GAMES
Point-Based Representation

Qixing Huang

August 19th 2021

Acknowledgement

- The first part of the slides adopted from

Point-Based Computer Graphics
SIGGRAPH 2004 Course Notes

Marc Alexa, Darmstadt University of Technology
Markus Gross, ETH Zurich
Mark Pauly, Stanford University
Hanspeter Pfister, Mitsubishi Electric Research Laboratories
Marc Stamminger, University of Erlangen-Nuremberg
Matthias Zwicker, Massachusetts Institute of Technology

Goal

- Learn traditional stuff which will be useful for developing point-based neural networks

Tzu-Mao Li @tzumaoli · Aug 13

Read the books those papers cited. Learn the basics. Learn Monte Carlo methods. Learn finite element methods. Learn differential geometry. Learn differential equations. Learn how compilers work. Learn linear algebra. Don't say "it's irrelevant". It's usually relevant.

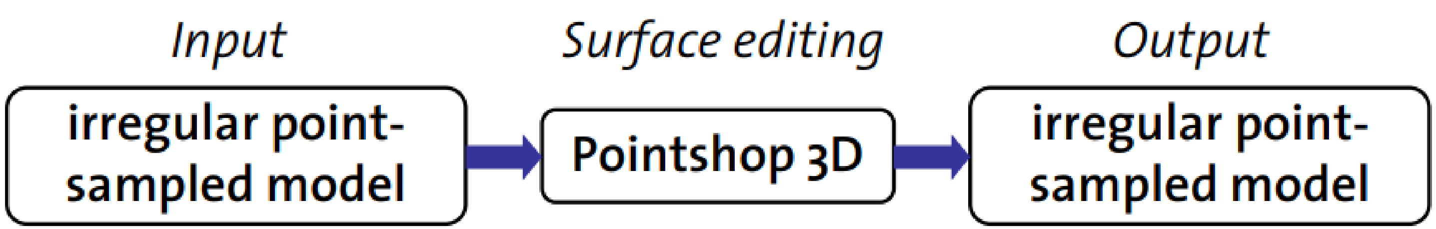
Pointshop 3D

Overview

- Introduction
- Pointshop3D System Components
  - Point Cloud Parameterization
  - Resampling Scheme
  - Editing Operations

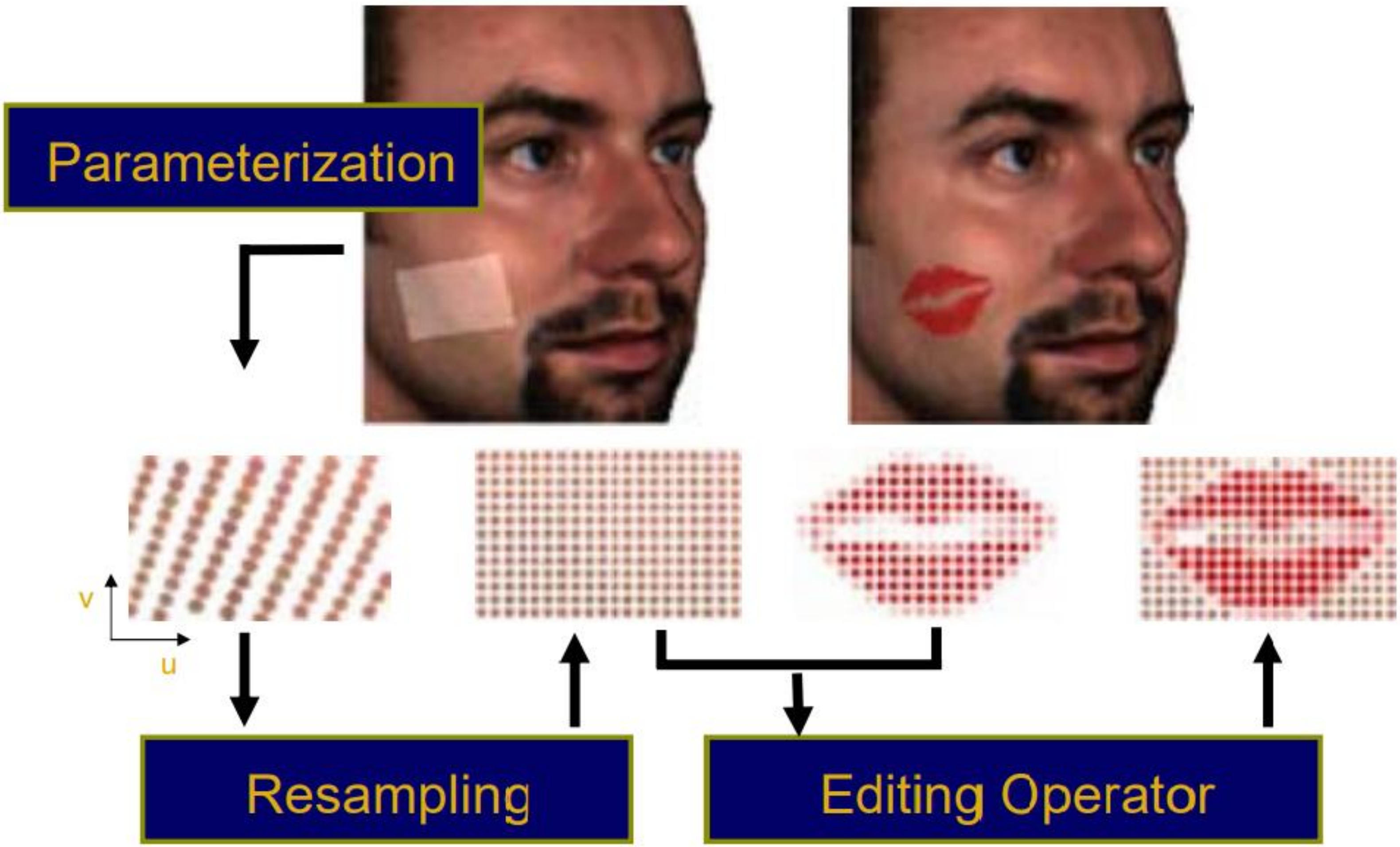
Pointshop 3D

- Interactive system for point-based surface editing
- Generalize 2D photo editing concepts and functionality to 3D point-sampled surfaces
- Use 3D surface pixels (*surfels*) as versatile display and modeling primitive



- Does not require intermediate triangulation

Concept



Key Components

- Point cloud parameterization Φ
  - brings surface and brush into common reference frame
- Dynamic resampling Ψ
  - creates one-to-one correspondence of surface and brush samples
- Editing operator Ω
  - combines surface and brush samples

$$\underbrace{S'}_{\text{modified surface}} = \Omega(\Psi(\Phi(\underbrace{S}_{\text{original surface}})), \Psi(\underbrace{B}_{\text{brush}}))$$

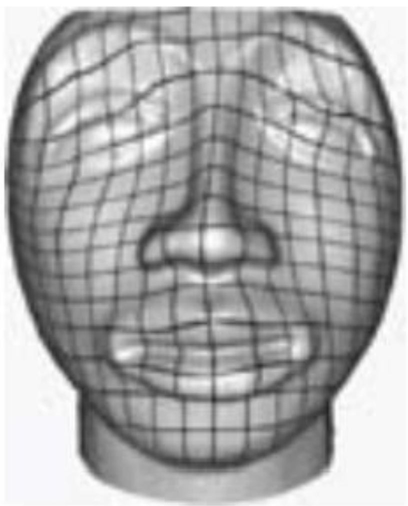
Parameterization

- Constrained minimum distortion parameterization of point clouds

$$\mathbf{u} \in [0,1]^2 \Rightarrow X(\mathbf{u}) = \begin{bmatrix} x(\mathbf{u}) \\ y(\mathbf{u}) \\ z(\mathbf{u}) \end{bmatrix} = \mathbf{x} \in P \subset R^3$$

Parameterization

contraints = matching
of feature points



minimum distortion =
maximum smoothness

Parameterization

- Find mapping X that minimizes objective function:

$$C(X) = \sum_{j \in M} \underbrace{(X(\mathbf{p}_j) - \mathbf{x}_j)^2}_{\text{fitting constraints}} + \underbrace{\varepsilon \int_P \gamma(\mathbf{u}) d\mathbf{u}}_{\text{distortion}}$$

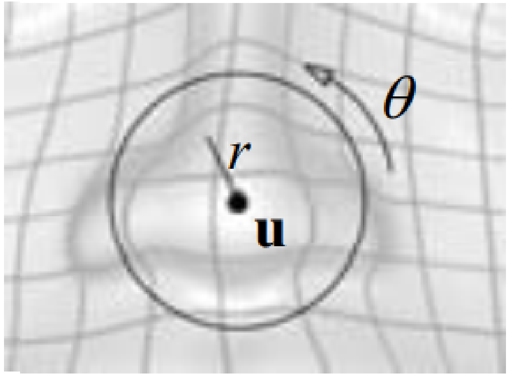
brush points

surface points

Parameterization

- Measuring distortion

$$\gamma(\mathbf{u}) = \int_{\theta} \left(\frac{\partial^2}{\partial r^2} X_{\mathbf{u}}(\theta, r) \right)^2 d\theta$$



  - Integrates squared curvature using local polar re-parameterization

$$X_{\mathbf{u}}(\theta, r) = X\left(\mathbf{u} + r \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \right)$$

Parameterization

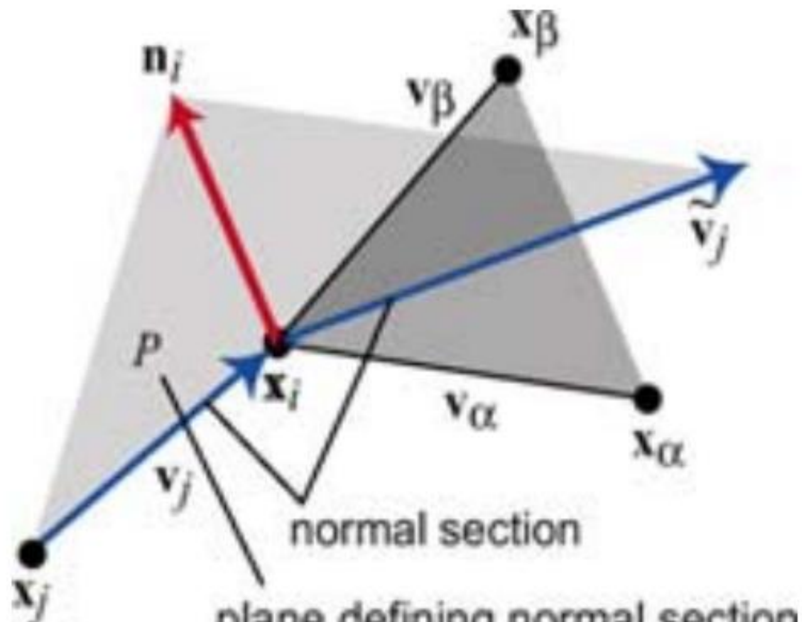
- Discrete formulation:

$$\widetilde{C}(U) = \sum_{j \in M} (\mathbf{p}_j - \mathbf{u}_j)^2 + \varepsilon \sum_{i=1}^{n} \sum_{j \in N_i} \left(\frac{\partial U(\mathbf{x}_i)}{\partial \mathbf{v}_j} - \frac{\partial U(\mathbf{x}_i)}{\partial \widetilde{\mathbf{v}}_j} \right)^2$$

  – Approximation: mapping is piecewise linear

Parameterization

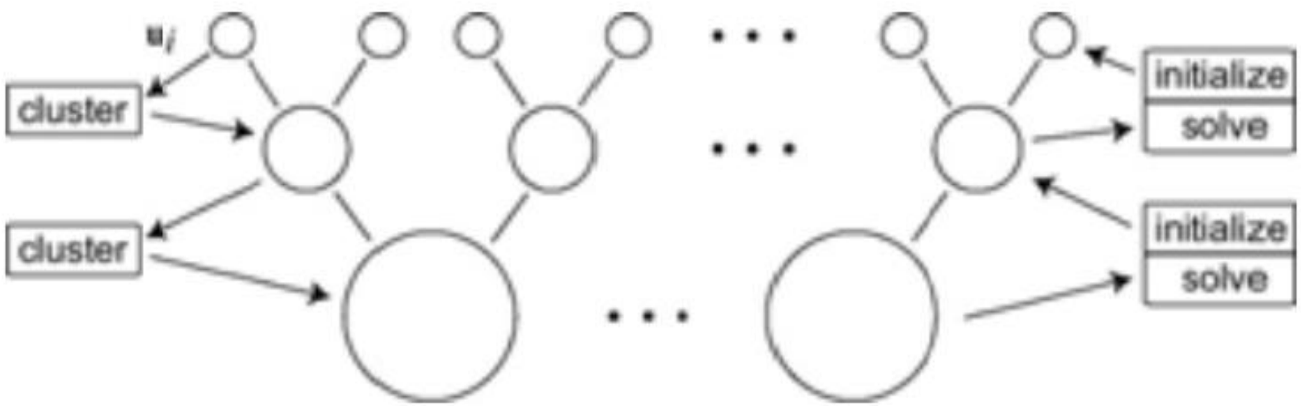
- Directional derivatives as extension of divided differences based on k-nearest neighbors



Parameterization

- HIerarchical solver for efficient computation of resulting sparse linear least squares problem

$$\widetilde{C}(U) = \sum_{j} \left(\mathbf{b}_j - \sum_{i=1}^{n} a_{j,i} \mathbf{u}_i \right)^2 = \left\| \mathbf{b} - A\mathbf{u} \right\|^2$$



Reconstruction

- Parameterized scattered data approximation

fitting functions

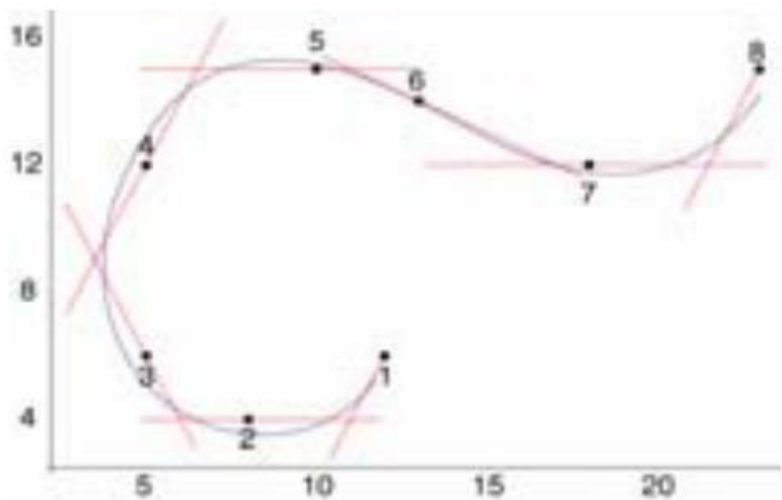
weight functions

$$X(\mathbf{u}) = \frac{\sum_i \Phi_i(\mathbf{u}) r_i(\mathbf{u})}{\sum_i r_i(\mathbf{u})}$$

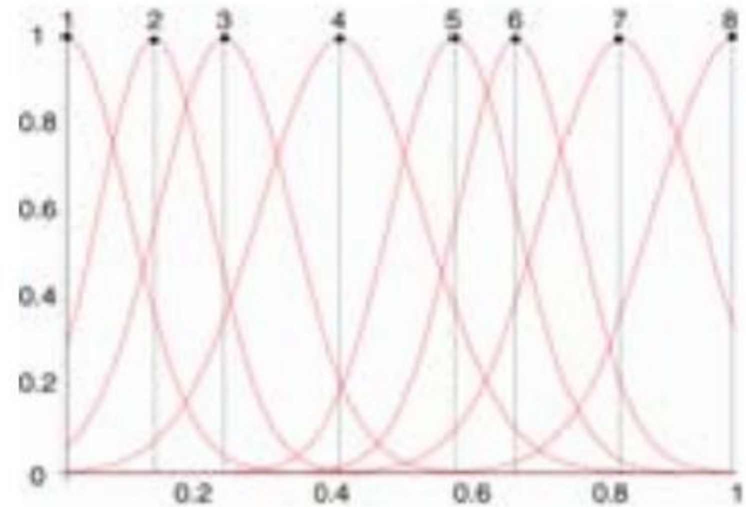
normalization

- Fitting functions
  - Compute local fitting functions using local parameterizations
  - Map to global parameterization using global parameter coordinates of neighboring points

Reconstruction



reconstruction with
linear fitting functions



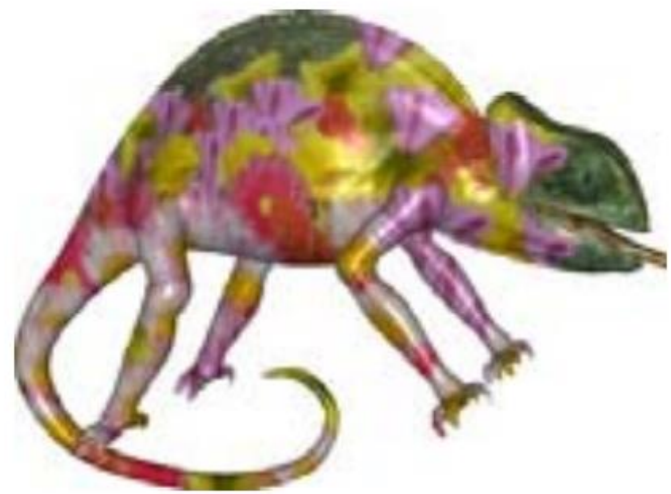
weight functions in
parameter space

Reconstruction

- Reconstruction with linear fitting functions is equivalent to surface splatting!
  - Use the surface splatting renderer to reconstruct our surface function (see chapter on rendering)
- Provides:
    - Fast evaluation
    - Anti-aliasing (Band-limit the weight functions before sampling using Gaussian low-pass filter)
  - Distortions of splats due to parameterization can be computed efficiently using local affine mappings

Editing Operators

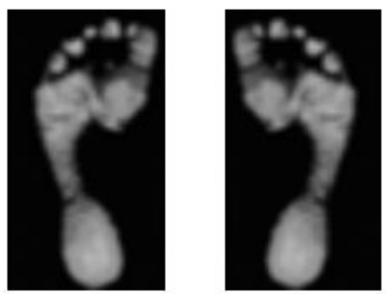
- Painting
  - Texture, material properties, transparency



Editing Operators

- Sculpting
  - Carving, normal displacement

texture map

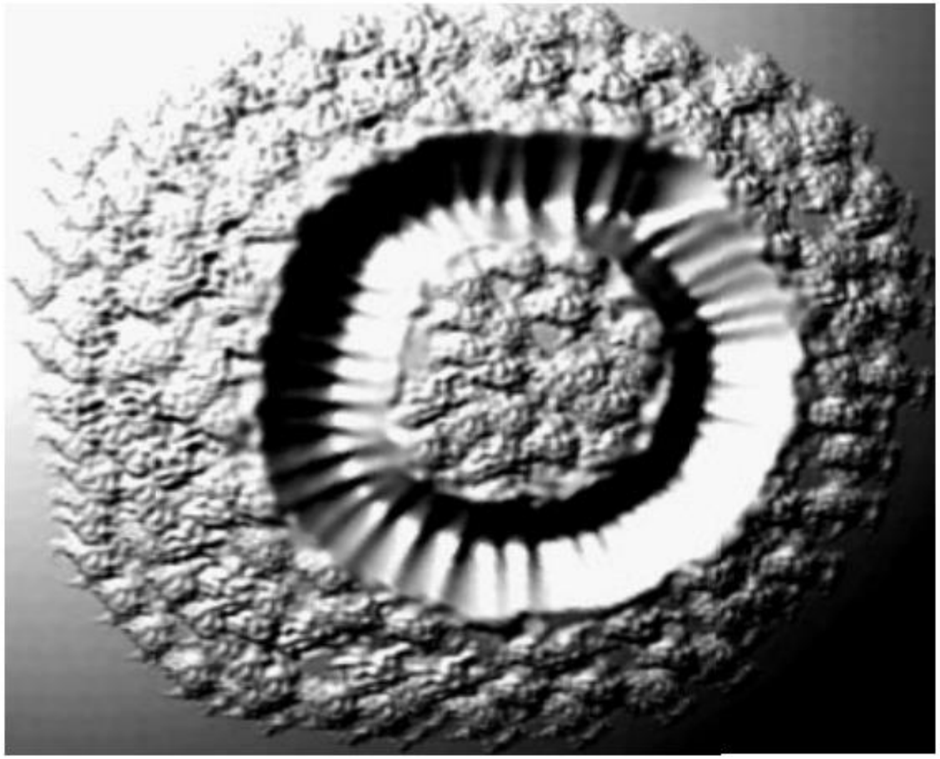


displacement maps

carved and texture mapped point-sampled surface

Editing Operators

- Engraving surface detail

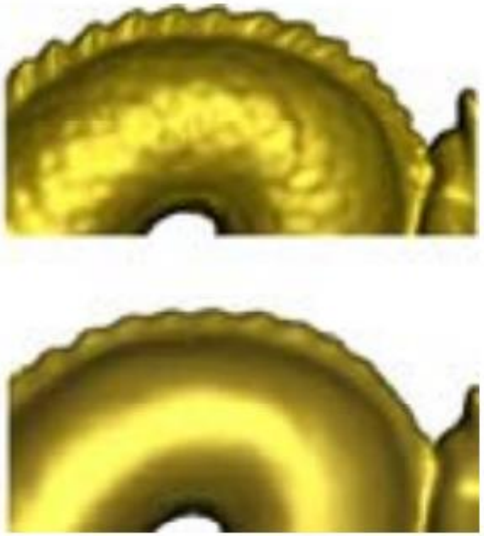


Editing Operators

- Filtering appearance and geometry

Editing Operators

- Filtering appearance and geometry
  - Scalar attributes, geometry



Advanced Processing

- Multiscale feature extraction

Point-Cloud Simplification

Overview

- Introduction
- Local surface analysis
- Simplification methods
- Error measurement
- Comparison

Introduction

- Point-based models are often sampled very densely
- Many applications require coarser approximations, e.g. for efficient
  - Storage
  - Transmission
  - Processing
  - Rendering
- We need simplification methods for reducing the complexity of point-based surfaces

Introduction

- Example: Level-of-detail (LOD) rendering

10k 20k 60k 200k 2000k

Introduction

- Different simplification methods from triangle meshes to point clouds:
  - Hierarchical clustering
  - Iterative simplification
  - Particle simulation

- Each method has its pros and cons

- We will talk about mesh simplifications later

Local Surface Analysis

- Cloud of point samples describes underlying (manifold) surface
- We need:
  - Mechanisms for locally approximating the surface -> MLS approach
  - Fast estimation of tangent plane and curvature -> principal component analysis of local neighborhood

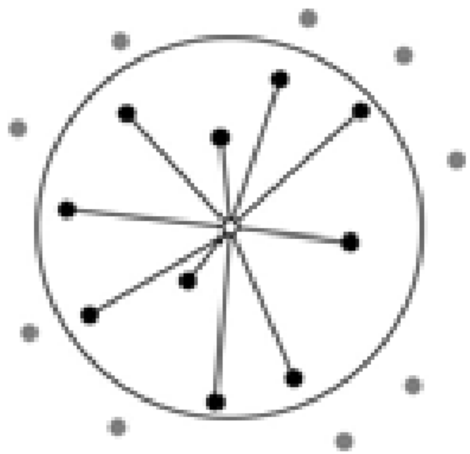
Neighborhood

- Replace geodesic proximity with spatial proximity (requires sufficiently high sampling density!)

- Compute neighborhood according to Euclidean distance

Neighborhood

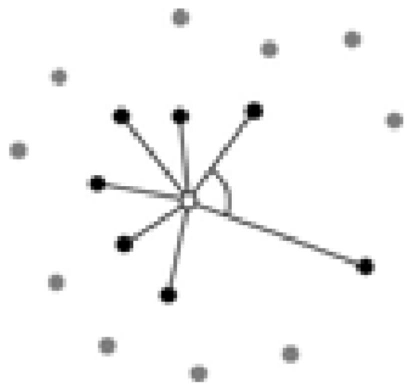
- K-nearest neighbors



- Can be quickly computed using spatial data-structures (e.g. kd-tree, octree, bsp-tree)
- Requires isotropic point distribution

Neighborhood

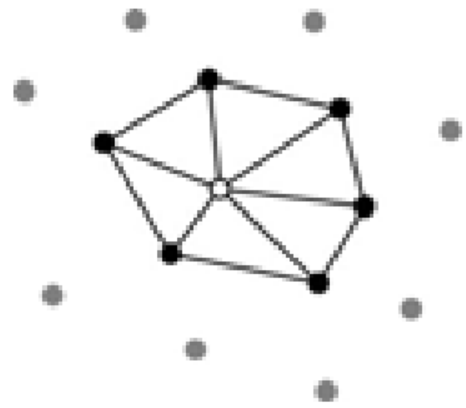
- Improvement: Angle criterion (Linsen)



  – Project points onto tangent plane
  – Sort neighbors according to angle
  – Include more points if angle between subsequent points is above some threshold

Neighborhood

- Local Delaunay triangulation (Floater)



  - Project points into tangent plane
  - Compute local Voronoi diagram

Covariance Analysis

- Covariance matrix of local neighborhood N:

$$\mathbf{C} = \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \cdots \\ \mathbf{p}_{i_n} - \bar{\mathbf{p}} \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{p}_{i_1} - \bar{\mathbf{p}} \\ \cdots \\ \mathbf{p}_{i_n} - \bar{\mathbf{p}} \end{bmatrix}, \qquad i_j \in N$$

- with centroid $\bar{\mathbf{p}} = \frac{1}{|N|} \sum_{i \in N} \mathbf{p}_i$

Covariance Analysis

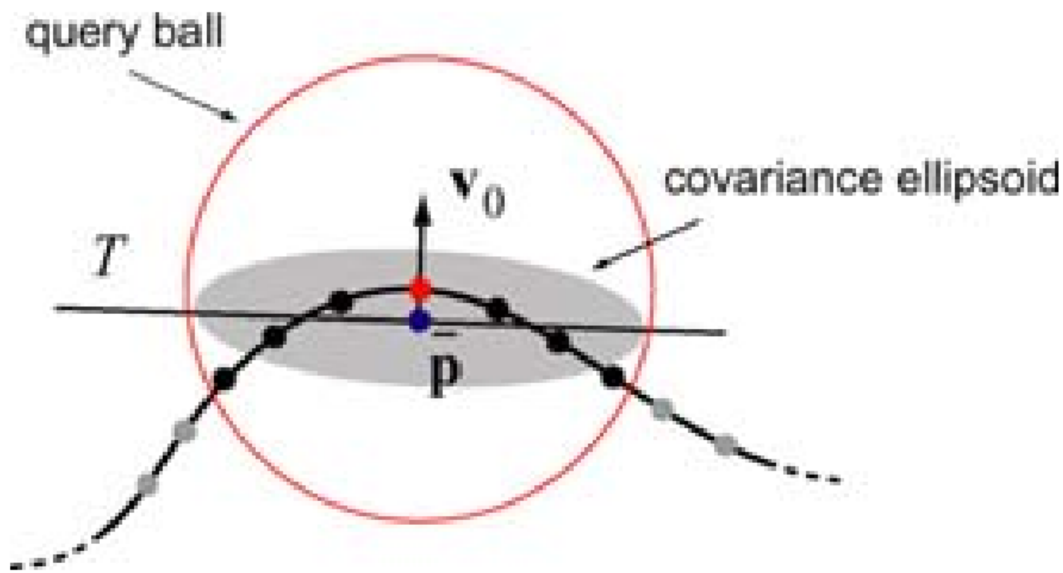
- Consider the eigenproblem:

$$\mathbf{C} \cdot \mathbf{v}_l = \lambda_l \cdot \mathbf{v}_l, \qquad l \in \{0,1,2\}$$

- C is a 3x3, positive semi-definite matrix
  - ⇨ All eigenvalues are real-valued
  - ⇨ The eigenvector with smallest eigenvalue defines the least-squares plane through the points in the neighborhood, i.e. approximates the surface normal

Covariance Analysis

- Covariance ellipsoid spanned by the eigenvectors scaled with corresponding eigenvalue



Covariance Analysis

- The total variation is given as:

$$\sum_{i \in N} \left| \mathbf{p}_i - \overline{\mathbf{p}} \right|^2 = \lambda_0 + \lambda_1 + \lambda_2$$

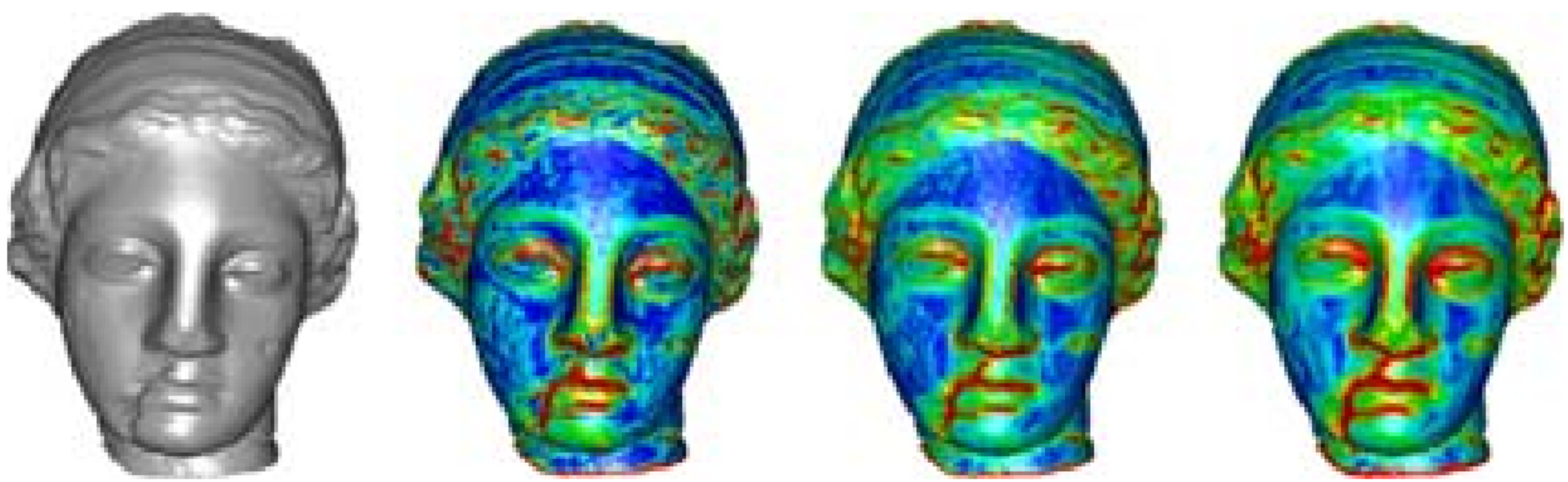
- We define surface variation as:

$$\sigma_n(\mathbf{p}) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \qquad \lambda_0 \leq \lambda_1 \leq \lambda_2$$

  - Measures the fraction of variation along the surface normal, i.e. quantifies how strong the surface deviates from the tangent plane ⇨ estimate for curvature

Covariance Analysis

- Comparison with curvature:



original | mean curvature | variation n=20 | variation n=50

Surface Simplification

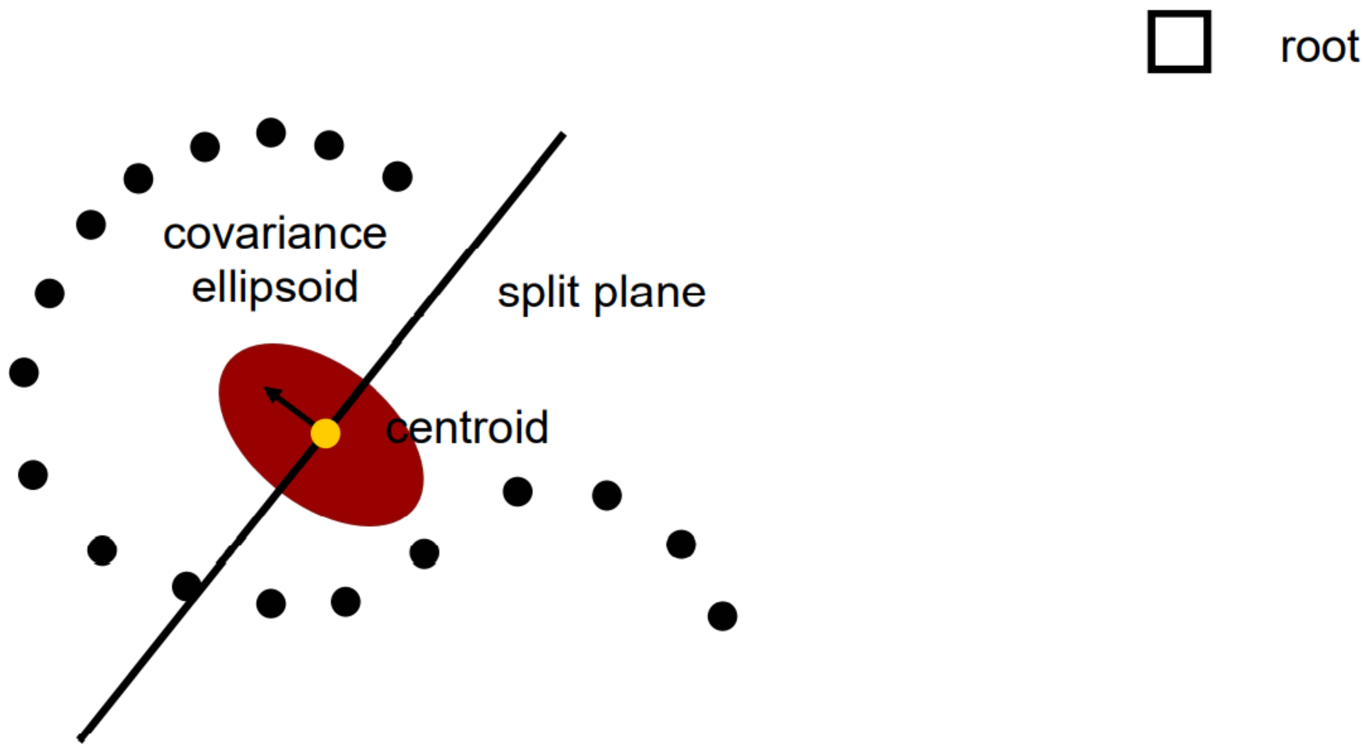
- Hierarchical clustering
- Iterative simplification
- Particle simulation

Hierarchical Clustering

- Top-down approach using binary space partition
- Split the point cloud if:
  - Size is larger than user-specified maximum or
  - Surface variation is above maximum threshold
- Split plane defined by centroid and axis of greatest variation (= eigenvector of covariance matrix with largest associated eigenvector)

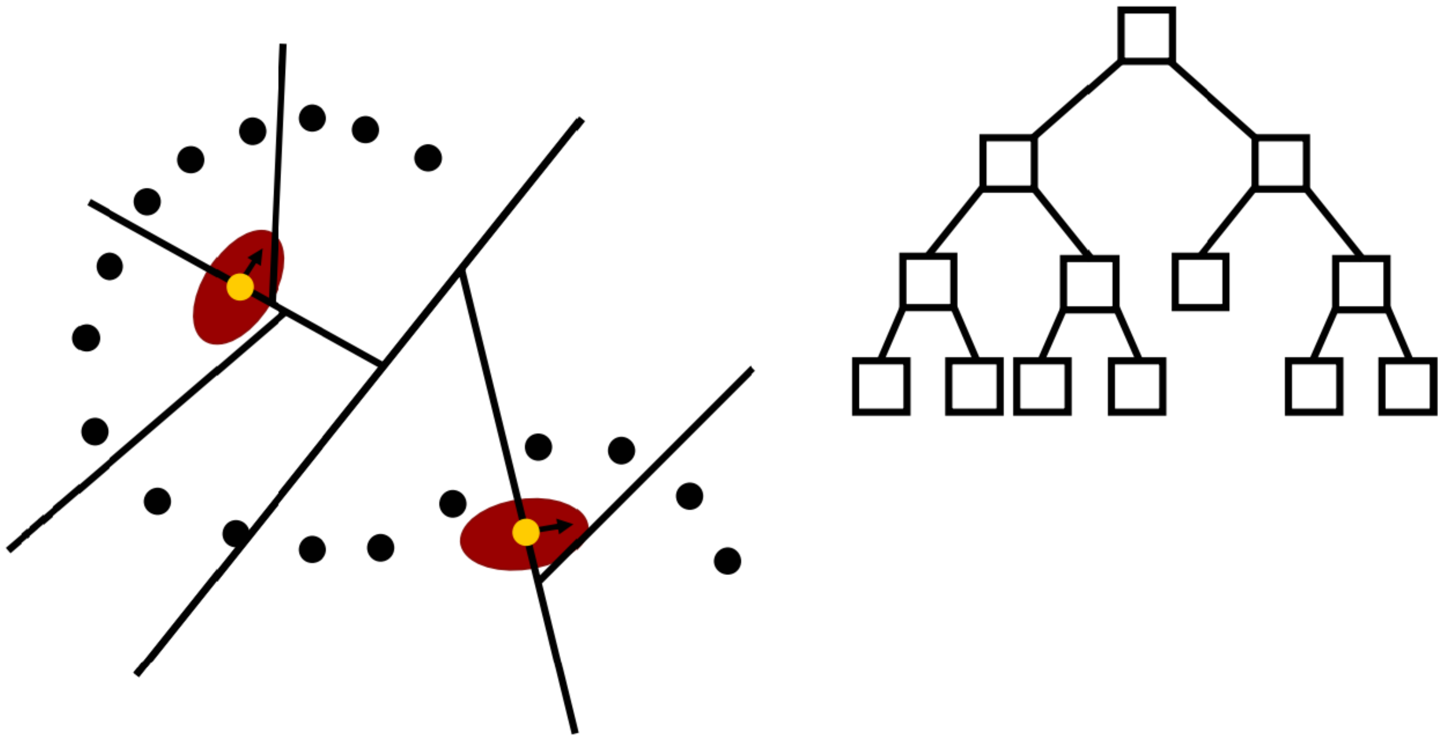
- Leaf nodes of the tree correspond to clusters
- Replace clusters by centroid

Hierarchical Clustering



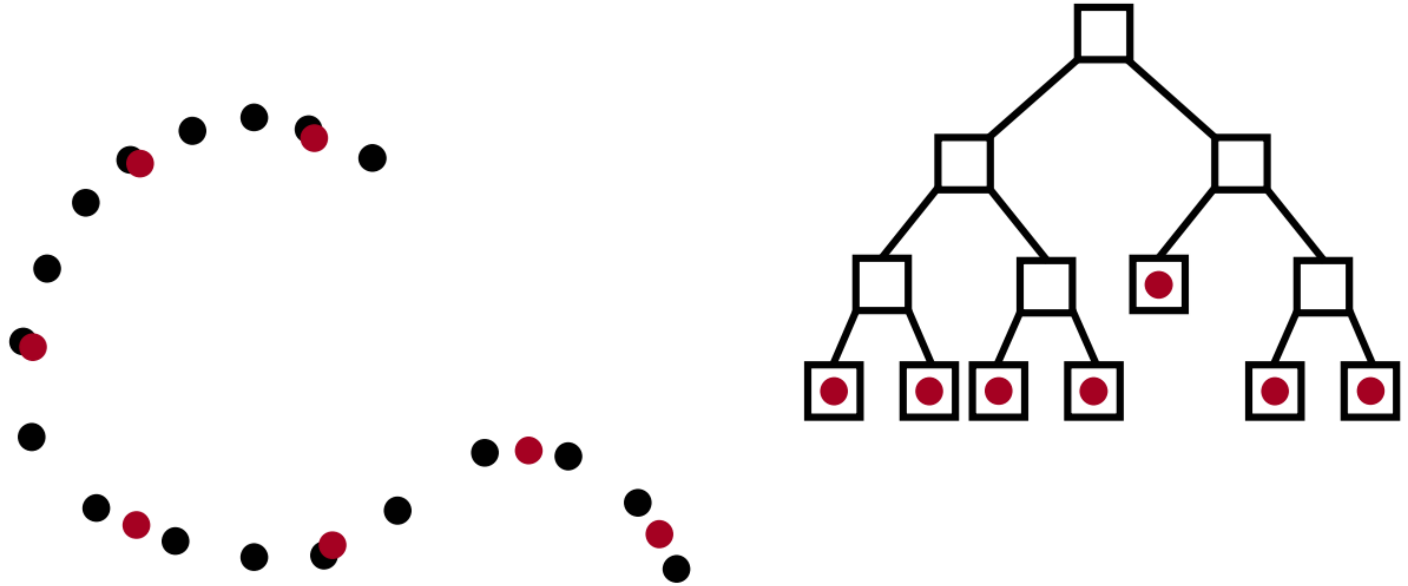
Hierarchical Clustering

- 2D example



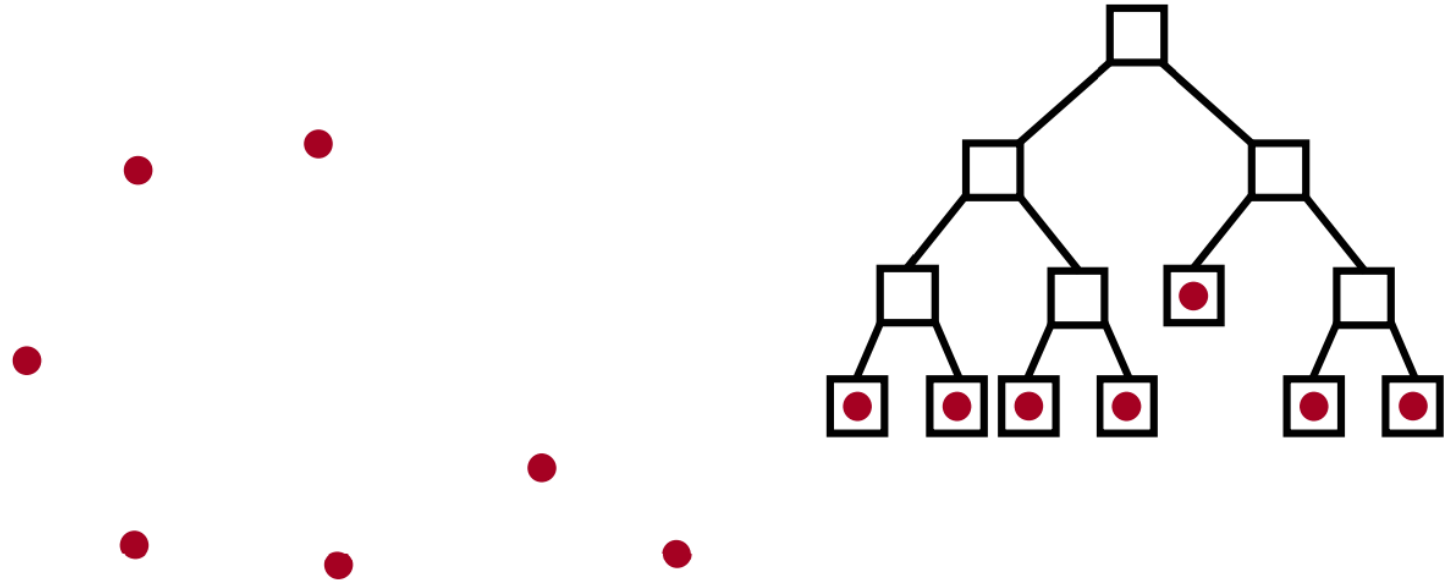
Hierarchical Clustering

- 2D example



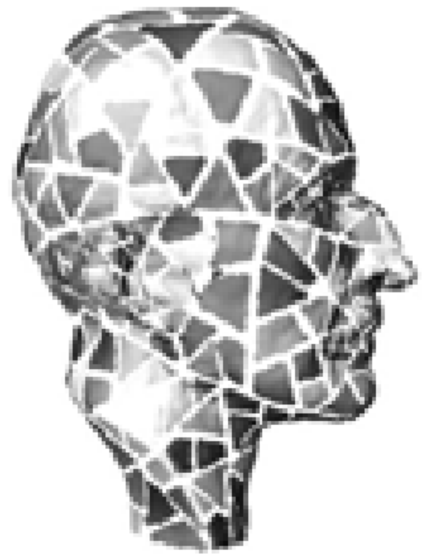
Hierarchical Clustering

- 2D example



Hierarchical Clustering

43 Clusters

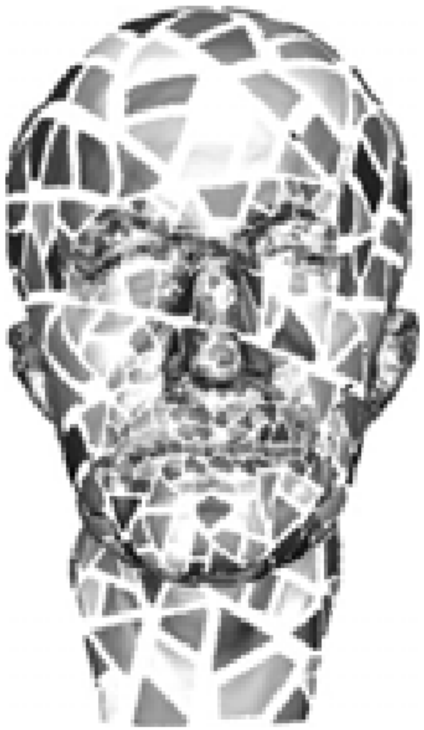


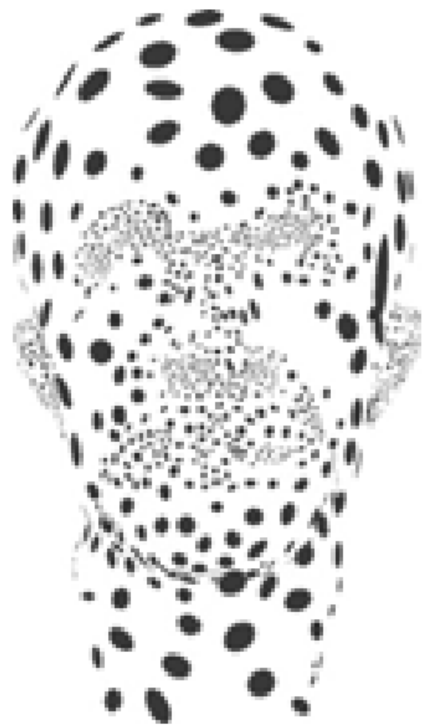
436 Clusters

4,280 Clusters

Hierarchical Clustering

- Adaptive Clustering



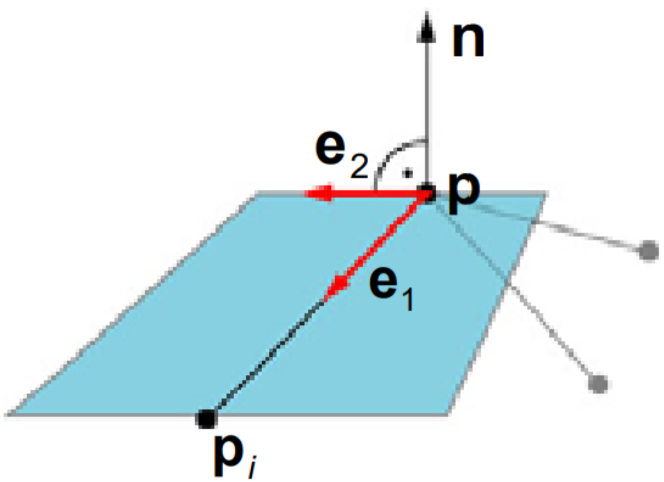


Iterative Simplification

- Iteratively contracts point pairs
  - Each contraction reduces the number of points by one
- Contractions are arranged in priority queue according to quadric error metric (Garland and Heckbert)
- Quadric measures cost of contraction and determines optimal position for contracted sample
- Equivalent to QSlim except for definition of approximating planes

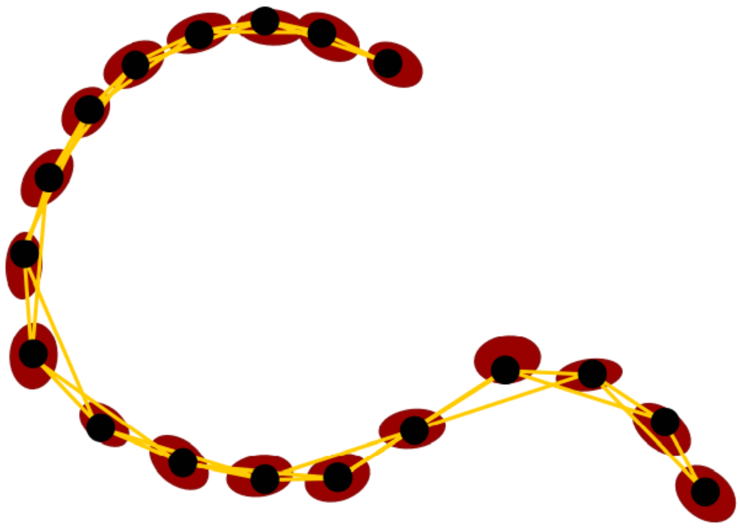
Iterative Simplification

- Quadric measures the squared distance to a set of planes defined over *edges* of neighborhood
  - plane spanned by vectors $\mathbf{e}_1 = \mathbf{p}_i - \mathbf{p}$ and $\mathbf{e}_2 = \mathbf{e}_1 \times \mathbf{n}$



Iterative Simplification

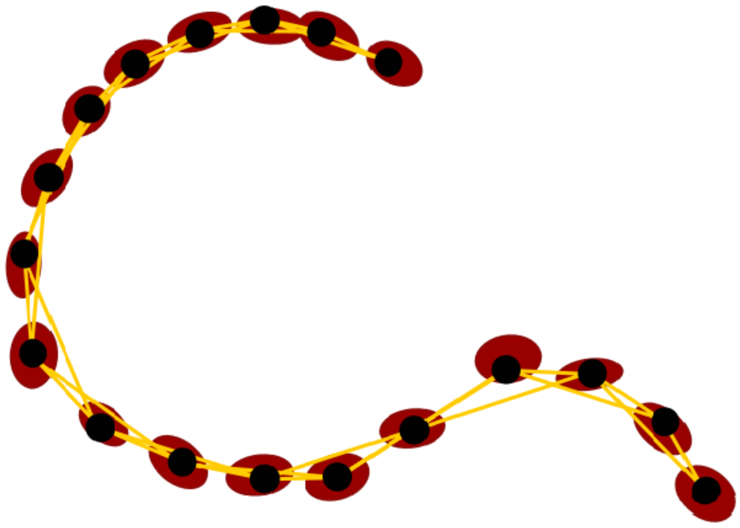
- 2D example



- Compute initial point-pair contraction candidates
- Compute fundamental quadrics
- Compute edge costs

Iterative Simplification

- 2D example

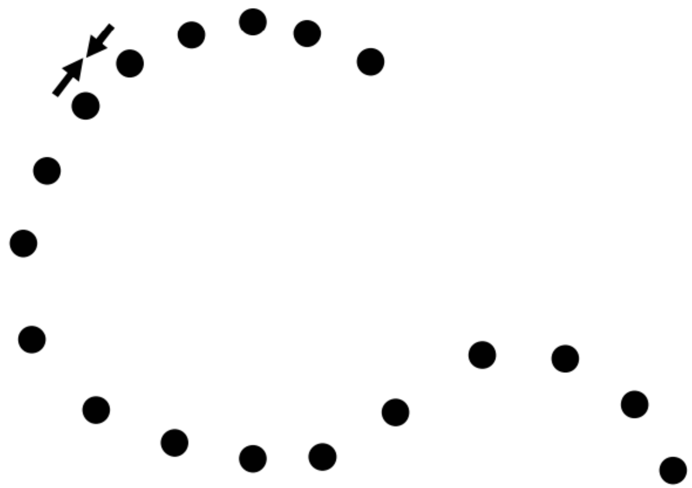


priority queue

edge	cost
6	0.02
2	0.03
14	0.04
5	0.04
9	0.09
1	0.11
13	0.13
3	0.22
11	0.27
10	0.36
7	0.44
4	0.56

Iterative Simplification

- 2D example

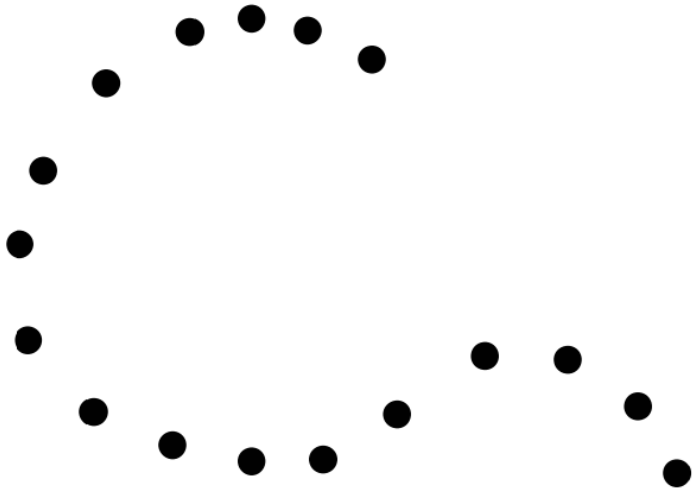


priority queue

edge	cost
6	0.02
2	0.03
14	0.04
5	0.04
9	0.09
1	0.11
13	0.13
3	0.22
11	0.27
10	0.36
7	0.44
4	0.56

Iterative Simplification

- 2D example

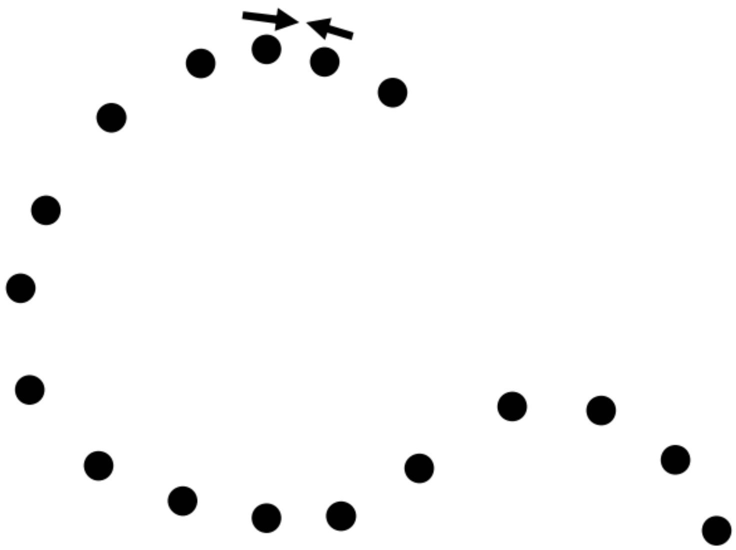


priority queue

edge	cost
6	0.02
2	0.03
14	0.04
5	0.06
9	0.09
1	0.11
13	0.13
3	0.23
11	0.27
10	0.36
7	0.49
4	0.56

Iterative Simplification

- 2D example

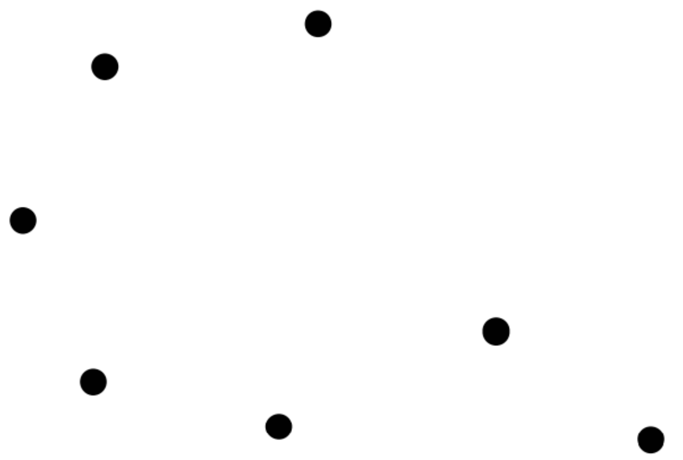


priority queue

edge	cost
2	0.03
14	0.04
5	0.06
9	0.09
1	0.11
13	0.13
3	0.23
11	0.27
10	0.36
7	0.49
4	0.56

Iterative Simplification

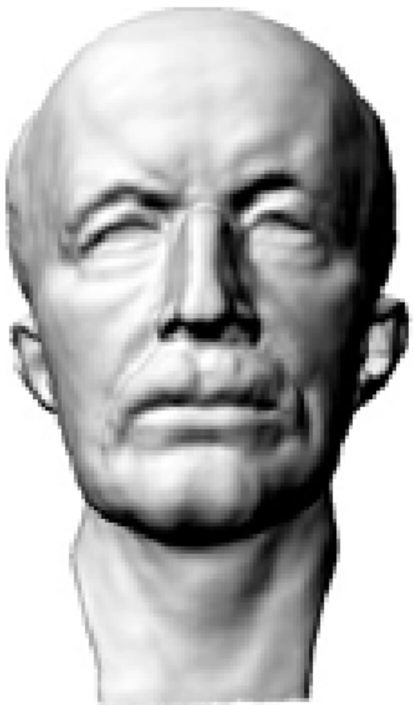
- 2D example



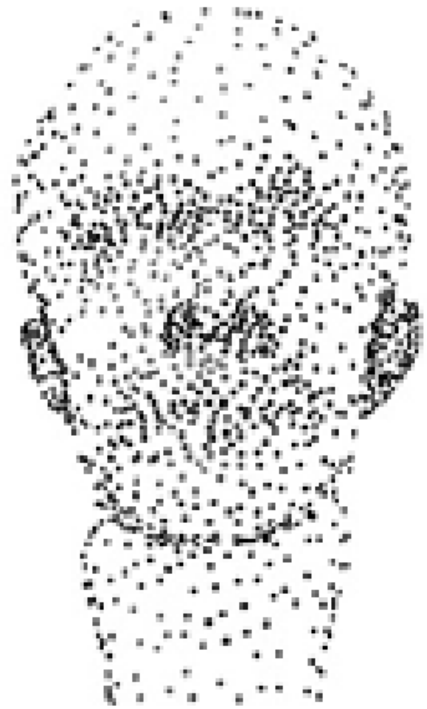
priority queue

edge	cost
11	0.27
10	0.36
7	0.49
4	0.56

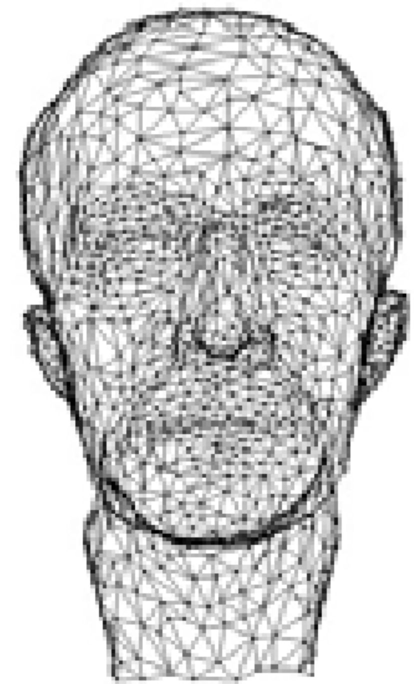
Iterative Simplification



original model
(296,850 points)



simplified model
(2,000 points)



remaining point pair
contraction candidates

Particle Simulation

- Resample surface by distributing particles on the surface
- Particles move on surface according to interparticle repelling forces
- Particle relaxation terminates when equilibrium is reached (requires damping)
- Can also be used for up-sampling!

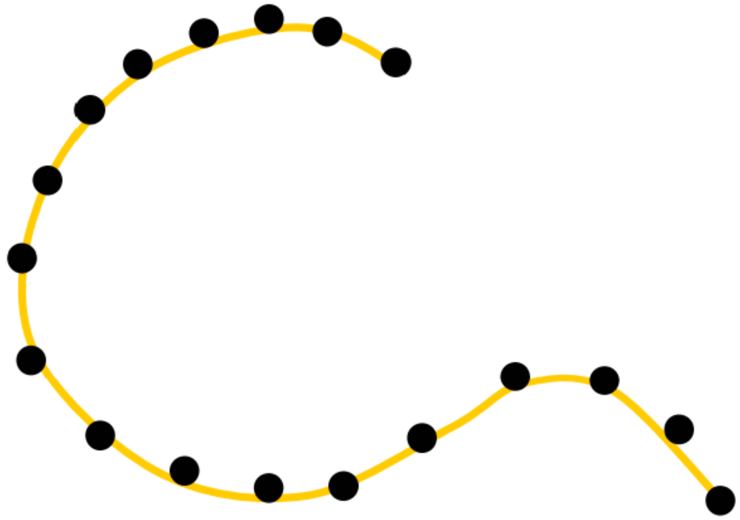
Particle Simulation

- Initialization
  - Randomly spread particles
- Repulsion
  - Linear repulsion force $F_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|) \cdot (\mathbf{p} - \mathbf{p}_i)$

  ⇨ only need to consider neighborhood of radius r
- Projection
  - Keep particles on surface by projecting onto tangent plane of closest point
  - Apply full MLS projection at end of simulation

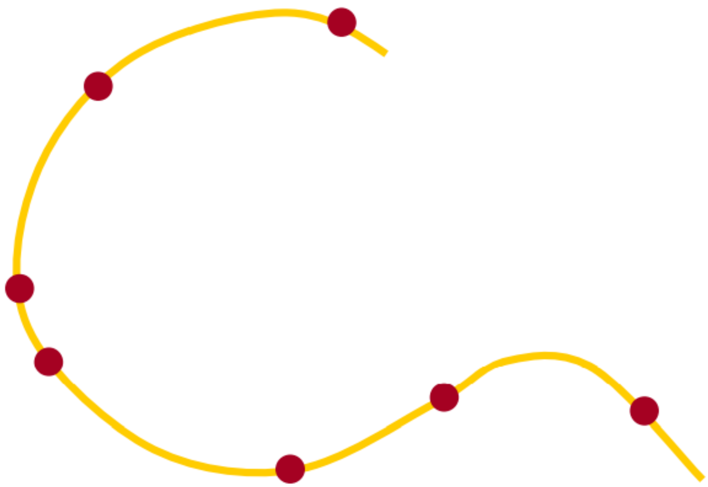
Particle Simulation

- 2D example



Particle Simulation

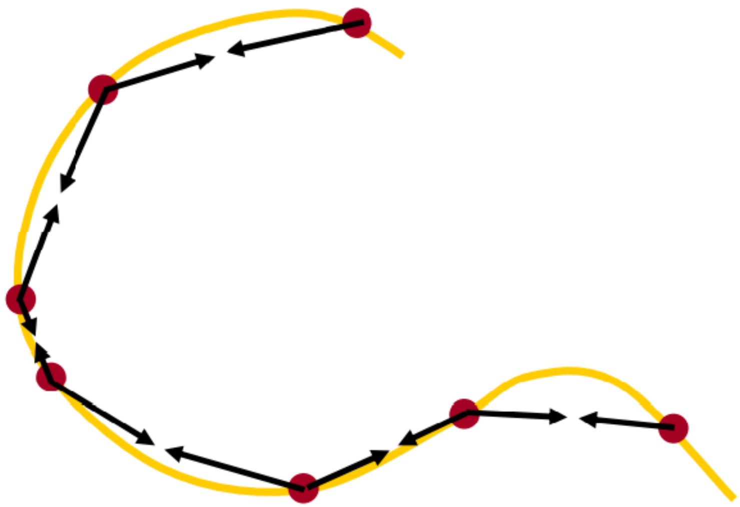
- 2D example



- Initialization
  - randomly spread particles

Particle Simulation

- 2D example

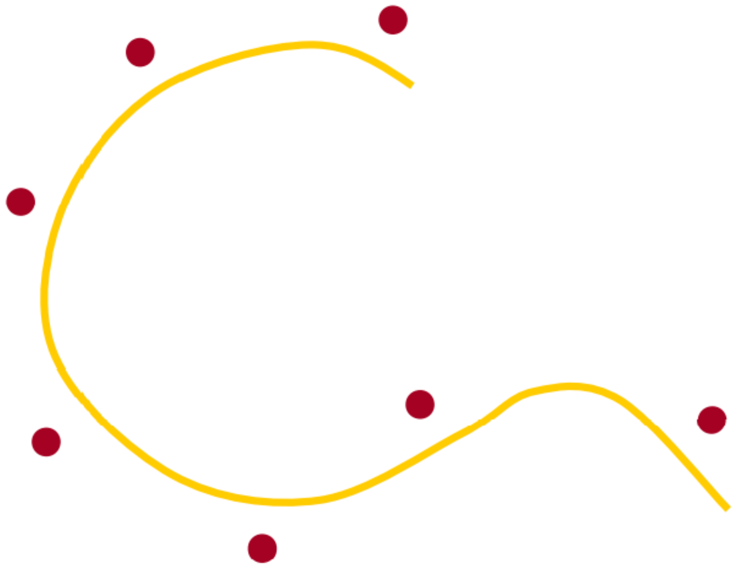


- Initialization
  - randomly spread particles
- Repulsion
  - linear repulsion force

$$F_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|) \cdot (\mathbf{p} - \mathbf{p}_i)$$

Particle Simulation

- 2D example

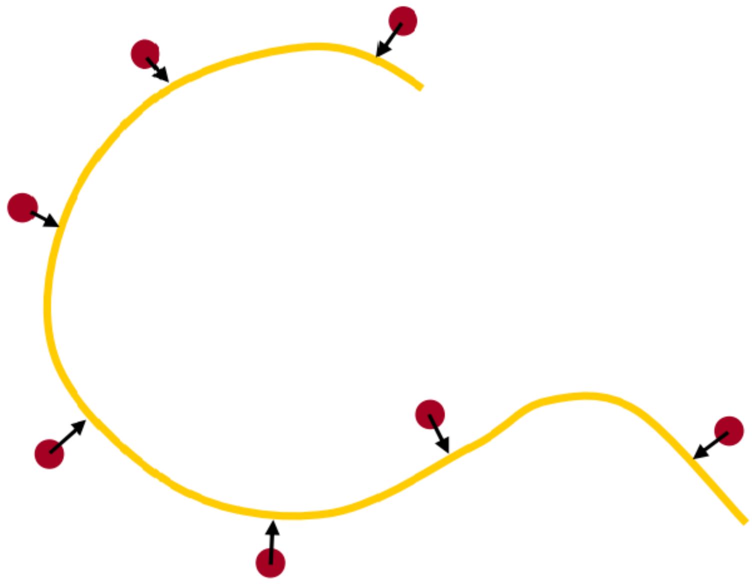


- Initialization
  - randomly spread particles
- Repulsion
  - linear repulsion force

$$F_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|) \cdot (\mathbf{p} - \mathbf{p}_i)$$

Particle Simulation

- 2D example



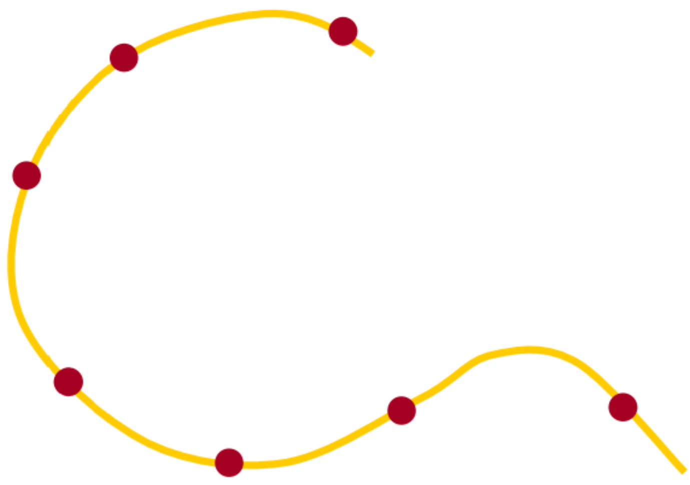
- Initialization
  - randomly spread particles
- Repulsion
  - linear repulsion force

$$F_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|) \cdot (\mathbf{p} - \mathbf{p}_i)$$

- Projection
  - project particles onto surface

Particle Simulation

- 2D example



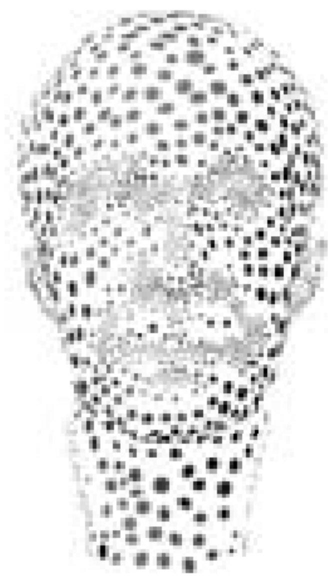
- Initialization
  - randomly spread particles
- Repulsion
  - linear repulsion force

  $$F_i(\mathbf{p}) = k(r - \|\mathbf{p} - \mathbf{p}_i\|) \cdot (\mathbf{p} - \mathbf{p}_i)$$
- Projection
  - project particles onto surface

Particle Simulation

- Adjust repulsion radius according to surface variation ⇨ more samples in regions of high variation

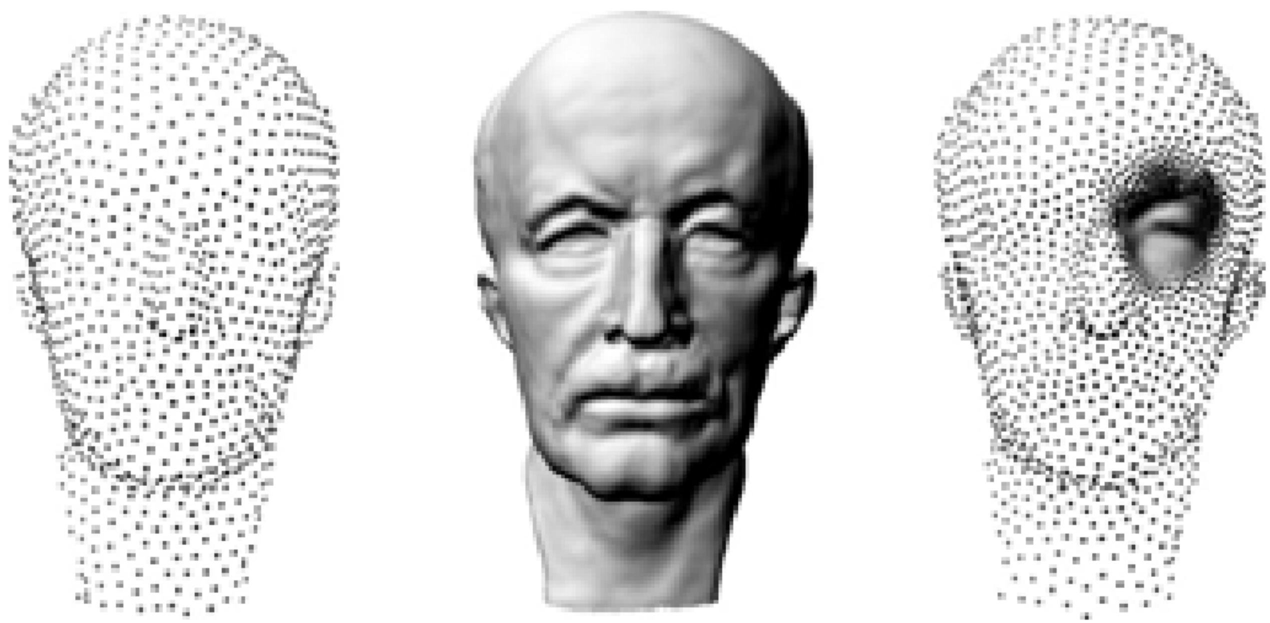
variation
estimation



simplified model
(3,000 points)

Particle Simulation

- User-controlled simulation
  - Adjust repulsion radius according to user input



uniform original selective

Measuring Error

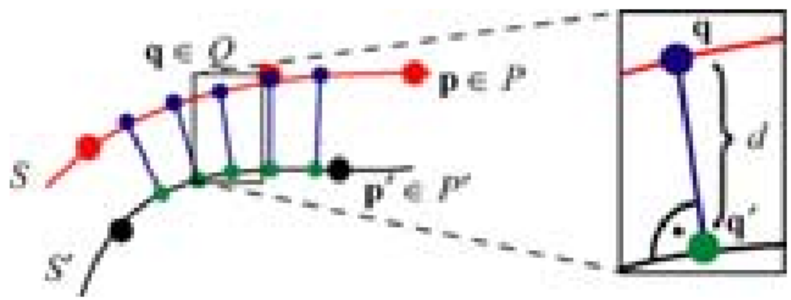
- Measure the distance between two point-sampled surfaces using a sampling approach
- Maximum error: $\Delta_{\max}(S, S') = \max_{\mathbf{q} \in Q} d(\mathbf{q}, S')$

  ⇨ Two-sided Hausdorff distance
- Mean error: $\Delta_{\text{avg}}(S, S') = \frac{1}{|Q|} \sum_{\mathbf{q} \in Q} d(\mathbf{q}, S')$

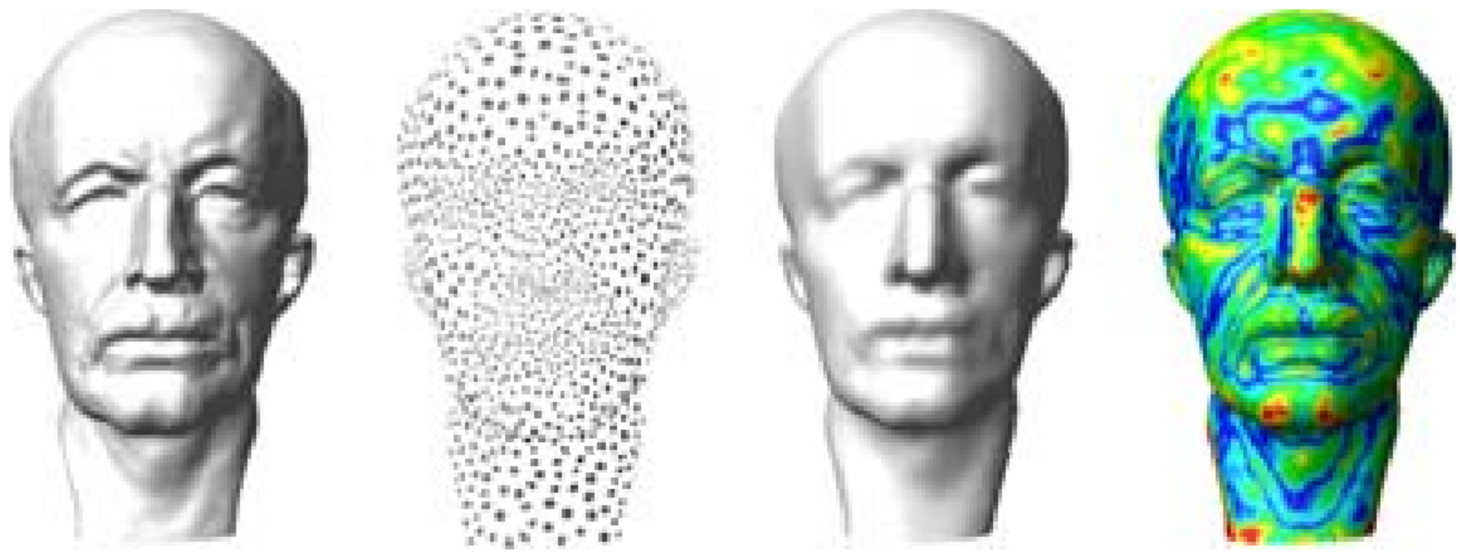
  ⇨ Area-weighted integral of point-to-surface distances
- Q is an up-sampled version of the point cloud that describes the surface S

Measuring Error

- $d(\mathbf{q},S)$ approximates the distance of point $\mathbf{q}$ to surface S using the MLS projection operator



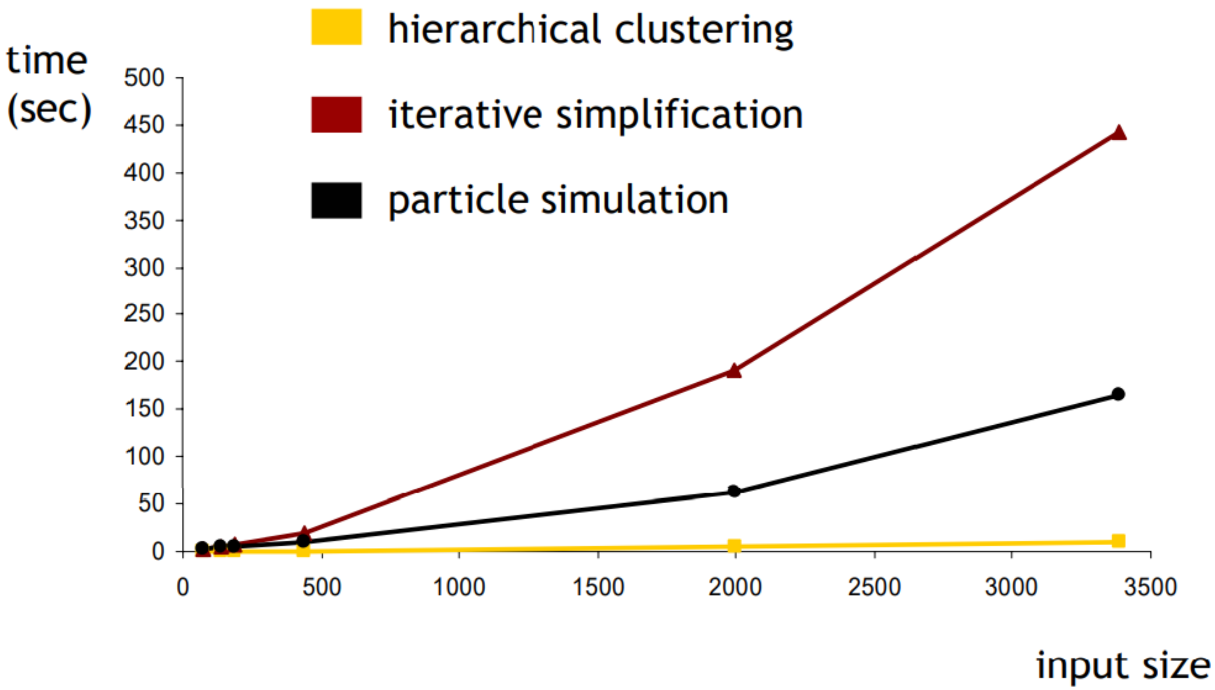
Measuring Error



original simplified upsampled error

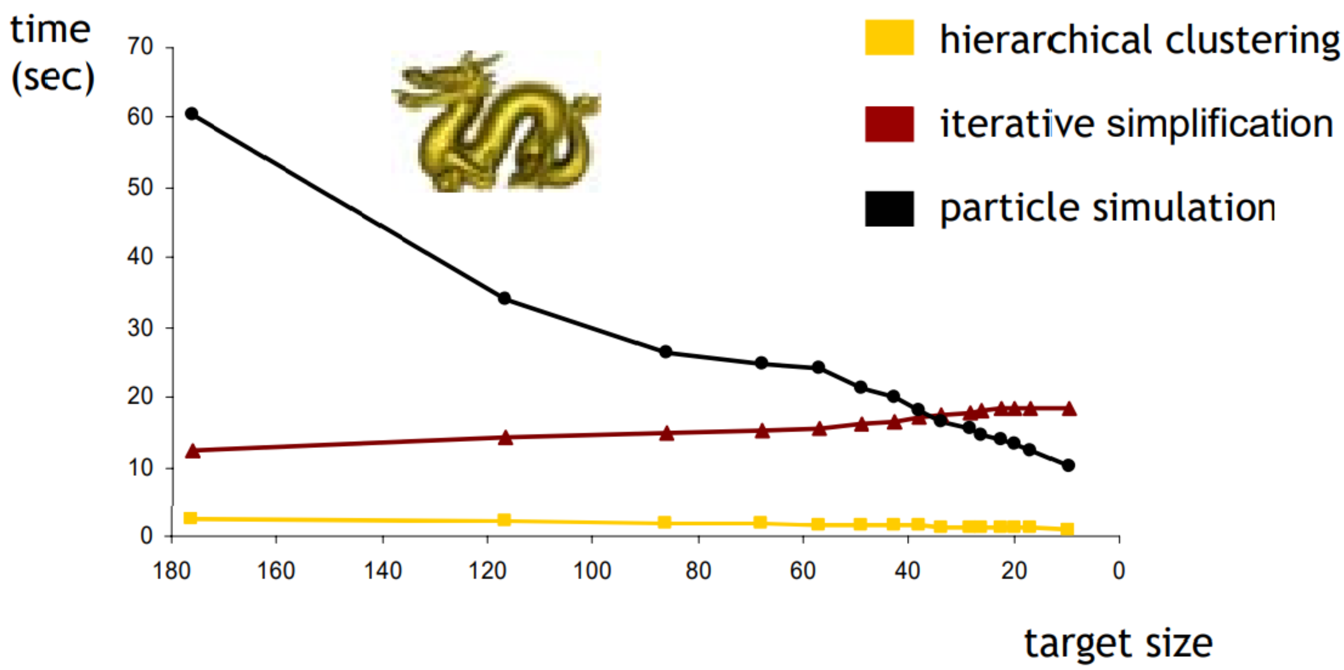
Comparison

- Execution time as a function of input model size (reduction to 1%)



Comparison

- Execution time as a function of target model size (input: dragon, 535,545 points)



Comparison

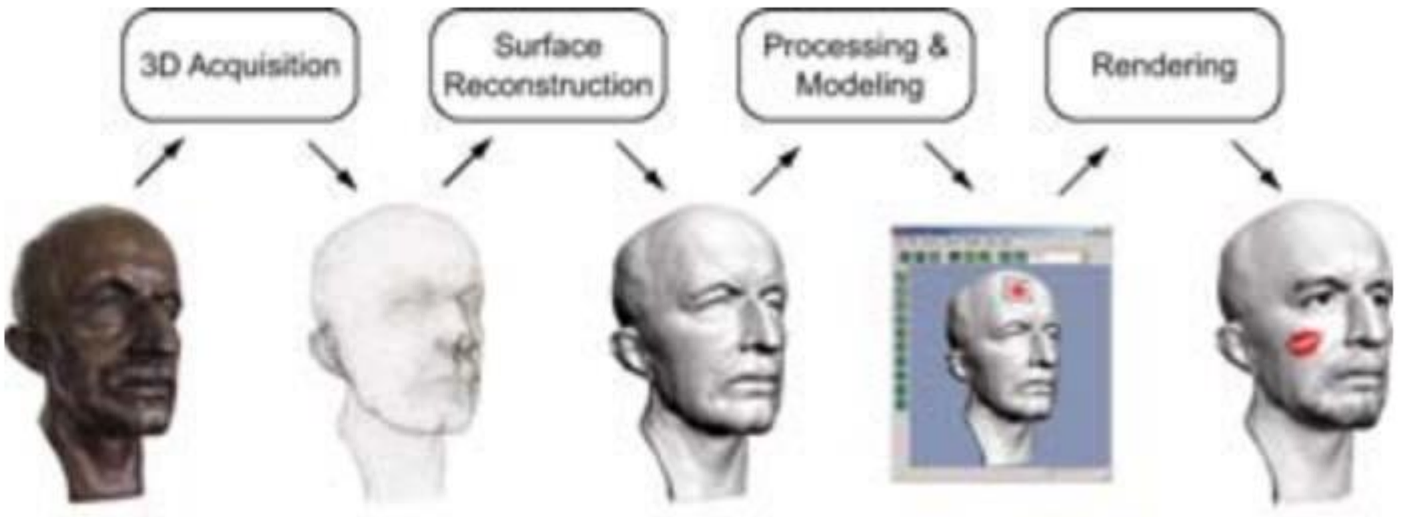
- Summary

	Efficiency	Surface Error	Control	Implementation
Hierarchical Clustering	+	-	-	+
Iterative Simplification	-	+	o	o
Particle Simulation	o	+	+	-

Shape Modeling

Motivation

- 3D content creation pipeline



physical model | raw point cloud | model representation | modified model | final model

Surface Representations

- Explicit surfaces (B-reps)
  - Polygonal meshes
  - Subdivision surfaces
  - NURBS
- Limitations
  - Efficient rendering
  - Sharp edges
  - Intuitive editing

Surface Representations

- Implicit surfaces
  - Level sets
  - Radial basis functions
  - Algebraic surfaces
- Limitations
  - Boolean operations
  - Changes of topology
  - Extreme deformations

Hybrid Representation

- Goals
  - Explicit cloud of point samples
  - Implicit dynamic surface model
- Point cloud representation
  - Minimal consistency requirements for extreme deformations (dynamic re-sampling)
  - Fast inside/outside classification for boolean operations and collision detection
  - Explicit modeling and rendering of sharp feature curves
  - Integrated, intuitive editing of shape and appearance

Interactive Modeling

- Interactive design and editing of point-sampled models
  - Shape modeling
    - Boolean operations
    - Free-form deformation
  - Appearance modeling
    - Painting & texturing
    - Embossing & engraving

Surface Model

- Goal: Define continuous surface from a set of discrete point samples

discrete set of point samples $\boldsymbol{P} = \{ \boldsymbol{p}_i, \boldsymbol{c}_i, \boldsymbol{m}_i, \dots \}$

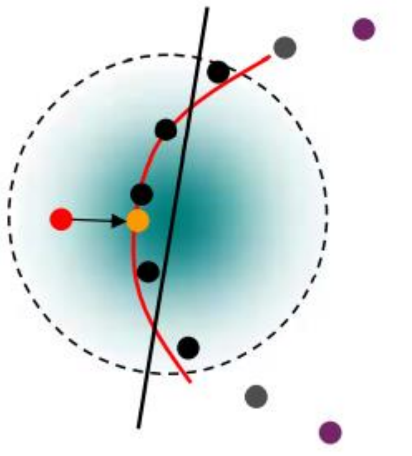
continuous surface $\boldsymbol{S}$ interpolating or approximating $\boldsymbol{P}$

Surface Model

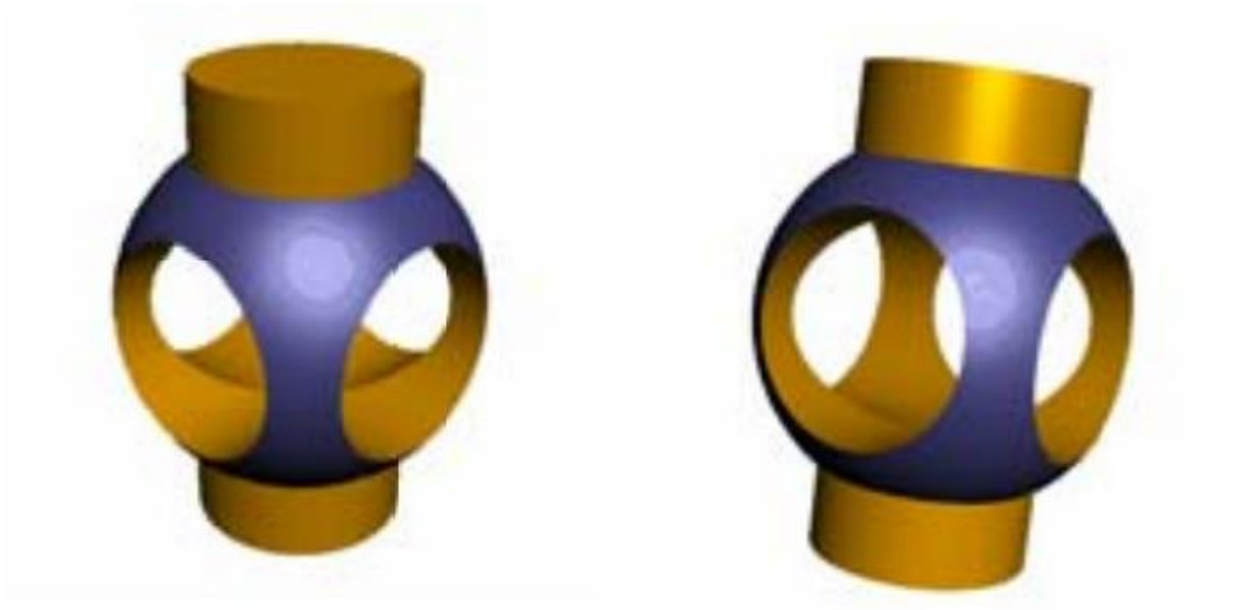
- Moving least squares (MLS) approximation (Levin, Alexa et al.)
  - Surface defined as stationary set of projection operator

$$S_P = \left\{ x \in \mathbf{R}^3 \middle| \Psi_P(x) = x \right\}$$

  - Weighted least squares optimizatio
    - Gaussian kernel function
      - local, smooth
      - mesh-less, adaptive



Boolean Operations



Boolean Operations

- Create new shapes by combining existing models using union, intersection, or difference operations

- Powerful and flexible editing paradigm mostly used in industrial design applications (CAD/CAM)

Boolean Operations

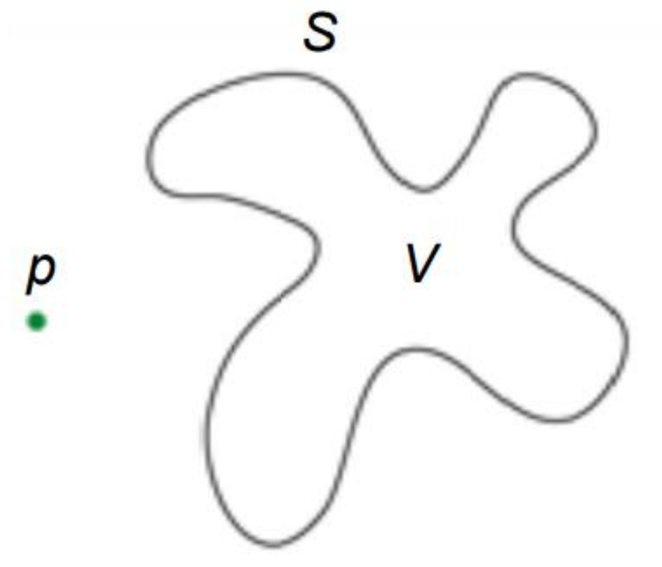
- Easily performed on implicit representations
  - Requires simple computations on the distance function
- Difficult for parametric surfaces
  - Requires surface-surface intersection
- Topological complexity of resulting surface depends on geometric complexity of input models

Boolean Operations

- Point-Sampled Geometry
  - Classification
    - Inside-outside test using signed distance function induced by MLS projection
  - Sampling
    - Compute exact intersection of two MLS surfaces to sample the intersection curve
  - Rendering
    - Accurate depiction of sharp corners and creases using point-based rendering

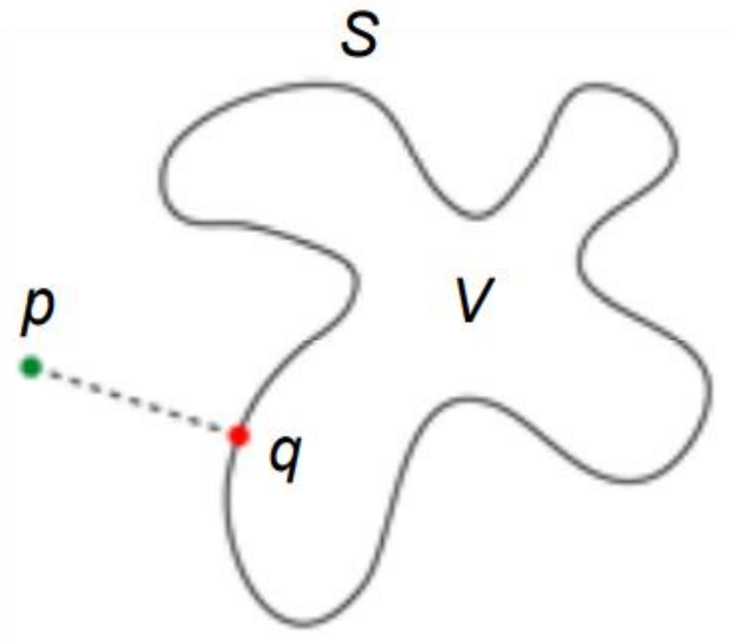
Boolean Operations

- Classification:
  - Given a smooth, closed surface S and point p. Is p inside or outside of the volume V bounded by S?



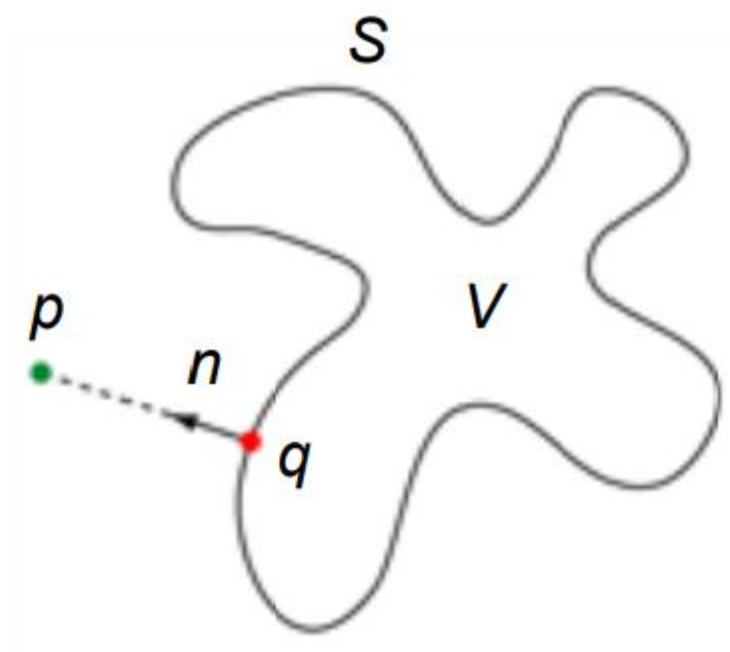
Boolean Operations

- Classification:
  - Given a smooth, closed surface S and point p. Is p inside or outside of the volume V bounded by S?
  - 1. find closest point q on S



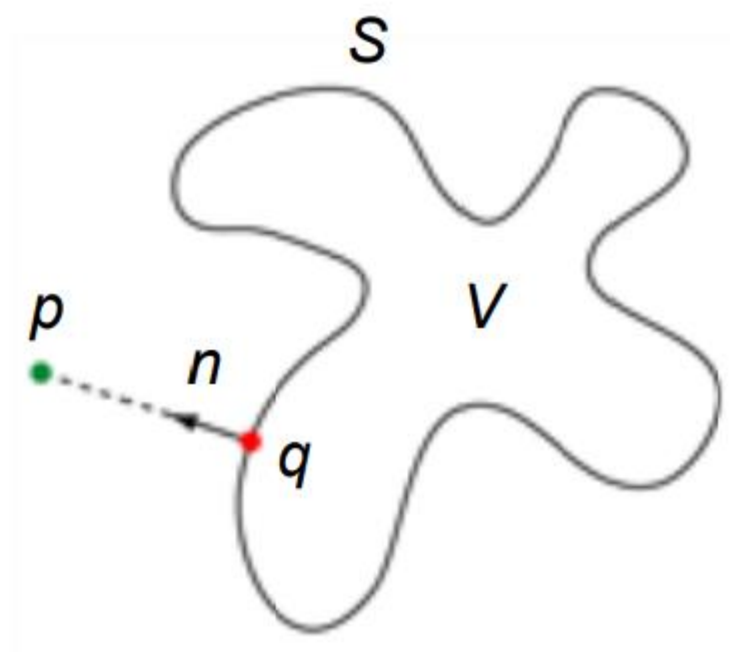
Boolean Operations

- Classification:
  - Given a smooth, closed surface S and point p. Is p inside or outside of the volume V bounded by S?
  - 1. find closest point q on S
  - 2. d = (p-q)’*n defines signed distance of p to S



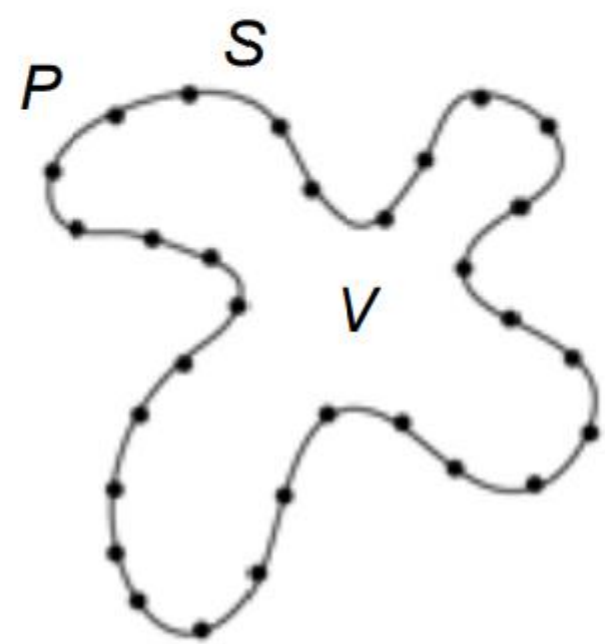
Boolean Operations

- Classification:
  - Given a smooth, closed surface S and point p. Is p inside or outside of the volume V bounded by S?
  - 1. find closest point q on S
  - 2. d = (p-q)’*n defines signed distance of p to S
  - 3. classify p as
    - inside V, if d < 0
    - Outside V, if d > 0



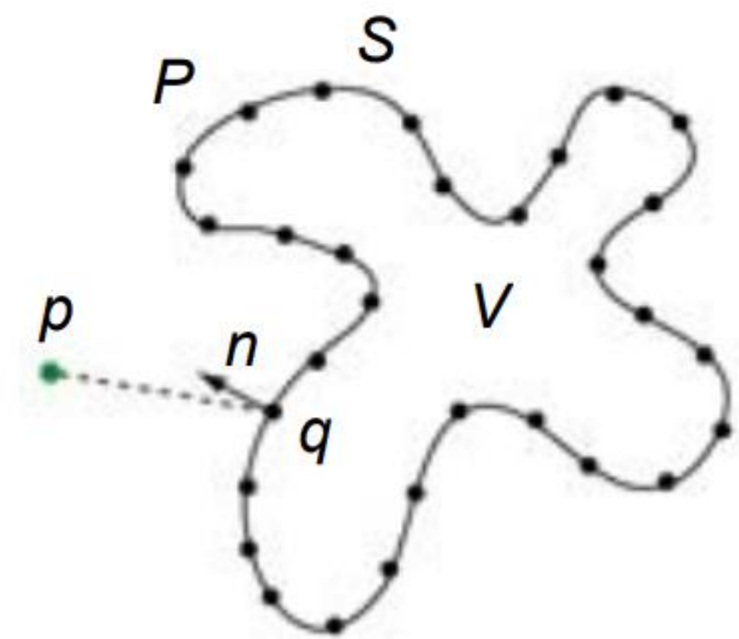
Boolean Operations

- Classification:
  - Represent smooth surface S by point cloud P



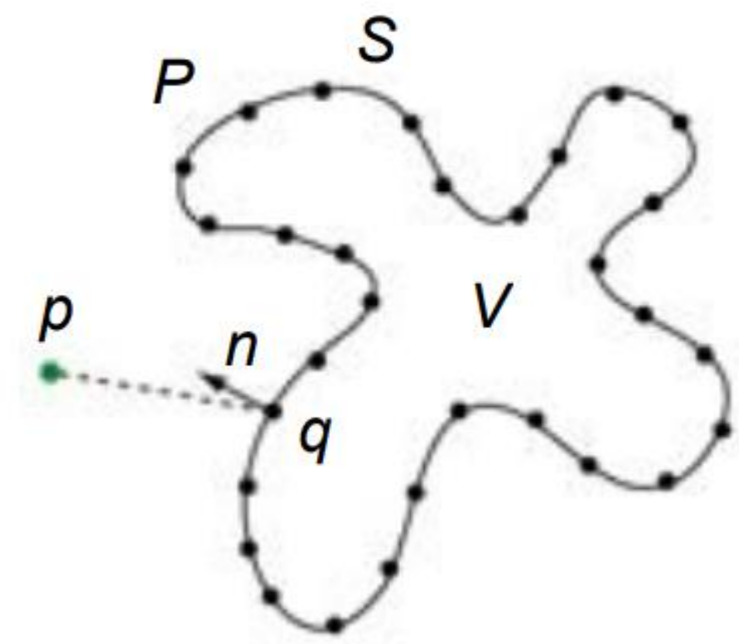
Boolean Operations

- Classification:
  - Represent smooth surface S by point cloud P
  - 1. find closest point q in P



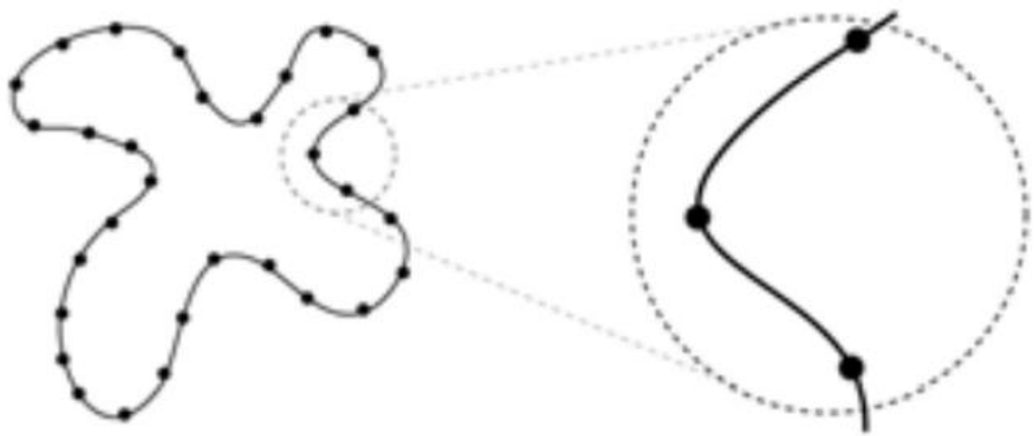
Boolean Operations

- Classification:
  - Represent smooth surface S by point cloud P
  - 1. find closest point q in P
  - 2. classify p as
    - inside V, if (p-q)’*n < 0
    - outside V, if (p-q)’*n > 0



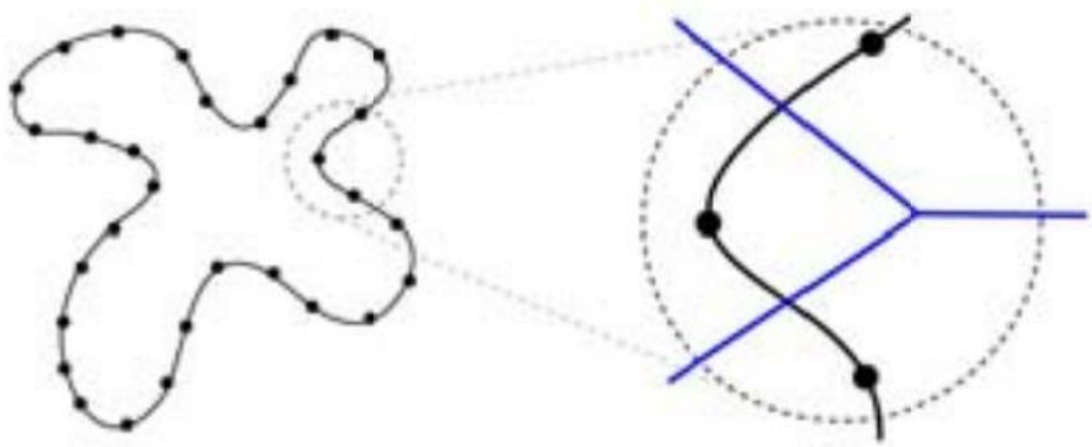
Boolean Operations

- Classification:
  - piecewise constant surface approximation leads to false classification close to the surface



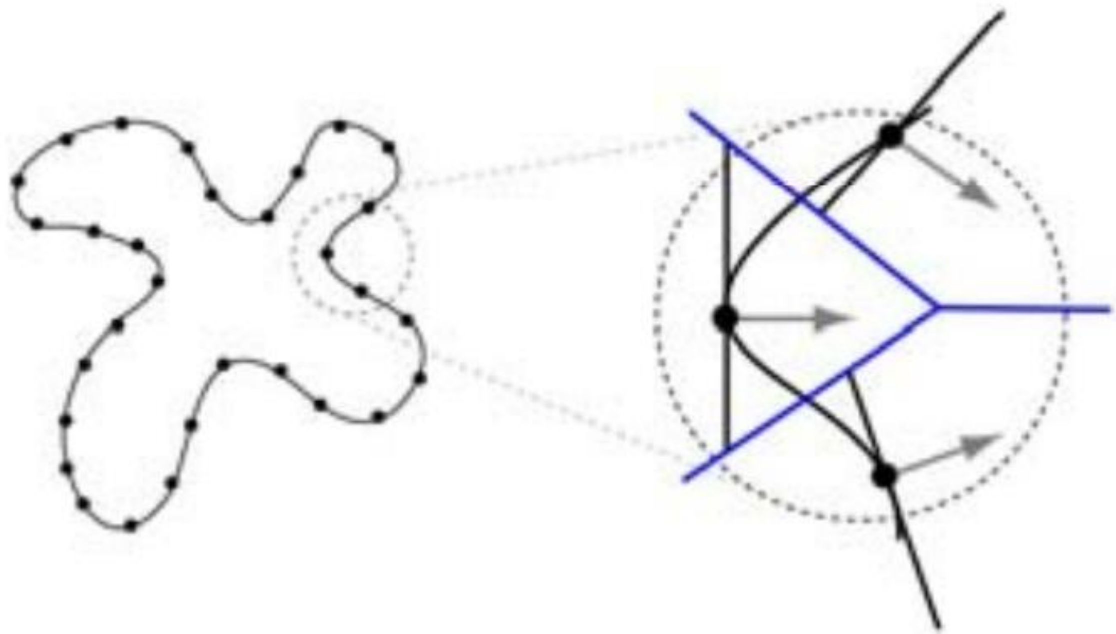
Boolean Operations

- Classification:
  - piecewise constant surface approximation leads to false classification close to the surface



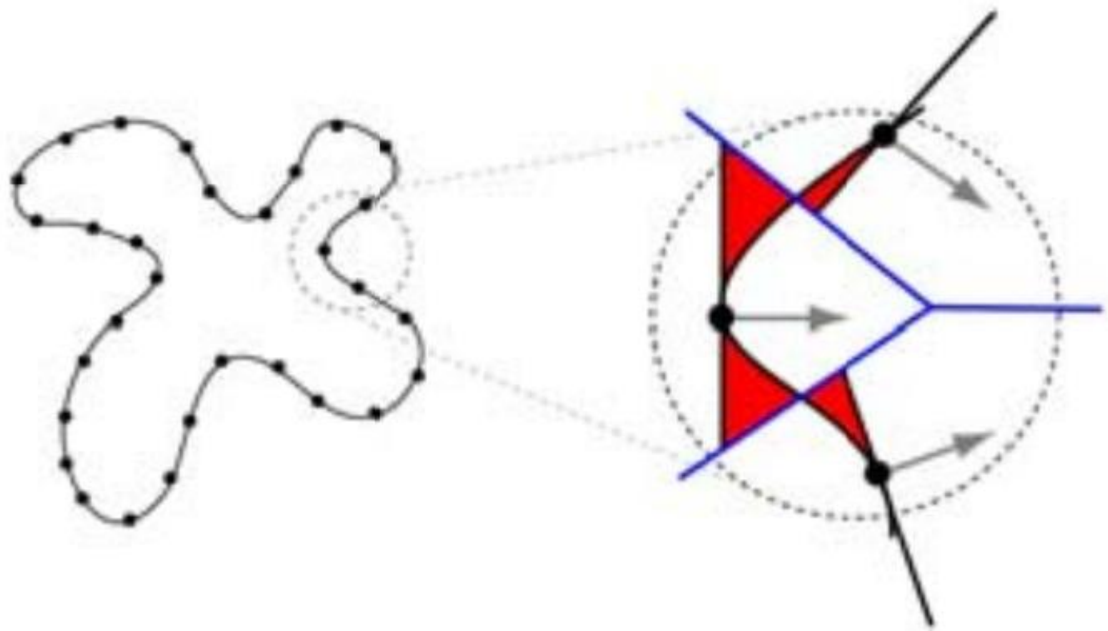
Boolean Operations

- Classification:
  - piecewise constant surface approximation leads to false classification close to the surface



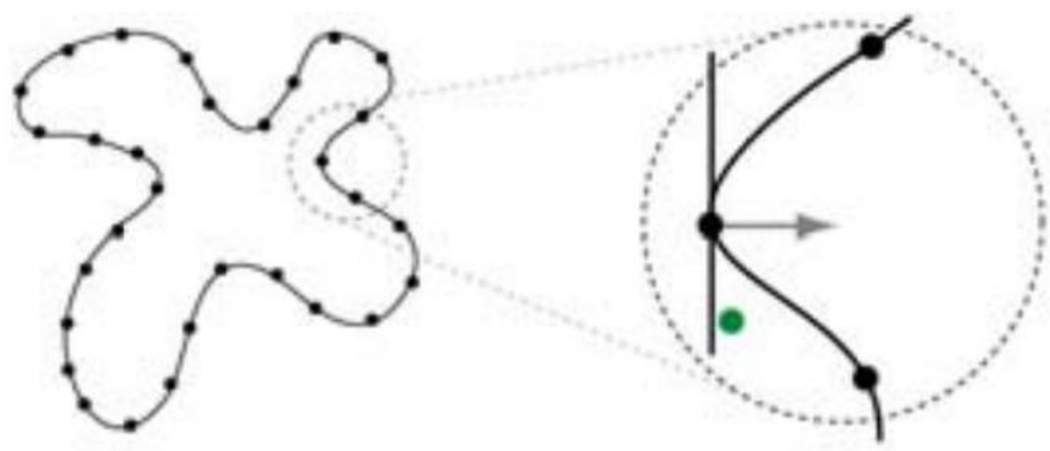
Boolean Operations

- Classification:
  - piecewise constant surface approximation leads to false classification close to the surface



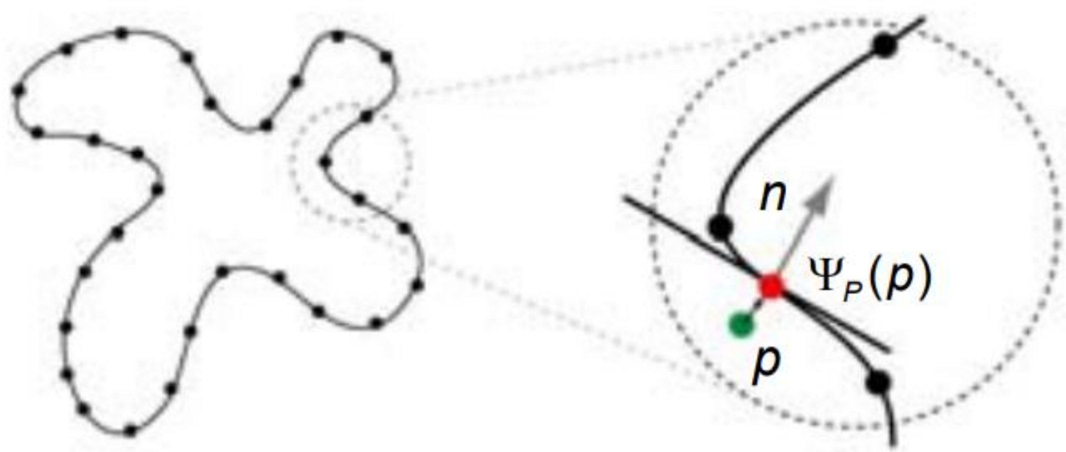
Boolean Operations

- Classification:
  - piecewise constant surface approximation leads to false classification close to the surface



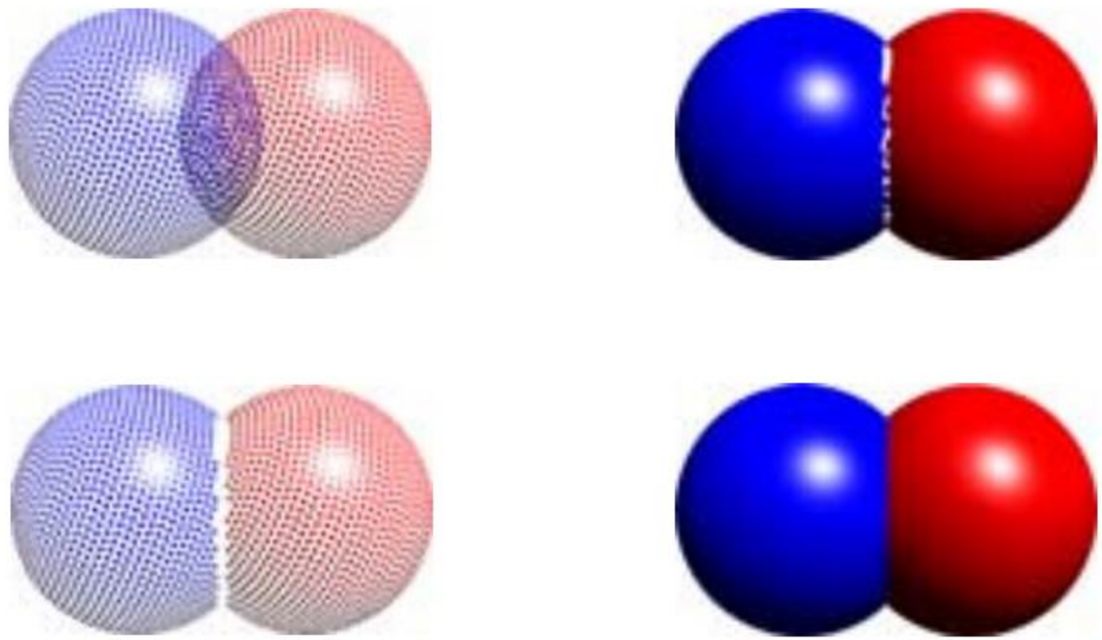
Boolean Operations

- Classification:
  - use MLS projection of p for correct classification



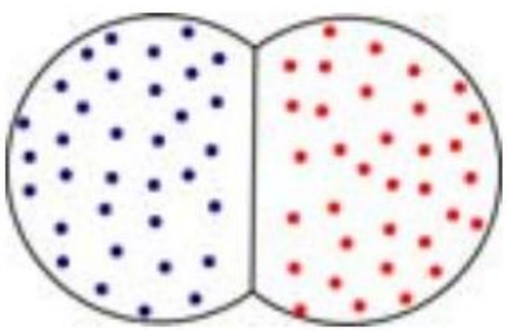
Boolean Operations

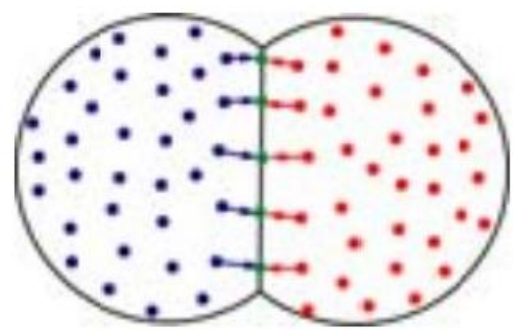
- Sampling the intersection curve



Boolean Operations

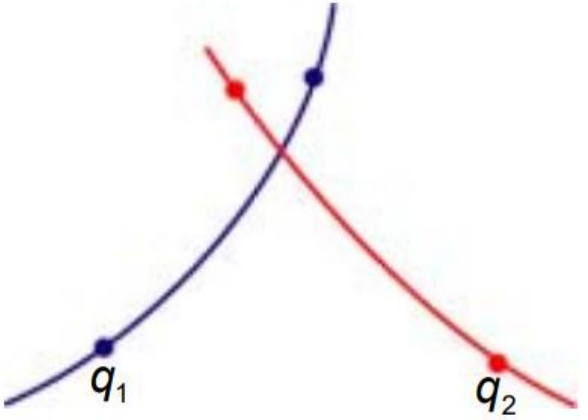
- Newton scheme:
  - Identify pairs of closest points





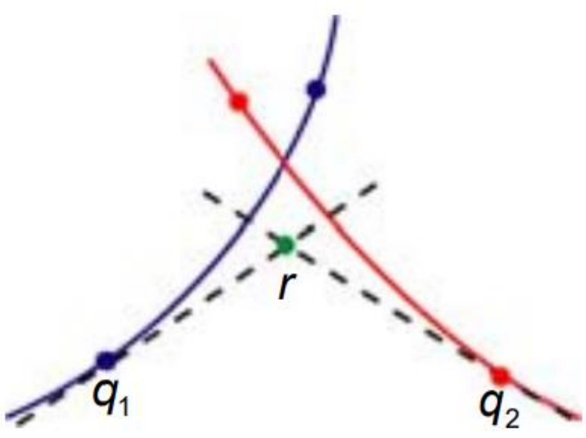
Boolean Operations

- Newton scheme:
  - Identify pairs of closest points



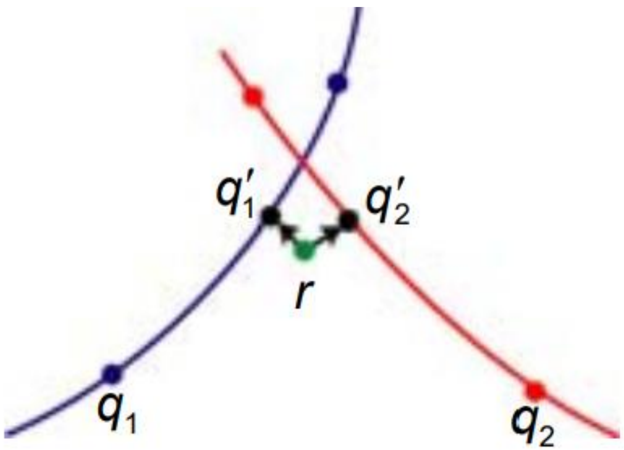
Boolean Operations

- Newton scheme:
  - Identify pairs of closest points
  - Compute closest point on intersection of tangent spaces



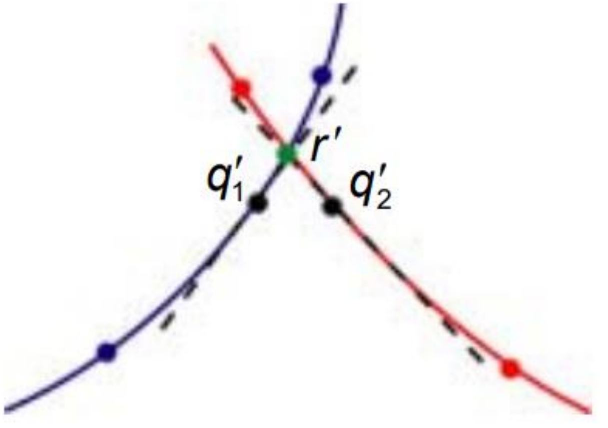
Boolean Operations

- Newton scheme:
  - Identify pairs of closest points
  - Compute closest point on intersection of tangent spaces
  - Re-project point on both surfaces



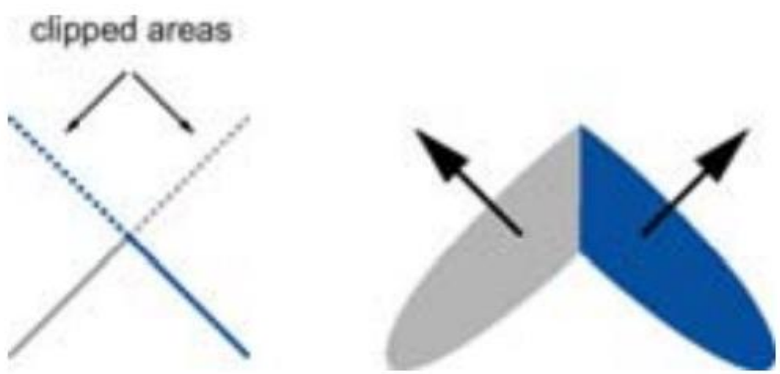
Boolean Operations

- Newton scheme:
  - Identify pairs of closest points
  - Compute closest point on intersection of tangent spaces
  - Re-project point on both surfaces
  - Iterate



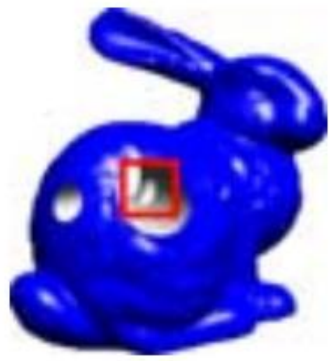
Boolean Operations

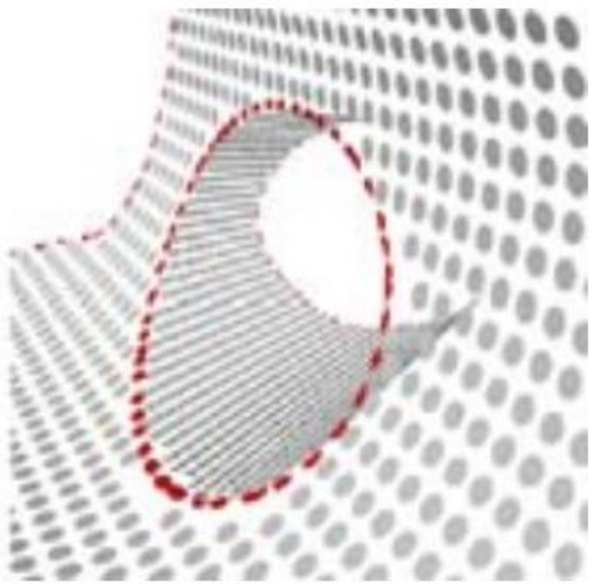
- Rendering sharp creases
  - represent points on intersection curve with two surfels that mutually clip each other



Boolean Operations

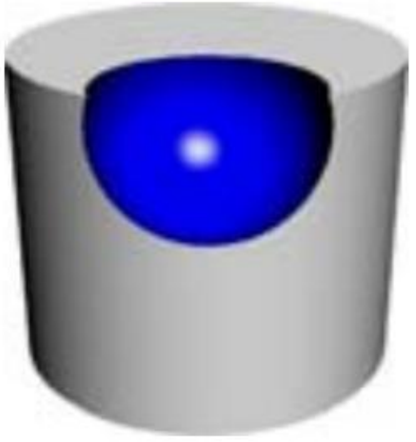
- Rendering sharp creases





Boolean Operations

- Rendering sharp creases
  - easily extended to handle corners by allowing multiple clipping



Boolean Operations

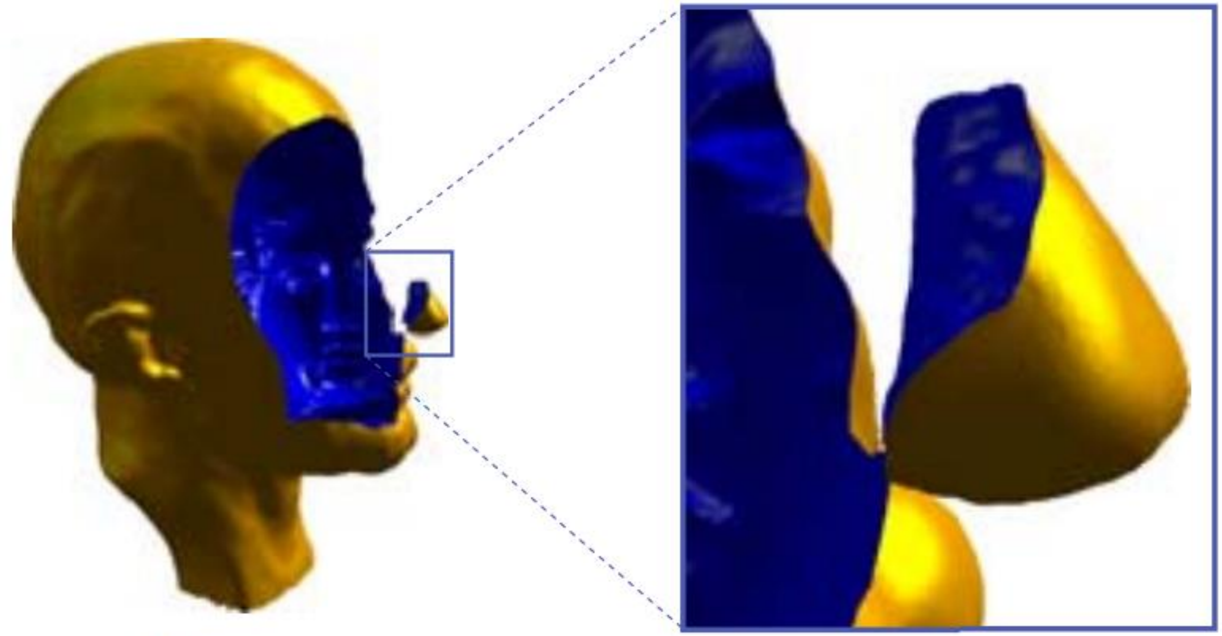
- Rendering sharp creases
  - easily extended to handle corners by allowing multiple clipping

Difference

Union

Boolean Operations

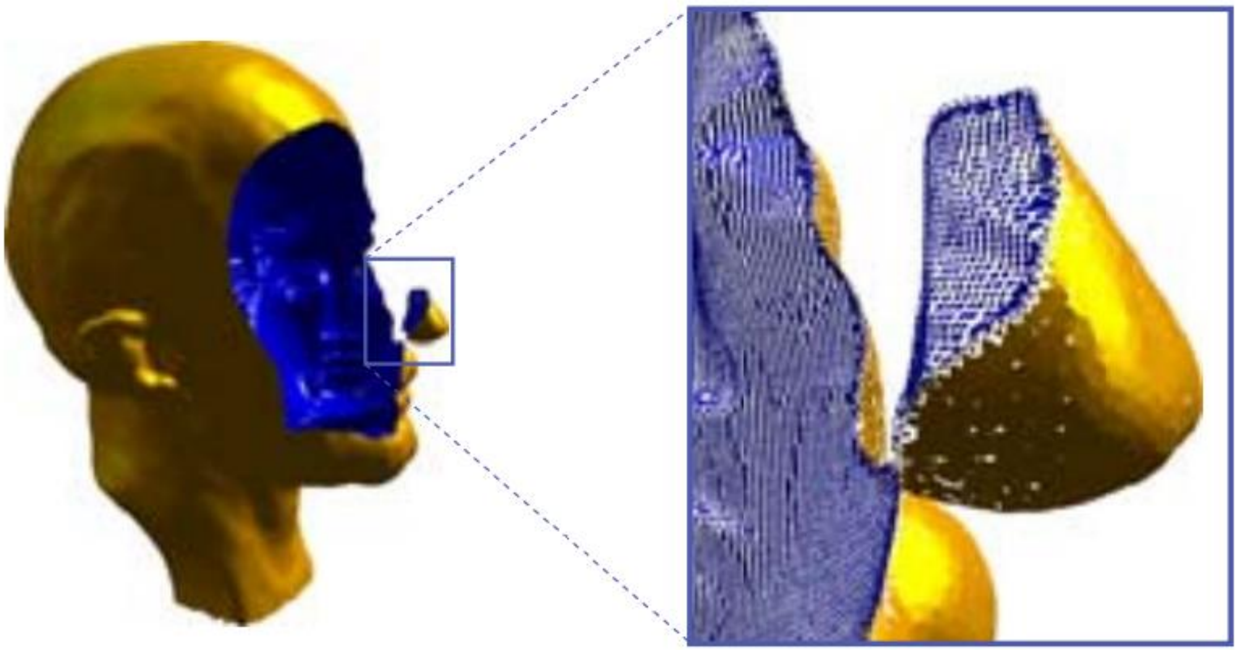
- Rendering sharp creases



Difference

Boolean Operations

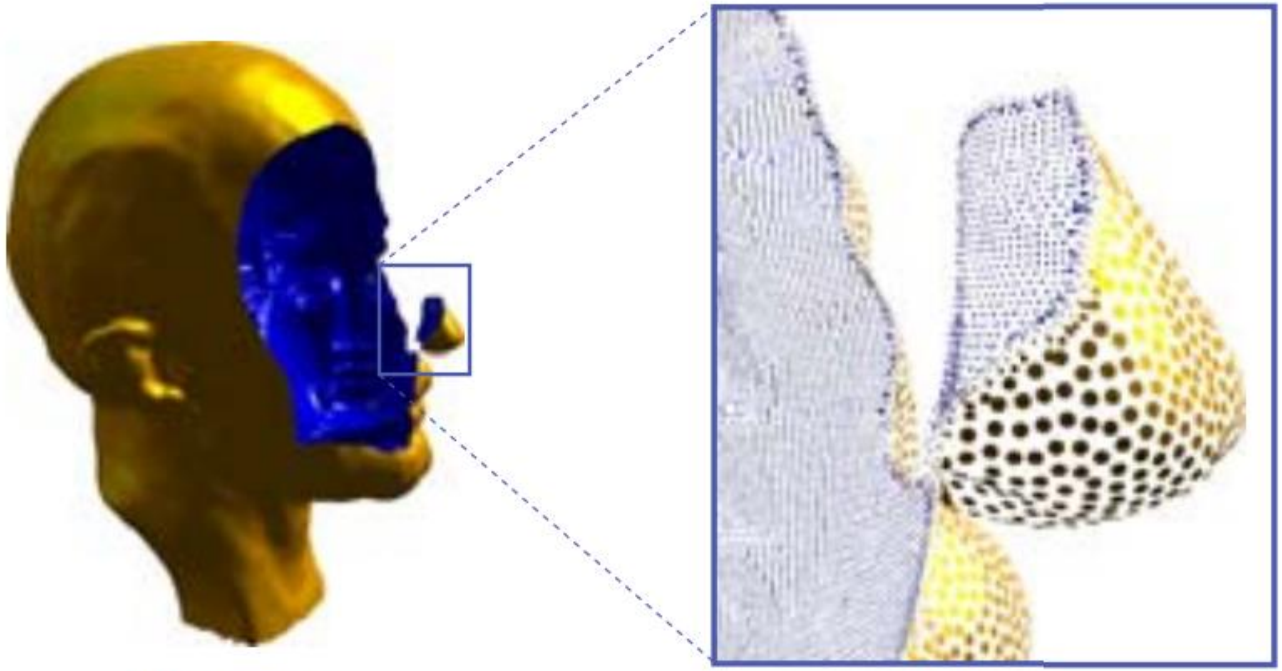
- Rendering sharp creases



Difference

Boolean Operations

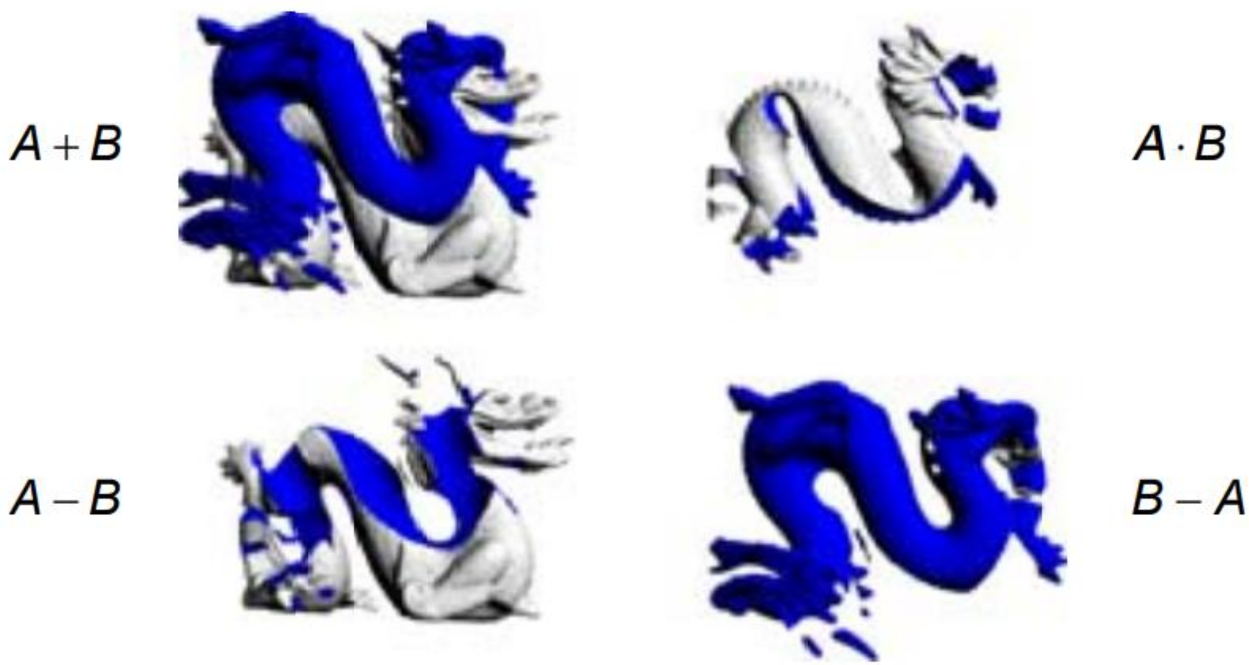
- Rendering sharp creases



Difference

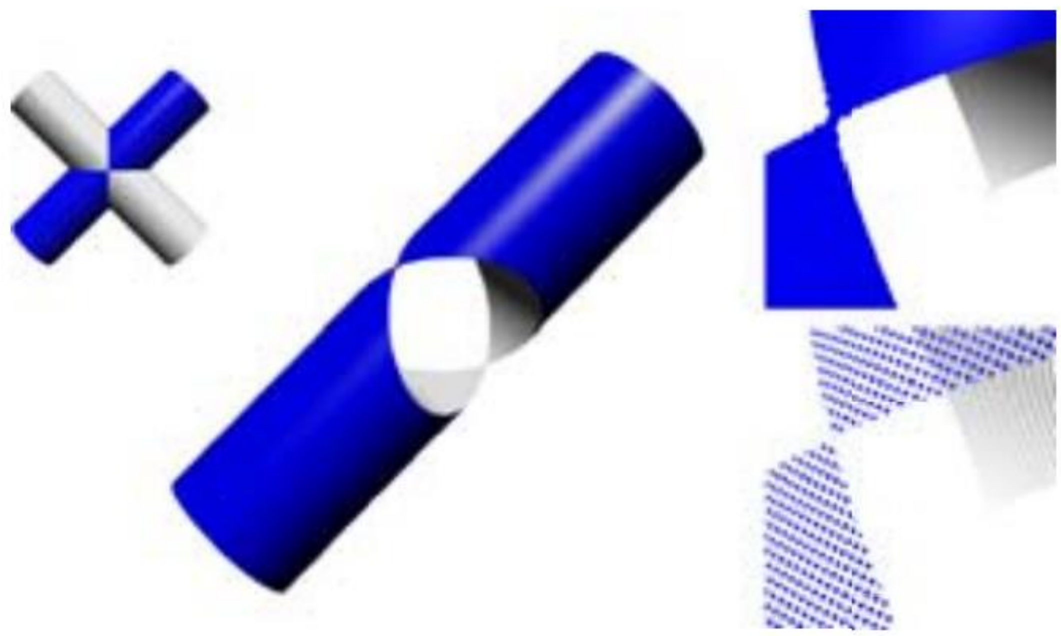
Boolean Operations

- Boolean operations can create intricate shapes with complex topology



Boolean Operations

- Singularities lead to numerical instabilities (intersection of almost parallel planes)



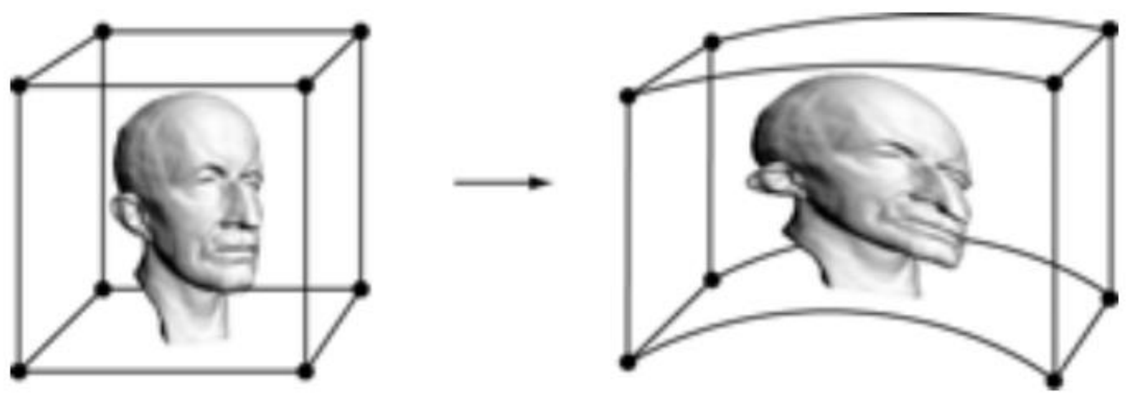
Particle-based Blending

- Boolean operations create sharp intersection curves
- Particle simulation to create smooth transition
  - Repelling force to control particle distribution
  - Normal potentials to control particle orientation

Free-form deformation

Free-form Deformation

- Smooth deformation field $F:\mathbf{R}^3 \rightarrow \mathbf{R}^3$ that warps 3D space
- Can be applied directly to point samples



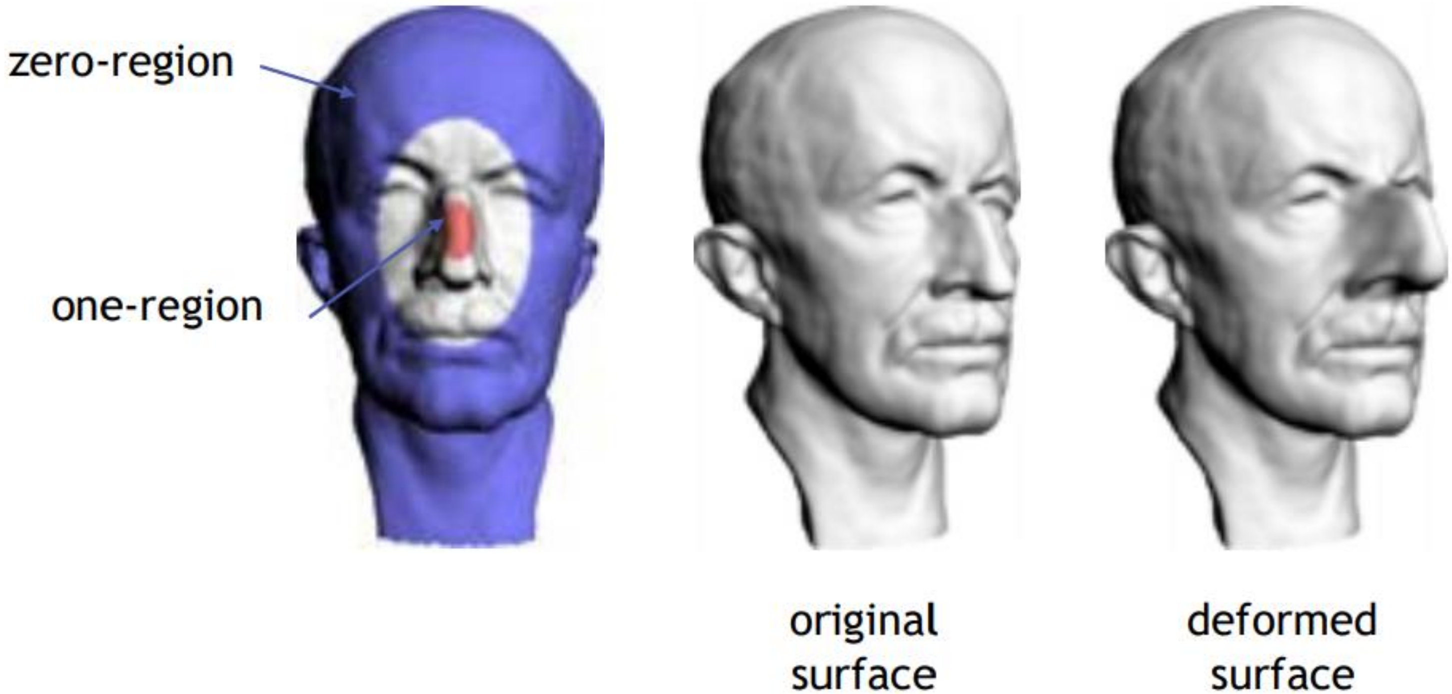
Free-form Deformation

- How to define the deformation field?
  - Painting metaphor
- How to detect and handle self-intersections?
  - Point-based collision detection, boolean union, particle-based blending
- How the handle strong distortions?
  - Dynamic re-sampling

Free-form Deformation

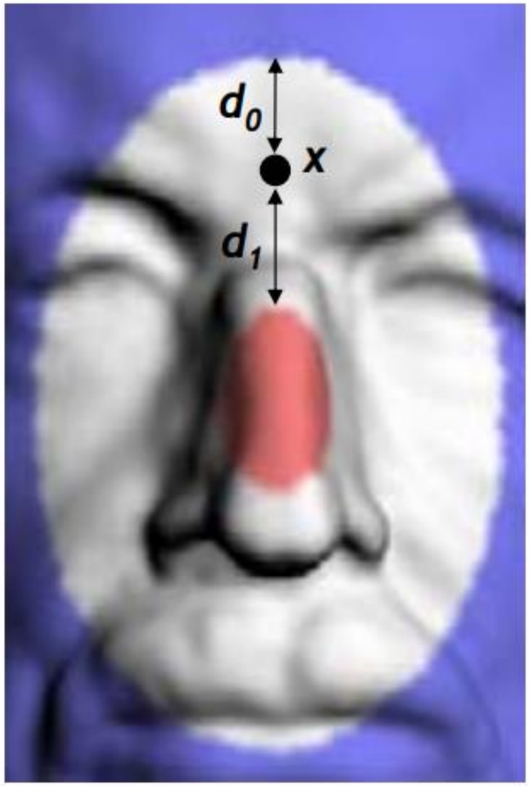
- Intuitive editing paradigm using painting metaphor
  - Define rigid surface part (zero-region) and handle (one-region) using interactive painting tool
  - Displace handle using combination of translation and rotation
  - Create smooth blend towards zero-region

Free-form Deformation



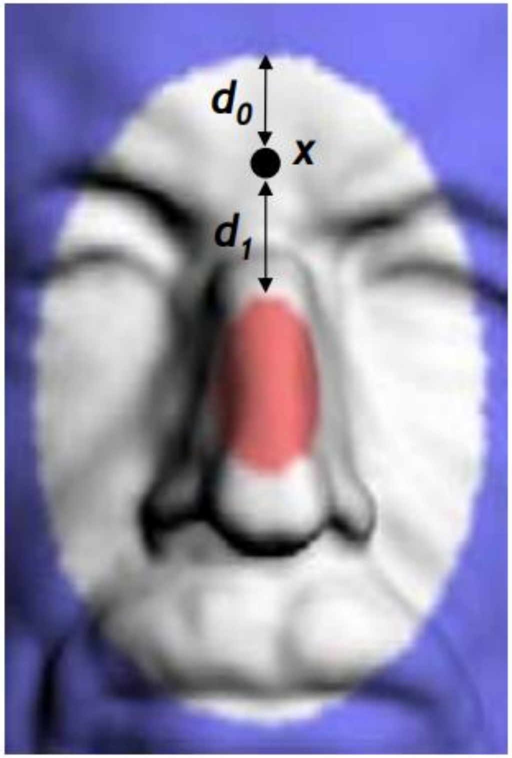
Free-form Deformation

- Definition of deformation field:
  - Continuous scale parameter t_x
    - $t_x = \beta(d_0 / (d_0 + d_1))$
    - d_0 : distance of x to zero-region
    - d_1 : distance of x to one-region
  - Blending function
    - $\beta : [0,1] \rightarrow [0,1]$
    - $\beta \in C^0$, $\beta(0) = 0$, $\beta(1) = 1$
  - $t_x = 0$ if x in zero-region
  - $t_x = 1$ if x in one-region



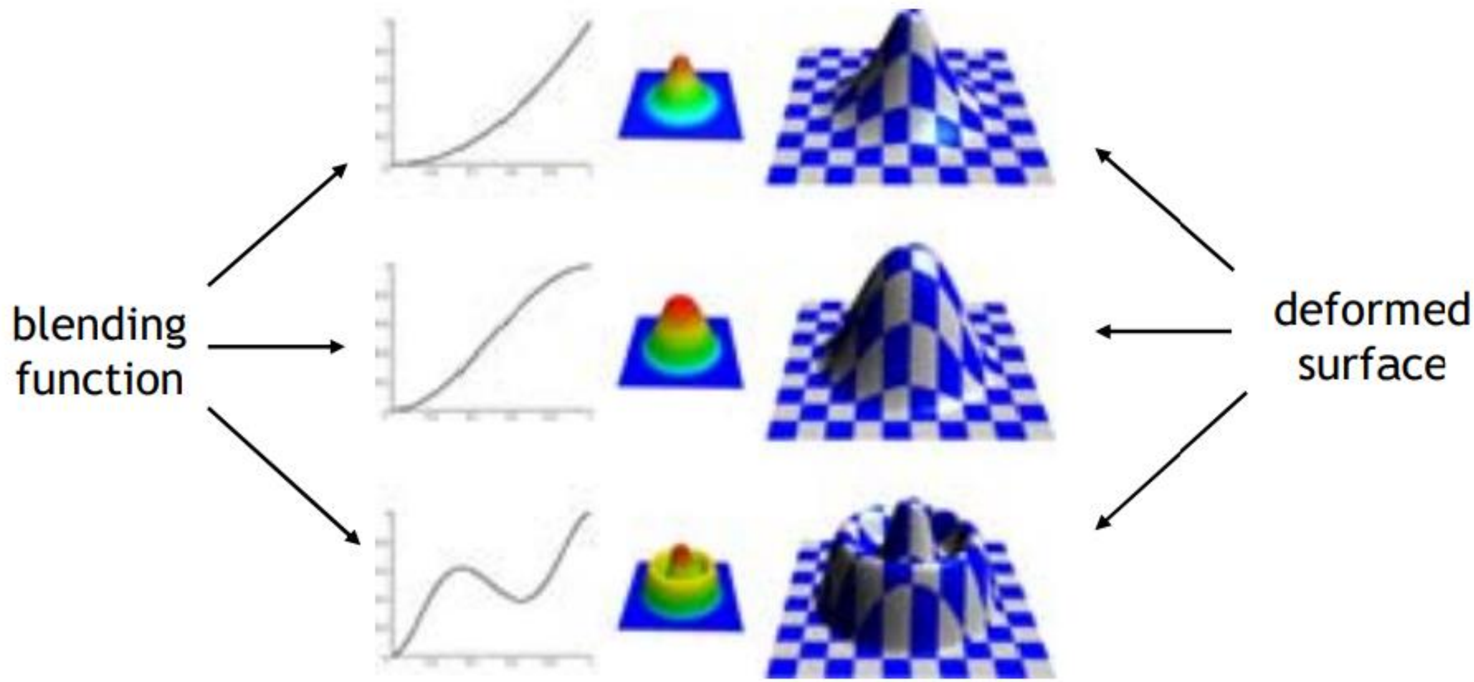
Free-form Deformation

- Definition of deformation field:
  - Deformation function
    - $F(x) = F_T(x) + F_R(x)$
  - Translation
    - $F_T(x) = x + t_x \cdot v$
  - Rotation
    - $F_R(x) = M(t_x) \cdot x$



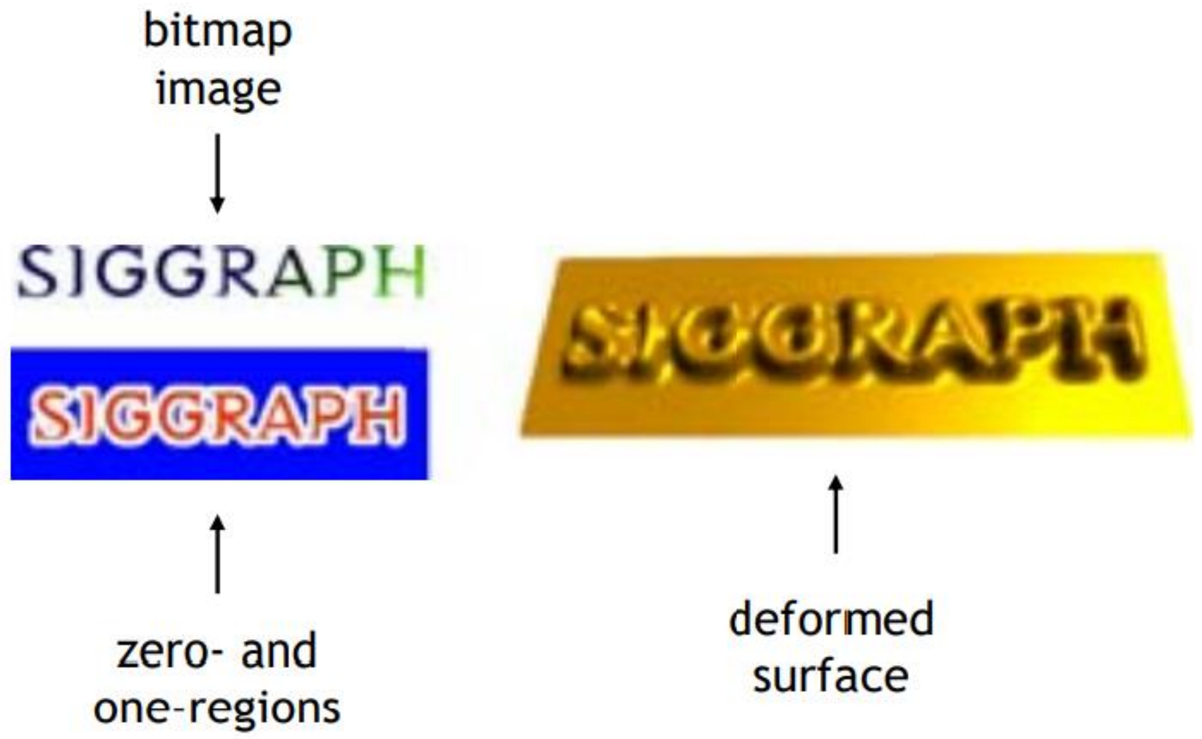
Free-form Deformation

- Translation for three different blending functions



Free-form Deformation

- Embossing effect

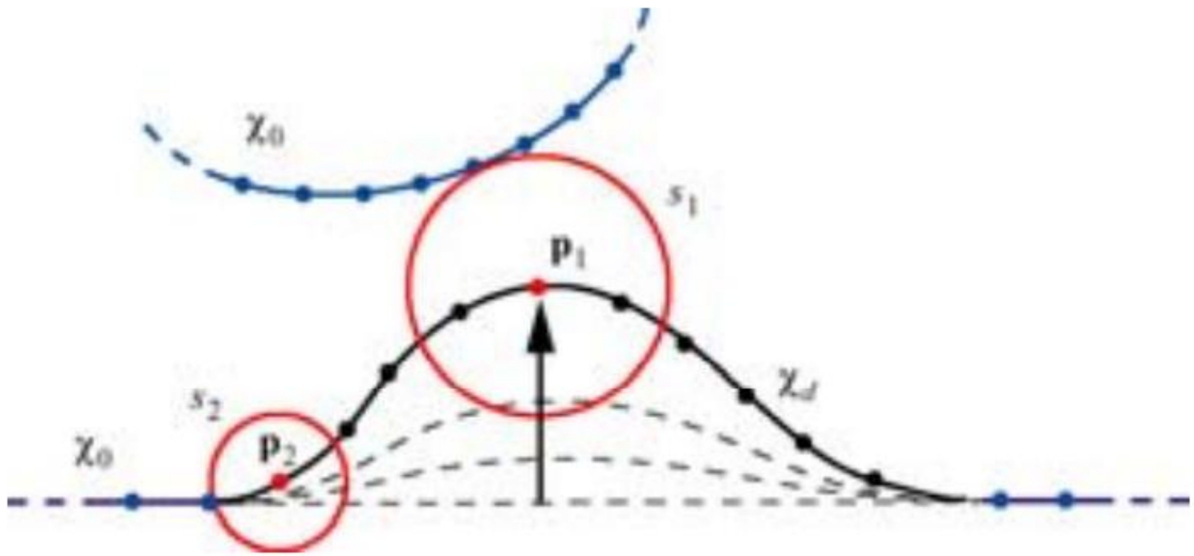


Collision Detection

- Deformations can lead to self-intersections
- Apply boolean inside/outside classification to detect collisions
- Restricted to collisions between deformable region and zero-region to ensure efficient computations

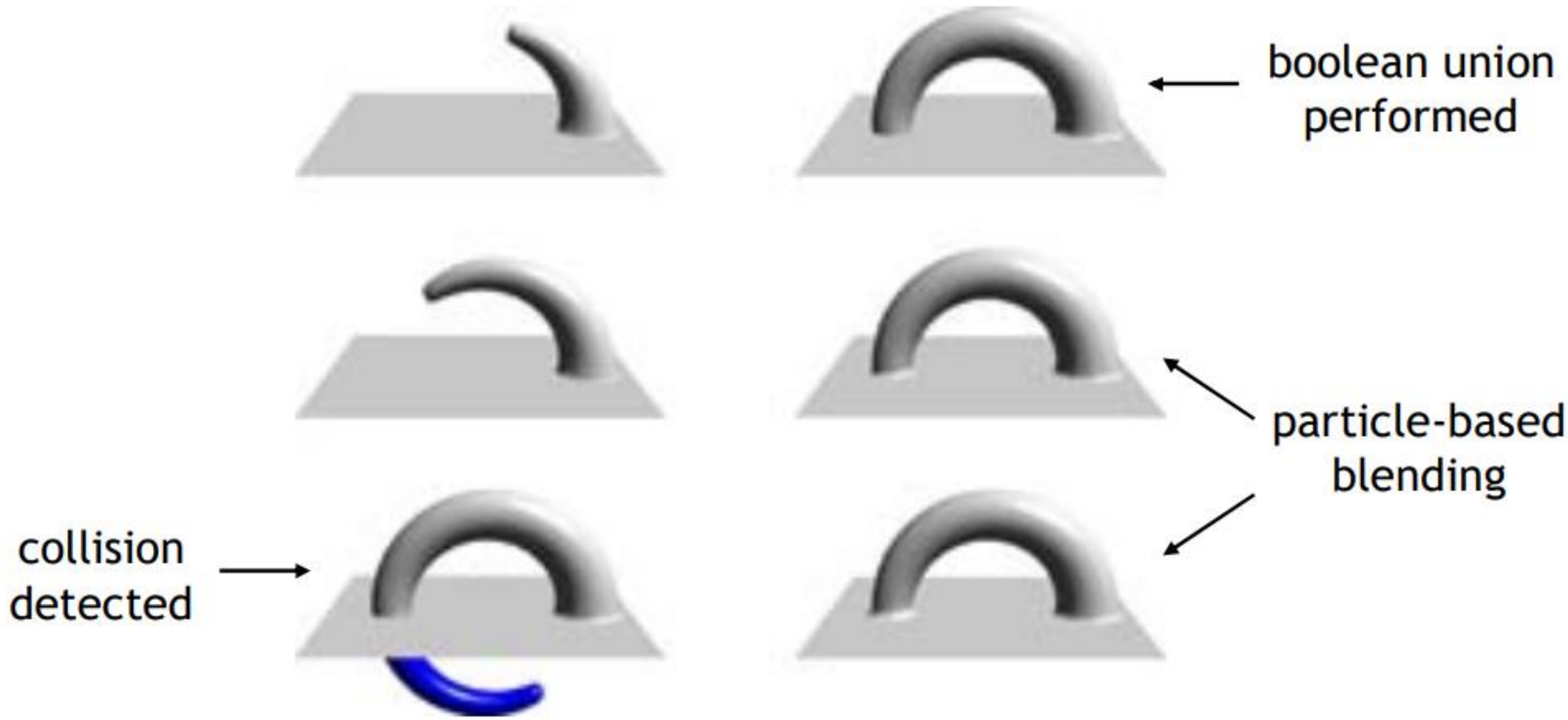
Collision Detection

- Exploiting temporal coherence



Collision Detection

- Interactive modeling session



Dynamic Sampling

10,000 points

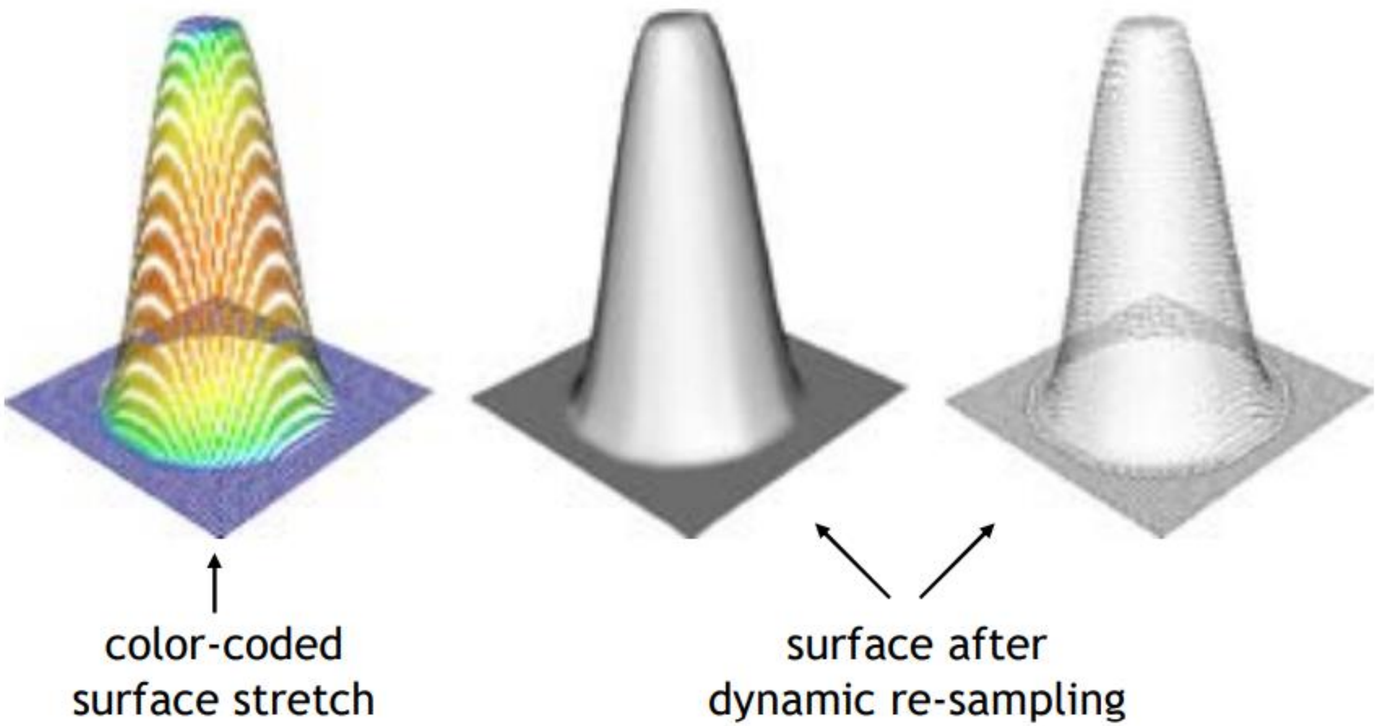
271,743 points

Dynamic Sampling

- Large model deformations can lead to strong surface distortions
- Requires adaptation of the sampling density
- Dynamic insertion and deletion of point samples

Dynamic Sampling

- Surface distortion varies locally

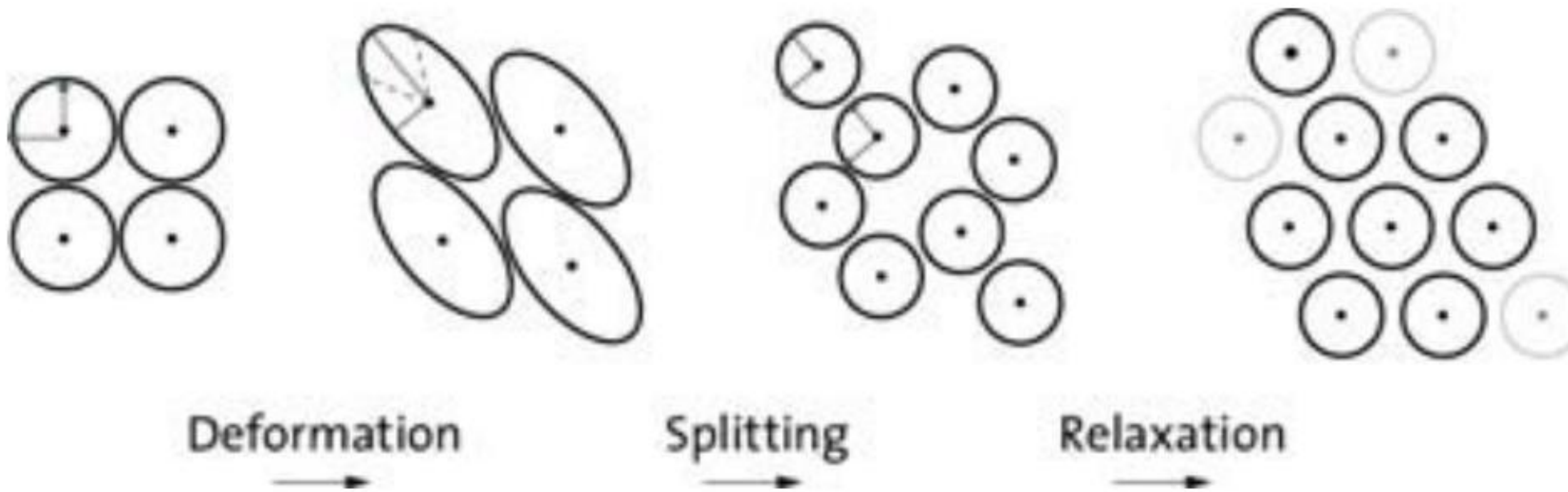


Dynamic Sampling

- Measure local surface stretch from first fundamental form
- Split samples that exceed stretch threshold
- Regularize distribution by relaxation
- Interpolate scalar attributes

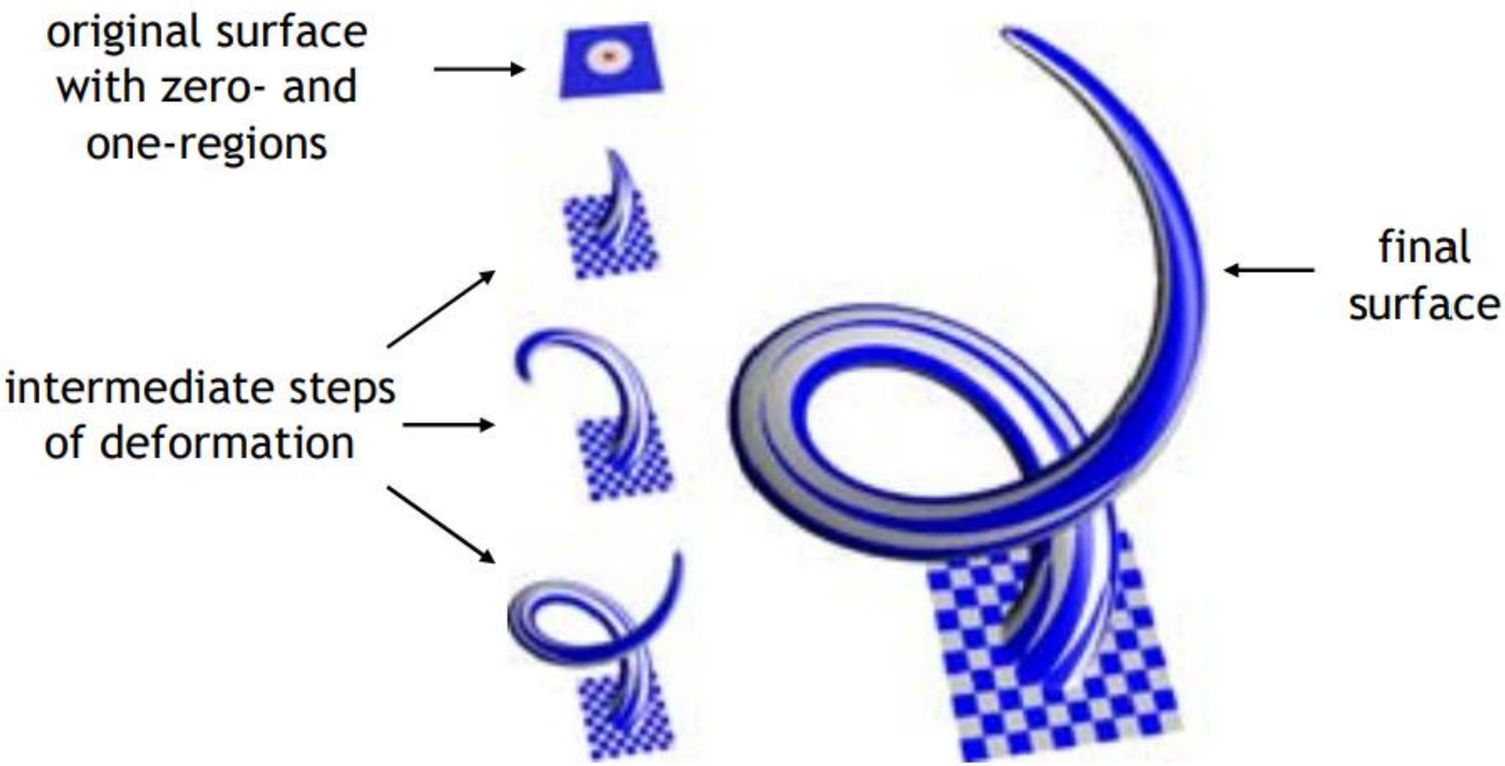
Dynamic Sampling

- 2D illustration



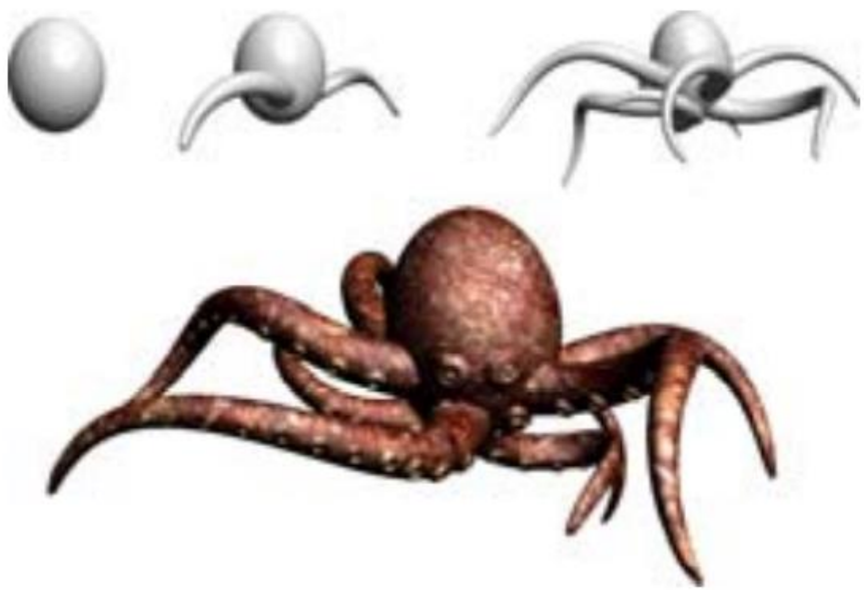
Free-form Deformation

- Interactive modeling session with dynamic sampling



Results

- Ab-initio design of an Octopus
  - Free-form deformation with dynamic sampling from 69,706 to 295,222 points

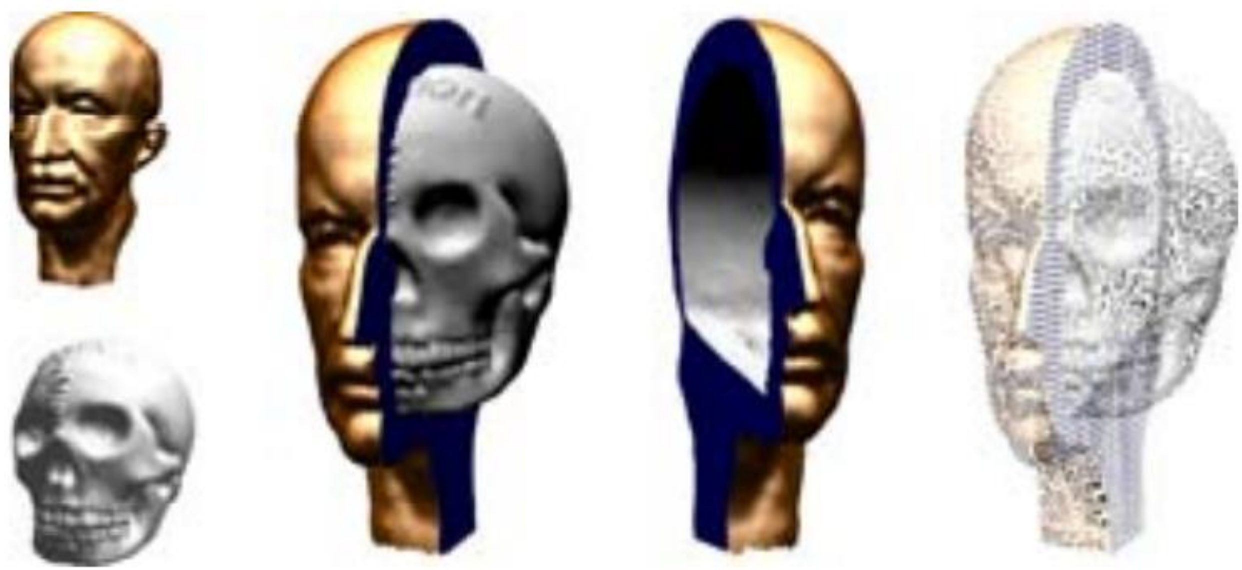


Results

- Modeling with synthetic and scanned data
  - Combination of free-form deformation with collision detection, boolean operations, particle-based blending, embossing and texturing

Results

- Boolean operations on scanned data
  - Irregular sampling pattern, low resolution models



Discussion

- Points are a versatile shape modeling primitive
  - Combines advantages of implicit and parametric surfaces
  - Integrates boolean operations and free-form deformation
  - Dynamic restructuring
  - Time and space efficient implementations

Discussion

- The power of point clouds as a shape representation for 3D deep learning is not fully utilized
  - Dynamic geometry