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
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

Parameterization

Resampling

Editing Operator
Key Components

- **Point cloud parameterization** $\Phi$
  - brings surface and brush into common reference frame
- **Dynamic resampling** $\Psi$
  - creates one-to-one correspondence of surface and brush samples
- **Editing operator** $\Omega$
  - combines surface and brush samples

$$S' = \Omega(\Psi(\Phi(S)), \Psi(B))$$

modified surface  | original surface  | brush
Parameterization

- Constrained minimum distortion parameterization of point clouds

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

correlations = matching of feature points

minimum distortion = maximum smoothness
Parameterization

- Find mapping $X$ that minimizes objective function:

$$C(X) = \sum_{j \in M} (X(p_j) - x_j)^2 + \varepsilon \int_P \gamma(u) du$$

Where:
- $p_j$ are brush points
- $x_j$ are surface points
- $\gamma(u)$ represents fitting constraints
- $\int_P \gamma(u) du$ represents distortion
Parameterization

- Measuring distortion

\[ \gamma(u) = \int_{\theta} \left( \frac{\partial^2}{\partial r^2} X_u(\theta, r) \right)^2 \, d\theta \]

- Integrates squared curvature using local polar re-parameterization

\[ X_u(\theta, r) = X \left( u + r \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \right) \]
Parameterization

- Discrete formulation:

\[
\tilde{C}(U) = \sum_{j \in M} (p_j - u_j)^2 + \varepsilon \sum_{i=1}^{n} \sum_{j \in N_i} \left( \frac{\partial U(x_i)}{\partial \tilde{v}_j} - \frac{\partial U(x_i)}{\partial v_j} \right)^2
\]

- 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

\[
\tilde{C}(U) = \sum_{j} \left( b_j - \sum_{i=1}^{n} a_{j,i} u_i \right)^2 = \| b - Au \|^2
\]
Reconstruction

- Parameterized scattered data approximation

\[
X(u) = \frac{\sum_i \Phi_i(u)r_i(u)}{\sum_i r_i(u)}
\]

- 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
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 $N$:
  \[
  C = \left( \begin{array}{c}
  p_{i_1} - \bar{p} \\
  \vdots \\
  p_{i_n} - \bar{p}
  \end{array} \right)^T \cdot \left( \begin{array}{c}
  p_{i_1} - \bar{p} \\
  \vdots \\
  p_{i_n} - \bar{p}
  \end{array} \right), \quad i_j \in N
  \]

- with centroid
  \[
  \bar{p} = \frac{1}{|N|} \sum_{i \in N} p_i
  \]
Covariance Analysis

• Consider the eigenproblem:

\[ \mathbf{C} \cdot \mathbf{v}_i = \lambda_i \cdot \mathbf{v}_i, \quad i \in \{0,1,2\} \]

• \( \mathbf{C} \) 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:
  \[ \sum_{i \in N} |p_i - \bar{p}|^2 = \lambda_0 + \lambda_1 + \lambda_2 \]

- We define surface variation as:
  \[ \sigma_n(p) = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}, \quad \lambda_0 \leq \lambda_1 \leq \lambda_2 \]

  - 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

covariance ellipsoid
split plane
centroid

root
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 $e_1 = p_i - p$ and $e_2 = e_1 \times n$
Iterative Simplification

- 2D example

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

- 2D example

<table>
<thead>
<tr>
<th>priority queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>edge</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>
Iterative Simplification

- 2D example
Iterative Simplification

- 2D example

```
<table>
<thead>
<tr>
<th>edge</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.02</td>
</tr>
<tr>
<td>2</td>
<td>0.03</td>
</tr>
<tr>
<td>14</td>
<td>0.04</td>
</tr>
<tr>
<td>5</td>
<td>0.06</td>
</tr>
<tr>
<td>9</td>
<td>0.09</td>
</tr>
<tr>
<td>1</td>
<td>0.11</td>
</tr>
<tr>
<td>13</td>
<td>0.13</td>
</tr>
<tr>
<td>3</td>
<td>0.23</td>
</tr>
<tr>
<td>11</td>
<td>0.27</td>
</tr>
<tr>
<td>10</td>
<td>0.36</td>
</tr>
<tr>
<td>7</td>
<td>0.49</td>
</tr>
<tr>
<td>4</td>
<td>0.56</td>
</tr>
</tbody>
</table>
```
Iterative Simplification

- 2D example
Iterative Simplification

- 2D example

<table>
<thead>
<tr>
<th>edge</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0.27</td>
</tr>
<tr>
<td>10</td>
<td>0.36</td>
</tr>
<tr>
<td>7</td>
<td>0.49</td>
</tr>
<tr>
<td>4</td>
<td>0.56</td>
</tr>
</tbody>
</table>
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 $F_i(p) = k(r - \|p - p_i\|) \cdot (p - p_i)$
  - 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

\[ F_i(p) = k(r - \|p - p_i\|) \cdot (p - p_i) \]
Particle Simulation

- 2D example

- Initialization
  - randomly spread particles

- Repulsion
  - linear repulsion force

\[ F_i(p) = k(r - \|p - p_i\|) \cdot (p - p_i) \]
Particle Simulation

- 2D example

- Initialization
  - randomly spread particles

- Repulsion
  - linear repulsion force
    \[ F_i(p) = k(r - \|p - p_i\|) \cdot (p - p_i) \]

- Projection
  - project particles onto surface
Particle Simulation

- 2D example

  - Initialization
    - randomly spread particles

  - Repulsion
    - linear repulsion force
    \[ F_i(p) = k(r - \|p - p_i\|) \cdot (p - p_i) \]

  - Projection
    - project particles onto surface
Particle Simulation

- Adjust repulsion radius according to surface variation \Rightarrow more samples in regions of high variation
Particle Simulation

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

- Measure the distance between two point-sampled surfaces using a sampling approach.

- Maximum error: $\Delta_{\text{max}}(S, S') = \max_{q \in Q} d(q, S')$
  - Two-sided Hausdorff distance

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

- $d(q, S)$ approximates the distance of point $q$ 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**

<table>
<thead>
<tr>
<th></th>
<th>Efficiency</th>
<th>Surface Error</th>
<th>Control</th>
<th>Implementation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical Clustering</td>
<td>+</td>
<td>-</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Iterative Simplification</td>
<td>-</td>
<td>+</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Particle Simulation</td>
<td>0</td>
<td>+</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>
Shape Modeling
Motivation

- 3D content creation pipeline
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

\[ P = \{ p_i, c_i, m_i, \ldots \} \]  

\[ \text{continuous surface } S \text{ interpolating or approximating } P \]
Surface Model

- Moving least squares (MLS) approximation (Levin, Alexa et al.)
  - Surface defined as stationary set of projection operator
    \[ S_P = \{ x \in \mathbb{R}^3 \mid \Psi_P(x) = x \} \]
  - Weighted least squares optimization
    - 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
Boolean Operations

- Rendering sharp creases
Boolean Operations

- Rendering sharp creases
Boolean Operations

- Boolean operations can create intricate shapes with complex topology

\[ A + B \quad A \cdot B \]
\[ A - B \quad B - A\]
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 $F : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ 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

zero-region

one-region

original surface
deformed surface
Free-form Deformation

- **Definition of deformation field:**
  - Continuous scale parameter $t_x$
    - $t_x = \beta \left( \frac{d_0}{d_0 + d_1} \right)$
    - $d_0$: 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

bitmap image

SIGGRAPH

zero- and one-regions

deformed surface

SIGGRAPH
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

- Boolean union performed

- Particle-based blending

- Collision detected
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

- color-coded surface stretch

- surface after dynamic re-sampling
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

original surface with zero- and one-regions

intermediate steps of deformation

final surface
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