CSE528 Computer Graphics: Theory, Algorithms, and Applications

Hong Qin
Rm. 151, NEW CS Building
Department of Computer Science
Stony Brook University (SUNY at Stony Brook)
Stony Brook, New York 11794-2424
Tel: (631)632-8450; Fax: (631)632-8334
qin@cs.stonybrook.edu; or qin@cs.sunysb.edu;
http://www.cs.stonybrook.edu/~qin

Introduction to Geometric Modeling

- What is geometric modeling
  - Representation of existing objects (mathematical tools to represent shape geometry of real-world objects, both natural and manufactured ones)
  - Reverse engineering (from physical prototypes to digital prototypes)
  - Design of new objects (shape editing, deformation, manipulation)
  - Rendering - leading to visual interpretation
- Application of geometric modeling
  - Graphics, CAD, CAGD, CAM/CAE, robotics, vision, virtual reality, scientific visualization, animation, physical simulation, computer games, etc.

Geometry Representations

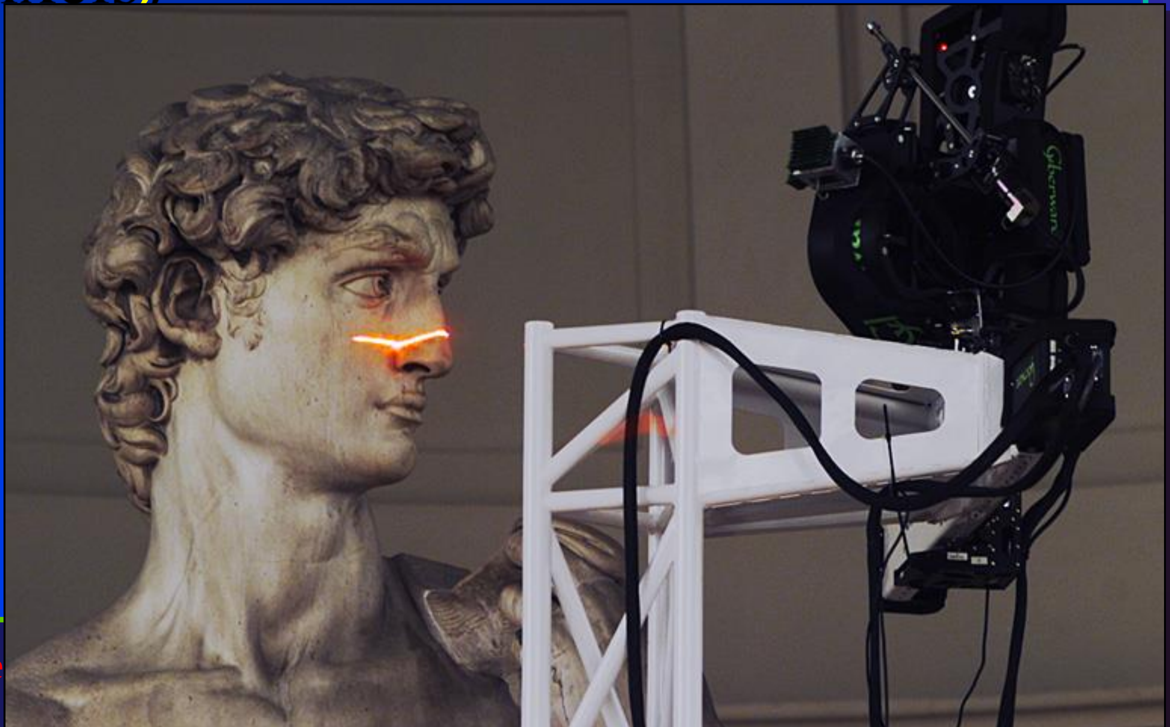
- Various strengths and weaknesses
  - Ease of use for design
  - Ease/speed for rendering
  - Simplicity
  - Smoothness
  - Collision detection
  - Flexibility (in more than one sense)
  - Suitability for simulation, and *many others*...

Point-based Graphics

- Modeling
- Rendering
- Animation / Simulation (Physics-based modeling)

Data Acquisition

- Laser scanners – obtaining millions to billions of points (x, y, z coordinates)
- Consider regular images – a few million pixels
- More points (than pixels)



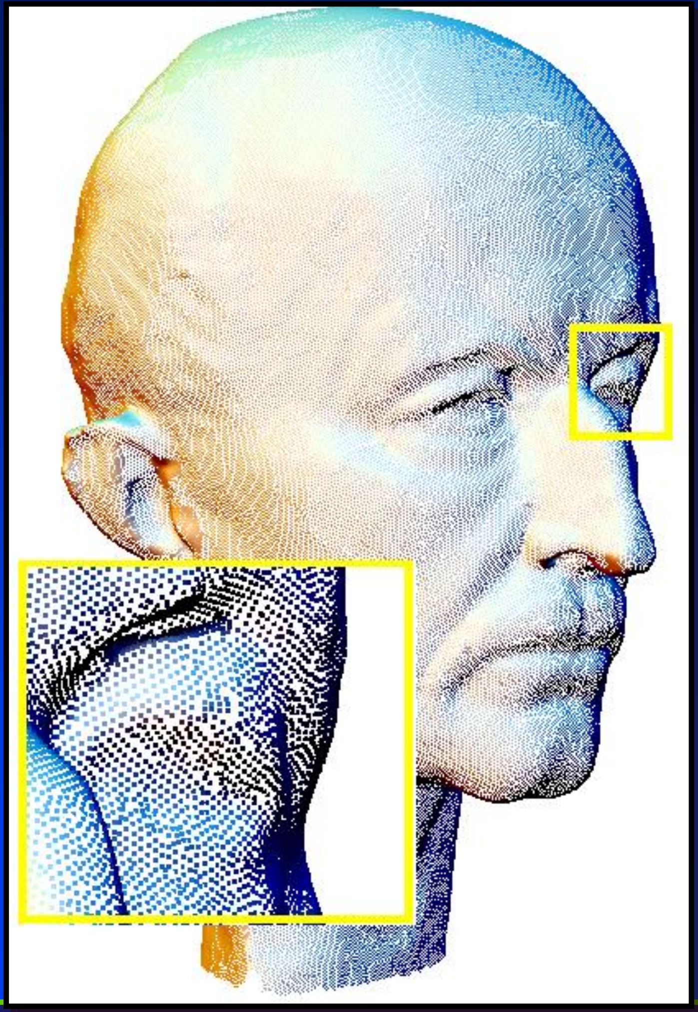
Point-based Graphics

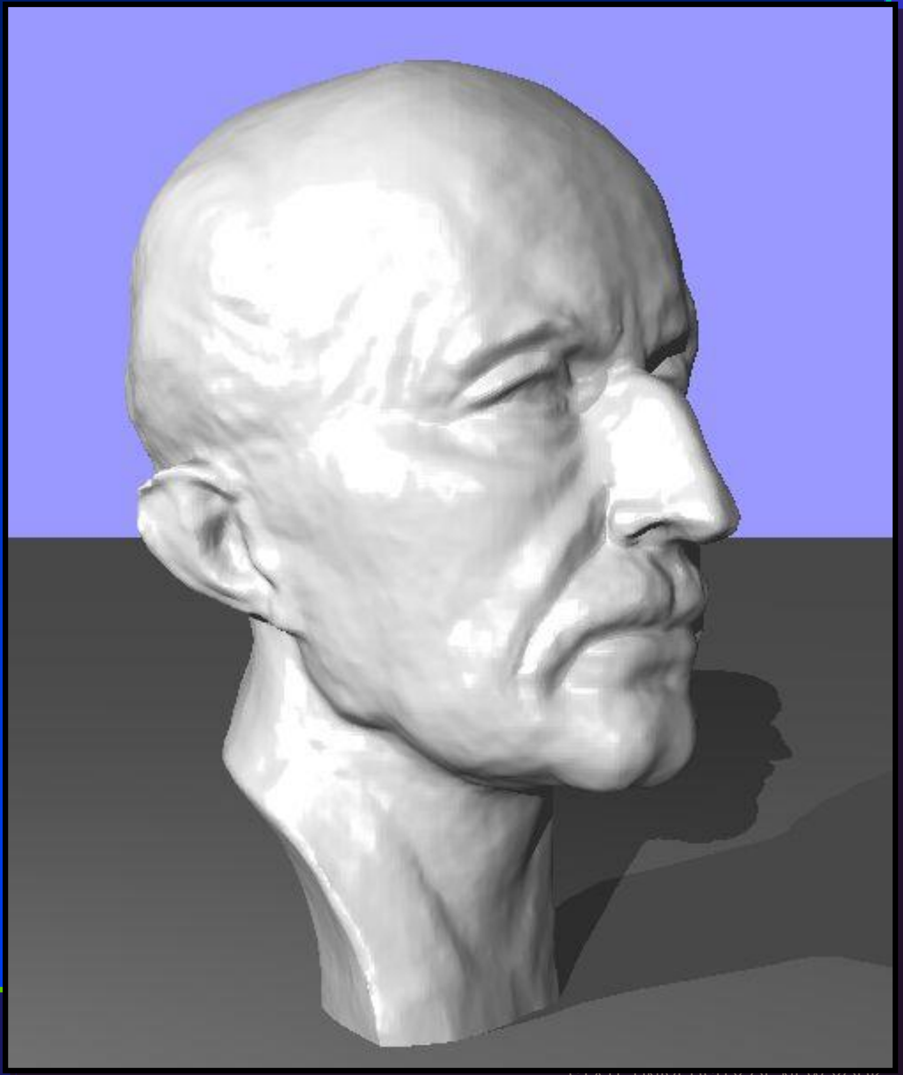
- Laser scanners
  - Millions to *billions* of points
- Typical image
  - At most a few million pixels
- More points than pixels...

Point-based Graphics

- Surfaces represented only by points
  - There are also normals (in addition to (x,y,z) coordintes)
  - No topology – no connectivity information
- How can we do
  - Rendering
  - Modeling operations
  - Simulation

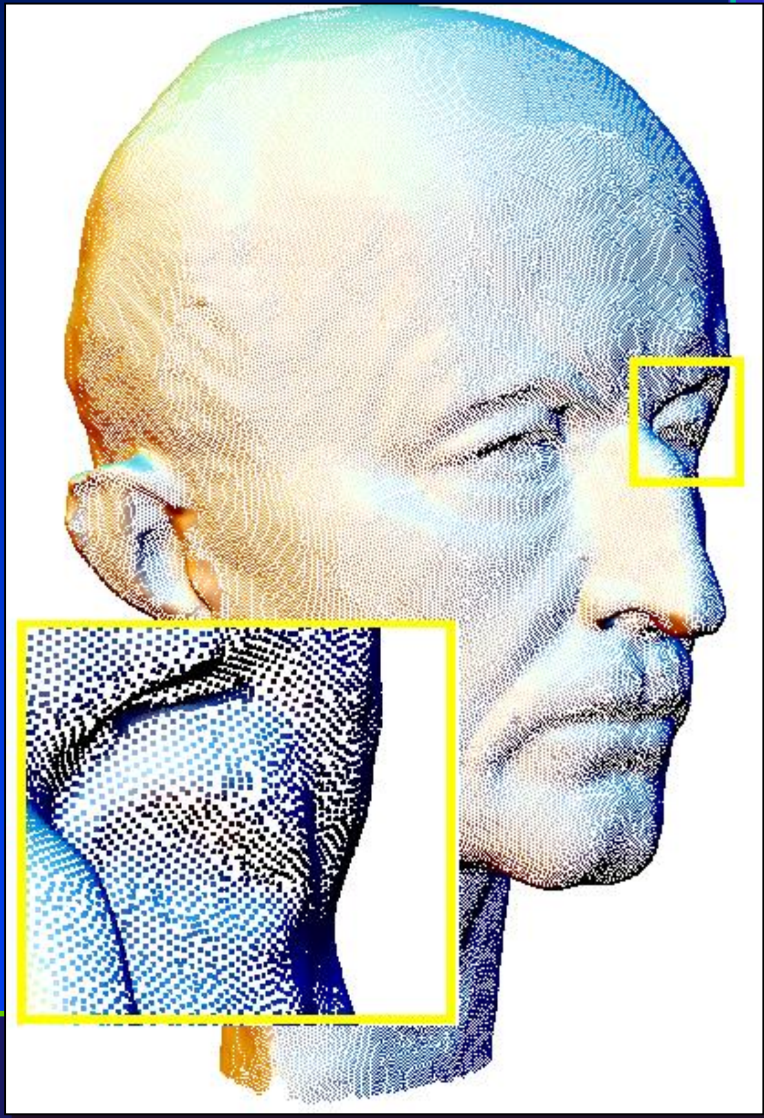
Point-based Surface Descriptions





Point Rendering

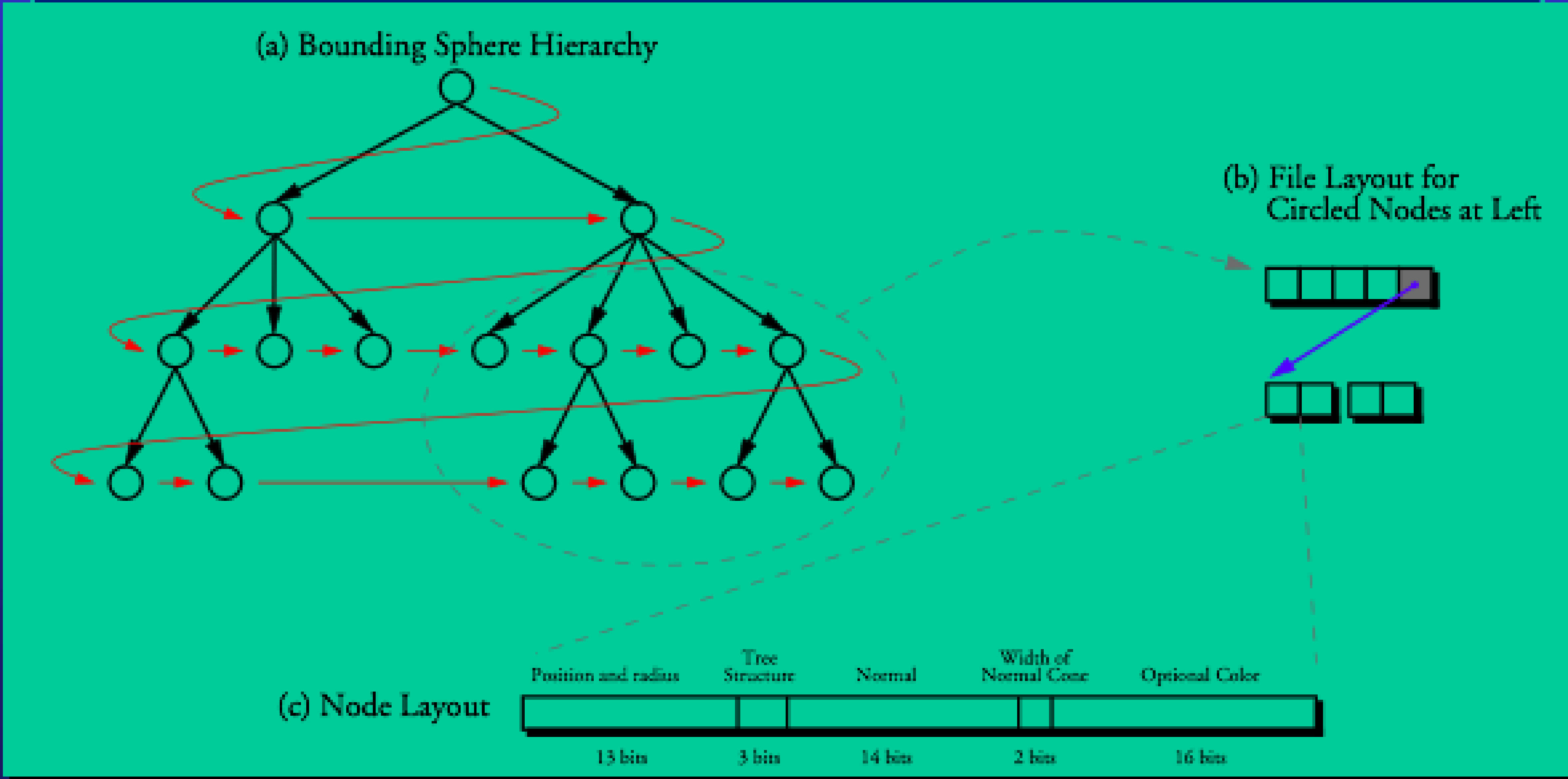
- For each point draw a little "splat"
  - Use associated normal for shading
  - Possibly apply texture
- If "splats" are small compared to spacing then gaps result
- Splatting too many points would waste time



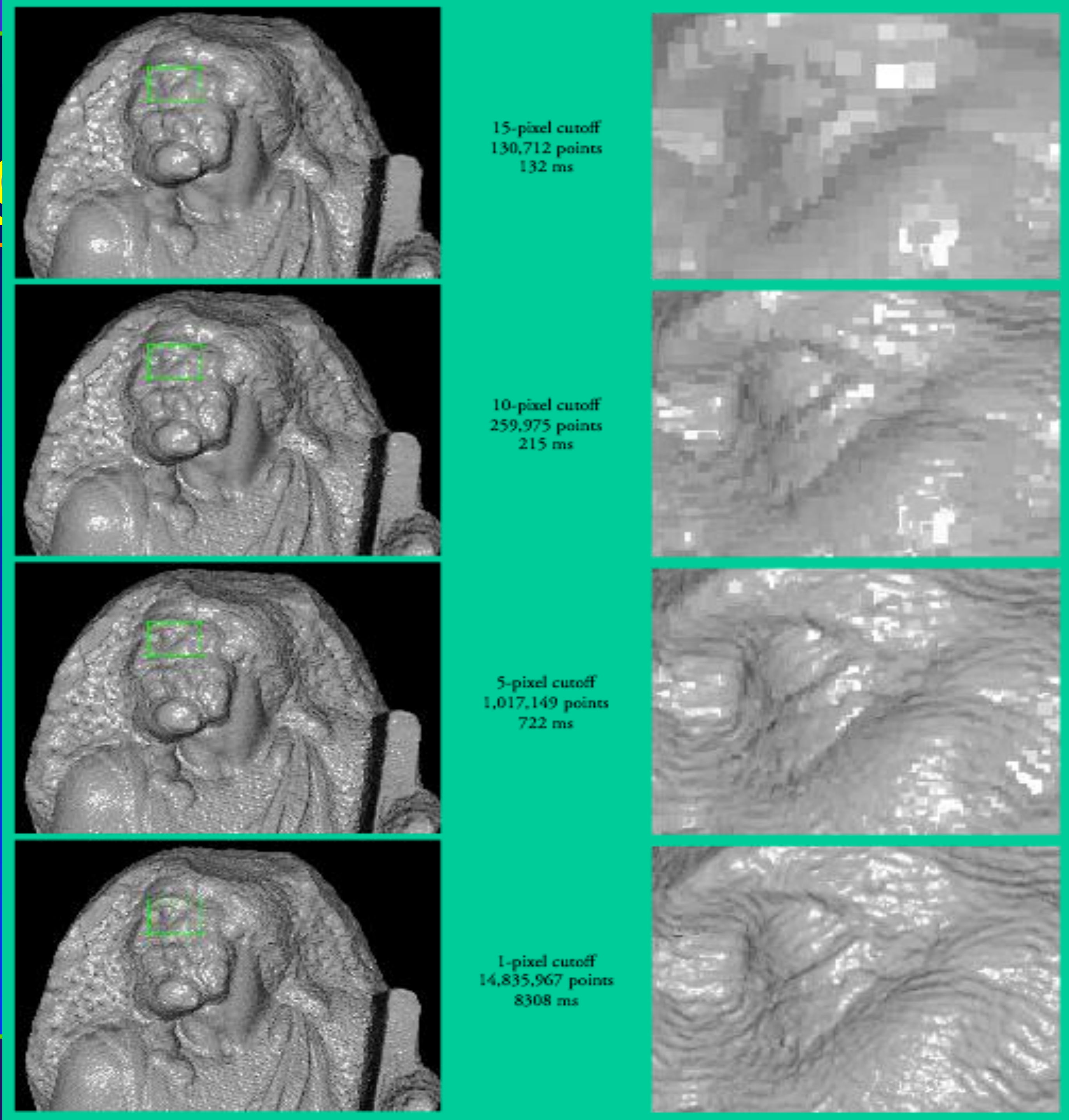
Rendering

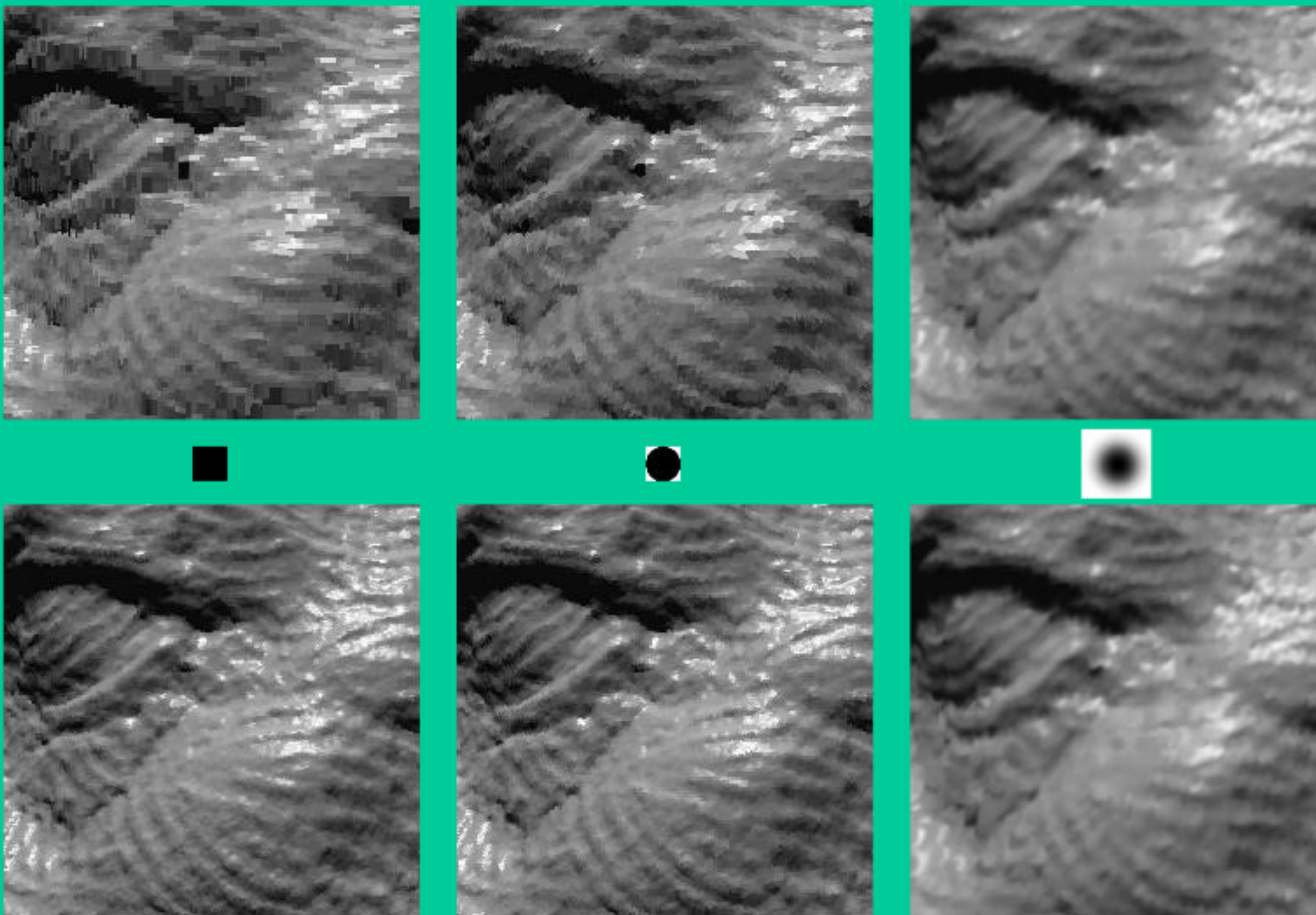
- "QSplat" algorithm
  - Build hierarchical tree of the points
  - Use bounding spheres to estimate size of clusters
  - Render clusters based on screen size
  - Use cluster-normals for internal nodes

Rendering – Qsplat Algorithm

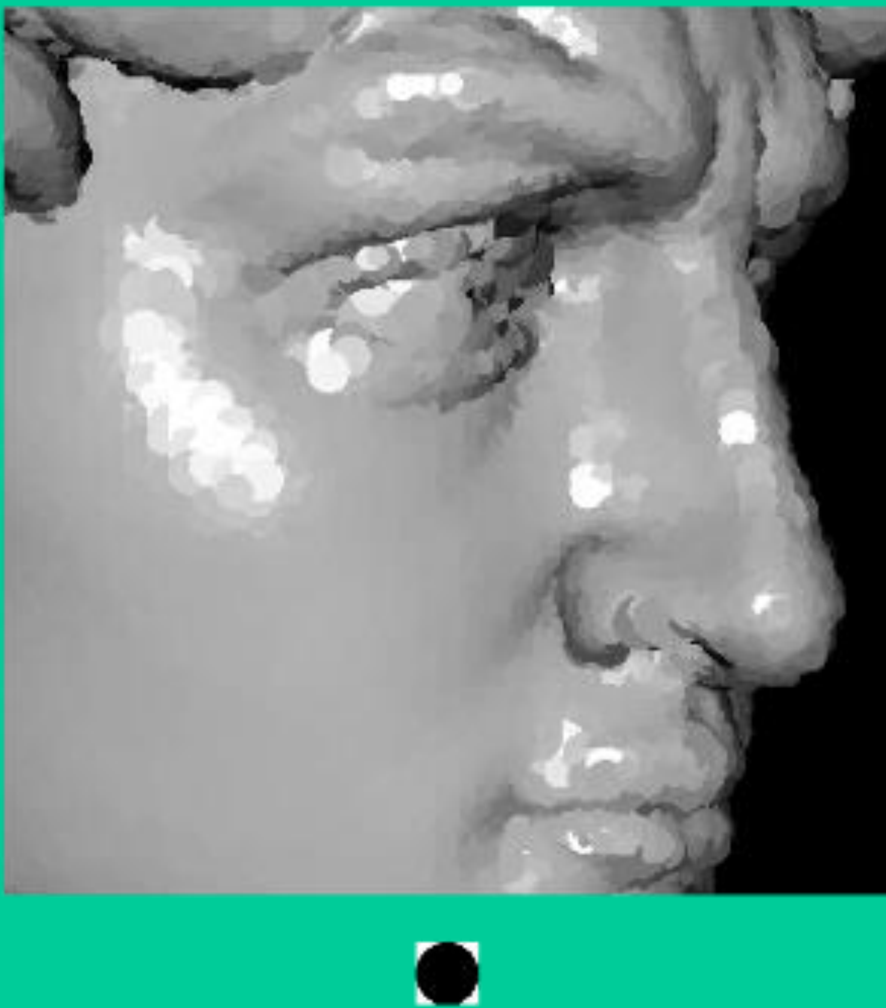


Renderin


15-pixel cutoff
130,712 points
132 ms
10-pixel cutoff
259,975 points
215 ms
5-pixel cutoff
1,017,149 points
722 ms
1-pixel cutoff
14,835,967 points
8308 ms



Rendering



Rendering

(a)
Points

(b)
Polygons – same number of primitives as (a)
Same rendering time as (a)

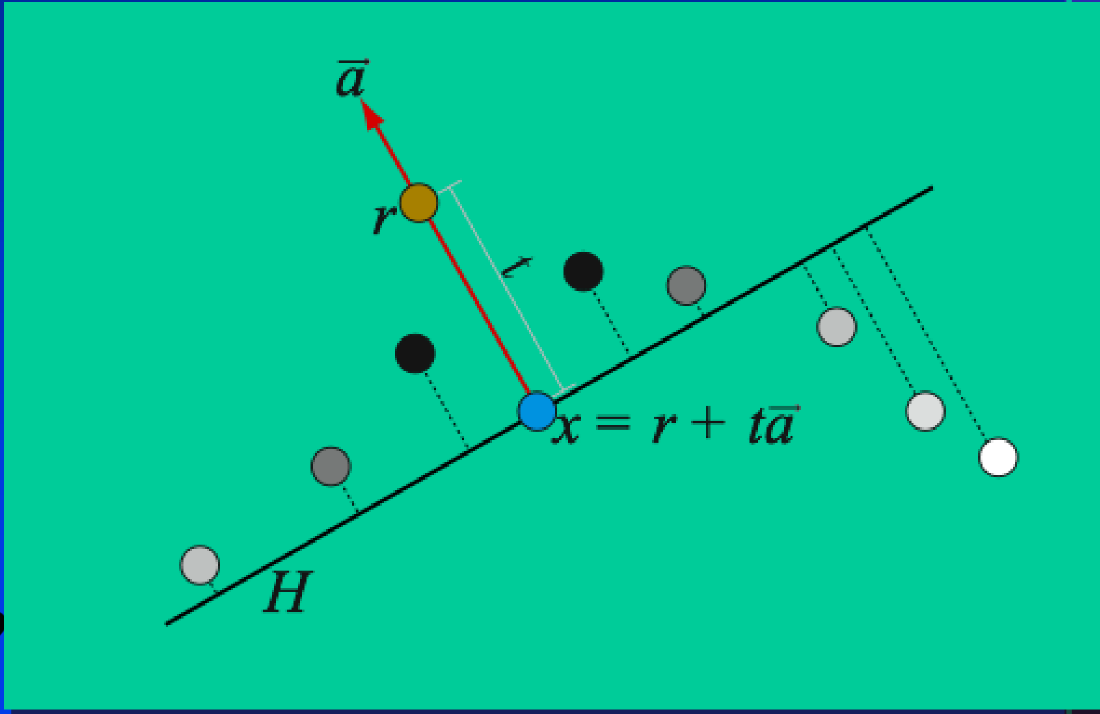
(c)
Polygons – same number of vertices as (a)
Twice the rendering time of (a)

Defining a Point Cloud Surface

- Modeling a point-cloud surface
- Two related methods
  - Surface is a point attractor
    - Point-set surfaces
  - Implicit surface
    - Multi-level Partition of Unity Implicits
    - Implicit Moving Least-Squares

Point-Set Surfaces

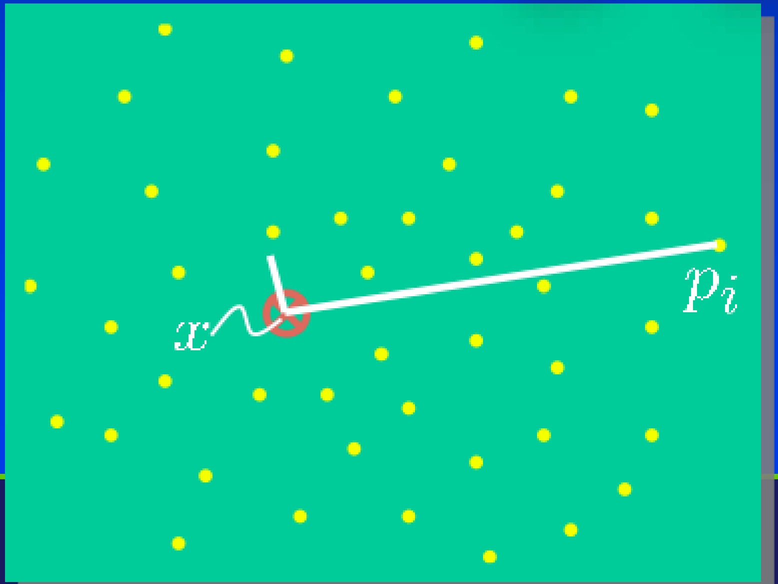
- Surface is the attractor of a repeated projection process
  - Find nearby points
  - Fit plane (weighted)
  - Project into plane
  - Repeat
- Does it converge?
- How to weight points?



Standard Least-Squares Fitting

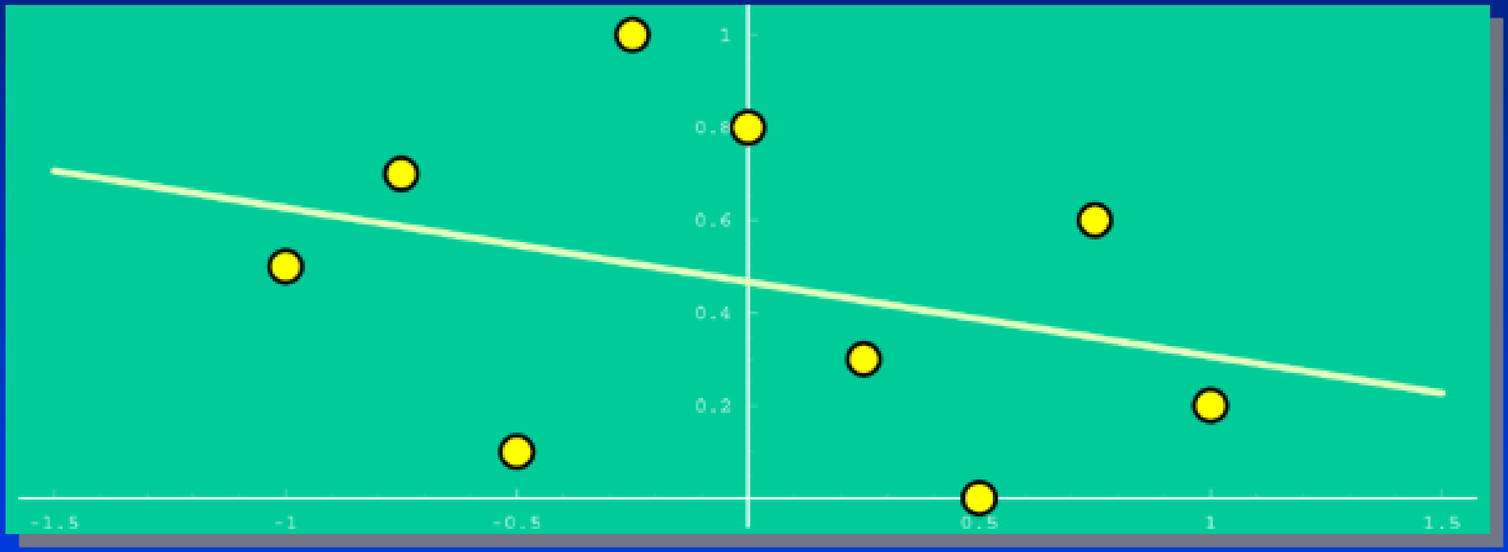
$$\begin{bmatrix} b^{\mathsf{T}}(p_1) \\ \vdots \\ b^{\mathsf{T}}(p_N) \end{bmatrix} c = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}$$

$$B^{\mathsf{T}} B\, c = B^{\mathsf{T}} \phi$$

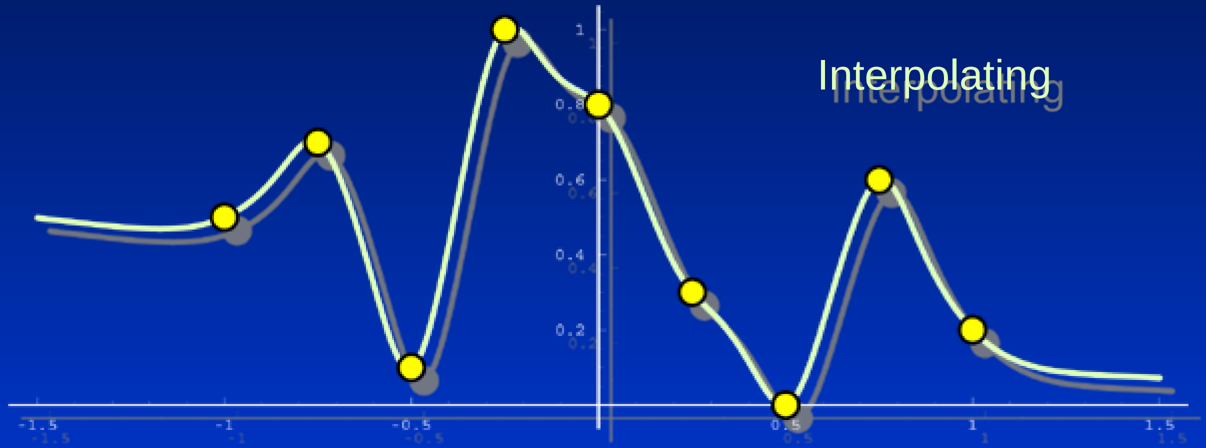


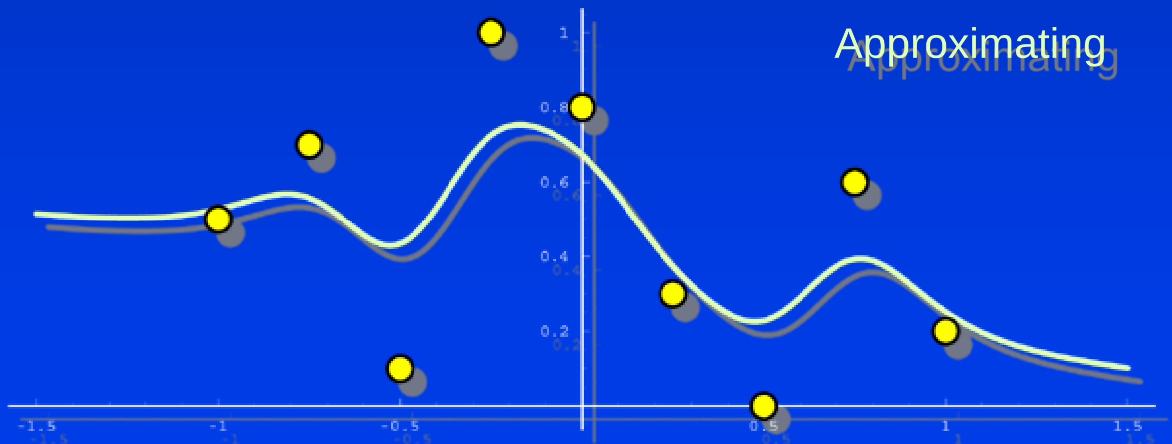
Moving Least-Square Interpolation

Least Squares



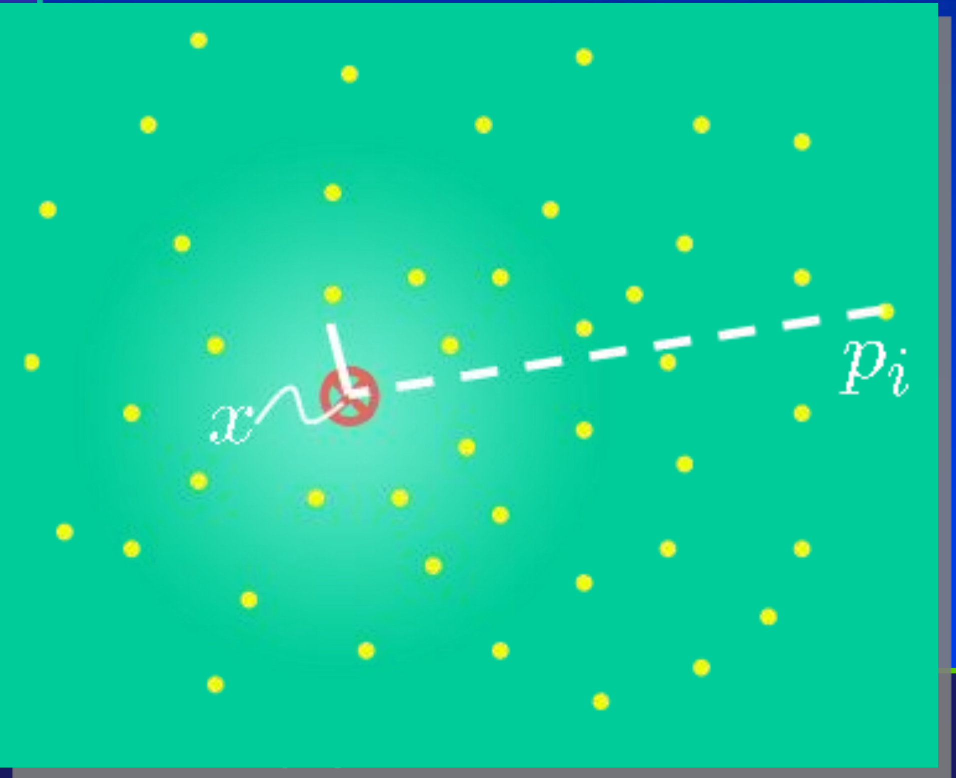
Moving Least-Squares Fitting





Moving Least-Squares Fitting

$$\begin{bmatrix} w(x,p_1) & & \\ & \ddots & \\ & & w(x,p_N) \end{bmatrix} \begin{bmatrix} b^\top(p_1) \\ \vdots \\ b^\top(p_N) \end{bmatrix} c = \begin{bmatrix} w(x,p_1) & & \\ & \ddots & \\ & & w(x,p_N) \end{bmatrix} \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}$$

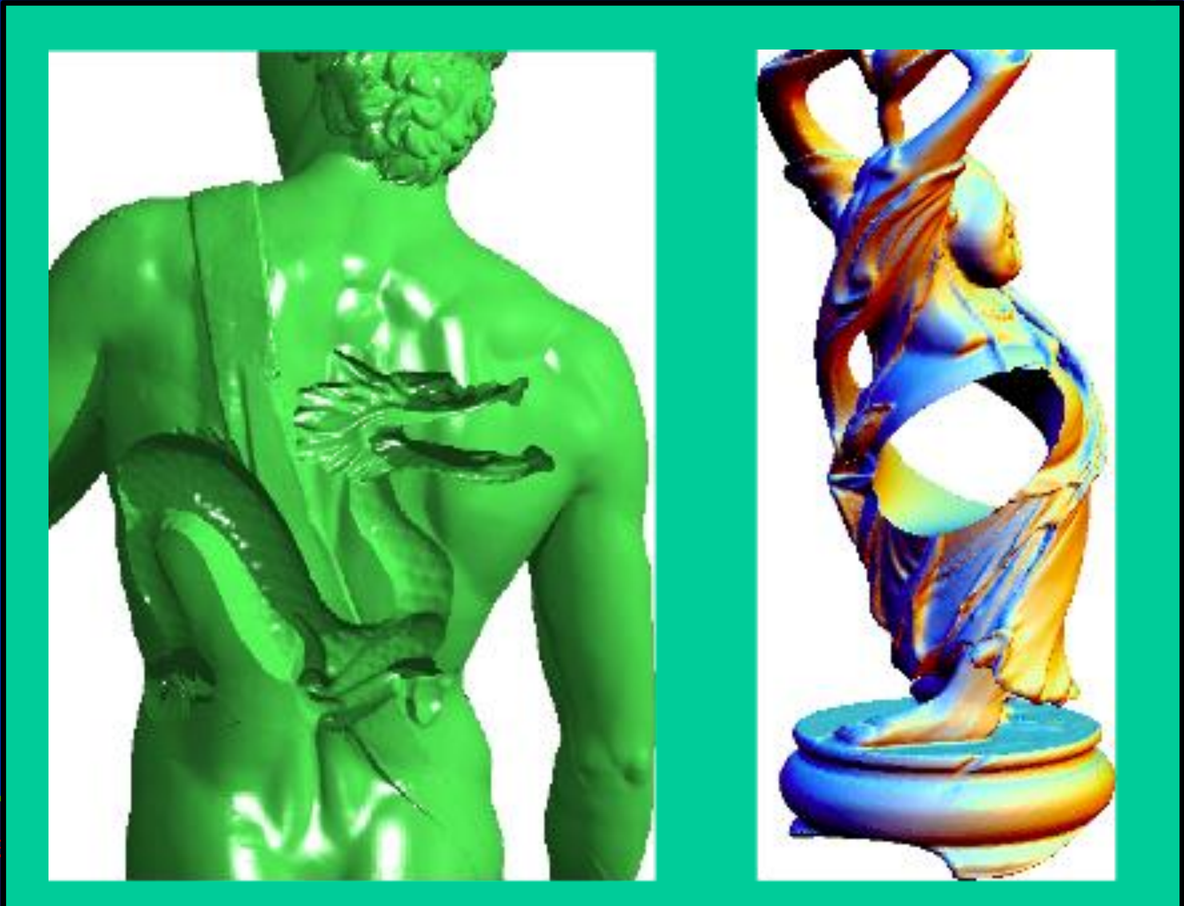


$$w(r) = \frac{1}{\left(r^2 + \epsilon^2\right)}$$

$$B^\top \, (W(x))^2 \, B \, c(x) = B^\top \, (W(x))^2 \, \phi$$

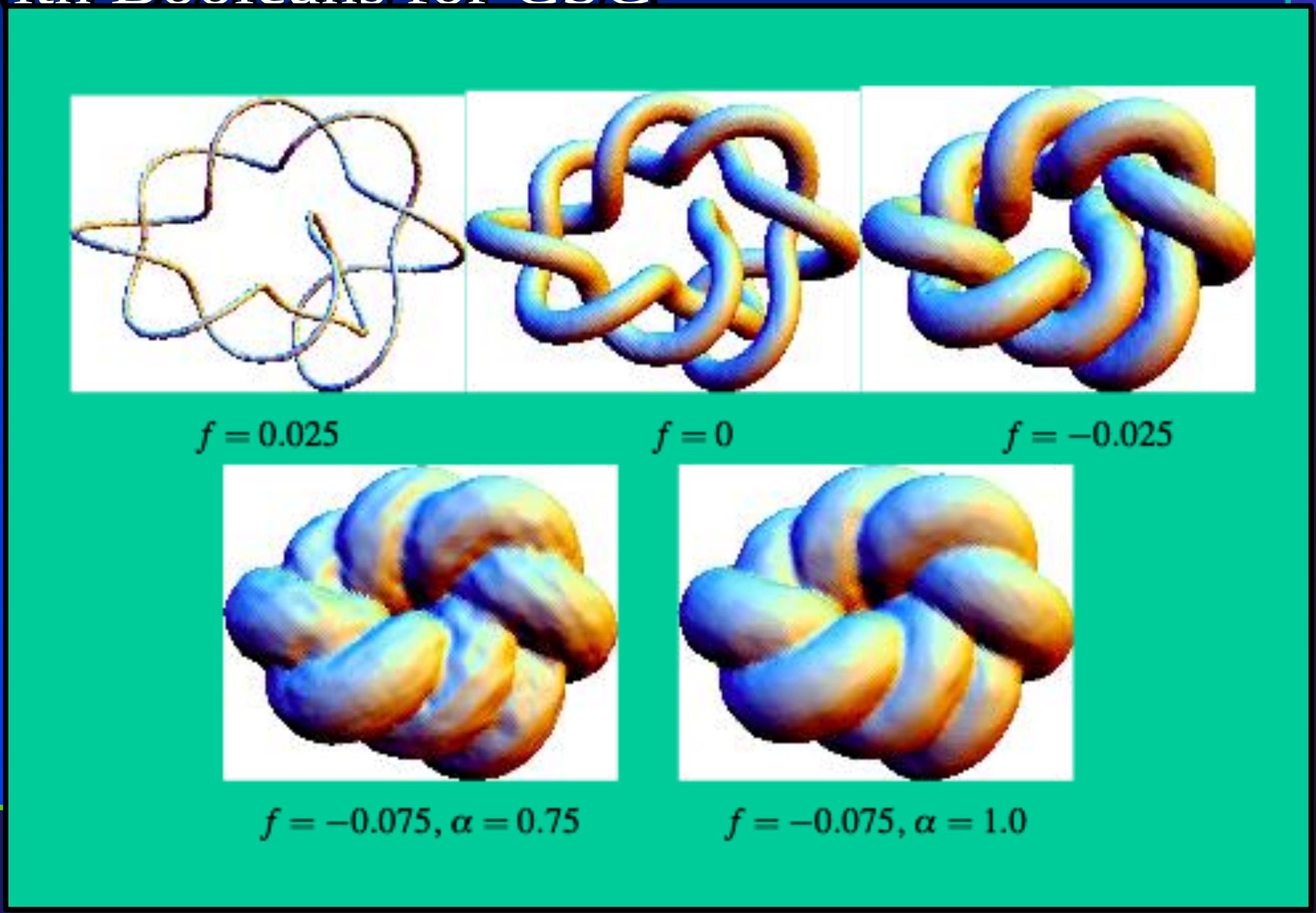
Editing Operations

- Implicit functions can be
  - Combined with booleans for CSG
  - Warped
  - Offset
  - Composed
  - And more...



Editing Operations

- Implicit functions can be
  - Combined with Booleans for CSG
  - Warped
  - Offset
  - Composed
  - And more…

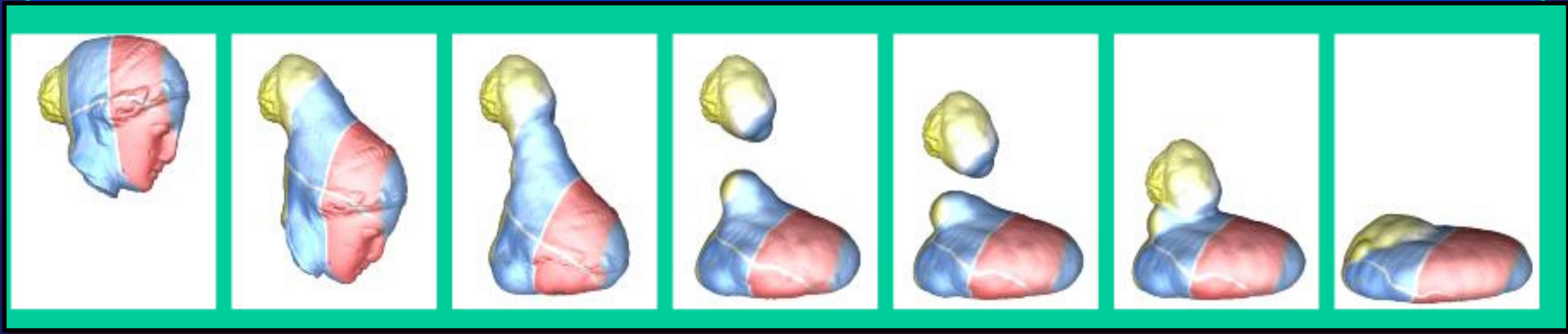


Editing Operations

- Implicit functions can be
  - Combined with Booleans for CSG
  - Warped
  - Offset
  - Composed
  - And more...

Point-based Simulation

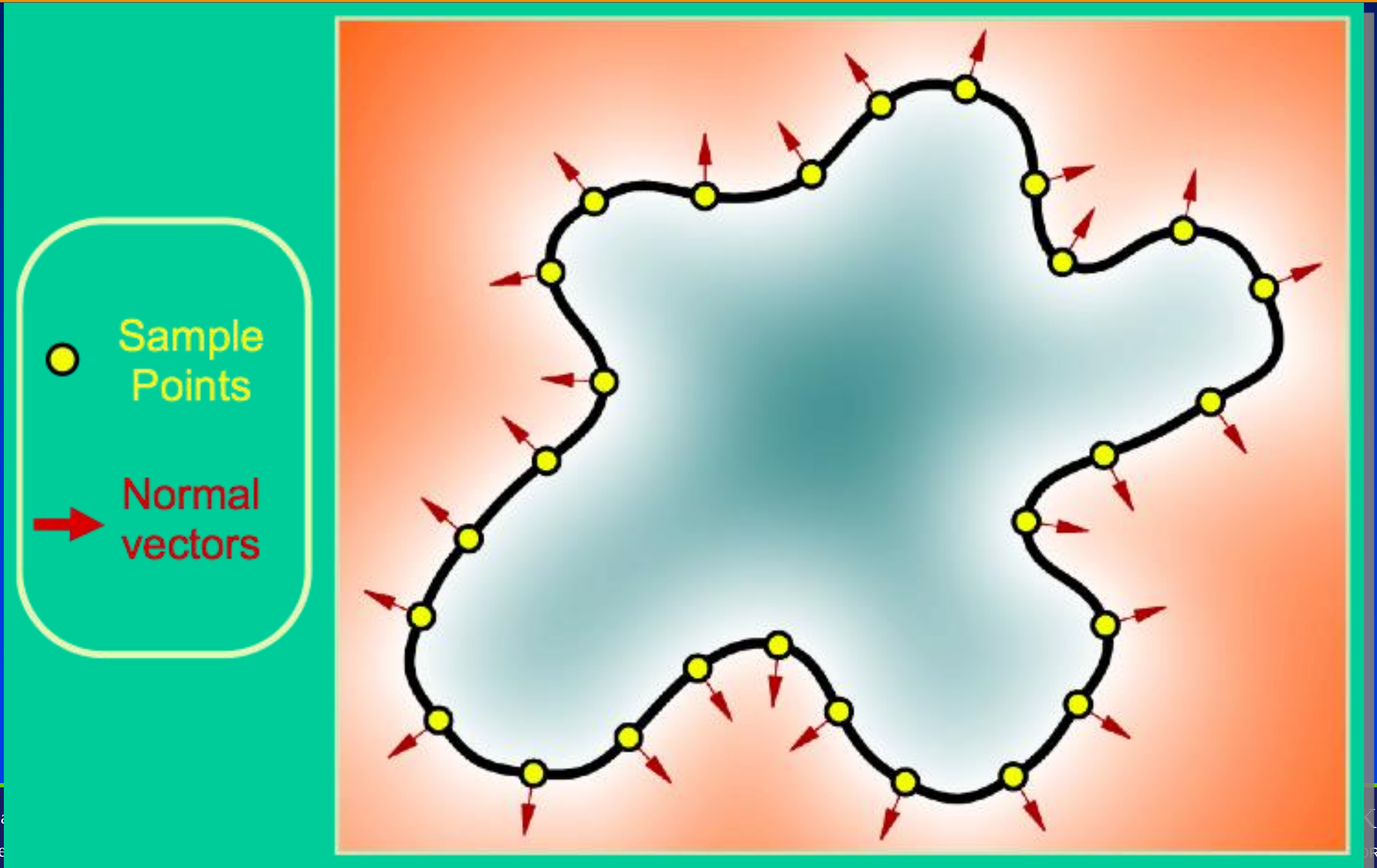
- MLS originated in mechanics literature
- Natural use in graphics for the animation purpose



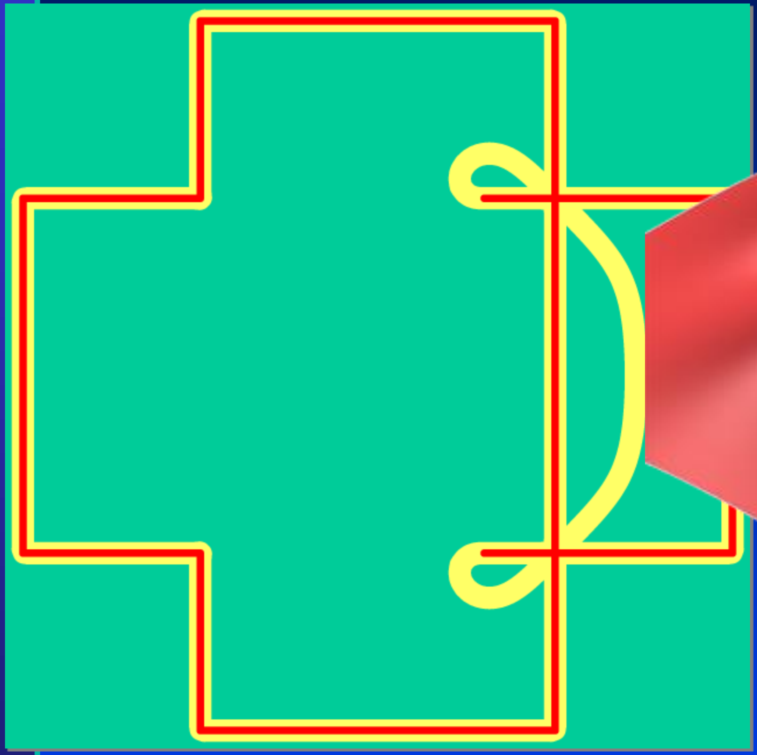
Implicit Moving Least-Squares

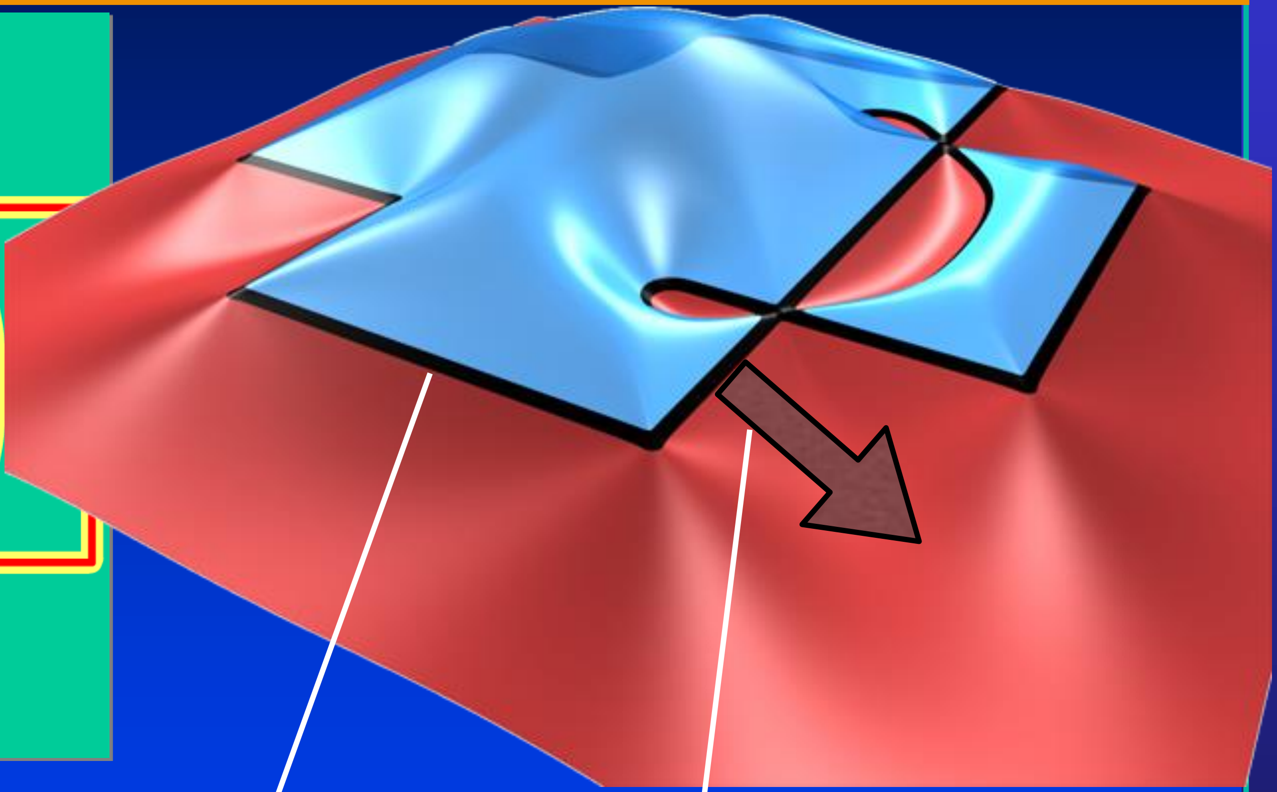
- Shape (surface) is implicitly defined by point cloud
- Define a scalar function that is zero passing through all the points

Implicit Moving Least-Squares



Implicit Moving Least-Squares



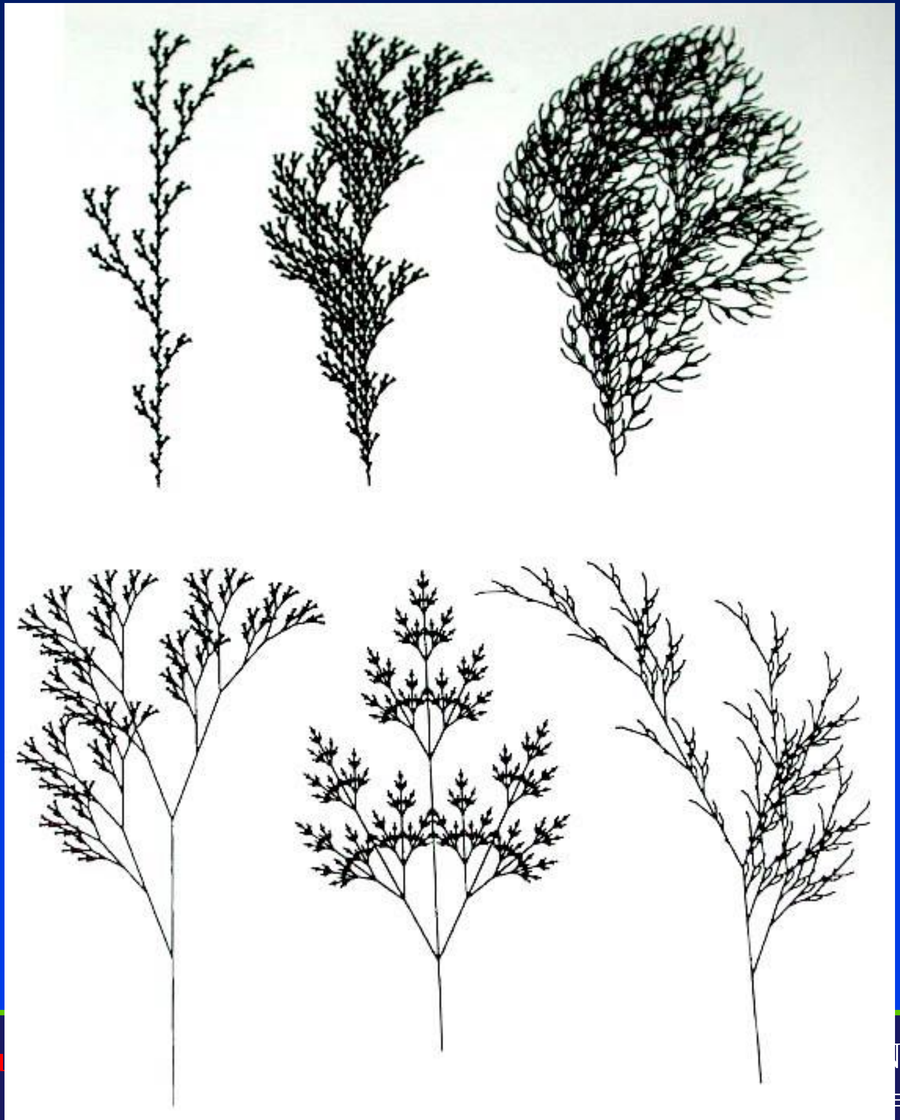


Function is zero on boundary
Decreases in outward direction

Particles

Algorithmic Primitives

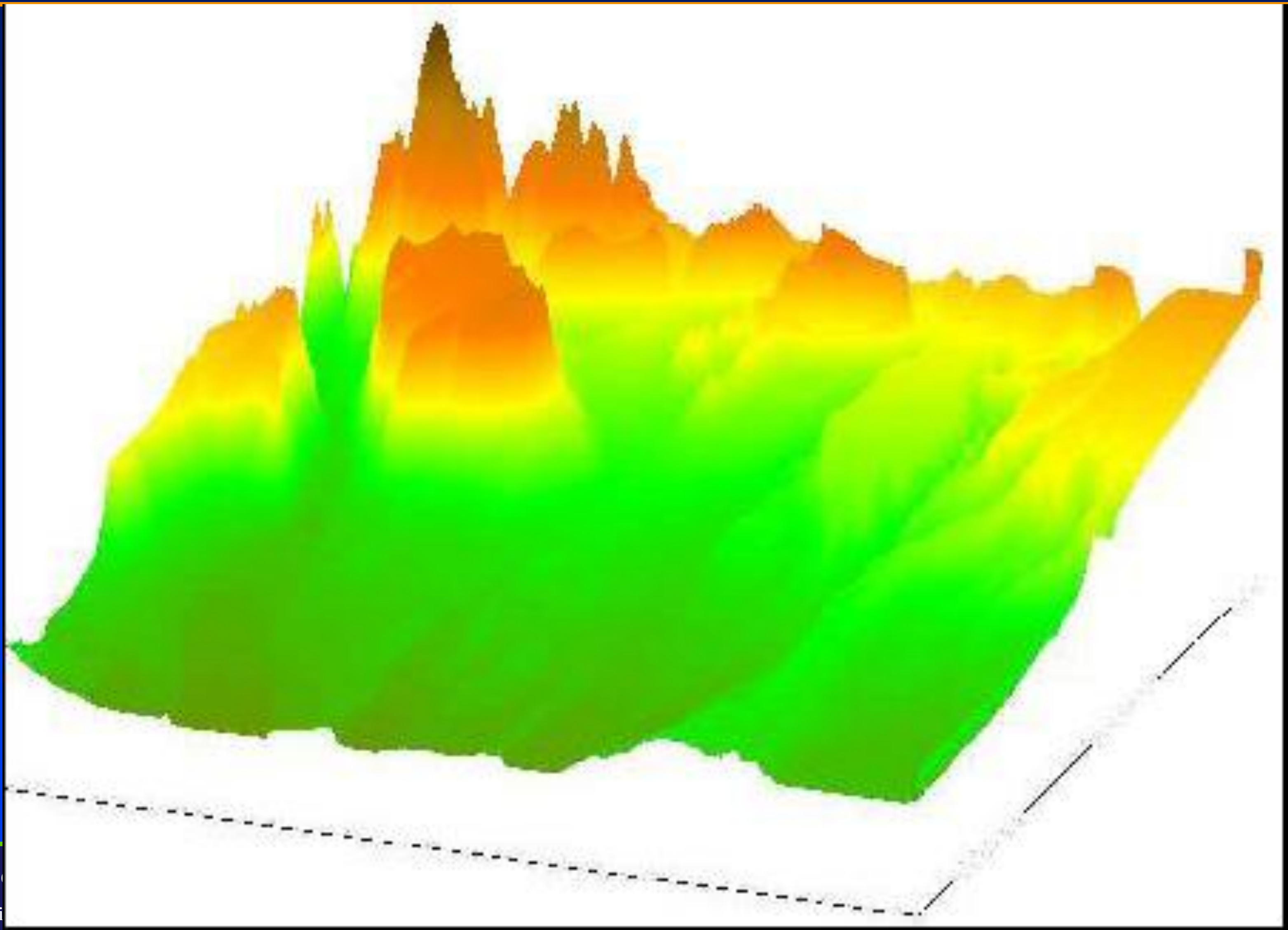
- Algorithms for trees, mountains, grass, fur, lightning, fire, …



Moving Least Squares

- Theory, and
- Techniques

Scattered Data Fitting



Scattered Data Approximation and Interpolation

- Scattered data: an arbitrary set of points in Rd space, and these scattered data carry scalar quantities (i.e., a scalar field in d dimensional parametric space)

Least Squares Approximation

- Commonly-used basis functions include: quadratic, linear, constant terms
- For example:

$$\mathbf{b}(\mathbf{x}) = \begin{bmatrix} 1 & x & y & x^2 & xy & y^2 \end{bmatrix}^T$$

$$\mathbf{b}(\mathbf{x}) = \begin{bmatrix} 1 & x & y & z \end{bmatrix}^T$$

$$\mathbf{b}(\mathbf{x}) = \begin{bmatrix} 1 \end{bmatrix}$$

Least Squares Approximation

- Problem statement: we have n points in Rd space, and we want to obtain a globally defined function f(x) that can approximate the given scalar values at these points in the least-squares senses
- We are considering the space of polynomials of total degree m in d spatial dimensions

$$\min_{f \in P_m^d} \sum_i \left\| \left(f(x_i) - f_i \right\| \right.^2$$

$$f(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \bullet \mathbf{c}$$

$$\mathbf{b}(\mathbf{x}) = \left[b_1(\mathbf{x}) \quad b_2(\mathbf{x}) \quad \ldots \quad b_k(\mathbf{x}) \right]^T$$

$$\mathbf{c} = \left[c_1 \quad c_2 \quad \ldots \quad c_k \right]^T$$

Solution

- Function minimization: the partial derivatives of the error functional must be set to zero
- We now obtain a linear system of equations

$$\frac{\partial E}{\partial \mathbf{c}} = \mathbf{0}$$

$$\sum_i 2b_j(\mathbf{x}_i)\left[\mathbf{b}(\mathbf{x}_i)^T\mathbf{c} - f_i\right] = 0$$

Solution

$$\sum_i \left[\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T \mathbf{c} - \mathbf{b}(\mathbf{x}_i) f_i \right] = \mathbf{0}$$

$$\mathbf{c} = \left[\sum_i \mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T \right]^{-1} \sum_i \mathbf{b}(\mathbf{x}_i) f_i$$

Weighted Least Squares Approximation

- In the weighted least squares formulation, we will have to use a different error functional that now has a weighting function term inside the formulation

$$\min_{f \in P_m^d} \sum_i \theta(\|\bar{\mathbf{x}} - \mathbf{x}_i\|) \|(f(\mathbf{x}_i) - f_i)\|^2$$

Weighting Function Choices

- The weighting function should be locally defined

$$\theta(d) = e^{-\frac{d^2}{h^2}}$$

$$\theta(d) = (1 - d/h)^4 (4d/h + 1)$$

$$\theta(d) = \frac{1}{d^2 + \varepsilon^2}$$

Solution

- Once again, we take partial derivatives of the error functional
- Function minimization: the partial derivatives of the error functional must be set to zero
- We now obtain a linear system of equations

$$\frac{\partial E}{\partial \mathbf{c}(\bar{\mathbf{x}})} = \mathbf{0}$$

$$\sum_i \theta(d_i)\, 2 b_j(\mathbf{x}_i)\left[\mathbf{b}(\mathbf{x}_i)^T \mathbf{c}(\bar{\mathbf{x}}) - f_i\right] = 0$$

Solution

- The weighting functions participate in the solution
- Note that, this solution is actually locally meaningful, and it is applicable in a small neighborhood

$$\sum_i \left[\theta(d_i)\mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T \mathbf{c}(\bar{\mathbf{x}}) - \theta(d_i)\mathbf{b}(\mathbf{x}_i) f_i\right] = \mathbf{0}$$

$$\mathbf{c}(\bar{\mathbf{x}}) = \left[\theta(d_i)\sum_i \mathbf{b}(\mathbf{x}_i)\mathbf{b}(\mathbf{x}_i)^T\right]^{-1} \sum_i \theta(d_i)\mathbf{b}(\mathbf{x}_i) f_i$$

Global Approximation

- The concept of Partition-of-Unity (POU)

$$\varphi_j(x) = \frac{\theta_j(x)}{\sum_1^n \theta_i(x)}$$

$$f(\mathbf{x}) = \sum_j \varphi_j(\mathbf{x})\mathbf{b}(\mathbf{x})^T \mathbf{c}(\bar{\mathbf{x}})$$

Moving Lease Squares

- Moving Least Squares Approximants

$$f(x) = \sum_i \phi_i(x) f_i = \sum_j b_j(x) c_j(x)$$

$$\min_c \sum_i \theta(\|\mathbf{x} - \mathbf{x}_i\|) \left\| (\mathbf{b}(\mathbf{x}_i)^T \mathbf{c}(\mathbf{x}) - f_i) \right\|^2$$

MLS Basis Functions

$$\phi_i(\mathbf{x}) = \mathbf{b}(\mathbf{x})^T \mathbf{A}(\mathbf{x})^{-1} \mathbf{B}_i(\mathbf{x})$$

$$\mathbf{A}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i(\mathbf{x}) \mathbf{b}(\mathbf{x}_i) \mathbf{b}(\mathbf{x}_i)^T$$

$$\mathbf{B}(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x})\mathbf{b}(\mathbf{x}_1) & \theta_2(\mathbf{x})\mathbf{b}(\mathbf{x}_2) & \ldots\ldots & \theta_n(\mathbf{x})\mathbf{b}(\mathbf{x}_n) \end{bmatrix}$$

Weighing Functions

- A cubic spline weight function is a good choice

STONY BROOK
STATE UNIVERSITY OF NEW YORK

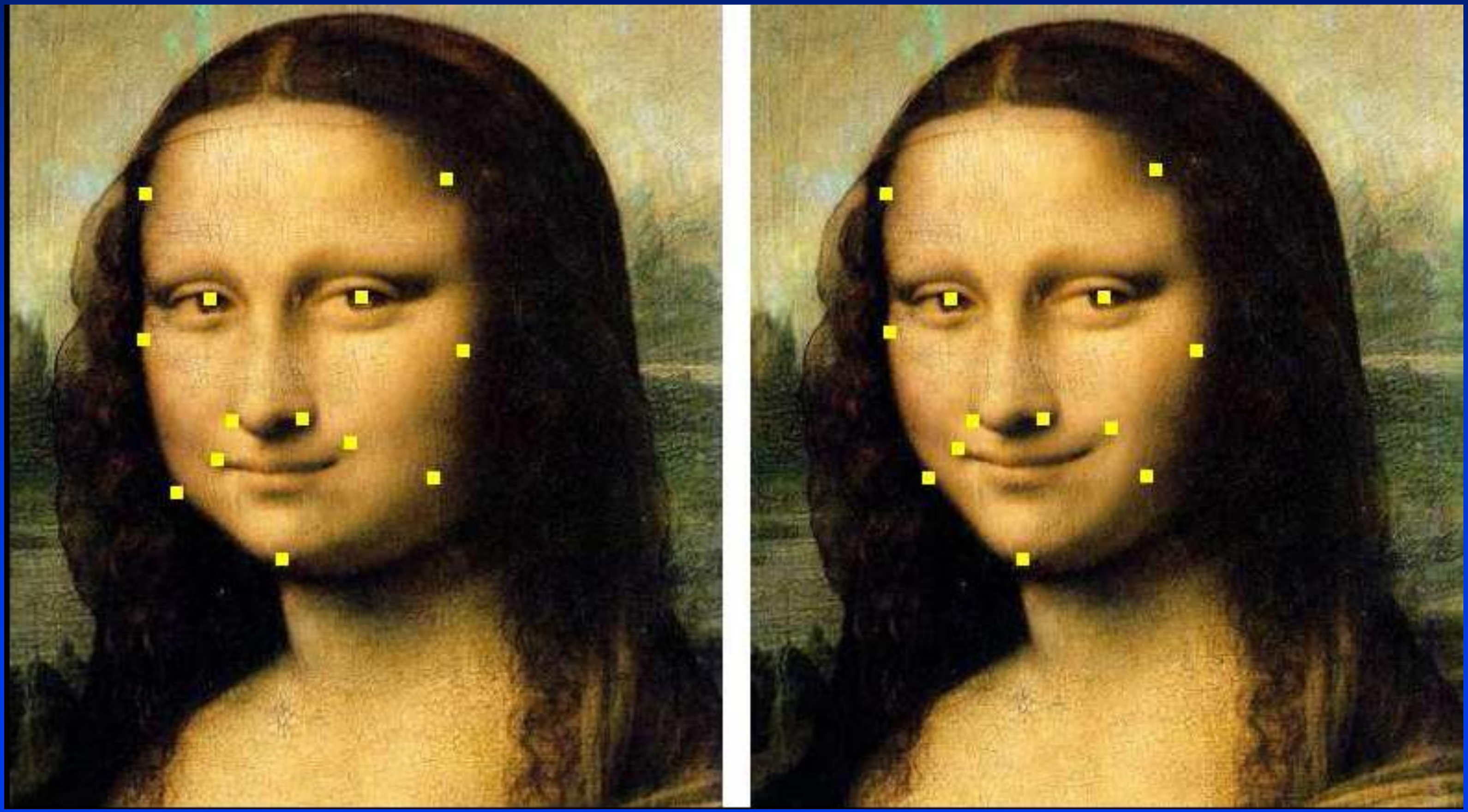
Partition of Unity

- When b is a constant term, MLS basis functions reduce to partition-of-unity basis functions for all the weighting functions

Other Applications

- In addition to point-based Graphics (discussed earlier), MLS is a widespread and very powerful tool in Graphics, with many applications
- Surface reconstruction from points
- Interpolating or approximating implicit surfaces
- Simulation
- Animation
- Partition of Unity
- Physics-based modeling, simulation, and animation

Image Editing



Surface Reconstruction