Freeform Shape Representations for Efficient Geometry Processing

Eurographics 2003

Granada, Spain

Motivation

- Geometry Processing
  (points, wireframes, patches, volumes)
- Efficient algorithms always have to be supported by efficient data structures
  (efficient implementation of mathematical representations)
- Which representation to use for what ?

Operations on Geometric Objects

- Evaluation
  sampling positions and normals (e.g. for rendering)
- Query
  inside/outside, distance, closest point
- Modification
  geometry, topology (inner / outer)

Functional Representations

- Explicit ... $f : IN \rightarrow IR^3$
  points (or polygons)
- Parametric ... $f : IR^2 \rightarrow IR^3$
  polygons or patches
- Implicit ... $f : IR^3 \rightarrow IR$
  volumes

Functional Representations

- Explicit ... $f : IN \rightarrow IR^3$
  no consistency conditions : easy modification
- Parametric ... $f : IR^2 \rightarrow IR^3$
  domain structure: easy evaluation
- Implicit ... $f : IR^3 \rightarrow IR$
  domain structure: easy query

Basic Data Structures

- List / Graph
  sequential / topological ordering
- Array
  global index structure
- Tree / DAG
  hierarchical clustering

Basic Data Structures

- List / Graph
  „unstructured“
- Array
  „structured“
- Tree / DAG
  „hierarchical“

Basic Data Structures

- List / Graph
  difficult query
- Array
  difficult modification
- Tree / DAG
  range-based ⇒ difficult evaluation/modification
  domain-based ⇒ difficult query

Overview

- Basic types of geometry representations
- Hybrid geometry representations

Types of Representations

- functional approximation
- piecewise definition → patch size O(h)
- segments in Π_k → error $O(h^{k+1})$
  (local Taylor expansion of a smooth function)
- domain in IR^d → complexity $O(h^{-d})$
- asymptotic efficiency class : k+1 / d

Types of Representations

	unstructured	structured	hierarchical
explicit			
parametric			
implicit			

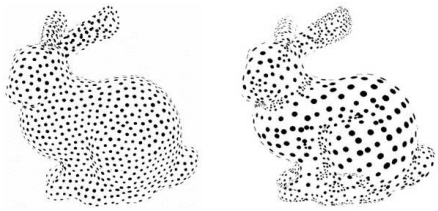
Unstructured Point Clouds

- piecewise **constant** approximation
  – sampling density = h
  – error = O(h)
  – adjust sampling density to ***surface area***

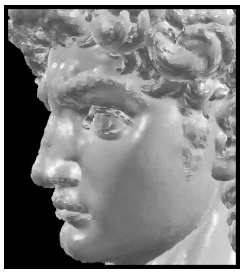
Unstructured Point Clouds

- piecewise **constant** approximation
  - sampling density = h
  - error = O(h)
  - adjust sampling density to ***surface area***
- piecewise **linear** approximation
  - surface splat = point + normal + radius
  - error $\approx \kappa_{max} h^2 \rightarrow r \approx \sqrt{\text{error} / \kappa_{max}}$
  - adjust sampling density to ***local curvature***

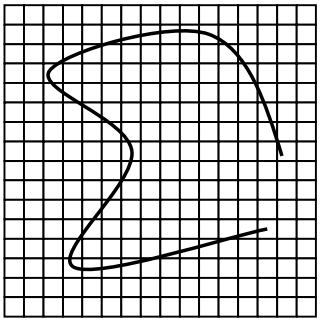
Curvature Dependent Sampling



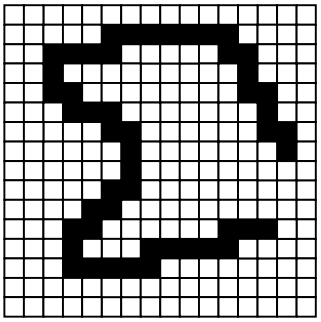
Surface Splatting



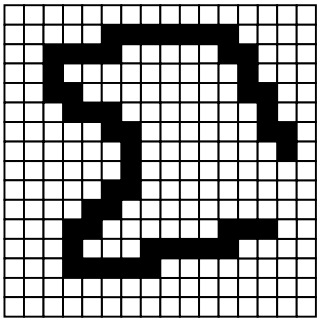
Structured Point Clouds



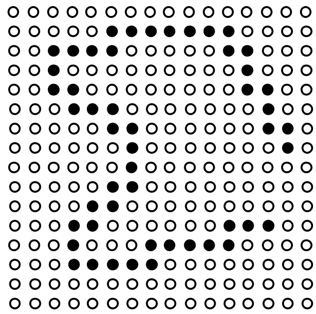
Structured Point Clouds



Structured Point Clouds



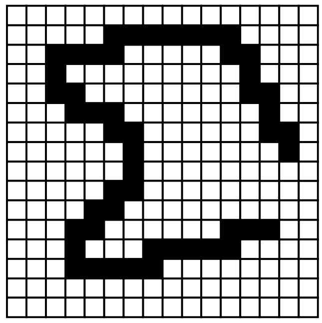
Structured Point Clouds



Quantization:
O(h)

Density:
O(h)

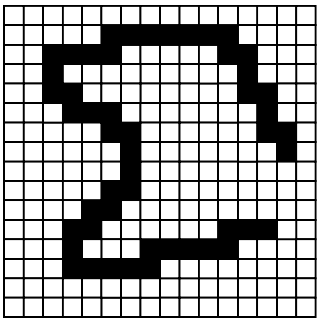
Binary Voxel Grid



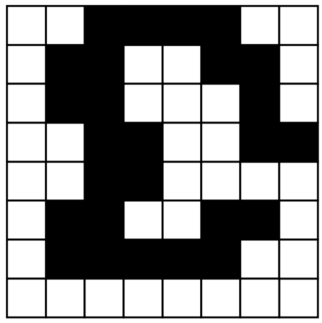
Precision:
O(h)

Complexity:
$O(h^{-3})$

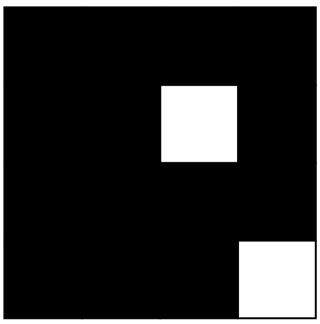
Octree Point Clouds



Octree Point Clouds



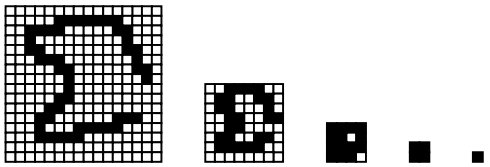
Octree Point Clouds



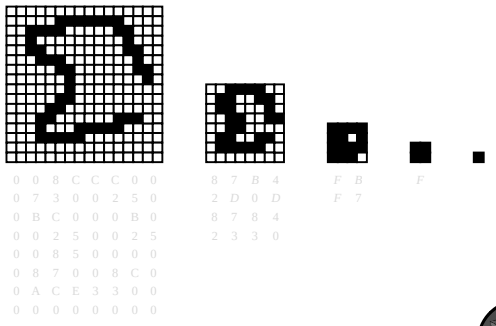
Octree Point Clouds

Octree Point Clouds

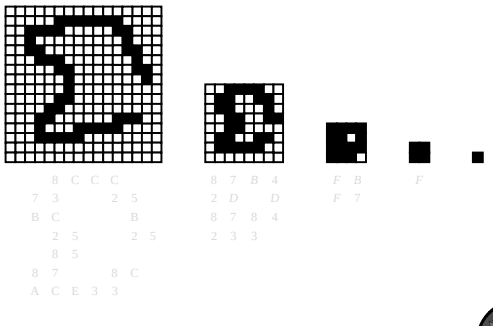
Zero Tree Coding



Zero Tree Coding



Zero Tree Coding



Octree Point Clouds

- Uniform clustering
- Sampling density = quantization precision
- Down-sample binary voxel grid
- Efficient octree representation
  - 2.67 bit per point (uncompressed)
  - 1 bit - 1.5 bit per point (entropy encoded)
  - independent of the quantization resolution

Octree Point Clouds

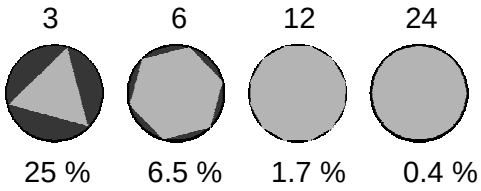
- Uniform clustering
- Sampling density = quantization precision
- Down-sample binary voxel grid
- Efficient octree representation
- Efficient rendering by octree traversal
  - 4 scalar additions + 2 divisions per point
  - precomputed lighting per normal per frame
  - 15M points per second by pure software renderer

Types of Representations

	unstructured	structured	hierarchical
explicit	point clouds	binary voxel grid	octree
parametric			
implicit			

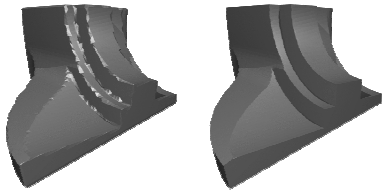
Unstructured Triangle Meshes

- Piecewise linear approximation
  - smooth surfaces



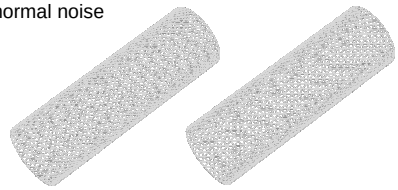
Unstructured Triangle Meshes

- Piecewise linear approximation
  - smooth surfaces
  - sharp features



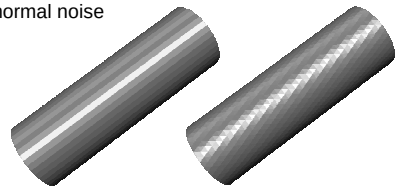
Unstructured Triangle Meshes

- Piecewise linear approximation
  - smooth surfaces
  - sharp features
  - normal noise



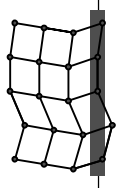
Unstructured Triangle Meshes

- Piecewise linear approximation
  - smooth surfaces
  - sharp features
  - normal noise



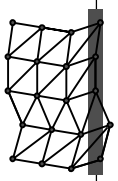
Structured Meshes → NURBS

- Piecewise polynomial / rational
- Higher approximation order
- Higher smoothness order
- Rectangular parameter domain
- Less flexibility
  - regular control mesh
  - continuity between patches
  - trimming



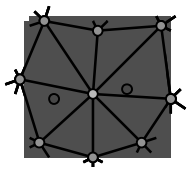
Structured Meshes → NURBS

- Piecewise polynomial / rational
- Higher approximation order
- Higher smoothness order
- Rectangular parameter domain
- Less flexibility
  - regular control mesh
  - continuity between patches
  - trimming



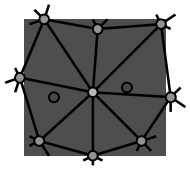
Mesh Hierarchy (F2C)

- Progressive Meshes
  - edge collapse

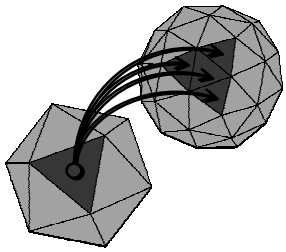


Mesh Hierarchy (F2C)

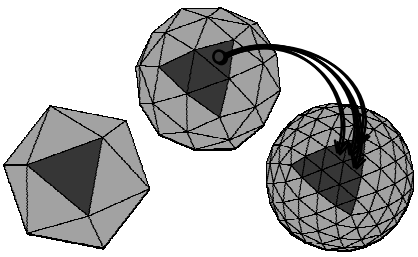
- Progressive Meshes
  - edge collapse
  - vertex tree



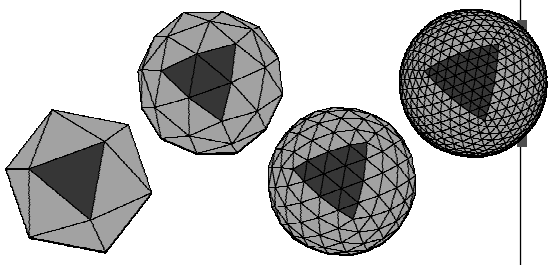
Mesh Hierarchy (C2F)



Mesh Hierarchy (C2F)



Mesh Hierarchy (C2F)



Subdivision Surfaces

- Uniform splitting operator
  - arbitrary control (base) meshes
  - 1:4 , 1:3 , 1:2, ...
  - semi-regular meshes
- Local smoothing operator
  - C^{2+} almost everywhere, isolated C^1 points
  - high approximation order
- Geometric hierarchies
  - prediction + displacement

Types of Representations

	unstructured	structured	hierarchical
explicit	point clouds	binary voxel grid	octree
parametric	triangle meshes	NURBS	subdivision surfaces
Implicit			

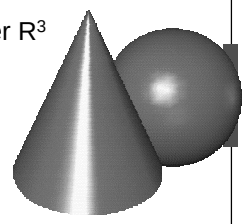
Implicit Surfaces

- Functions defined over R^3

  $x^2 + y^2 + z^2 = r^2$

  $x^2 + y^2 = z^2$

- $S = F^{-1}(0)$
- $F(x,y,z) = dist_S(x,y,z)$
- Scalar coefficients vs control vertices

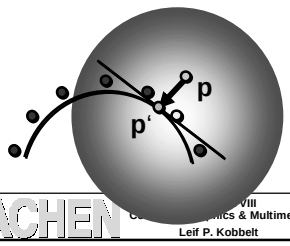


MLS Surfaces

- Projection operator $\psi : IR^3 \rightarrow IR^3$, $\psi^2 = \psi$
- Surface S = range(ψ) = $\{ \mathbf{p} \mid \psi(\mathbf{p}) = \mathbf{p} \}$
- ψ (and hence S) is defined by a set of unstructured control points $p_i \in IR^3$ (scattered over a surface)

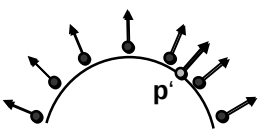
Signed Distance Formulation

- Use *linear* MLS surfaces for simplicity
  $\mathbf{p}' = \psi(\mathbf{p})$



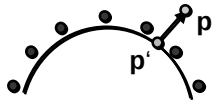
Signed Distance Formulation

- Use *linear* MLS surfaces for simplicity
  $\mathbf{p}' = \psi(\mathbf{p})$
- Normal vectors at $\mathbf{p}_i$ imply surface orientation

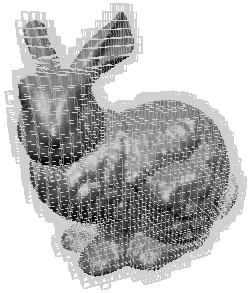


Signed Distance Formulation

- Use *linear* MLS surfaces for simplicity
  $\mathbf{p}' = \psi(\mathbf{p})$
- Normal vectors at $\mathbf{p}_i$ imply surface orientation
- $F(p) = \text{sign}(\mathbf{n}^T (\mathbf{p} - \mathbf{p}')) \parallel \mathbf{p} - \mathbf{p}' \parallel$
- $S = F^{-1}(0)$

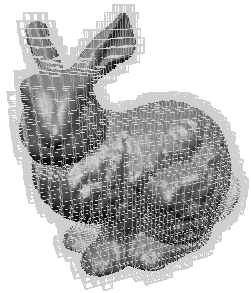


Structured Volumes → Voxel Grid



Grid topology is *trivial* and *independent* from surface topology

Structured Volumes → Voxel Grid



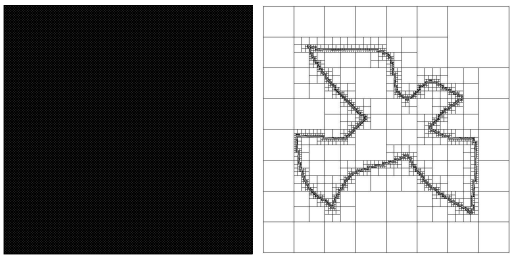
... per cell:

- binary value
- trilinear function
- higher order

Spatial Hierarchies

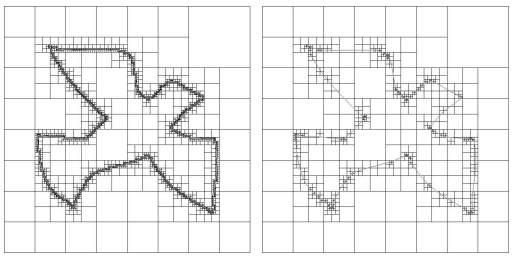
- Adaptive octree
  - Three color octree → $O(h^2)$
  - Adaptively sampled distance fields → $O(h)$
- Binary space partition
  - Partition the space along arbitrary planes
  - Piecewise **linear** C^{-1} approximation
  - Align cells to surface
    ... promote good splits
    ... prevent bad splits

Three Color Quadtree



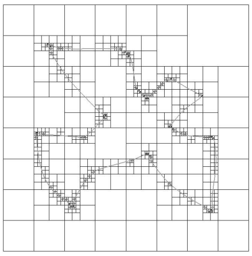
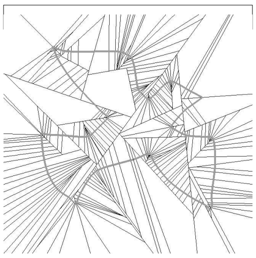
1048576 cells 12040 cells

Adaptively Sampled Distance Fields



12040 cells 895 cells

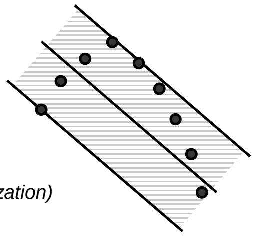
Binary Space Partitions

895 cells   254 cells

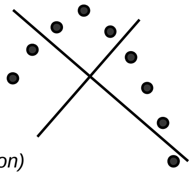
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)



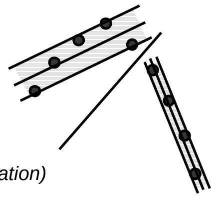
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)



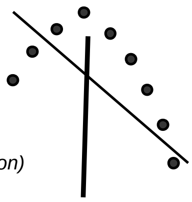
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)



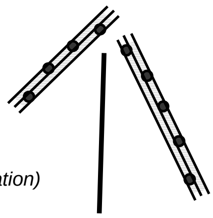
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)



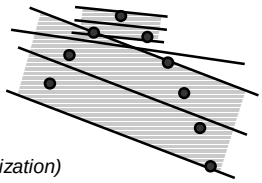
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)



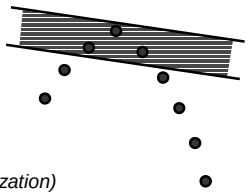
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)

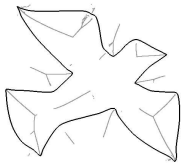


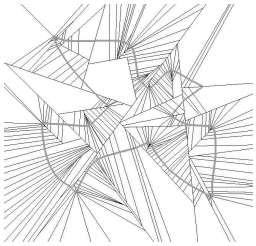
Binary Space Partitions

- Linear approximation
  (least squares fitting)
- Good splits
  (use medial axis)
- Bad splits
  (Hough space characterization)



Binary Space Partitions





254 cells

Types of Representations

	unstructured	structured	hierarchical
explicit	point clouds	binary voxel grid	octree
parametric	triangle meshes	NURBS	subdivision surfaces
implicit	MLS surfaces	3D grid	octree BSP

Overview

- Basic types of geometry representations
- Hybrid geometry representations

Combinations

- Keep several representations for the same geometry and use each for those operations where it performs best ...
  - Points
    (resampling, collision detection)
  - Patches
    (evaluation, neighborhood search, rendering)
  - Volumes
    (distance queries, topological changes, inside/outside tests)

Application Scenarios

- Meshes + Volumes
  - mesh processing with global error control
  - mesh repair
- Points + Meshes
  - r-snakes
- Points + Volumes
  - shape modeling with point clouds
  - topology control for level sets

Application Scenarios

- Meshes + Volumes
  - mesh processing with global error control
  - mesh repair
- Points + Meshes
  - r-snakes
- Points + Volumes
  - shape modeling with point clouds
  - topology control for level sets

RWTH AACHEN Informatik VIII Computer Graphics & Multimedia Leif P. Kobbelt

Mesh Processing

- Incremental decimation and smoothing
  - greedy optimization, iteration
  - take surface curvature into account
  - simple local operations
  - typical ***evaluation*** and ***modification*** task
- Global error control
  - measure deviation from original mesh
  - local checking after each operation
  - typical ***query*** task

Mesh Processing

Combine ...

– Meshes
  - surface geometry
  - incremental updating (e.g. edge collapses)

– Volumes
  - signed distance function
  - check maximum over each triangle

Mesh Processing

- Signed distance function representation
  - piecewise tri-linear, C^0, octree based *(voxelize each triangle, check in each voxel)*
  - exact maximum difficult to estimate *(piecewise cubic function)*
  - piecewise linear, C^{-1}, BSP based *(tree traversal, splitting triangle into convex pieces)*
  - exact maximum easy to estimate *(piecewise linear function)*

Mesh Processing

- Triangle mesh represents the current mesh during decimation and smoothing

  ⇨ easy evaluation and modification

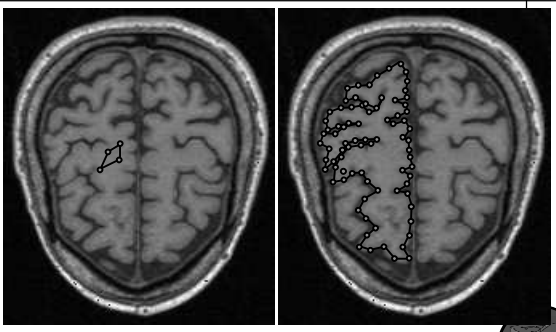
- Hierarchical signed distance function represents the (distance to) original input mesh

  ⇨ easy query

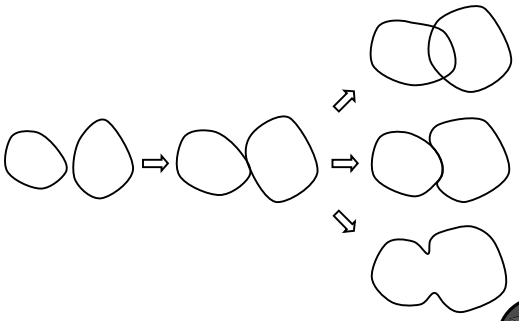
Application Scenarios

- Meshes + Volumes
  - mesh processing with global error control
  - mesh repair
- Points + Meshes
  - r-snakes
- Points + Volumes
  - shape modeling with point clouds
  - topology control for level sets

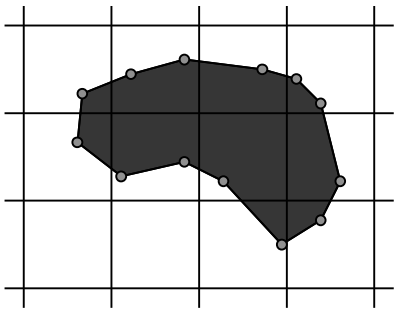
Resampling / Reparametrization



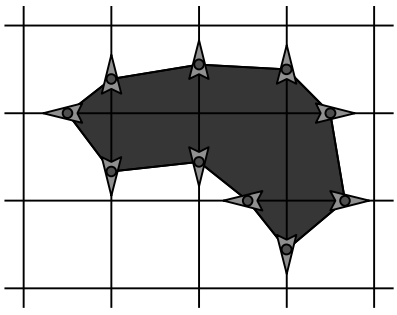
Collision Handling



Snakes

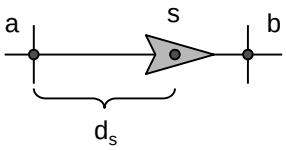


R-Snakes



R-Snakes

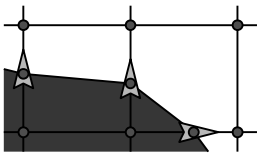
1. Each snaxel lives on a grid segment



$s = (1 - d_s)\, a + d_s\, b, \quad d_s \in [0,1)$

R-Snakes

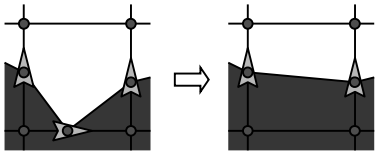
1. Each snaxel lives on a grid segment
2. All snaxels are consistently oriented



Each snaxel points from the inside to the outside.

R-Snakes

1. Each snaxel lives on a grid segment
2. All snaxels are consistently oriented



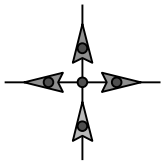
Reparametrization

When do we have to *insert* and/or *delete* snaxels from the contour ?

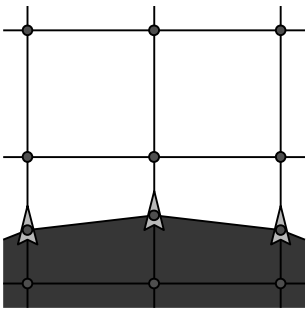
Snaxel Splitting

Snaxel splitting:

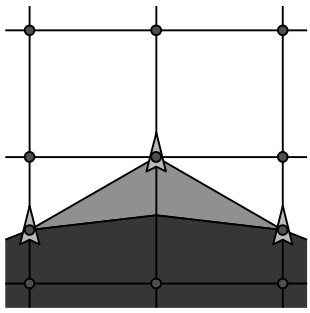
Whenever a snaxel crosses a grid point, it is split into 3 new snaxels.



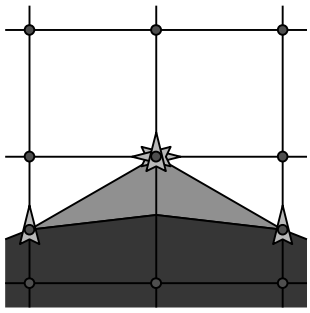
Snaxel Splitting



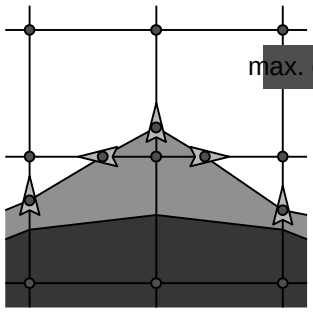
Snaxel Splitting



Snaxel Splitting



Snaxel Splitting



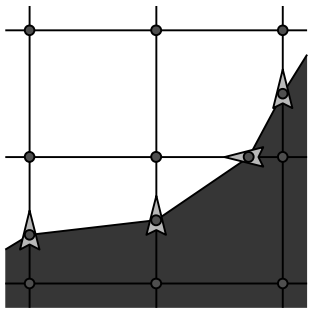
max. edge length
$\sqrt{2}\ h$

Snaxel Removal

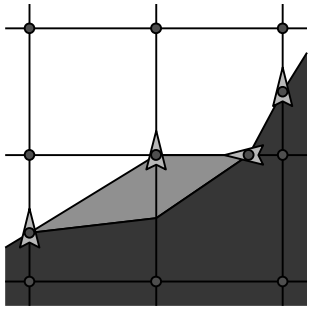
Snaxel removal:

After each snaxel split the consistency of the r-snake might be violated.
This is fixed by local snaxel removal.

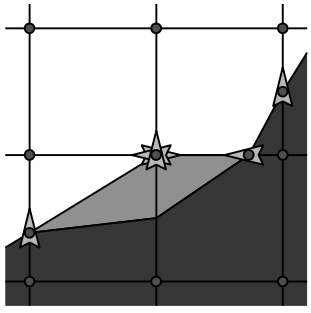
Snaxel Removal



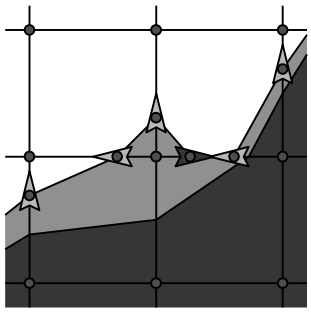
Snaxel Removal



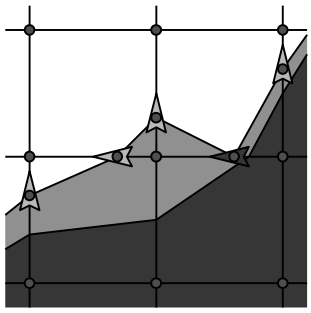
Snaxel Removal



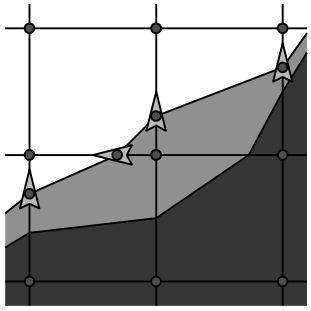
Snaxel Removal



Snaxel Removal



Snaxel Removal

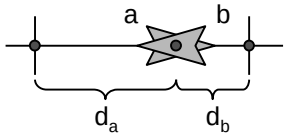


Collision Handling

Collision detection:

Two snaxels a and b collide, if

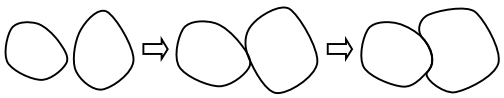
- they share the same grid segment
- $d_a + d_b \geq 1$



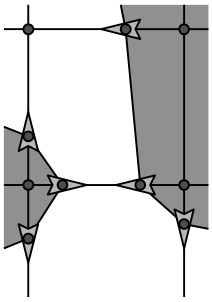
Topology Control

Topology *preservation*:

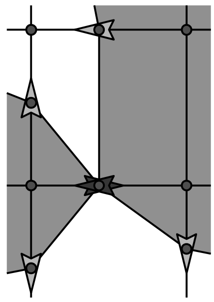
Colliding contours “touch” each other.



Topology Control

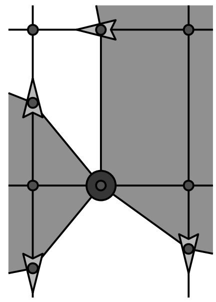


Topology Control



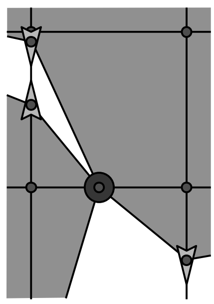
Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

Topology Control



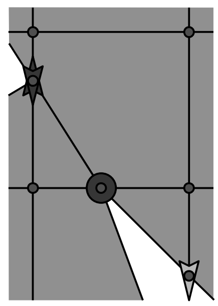
Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

Topology Control



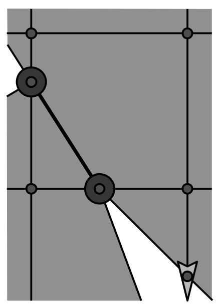
Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

Topology Control



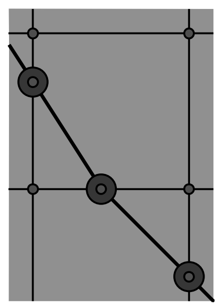
Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

Topology Control



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

Topology Control

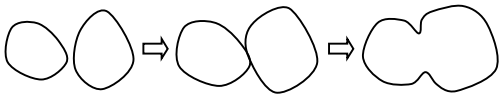


Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

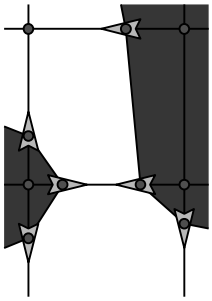
Topology Control

Topology *change*:

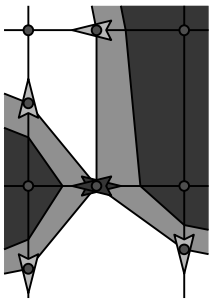
Colliding contours “merge”.



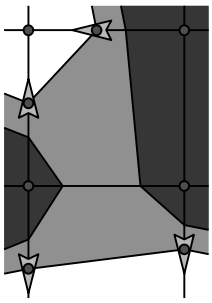
Topology Control



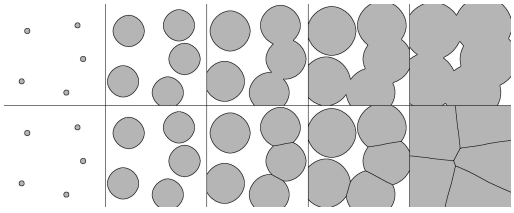
Topology Control



Topology Control



Results



R-Snakes

- Polygonal contour follows image and regularization forces

  ⇨ easy evaluation and modification

- Snakel-particles move on grid edges to detect collisions

  ⇨ easy queries and modification

Application Scenarios

- Meshes + Volumes
  - mesh processing with global error control
  - mesh repair
- Points + Meshes
  - r-snakes
- Points + Volumes
  - shape modeling with point clouds
  - topology control for level sets

Shape Modeling

- Local deformation
  - good on explicit and parametric representations
  - strong deformation requires re-structuring
- Boolean Operations
  - good on implicit representations
  - sharp feature preservation ?

Point-based Representation

- Randomly distribute particles on the given surface (per unit area)
- Repulsion forces generate uniform particle density (iterative relaxation)

  $\mathbf{p}_i \leftarrow P(\mathbf{p}_i + \alpha F(\mathbf{p}_i))$

  or curvature dependent particle density

  $\mathbf{p}_i \leftarrow P(\mathbf{p}_i + \alpha(C(\mathbf{p}_i)) F(\mathbf{p}_i))$

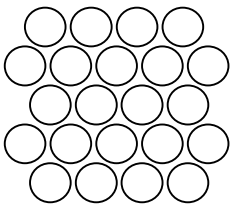
  $\mathbf{p}_i \leftarrow P(\mathbf{p}_i + \alpha F(\mathbf{p}_i) + \beta G(\mathbf{p}_i))$

Dynamic Resampling

- Shape deformation requires to update the sampling pattern
- Measure surface stretch
  (eigenvalues of the first fundamental form)
- Split & collapse particles to re-establish uniform sampling density

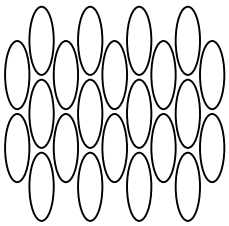
Dynamic Resampling

uniform sample distribution



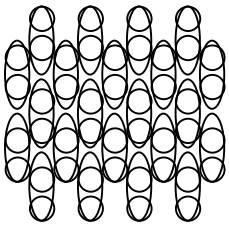
Dynamic Resampling

stretching



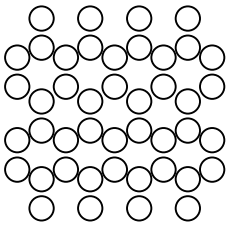
Dynamic Resampling

Splitting



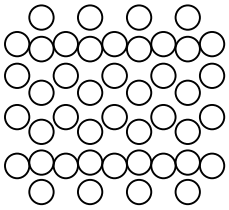
Dynamic Resampling

Relaxation



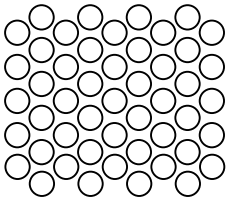
Dynamic Resampling

Relaxation



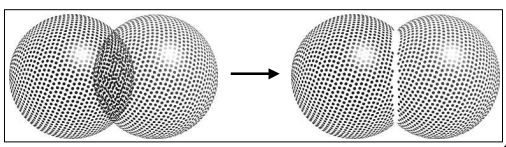
Dynamic Resampling

Relaxation



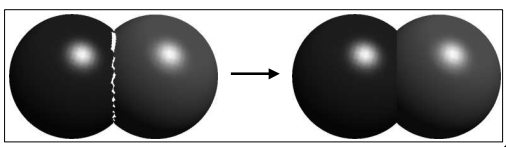
CSG for Point Clouds

- Boolean operations based on MLS signed distance function
  - union($\{a_i\},\{b_i\}$) $= \{a_i \mid SD_b(a_i) > 0\} \cup \{b_i \mid SD_a(b_i) > 0\}$
  - intersect($\{a_i\},\{b_i\}$) $= \{a_i \mid SD_b(a_i) \leq 0\} \cup \{b_i \mid SD_a(b_i) \leq 0\}$
  - ...



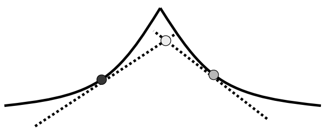
CSG for Point Clouds

- Sharp feature sampling
  - avoid alias error
  - tangent plane information $\rightarrow$ Newton iteration
  - render „clipped“ splats



CSG for Point Clouds

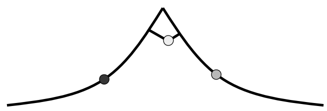
- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

CSG for Point Clouds

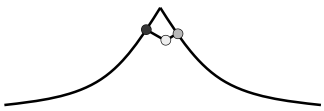
- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

CSG for Point Clouds

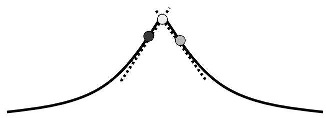
- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

CSG for Point Clouds

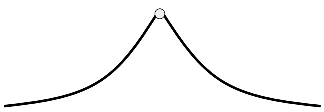
- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

CSG for Point Clouds

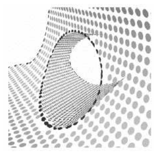
- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

CSG for Point Clouds

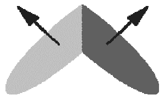
- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



Informatik VIII
Computer Graphics & Multimedia
Leif P. Kobbelt

CSG for Point Clouds

- Sharp feature sampling
  - avoid alias error
  - tangent plane information → Newton iteration
  - render „clipped“ splats



CSG for Point Clouds

- Point cloud does not have to preserve consistent manifold surface topology

  ⇨ easy modification

- MLS surface‘s signed distance function enables inside/outside tests

  ⇨ easy queries

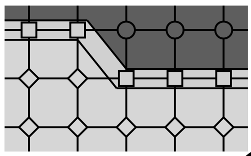
Topology Control for Level Sets

- Level sets are a *volumetric* representation
  - $\phi(x, y, z, \boldsymbol{t}) = 0$
  - $\partial\phi / \partial \boldsymbol{t} = -\,\text{speed}(x,y,z,\boldsymbol{t})\ \|\nabla_{x,y,z}\phi\|$
- Simple deformation without parametrization artifacts
  - offset surfaces (constant speed)
  - curvature flow evolution
- Topology control ?

  (surface sheets in an implicit representation merge accidentially if distance is less than voxel resolution)

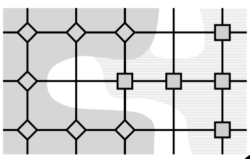
Topology Control for Level Sets

- Detect self-collisions of the surface
- Disallow sign changes in critical voxels
- Discretization artifacts
- Incorrect evolution



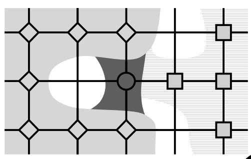
Topology Control for Level Sets

- Detect self-collisions of the surface
- Disallow sign changes in critical voxels
- Discretization artifacts
- Incorrect evolution



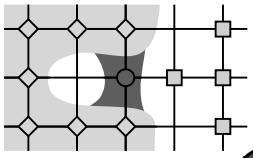
Topology Control for Level Sets

- Detect self-collisions of the surface
- Disallow sign changes in critical voxels
- Discretization artifacts
- Incorrect evolution



Topology Control for Level Sets

- Detect self-collisions of the surface
- Disallow sign changes in critical voxels
- Discretization artifacts
- Incorrect evolution



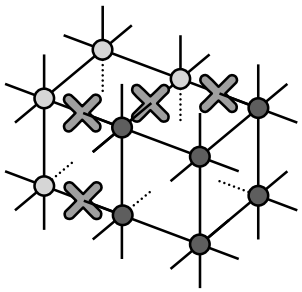
Topology Control for Level Sets

- Detect self-collisions of the surface
- Let *particles* evolve with the level set
- Make sure particles collide when self-collisions occur
- Number and position of particles ?
  (resampling, collision handling)

Cut Tag Grids

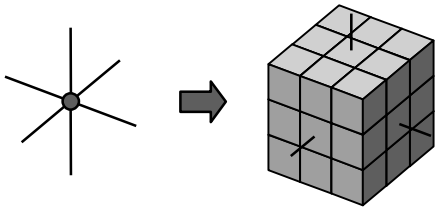
- Use standard level set evolution
- When a grid value changes its sign locally check for topology changes
- Place *particles* („cut-tags“) on grid edges to prevent topology changes

Cut Tag Grids



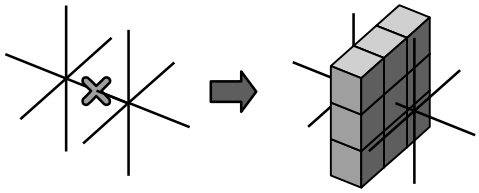
Topology Reconstruction

- Embed tagged grid into (binary) voxel grid with double resolution ...



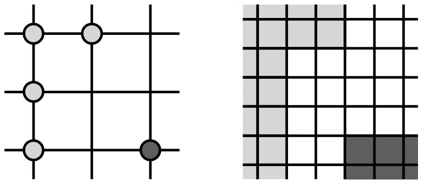
Topology Reconstruction

- Embed tagged grid into (binary) voxel grid with double resolution ...



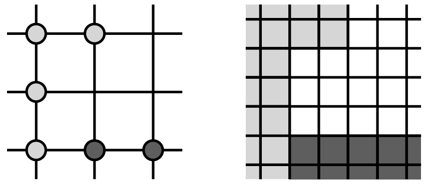
Topology Reconstruction

- *Set* all full voxel
- *Unset* all empty voxel
- *Unset* all cut-tags



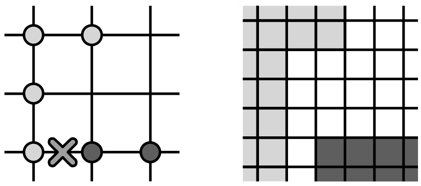
Topology Reconstruction

- *Set* all full voxel
- *Unset* all empty voxel
- *Unset* all cut-tags



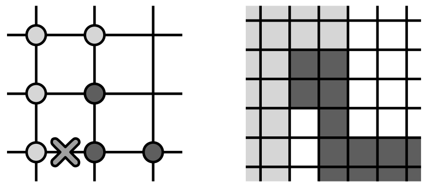
Topology Reconstruction

- *Set* all full voxel
- *Unset* all empty voxel
- *Unset* all cut-tags



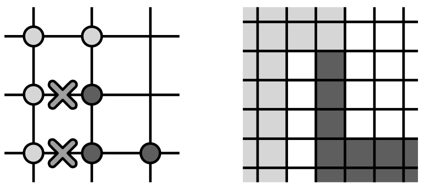
Topology Reconstruction

- *Set* all full voxel
- *Unset* all empty voxel
- *Unset* all cut-tags



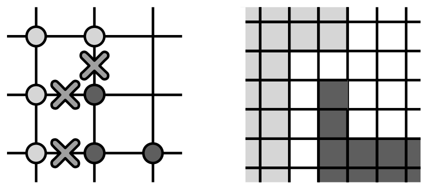
Topology Reconstruction

- *Set* all full voxel
- *Unset* all empty voxel
- *Unset* all cut-tags



Topology Reconstruction

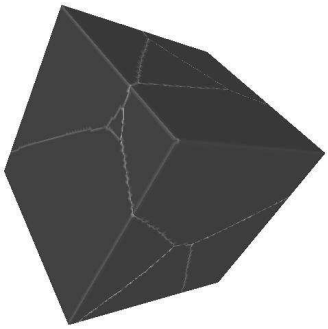
- *Set* all full voxel
- *Unset* all empty voxel
- *Unset* all cut-tags



Geometry Reconstruction

- Data structure
  - 3D voxel grid *(single resolution)*
  - binary 3D edge grid *(single resolution)*
  - double resolution embedding never has to be generated explicitly
- For each 3 x 3 x 3 block of the embedding
  - apply marching cubes
  - local mesh decimation
  - collision position estimation

Results



Topology Control for Level Sets

- Level sets can evolve freely without parameterization artifacts

  ⇨ easy modification

- Particles on 3D grid edges enable collision detection and topology control

  ⇨ easy evaluation

Application Scenarios

- Meshes + Volumes
  - mesh processing with global error control
  - mesh repair
- Points + Meshes
  - r-snakes
- Points + Volumes
  - shape modeling with point clouds
  - topology control for level sets

Conclusions

- Basic **representations**
  implicit *parametric* *explicit*
- Basic **operations**
  query *evaluation* *modification*
- Basic **data structures**
  unstructured *structured* *hierarchical*
- **Efficiency**
- **Robustness**

Thanks

- ACG team ...
  Sergey Alekseyev, Loic Barthe, Stephan Bischoff, David Bommes, Mario Botsch, Philipp Cerfontaine, Alexander Hornung, Thomas Huppertz, James Kessel, Junho Kim, Jonathan Klein, Martin Marinov, Jan Möbius, Darko Pavic, Oliver Schall, Abhijit Sovakar, Christoph Vogel, Jianhua Wu
- ETH Zürich
  Mark Pauly, Richard Keiser, Markus Gross