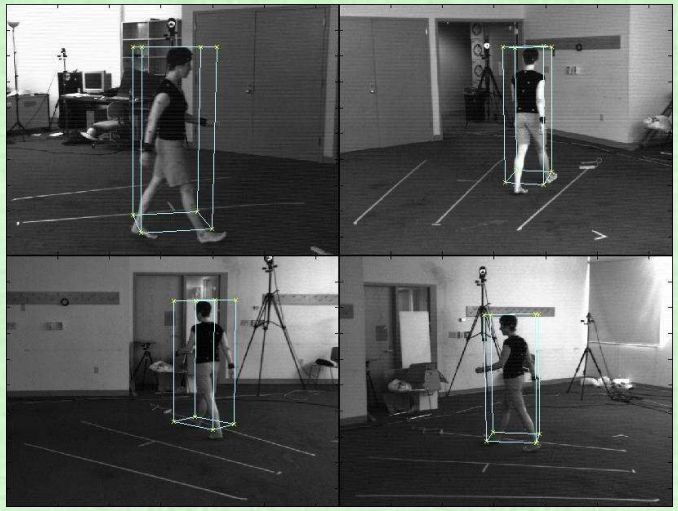




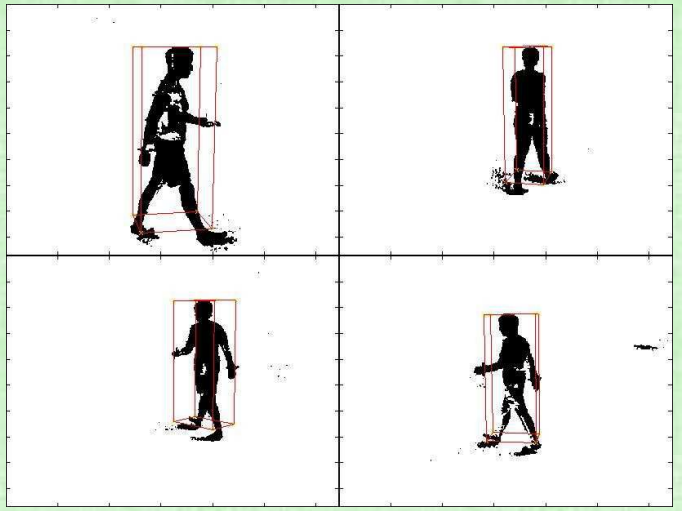
Color the voxel gray if silhouette is in every image

Voxel Carving

Original image

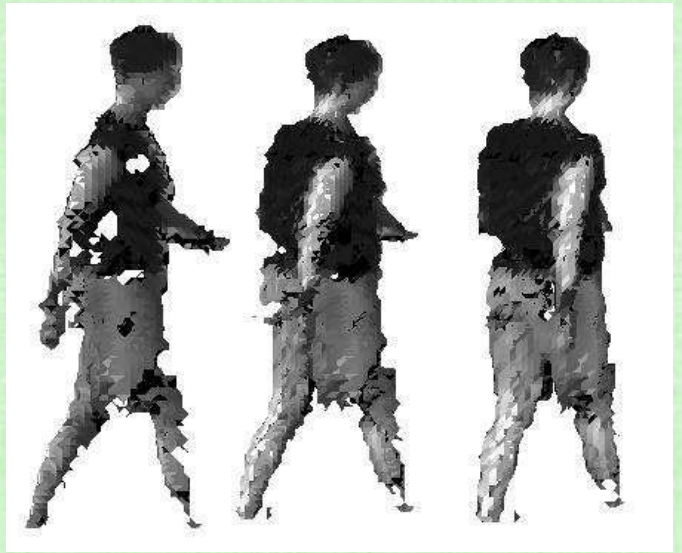


Extracted silhouettes



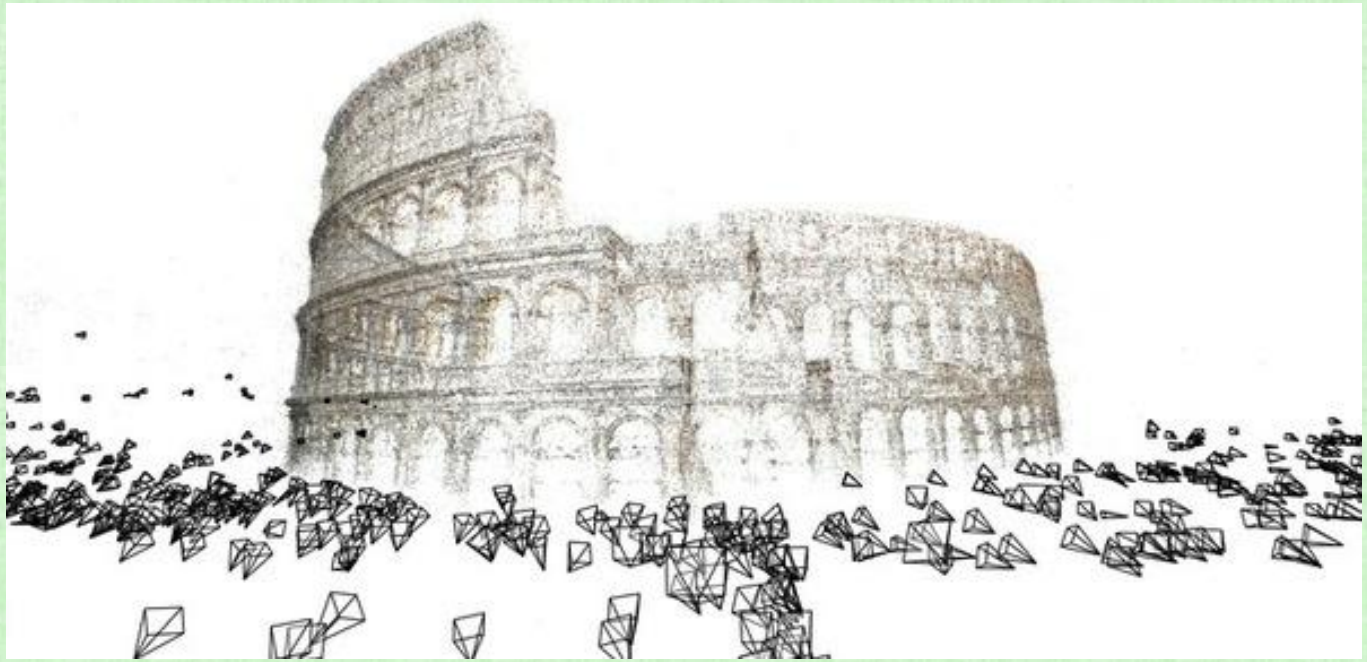
Carved out voxels

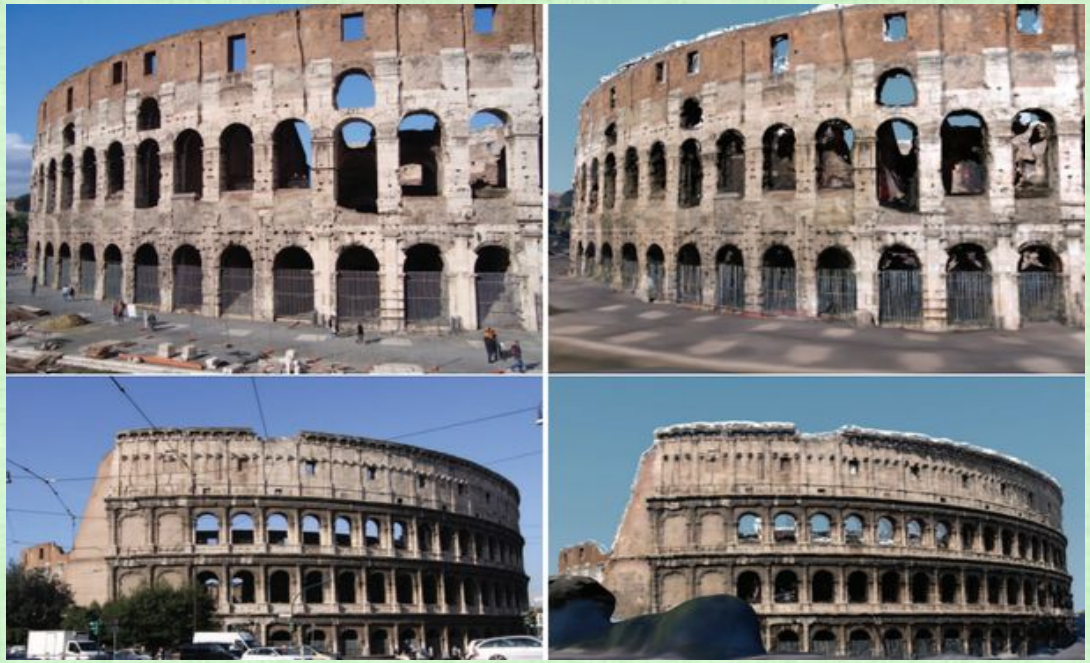
Back-projecting colors



Reconstruction from Large Photo Collections

- Collect a large number of photos (e.g. from google images)
- Predict relative camera position/orientation for each image
- The position of a point that is visible in multiple images can be triangulated
- Obtain a sparse point cloud representation of the object
- Dense reconstruction algorithms can be used to improve the results

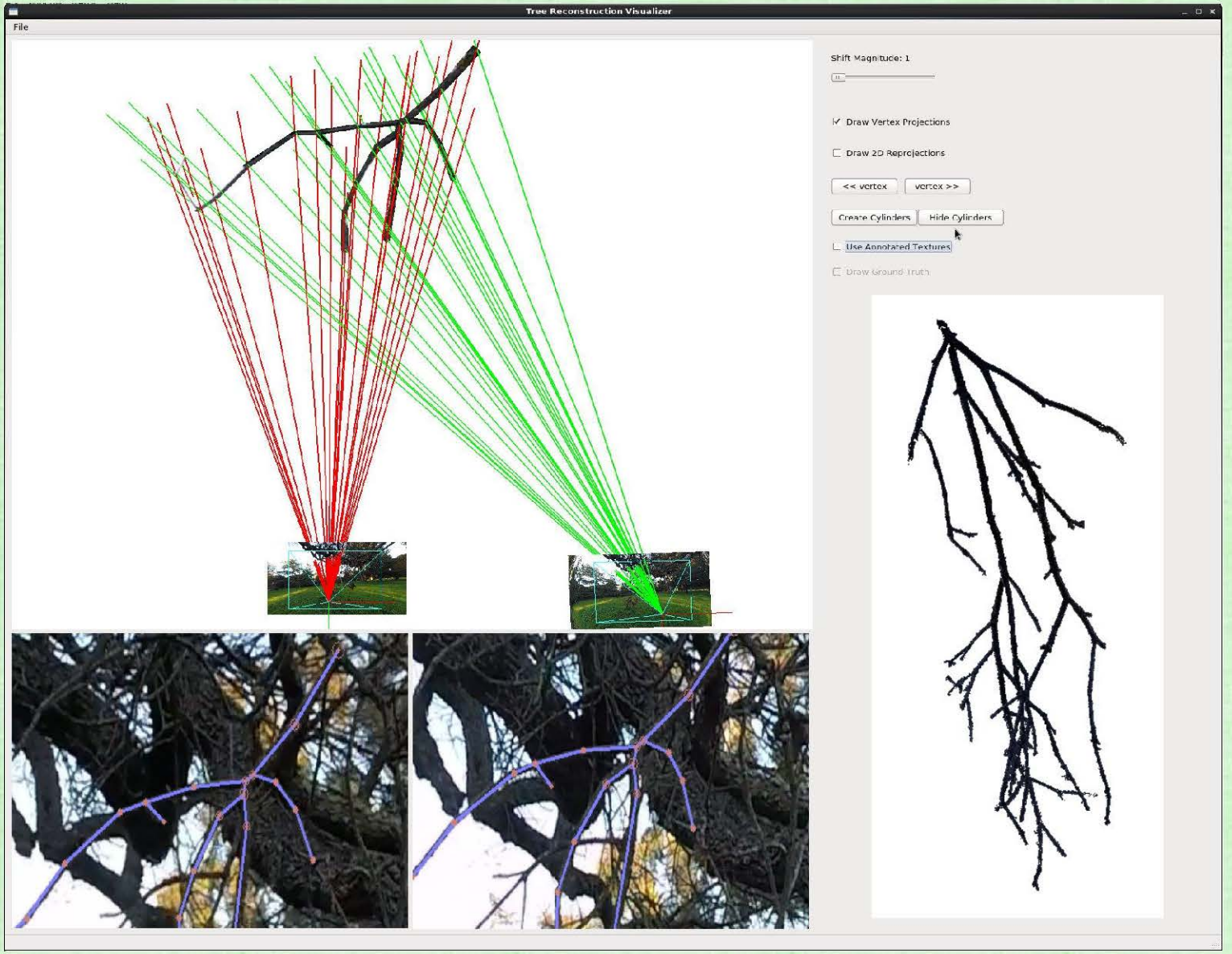




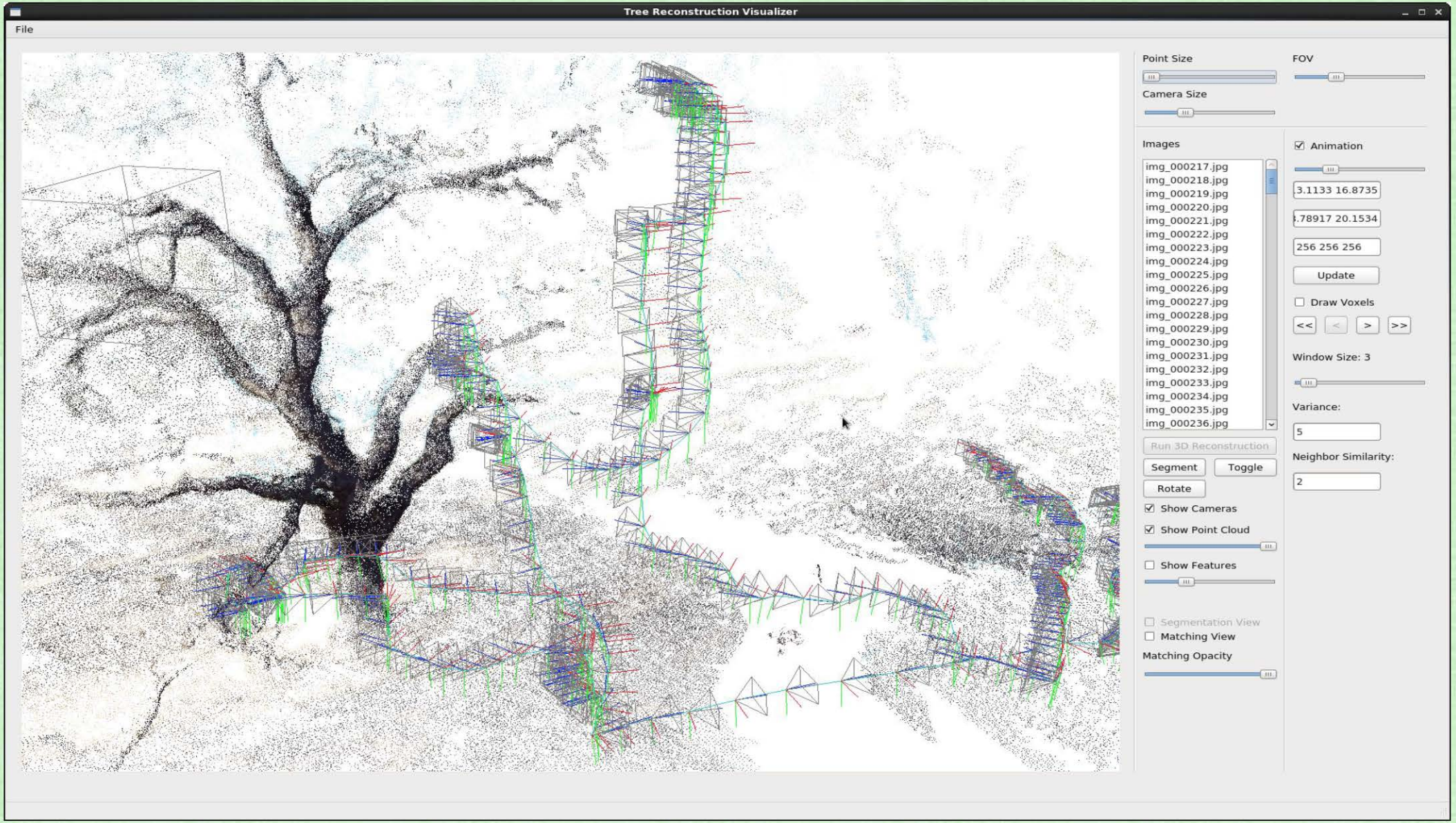
2D photos

3D geometry

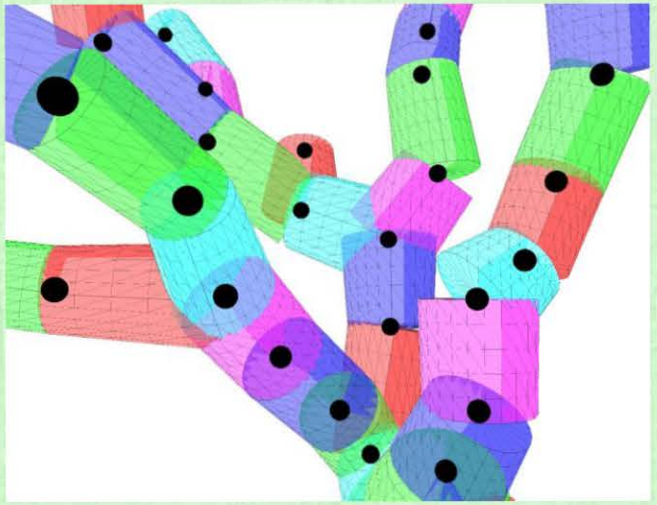
Drones & Trees



Drones & Trees



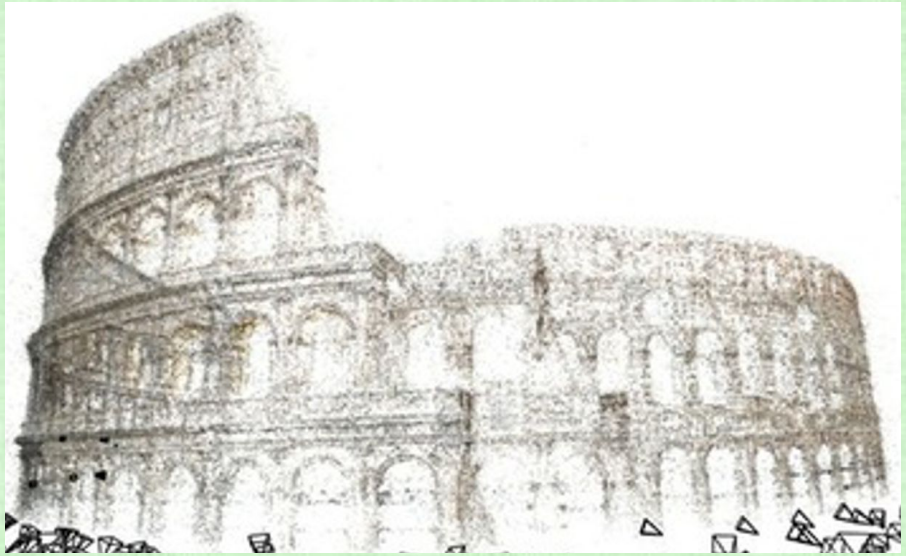
Drones & Trees

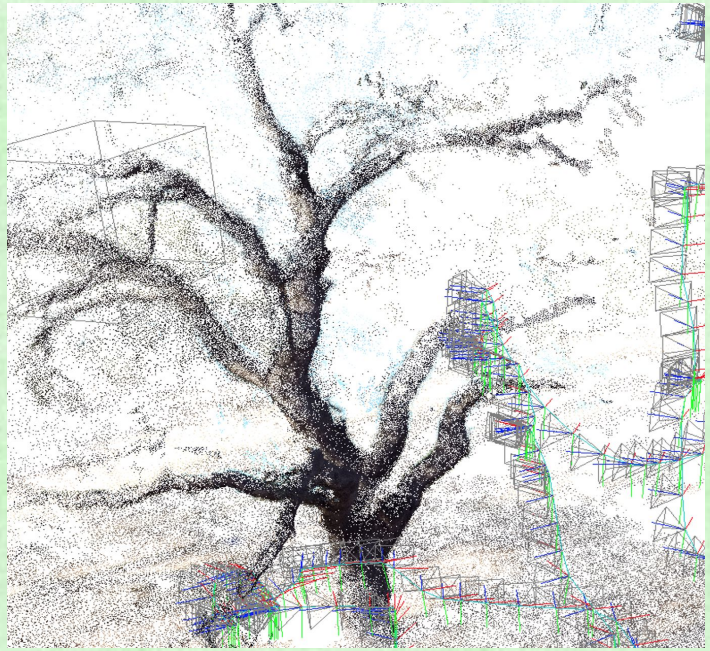


Drones & Trees

Noise & De-Noising

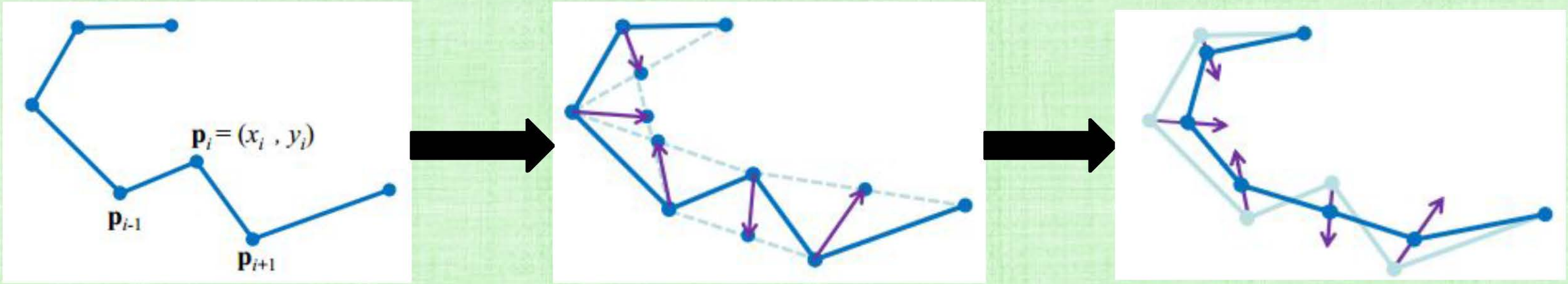
- Computer Vision algorithms use real-world sensors/data
- This results in noise, which is the biggest drawback to such methods
- Denoising/smoothing algorithms become very important





Laplacian Smoothing

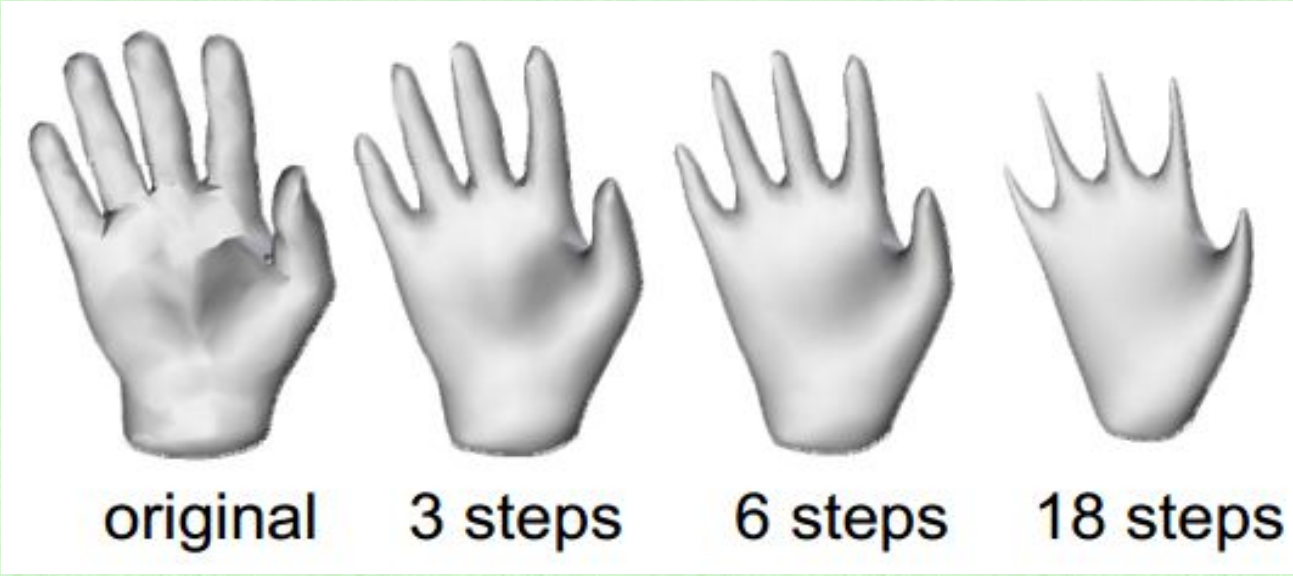
- Compute a Laplacian estimate using the one ring of vertices about a point
  - Similar to differential coordinates
- E.g., on a curve: $L(p_i) = \frac{1}{2}\left((p_{i+1} - p_i) + (p_{i-1} - p_i)\right)$
- Then, update $p_i^{new} = p_i + \lambda L(p_i)$ where $\lambda \in (0,1)$
- Repeat several iterations



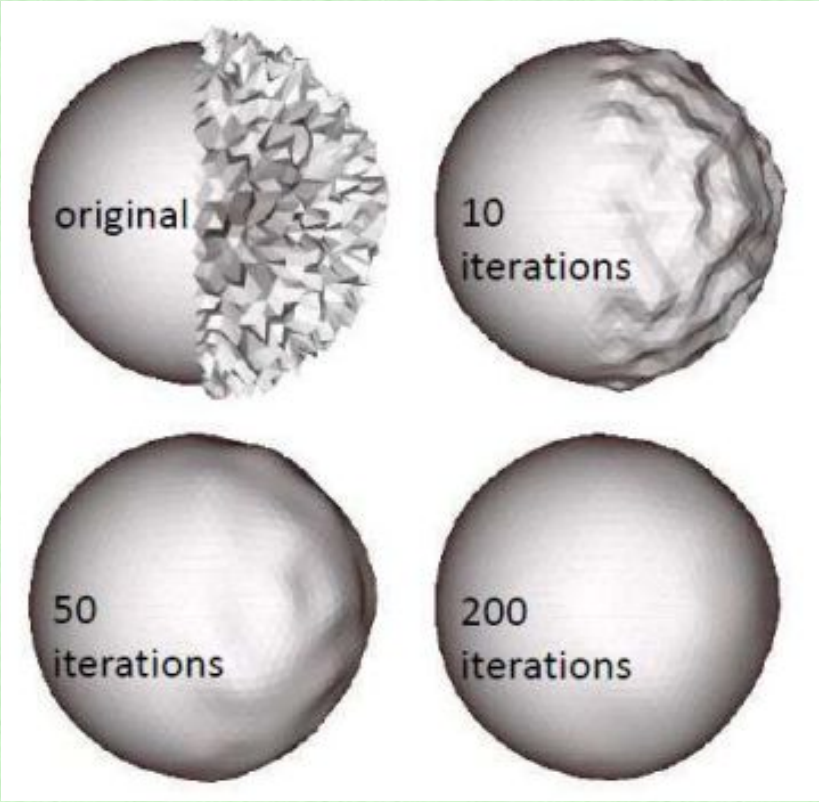
Taubin Smoothing

- Laplacian smoothing suffers from volume loss
- Taubin smoothing periodically performs an inflation step to add back volume:

$$p_i^{new} = p_i - \mu L(p_i) \quad \text{with} \quad \mu > 0$$



Laplacian smoothing (only)



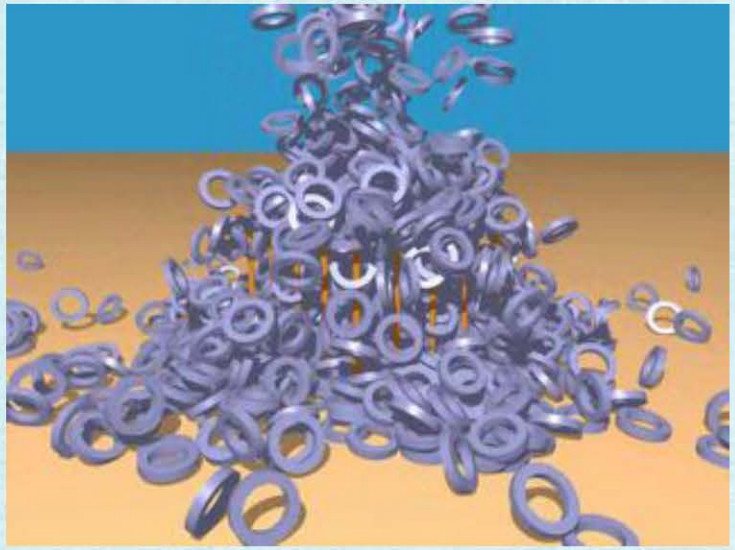
Taubin smoothing

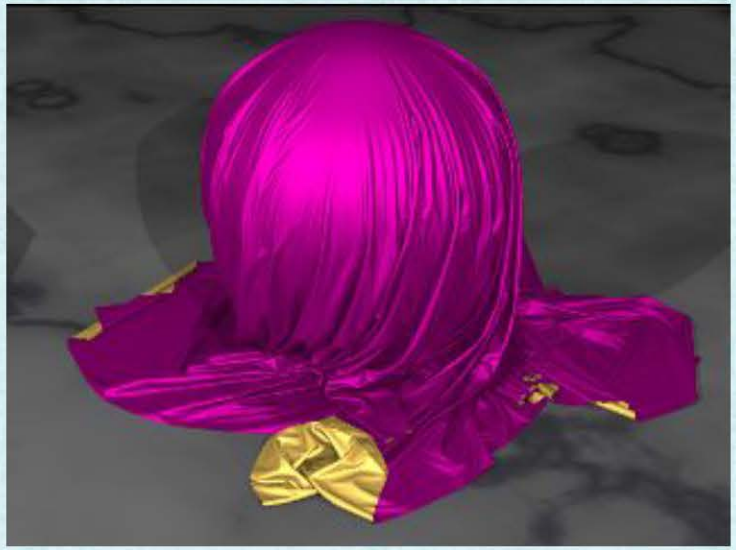
Physics Simulation

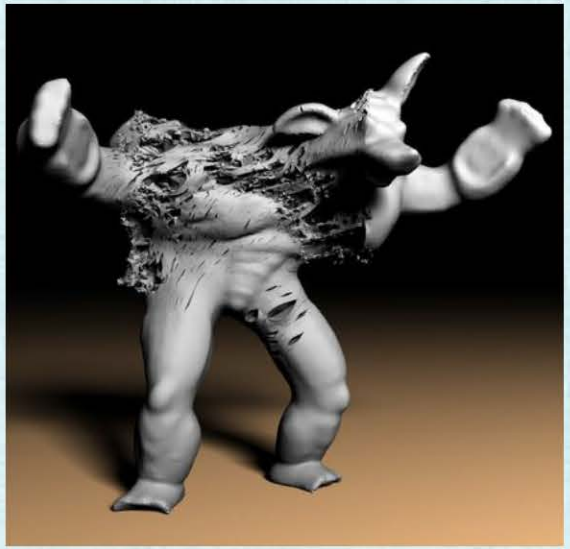
Netwon's Second Law

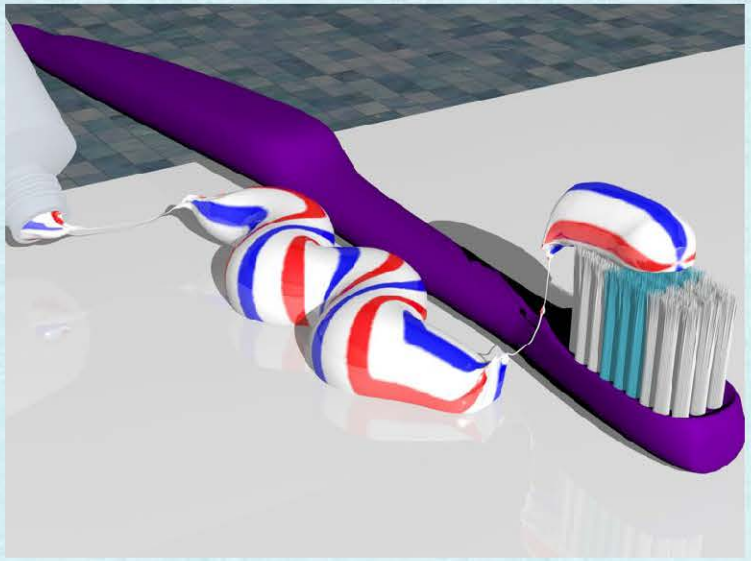
- Kinematics describe position $X(t)$ and velocity $V(t)$ as function of time t
  - $\frac{dX(t)}{dt} = V(t)$ or $X'(t) = V(t)$
- Dynamics describe responses to external stimuli
  - Newton's second law $F(t) = MA(t)$ is a dynamics equation
  - $V'(t) = A(t)$ implies $V'(t) = \frac{F(t)}{M}$ as well as $\frac{\mathrm{d}^2X(t)}{dt^2} = X''(t) = \frac{F(t)}{M}$
- Combining kinematics and dynamics gives: $\begin{pmatrix} X'(t) \\ V'(t) \end{pmatrix} = \begin{pmatrix} V(t) \\ \frac{F(t,X(t),V(t))}{M} \end{pmatrix}$
  - Note: forces often depend on position/velocity
- Much of the physical world can be simulated with computational mechanics (FEM) and computational fluid dynamics (CFD), using Newton's second law
  - Create degrees of freedom, specify forces, and solve the resulting ordinary differential equations (ODEs)
  - When the forces have spatial interdependence, one obtains partial differential equations (PDEs)

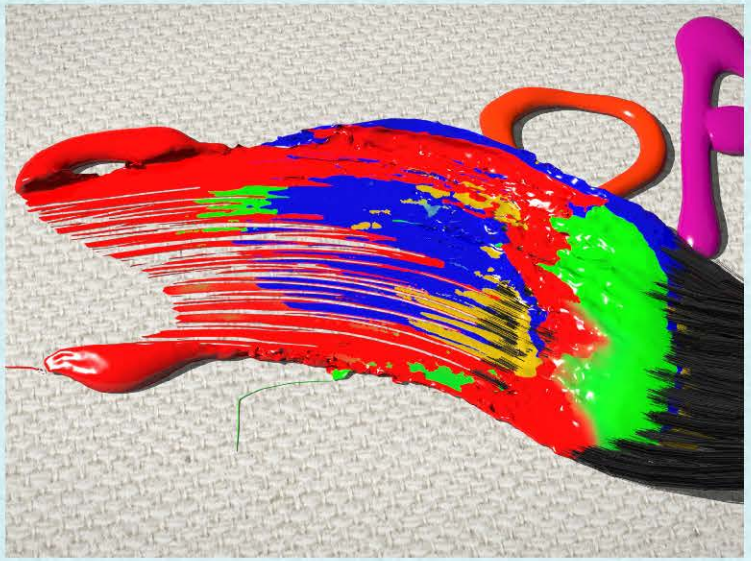
Computational Mechanics (FEM)

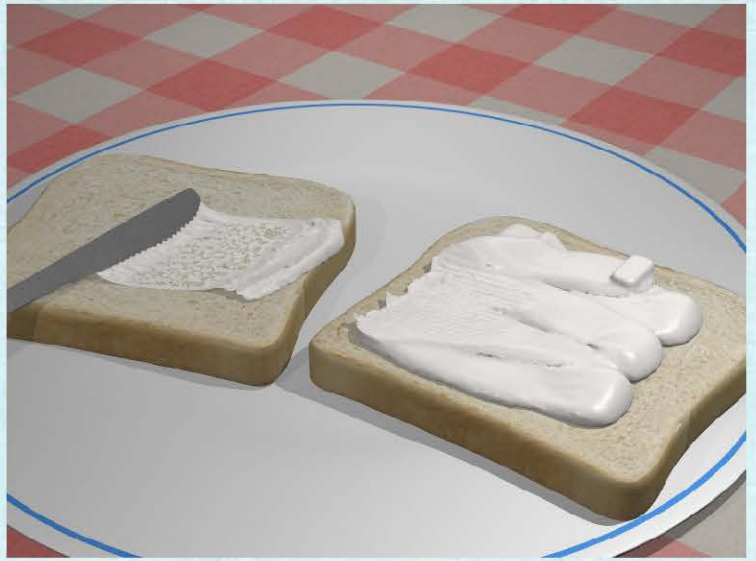




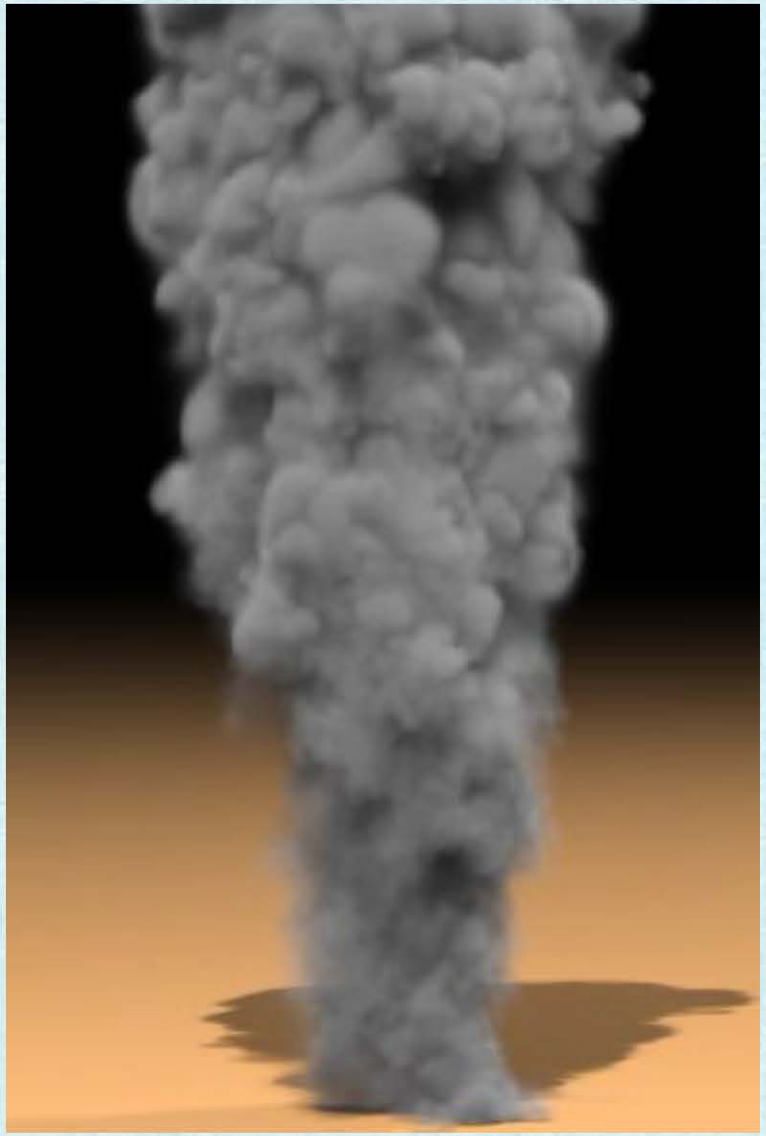


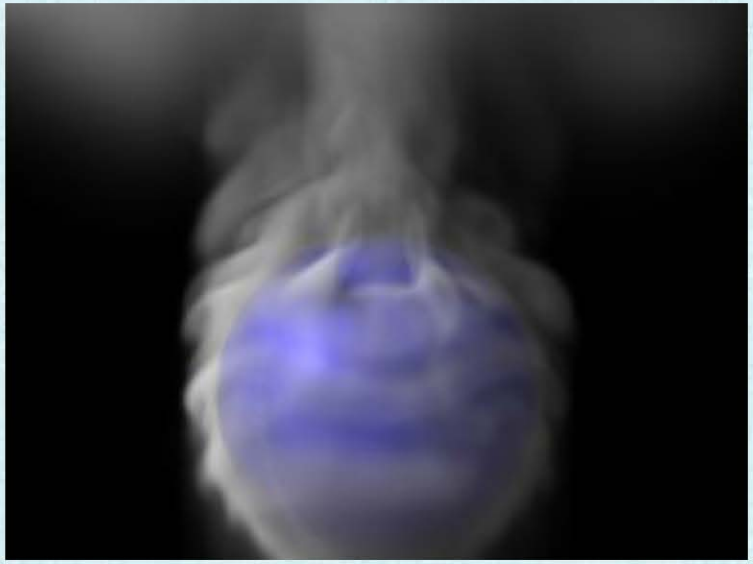


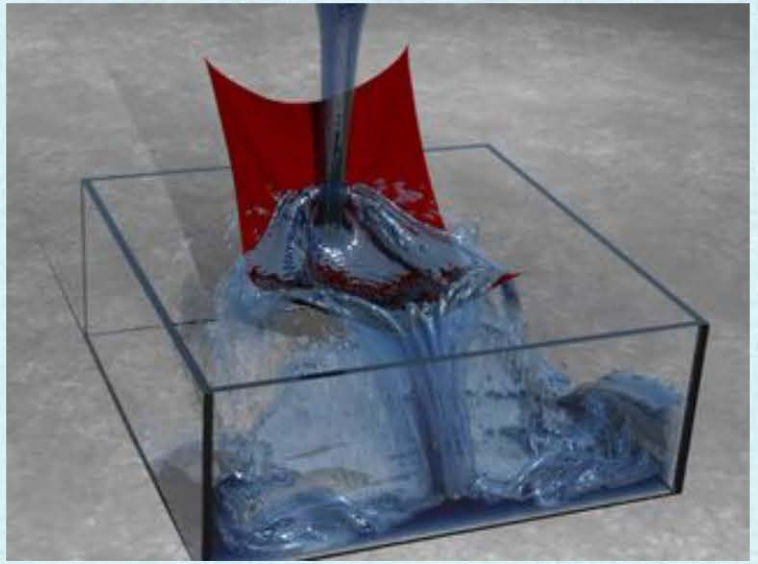




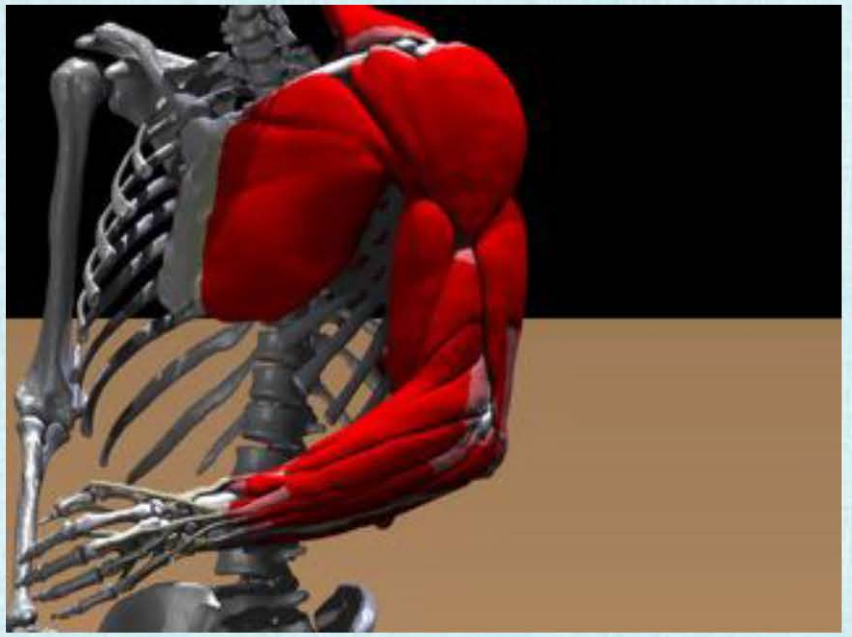
Computational Fluid Dynamics (CFD)

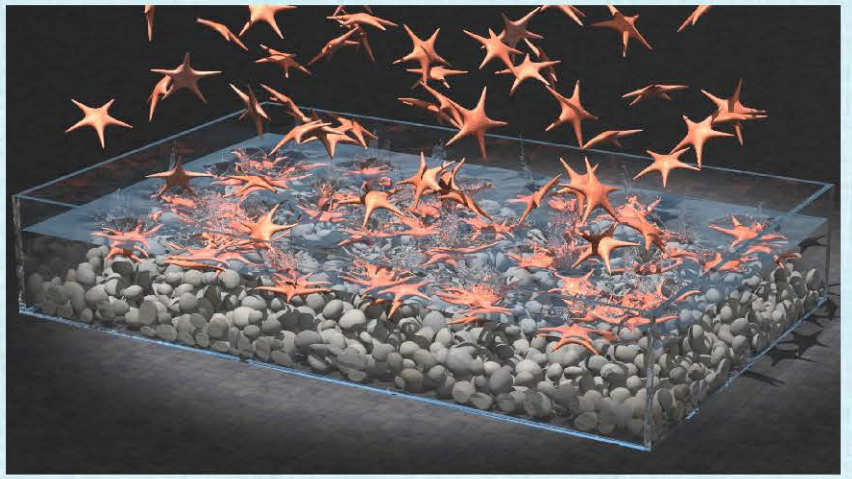






Computational Biomechanics

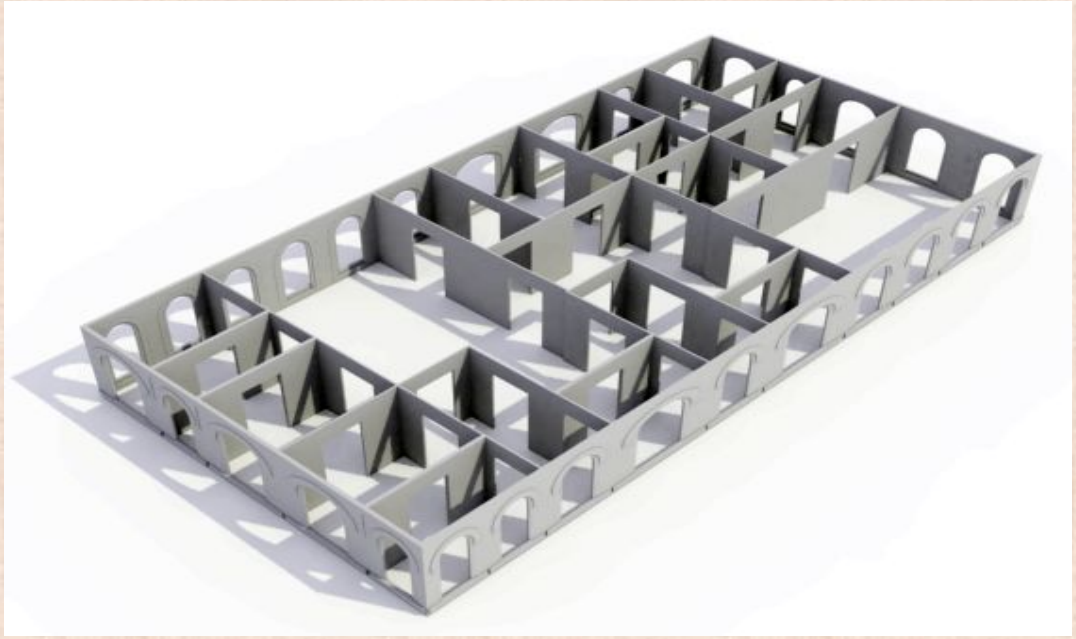


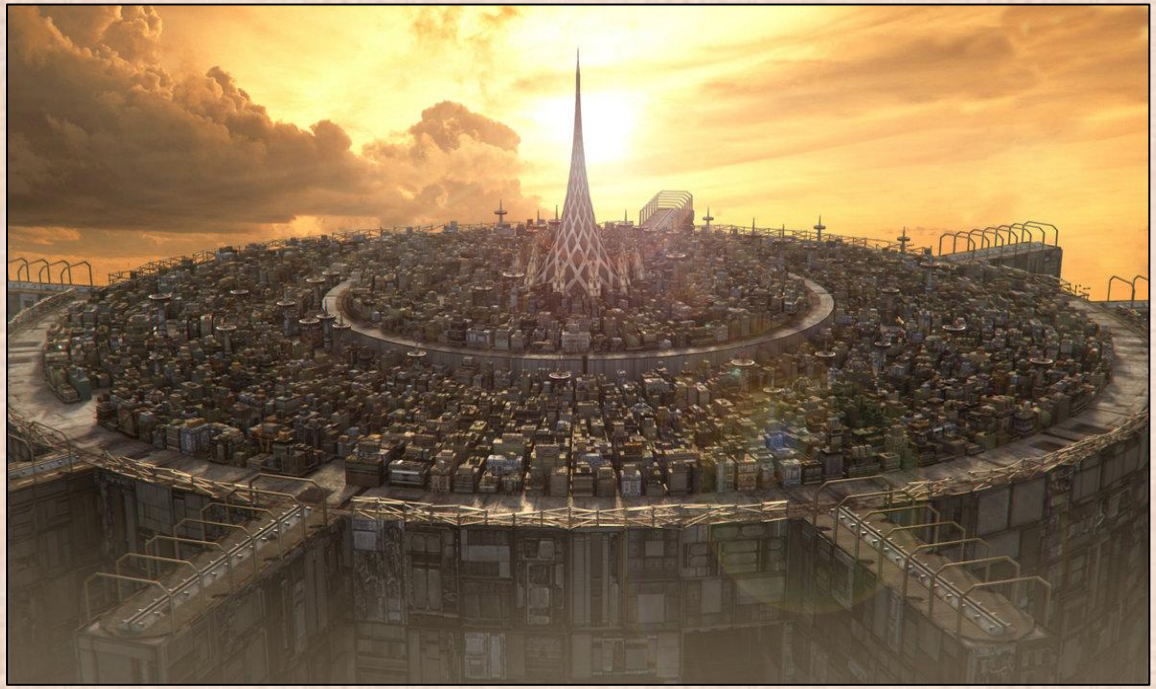


Procedural Methods

Procedural Geometry

- Generate geometry with an algorithm
  - Typically used for complex or tedious-to-create models
  - Perturb the algorithm to make variations of the geometry
- Start with a small set of data
- Use rules to describe high level properties of the desired geometry
- Add randomness, and use recursion



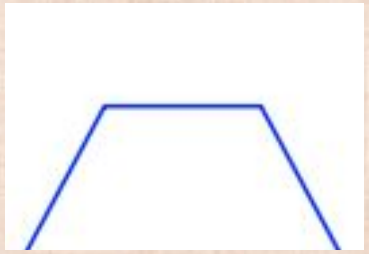


L-Systems

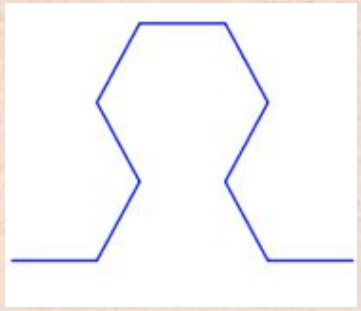
- Developed by a biologist (Lindenmayer) to study algae growth
- A recursive formal grammar:
  - An alphabet of symbols (terminal and non-terminal)
  - Production rules: non-terminal symbols recursively create new symbols (or sequences of symbols)
- Starts with an initial string (axiom), and apply production rules
- A translator turns symbols into geometric structures

Nonterminals: A and B both mean to “draw forward”
Terminals: +/- mean to turn right/left (respectively) by 60 degrees
Rules: $A \rightarrow B + A + B$ and $B \rightarrow A - B - A$

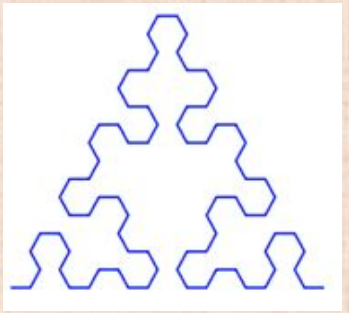
Intial Axiom: A



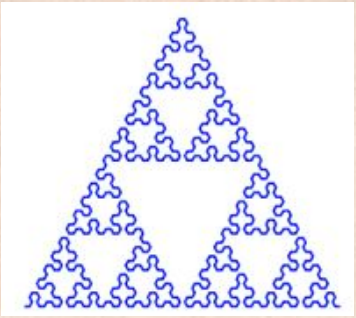
B+A+B



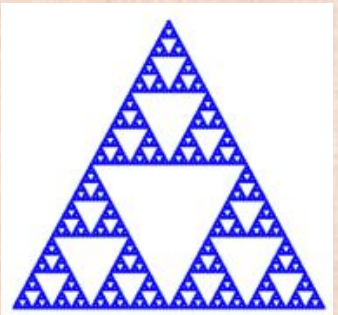
A-B-A + B+A+B + A-B-A



B+A+B-(A-B-A)-(B+A+B)
+A-B-A +B+A+B +A-B-A
+B+A+B-(A-B-A)-(B+A+B)



Etc.



Sierpinski Triangle

L-System + Stack = Branches

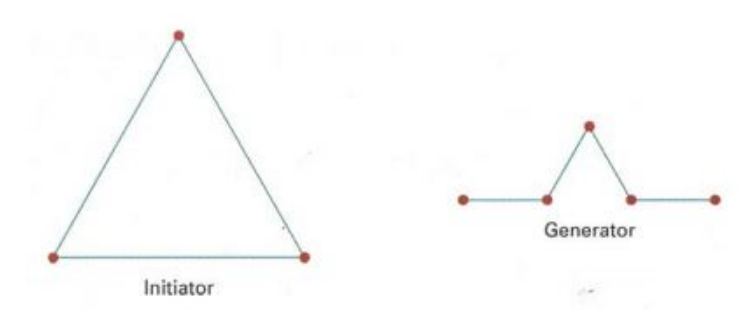
- Nonterminals: X is no action, F is draw forward
- Terminals: +/- means turn right/left by 25 degrees
  - **[means to store current state on the stack**
  - **] means to load the state from the stack**
- Initial Axiom: X
- Rules: X → F − [[X]+X] + F [+FX] − X, F → FF

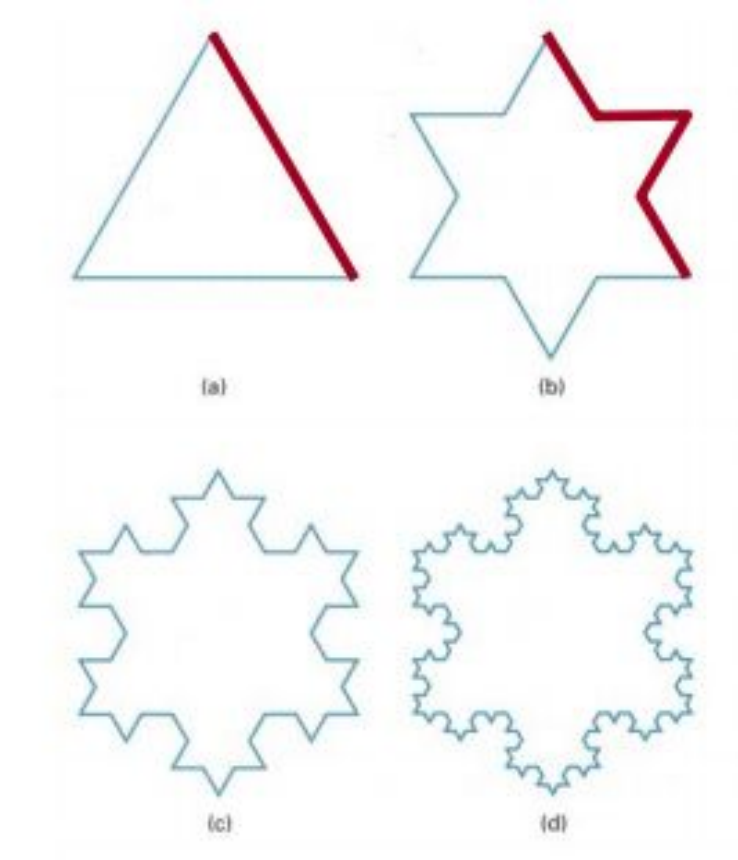
L-Systems

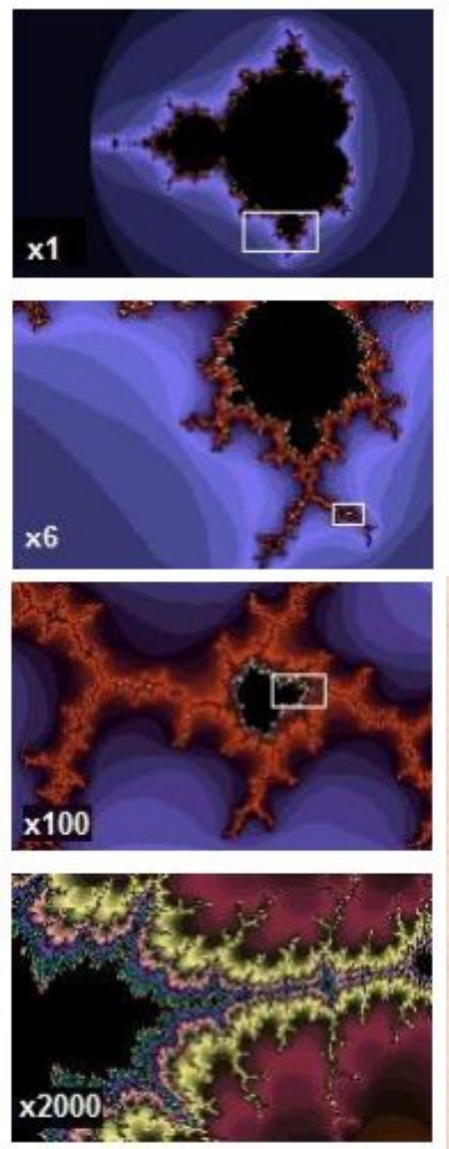
- Easily extended to 3D
  - Model trunk/branches as cylinders
  - As recursion proceeds:
    - Shrink cylinder size
    - Vary color (from brown to green)
  - Add more variety with a stochastic L-system
    - Multiple (randomly-chosen) rules for each symbol
  - Practice and experimentation is required in order to obtain good results
    - (just like for modern day ML)

Fractals

- Initiator: start with a shape
- Generator: replace subparts with scaled copy of the original
- Repeat

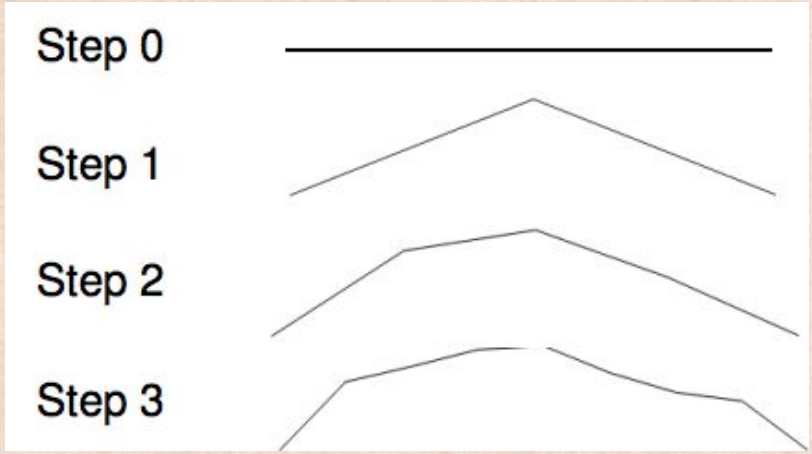






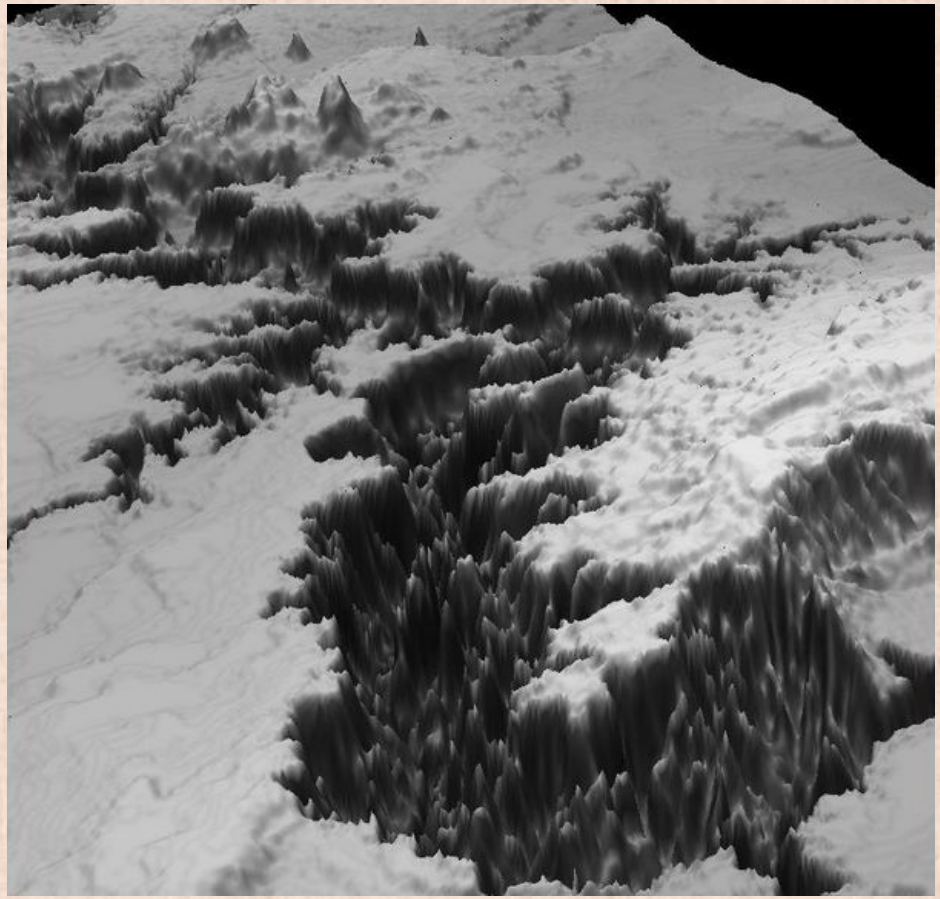
Fractals

- Add randomness (such as random midpoint location/displacement)
  - Create an irregular 2D silhouette (for far away mountains)



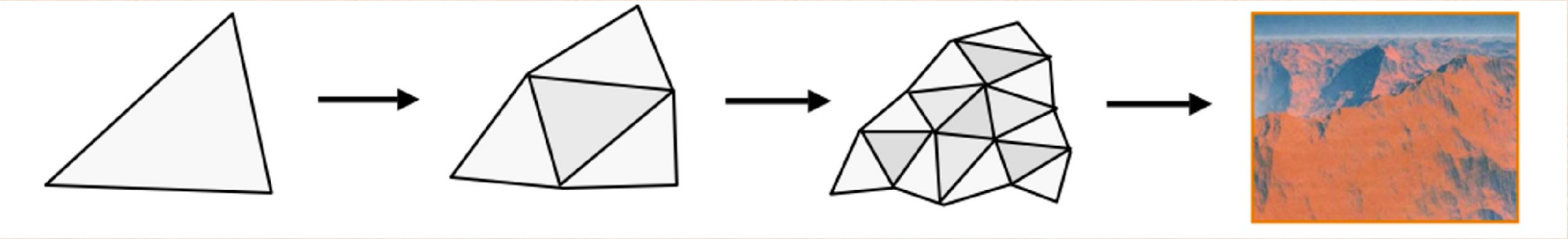
Height Fields

- Start with a 2D fractal (or any 2D grey-scale image)
- Place the image on top of a ground plane (subdivided into triangles)
- For each triangle vertex, vary the height based on pixel intensity



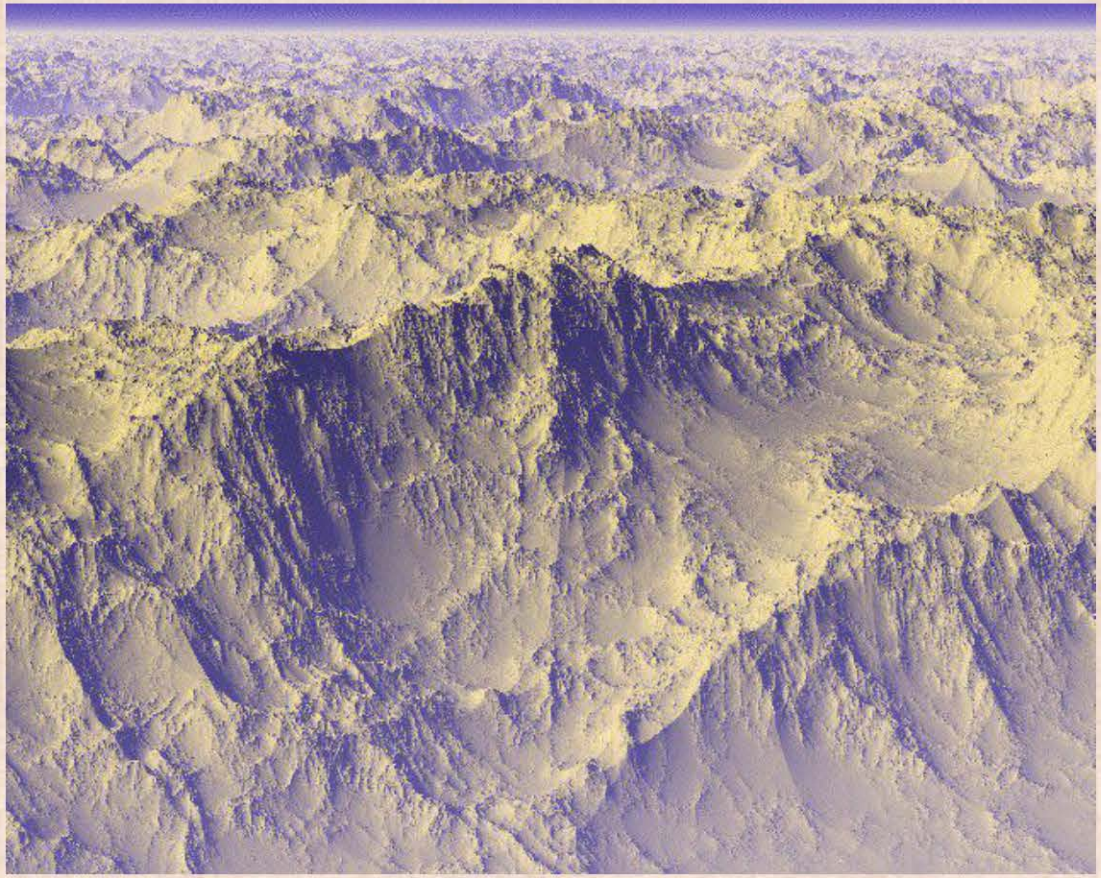
3D Landscapes

- Initiator: start with a shape
- Generator: replace random subparts with a self-similar random pattern

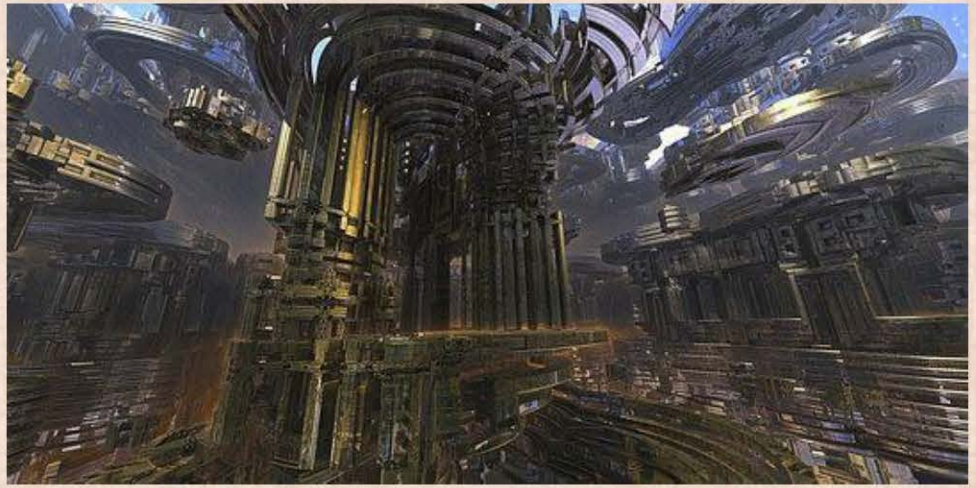


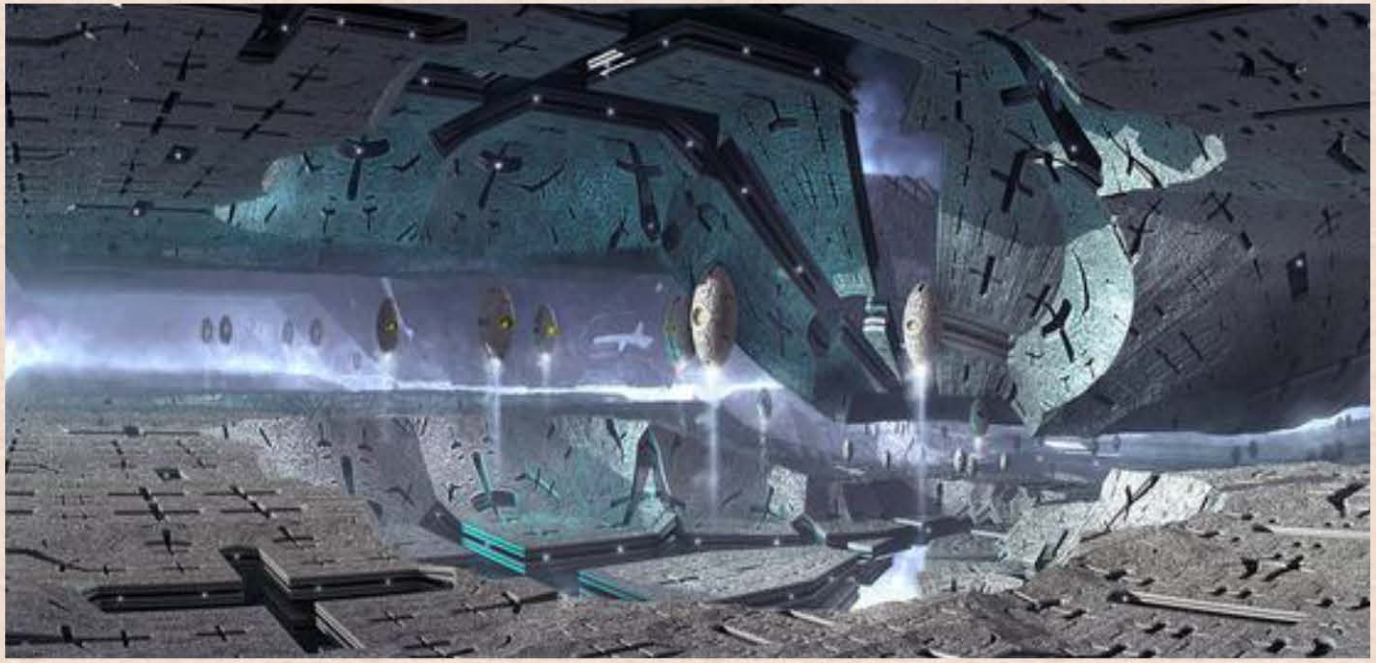
- Similar to subdivision, but with much more interesting rules for setting vertex positions

3D Landscapes



Fractal Worlds





Machine Learning

Machine Learning

- Interactive Example-Based Terrain Authoring with Generative Adversarial Networks
  - Siggraph 2017