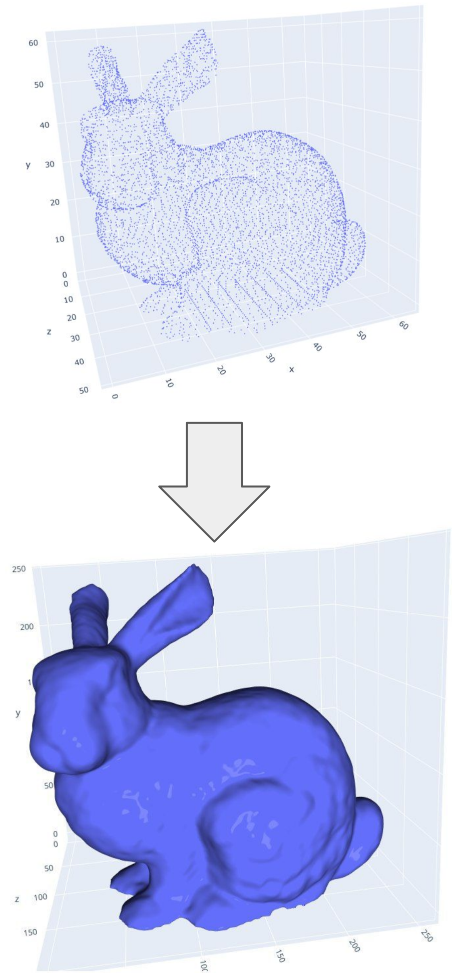
Implementation of “Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds Without Normal Information”

By

Jasmin Hoffmann
Michael Käser

Hornung, A. and Kobbelt, L., 2006, June. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Symposium on geometry processing* (pp. 41-50).



Objective

Given

Point cloud with irregularities

Goal

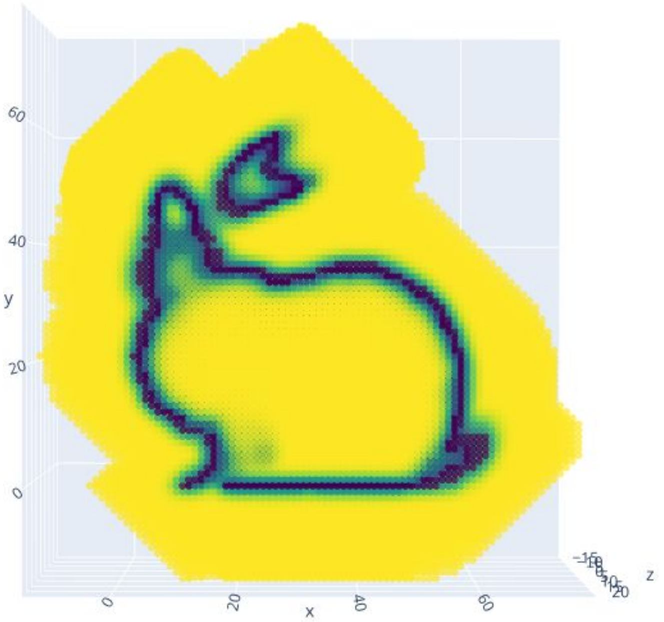
Watertight mesh of the object's surface

Core Idea

The **minimum** of an **unsigned distance function Φ** is the reconstructed mesh.

No information about normal orientation is needed.

Φ is constructed from a point cloud P.

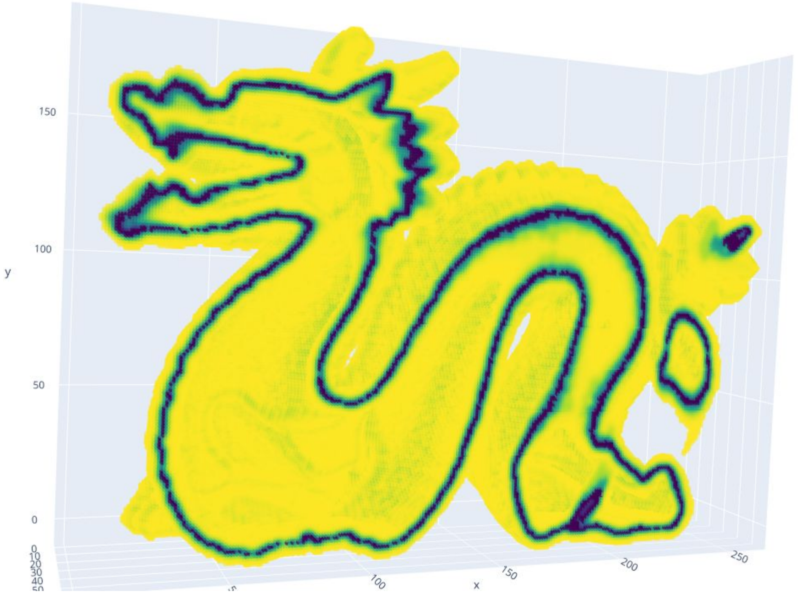


Cut through the volumetric unsigned distance function of a bunny

Approach of the Paper

Voxel-based approach to solve for the minimum of the distance function Φ

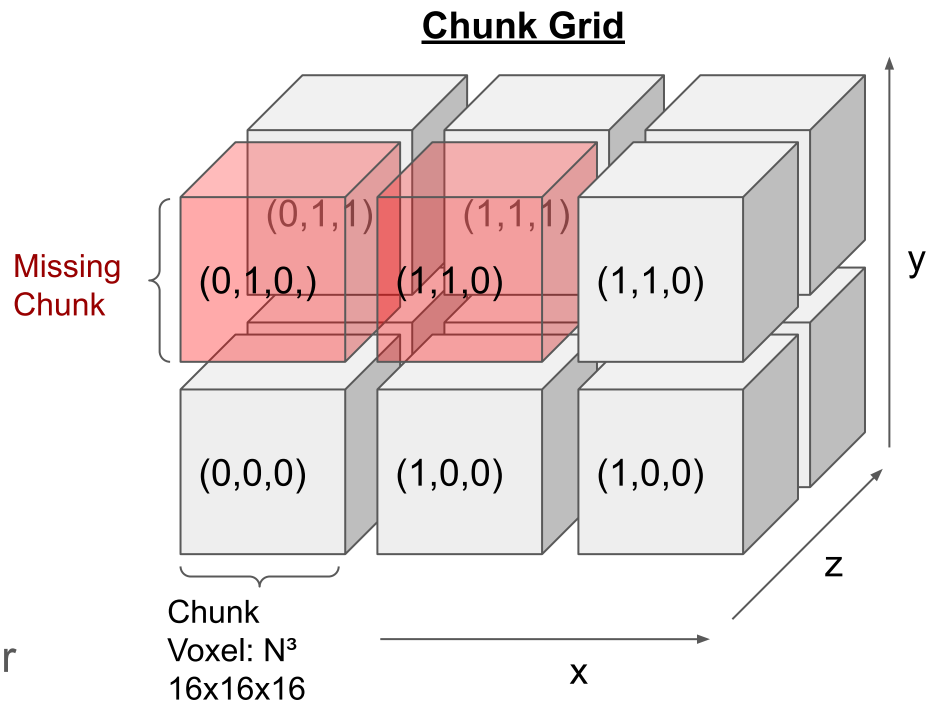
1. Insert point cloud
2. Grow a crust to close holes
3. Compute distance function Φ in crust
4. Solve Min-Cut of a graph through the crust
5. Repeat with higher resolution from 1,
   or
   Extract mesh from Min-Cut



Cut through the volumetric unsigned distance function of the dragon

Data Structure - Chunk-Based Voxel System

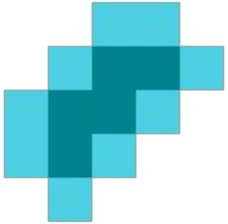
- Chunks indexed by tuple (int,int,int)
- Chunk Grid
  - Flat Chunk Map
    - ChunkIndex -> Chunk
  - Fill value
    - Value where a chunk is missing
- Chunk is either:
  - Filled with a single value
  - Voxel array N^3, (e.g. 16x16x16)
- Fast access to Data O(1)
- Numpy-array like: Operator, Setter, Getter



Surface Confidence Estimation

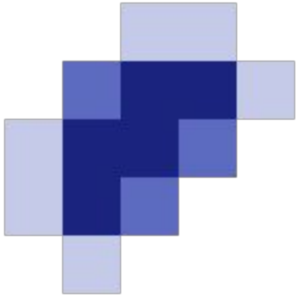
Build **crust V_{crust}**

- Consisting of surface candidates
- **Close holes** and create outer surface and inner core **V_{ext} and V_{int}**
- Add voxels with point samples to V_{crust}
- Iterate:
  - **Binary dilation** to expand crust
  - **Flood-fill** to estimate remaining components
- Stop iteration when |components| = 2
- Undo certain number of dilation steps (~3)



Surface Confidence Estimation

Diffusion

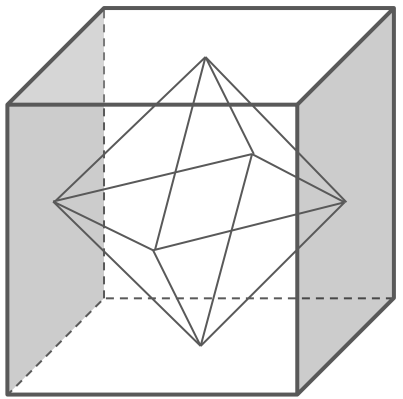


- Estimate **confidence** φ(v) of crust voxels
- Voxels with point samples fixed at $\varphi(v) = 0$
- Other voxels start with $\varphi(v) = 1$
- Iterative **averaging over 6-neighborhood** *N(v)* of each voxel (~3 iterations)

Graph-based Surface Extraction

Min-cut

- Construct min-cut problem from V_{crust}
- Graph consists of one **octahedron** per voxel
- Edge weight is $w(v) = \phi(v)^s + a$
  - *W.r.t* $s = 4, a = 10^{-5}$
- Solve (PyMaxflow)



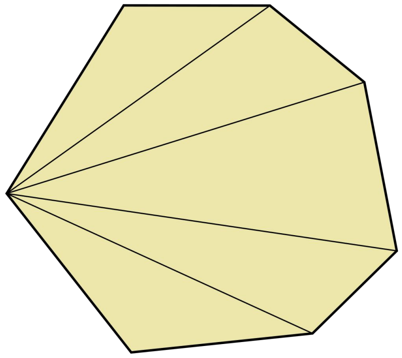
Extract **surface S_{Opt}**

- Voxels with at least one cut edge

Mesh Extraction

Extract faces from min-cut

- Iterate over all 2x2x2 blocks in crust
  - Create **polygonal face** if 3 voxels in S_{Opt}
- Convert polygonal faces to **triangle fans**



Triangle Fan (Wikipedia)

Bi-Laplacian Smoothing

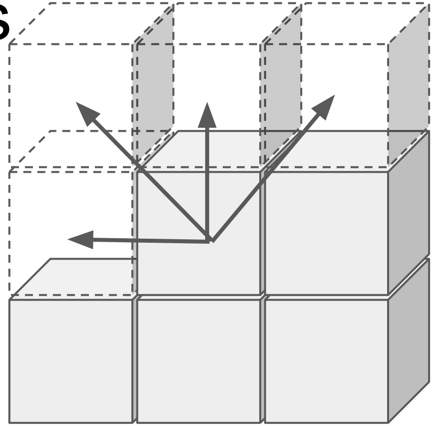
- Compute **Laplacian matrix** (PyTorch3D)
- Multiply with vertices to apply **umbrella operator**
- Stop movement earlier for **confident vertices**

$$v \leftarrow v - \frac{1}{d}\triangle^2 v \quad , \quad d = 1 + \frac{1}{n_v}\sum_j n_{v,j}$$

$$U(v) = \frac{1}{n}\sum_{i=0}^{n-1} v_i - v$$

$$U^2(v) = \frac{1}{n}\sum_{i=0}^{n-1} U(v_i) - U(v)$$

Medial Axis Approximation with Normal Cones



Problem:

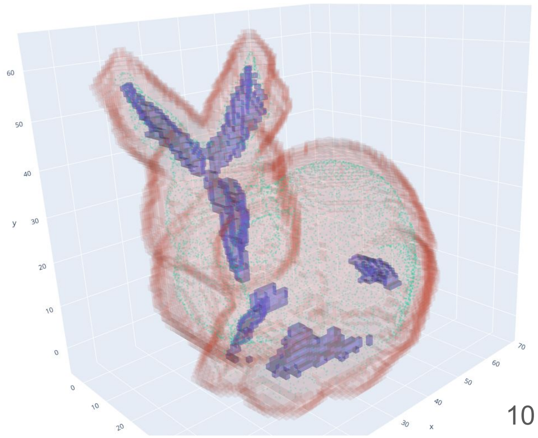
- The inner component can be missing in thin or sparse objects.

Solution:

- Label all voxels on the medial axis as V_{inner}

Approximate medial axis

- Propagate average normals inwards
- Mark voxel if opening angle of normal cone > 90°
- Estimate initial normals on the crust as average of the vectors to neighboring empty voxels.



Processor Intel(R) Core(TM) i7-9700KF CPU @ 3.60GHz, 3600 Mhz, 8 Core(s), 8 Logical Processor(s)
RAM 32.0 GB

Runtime

Bunny 30.571 point samples

Resolution 64^3 ~ 50 sec

- Dilation 11 sec
- Crust 12 sec
- Diffusion 2 sec
- Min-cut 6 sec
- Mesh Extraction 5 sec
- Smoothing 4 sec

Resolution 128^3 ~ 100 sec

- Volumetric Refinement 8 sec
- Crust 30 sec
- Diffusion 3 sec
- Min-cut 28 sec
- Mesh Extraction 8 sec
- Smoothing 20 sec

Dragon 437.645 point samples

Resolution 64^3 ~ 55 sec

- Dilation 10 sec
- Crust 21 sec
- Diffusion 3 sec
- Min-cut 8 sec
- Mesh Extraction 8 sec
- Smoothing 5 sec

Resolution 128^3 ~ 131 sec

- Volumetric Refinement 10 sec
- Crust 58 sec
- Diffusion 3 sec
- Min-cut 32 sec
- Mesh Extraction 8 sec
- Smoothing 20 sec

References

- Hornung, A. and Kobbelt, L., 2006, June. Robust reconstruction of watertight 3 d models from non-uniformly sampled point clouds without normal information. In *Symposium on geometry processing* (pp. 41-50).
- Desbrun, M., Meyer, M., Schröder, P. and Barr, A.H., 1999, July. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (pp. 317-324).
- Kobbelt, L., Campagna, S., Vorsatz, J. and Seidel, H.P., 1998, July. Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (pp. 105-114).

Images

- https://commons.wikimedia.org/w/index.php?curid=56535716 (19.1.2021)

Interactive Models

https://mickare.de/dff/reconstruction/

Appendix

Newer better approaches?

- Poranne, R., C. Gotsman, and D. Keren. "3D Surface Reconstruction Using a Generalized Distance Function." *Computer Graphics Forum* 29.8 (2010).

Libraries

● Numpy	- Matrix & Math
● Scipy	- Dilation, Binary Propagation
● PyTorch, PyTorch3D	- Laplacian Smoothing
● PyMaxflow	- MinCut in a Graph
● Plotly	- Plotting
● PyWavefront, PlyFile	- Model Loading
● tqdm	- Progress bar (printing in terminal)

Hierarchical Hole Filling and Detail Preservation

Computing crust and confidence impractical for **high resolutions.**

Approximate S_{Opt} on **low resolution** and refine:

- Start with low resolution and compute S_{Opt}
- **Split voxels** to higher resolution
- **Reinsert** point samples into V_{crust}
- Dilate reinserted voxels to **merge them with V_{crust}**
- Continue with diffusion

Mesh Extraction: Face Orientation

Reusing the normal estimation on a surface from the Medial axis approximation.

1. Pre-compute normal estimation in S_{opt}
2. Extract triangles for each block
3. Flip triangle face order when the average voxel normals is opposite to the face orientation

Data Structure - Operators

- Math operators
  - Applies the operator on the grid’s fill value and on each chunk
  - O(N)
  - Example Binary-AND:
    - ChunkGrid (fill=True) AND ChunkGrid(fill=False) ===>> ChunkGrid(fill=False)
    - Chunk(fill=True) AND Chunk(fill=True) ===>> Chunk(fill=True)
    - Chunk(voxels) AND Chunk(fill=True) ===>> Chunk(voxels)
- Custom filters
  - Iterate over chunks
  - Pad voxels if needed (take from neighboring voxels)
  - Example:
    - Dilation of a chunk takes the neighboring voxels of the neighboring chunks

Data Structure - Usage

- Filter Masks
  - Data type: Boolean
  - Usage in: Model, Dilation, Flood-Fill
- Component Detection
  - Data type: Integer
  - Usage in: Component detection with Flood-Fill
- Value Propagation
  - Data type: Float-Vector 3
  - Usage in: Normal propagation for medial axis approximation

Mesh Extraction: Block Traversal