A Fast and Simple Surface Reconstruction Algorithm

Siu-Wing Cheng, Jiongxin Jin and Man-Kit Lau

Department of Computer Science and Engineering
Hong Kong University of Science and Technology

Jun 17, 2012
Surface reconstruction

point cloud

triangular mesh
Surface reconstruction

**Given:** a “dense” point set sampled from an unknown surface $\Sigma \subset \mathbb{R}^3$

**Goal:** to compute a surface mesh that is

- topologically equivalent
  - homeomorphic to $\Sigma$
- geometrically faithful
  - small normal deviation
  - small Hausdorff distance

Applications:

- reverse engineering, medical imaging, computer graphics, ...
ε-sample

Medial axis $\mathcal{M}_\Sigma$

- closure of the set of points having at least two closest points in $\Sigma$

Local feature size $\text{lfs}(x)$

- $\text{lfs}(x) = d(x, \mathcal{M}_\Sigma)$

ε-sample $P$

- $\forall x \in \Sigma, \ d(x, P) \leq \varepsilon \text{lfs}(x)$
Previous results

- Crust, PowerCrust, Cocone: $O(n^2)$, need to compute 3D DT/VD

- Output-sensitive 3D VD algorithm: $O((n + f) \log^2 n)$
  - Uniform sample from generic smooth surface: $f = O(n \log n)$
    Attali, Boissonnat and Lieutier [SCG ’03]
  - Non-generic surface: $f = \Omega(n \sqrt{n})$ even for uniform sample
    Erickson [DCG ’03]

- Funke and Ramos [SODA ’02]: $O(n \log n)$ (not practical)
1. Extract a locally uniform $O(\varepsilon)$-sample $S \subseteq P$. 

Reconstruct a surface using $S$. $O(n \log n)$, Dey, Funke and Ramos [EuroCG '01]

Add back $P \setminus S$. $O(n \log n)$, Cheng, Jin and Lau (HKUST)
FR algorithm

1. Extract a locally uniform $O(\varepsilon)$-sample $S \subseteq P$.

2. Reconstruct a surface using $S$.
   - $O(n \log n)$, Dey, Funke and Ramos [EuroCG '01]

$O(n \log n)$, Dey, Funke and Ramos [EuroCG '01]

$O(n \log n)$, Dey, Funke and Ramos [EuroCG '01]
FR algorithm

1. Extract a locally uniform $O(\varepsilon)$-sample $S \subseteq P$.

2. Reconstruct a surface using $S$.
   - $O(n \log n)$, Dey, Funke and Ramos [EuroCG ’01]

3. Add back $P \setminus S$.
   - $O(n \log n)$
FR algorithm

1. Extract a locally uniform $O(\varepsilon)$-sample $S \subseteq P$.
   - $O(n \log n)$
   - well-separated pair decomposition
   - approximate range searching
   - approximate directional nearest neighbors

2. Reconstruct a surface using $S$.
   - $O(n \log n)$, Dey, Funke and Ramos [EuroCG ’01]

3. Add back $P \setminus S$.
   - $O(n \log n)$
FR algorithm

1. Extract a locally uniform $O(\varepsilon)$-sample $S \subseteq P$.
   - $O(n \log n)$
   - well-separated pair decomposition
   - approximate range searching
   - approximate directional nearest neighbors

   $\Rightarrow$ Octrees

2. Reconstruct a surface using $S$.
   - $O(n \log n)$, Dey, Funke and Ramos [EuroCG ’01]

3. Add back $P \setminus S$.
   - $O(n \log n)$
Our results

Given an $\varepsilon$-sample $P$, compute a locally uniform $O(\varepsilon)$-sample $S \subseteq P$ in $O(n \log n)$ time with a single octree.

surface reconstruction algorithm:

- $O(n \log n)$, optimal in the pointer machine model
- performance:
  - for non-uniform samples: 51% to 68% faster than Cocone
  - for locally uniform samples: still faster than Cocone
Locally uniform sample $S$

$B_q$: largest empty ball with center in $\Sigma$ and boundary through $q$

$$\forall q \in S, |B(q, \beta r_q) \cap S| = O(1)$$
Octree decomposition

Root cell
- smallest bounding cube of $P$

Splitting rule
- split a splittable leaf cell into eight children

Balancing rule
- split a leaf cell $C$ if it has a neighbor $C'$ s.t. $\ell_{C'} < \ell_C/2$

Apply the two rules alternately until the tree stops growing.
splitting …
splitting ...
splitting
splitting ...
splitting ...
splitting . . . balancing . . .
Example

splitting . . . balancing . . . splitting . . .
splitting . . . balancing . . . splitting . . . balancing . . .
splitting . . . balancing . . . splitting . . . balancing . . . done
Properties of the octree

- \(O(n)\) size, \(O(n \log n)\) construction time

- Balanced: side lengths of neighboring leaf cells differ by at most a factor 2

- Non-empty leaf cells have side lengths \(O(\varepsilon_{lfs})\)
  (but can be much smaller than \(\varepsilon_{lfs}\))
Trim the tree so that the sizes of non-empty leaf cells are “good”
- side lengths $O(\varepsilon \text{lfs})$
- the union of their const-factor expansions covers the surface

Smooth out the sizes of non-empty leaf cells

Pick one point from each non-empty leaf cell locally uniform $O(\varepsilon)$-sample
Trim the tree so that the sizes of non-empty leaf cells are “good”
- side lengths $O(\epsilon_{lfs})$
- the union of their const-factor expansions covers the surface
1. Trim the tree so that the sizes of non-empty leaf cells are “good”
   - side lengths $O(\varepsilon \text{lf}s)$
   - the union of their const-factor expansions covers the surface

2. Smooth out the sizes of non-empty leaf cells
   - no cell $C'$ intersecting $\kappa$-factor expansion of $C$ is smaller than half of $C$ ($\kappa = 2$ in the example)
   - can be done in linear time by processing non-empty leaf cells in decreasing order in their sizes
Trim the tree so that the sizes of non-empty leaf cells are “good”
- side lengths $O(\varepsilon_{lfs})$
- the union of their const-factor expansions covers the surface

Smooth out the sizes of non-empty leaf cells
- no cell $C'$ intersecting $\kappa$-factor expansion of $C$ is smaller than half of $C$ ($\kappa = 2$ in the example)
- can be done in linear time by processing non-empty leaf cells in decreasing order in their sizes
1. Trim the tree so that the sizes of non-empty leaf cells are “good”
   - side lengths $O(\varepsilon \text{lfs})$
   - the union of their const-factor expansions covers the surface

2. Smooth out the sizes of non-empty leaf cells
   - no cell $C'$ intersecting $\kappa$-factor expansion of $C$ is smaller than half of $C$ ($\kappa = 2$ in the example)
   - can be done in linear time by processing non-empty leaf cells in decreasing order in their sizes

3. Pick one point from each non-empty leaf cell
   - locally uniform $O(\varepsilon)$-sample
Overview

1. Trim the tree so that the sizes of non-empty leaf cells are “good”
   - side lengths $O(\varepsilon_{\text{Lfs}})$
   - the union of their const-factor expansions covers the surface

2. Smooth out the sizes of non-empty leaf cells
   - no cell $C'$ intersecting $\kappa$-factor expansion of $C$ is smaller than half of $C$ ($\kappa = 2$ in the example)
   - can be done in linear time by processing non-empty leaf cells in decreasing order in their sizes

3. Pick one point from each non-empty leaf cell
   - locally uniform $O(\varepsilon)$-sample
Tree trimming

Trim the tree so that the sizes of non-empty leaf cells are “good”:

1. estimate surface normal at \( p \in C \)
   \( H \): approximate tangent plane

2. look for an empty cube with center in \( H \) and side length \( \frac{1}{8} \ell_C \) in close neighborhood of \( C \)

3. if normal estimation fails or an empty cube is found, then make parent(\( C \)) a new leaf cell, and perform the checking on parent(\( C \)) (\( \ell_C = O(\varepsilon lfs) \) in this case)
Normal estimation

Normal estimation at $p \in C$:

1. Pick $p_i \in P \cap R_i$, for $i \in [1, 5^3]$.
2. Find $p_i, p_j$, s.t. $\angle p_i p p_j \in [\theta_0, \pi - \theta_0]$.
3. $\tilde{n}_p = n_{p_i p p_j}$.
Locally uniform subsample

Cocone on $P$

Cocone on $S$
Locally uniform subsample

Cocone on $P$

Cocone on $S$
Performance (non-uniform input)

- **Cocone on \( P \)**
- **Extraction**
- **Cocone on \( S \)**
- **Insertion**

Cheng, Jin and Lau (HKUST)

Surface reconstruction

Jun 17, 2012
Extensions

- multiple surfaces

- handling noise

- $k$-dimensional manifold in $\mathbb{R}^d$