The Ball Pivoting Algorithm

Ioannis Stamos

Points vs. Meshes

- Point-based vs. Mesh-based representation.
  - + Fast rendering for complicated point-sets.
  - + No loss of information (all points used).
  - + No need for modeling of every tiny scene detail.
  - - Rendering quality degrades when viewer zooms-in.
  - - Computation waste in mainly planar areas.
  - - Solid modeling operations not straightforward
  - - Not supported in commercial modeling/rendering software (Maya, etc.)

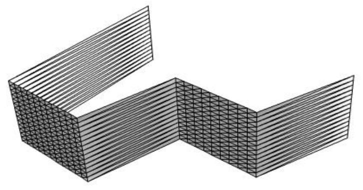
Meshing Algorithms

- **Input:** Range Images
  - Range Image = Set of 3D points
- **Output:** Single mesh = topologically correct set of triangles (not triangle soup)

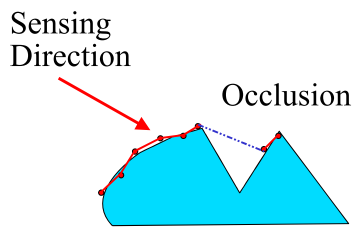
Range Sensor Imaging Characteristics

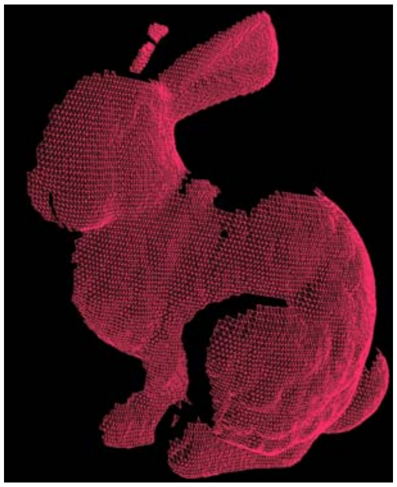
Range Image:
rectangular sampling

Image has structure
surface elements
occlusion elements



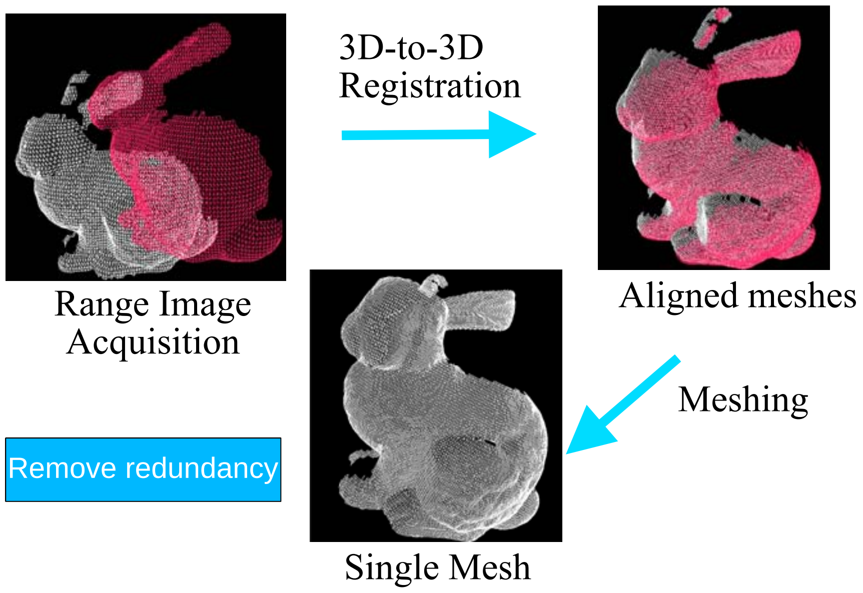
From Points to Mesh [Single Image]





Discard long edges ?
Threshold distance ?

3D-modeling pipeline

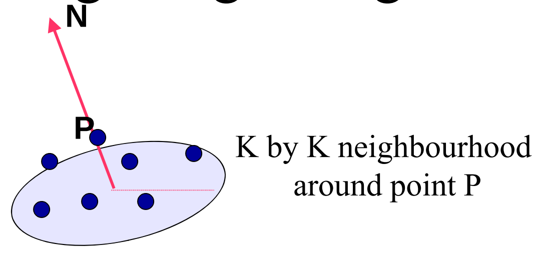


Methodology

- Each range image is preprocessed

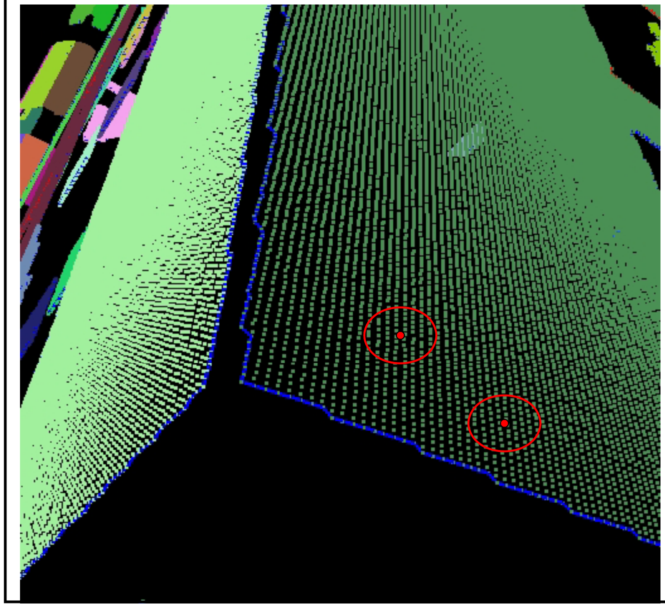
  3D points **P** => 3D points with associated normals: (**P**, **N**)
- Bounding box enclosing all images computed
- Ball Pivoting Algorithm:
  - Input: 3D points and normals from all range images
  - Output: Triangular mesh

Preprocessing Range Images



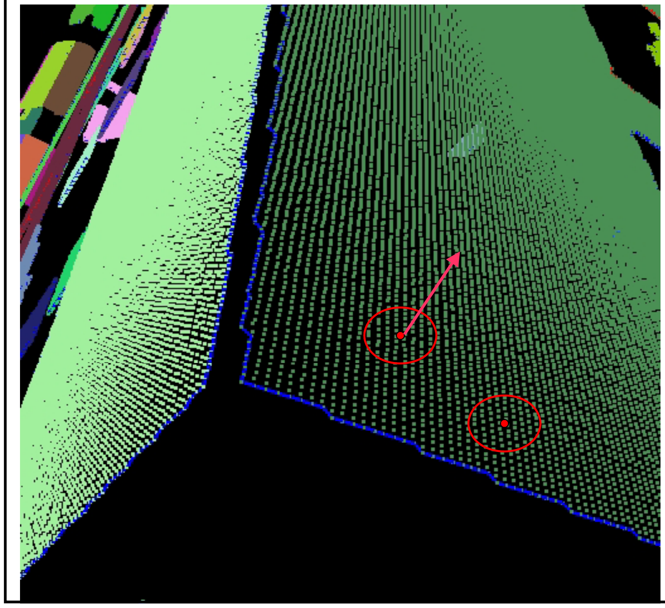
•Patch fit in K by K neighborhood around point P
•Eigenvector corresponding to smallest eigenvalue: normal
•Magnitude of smallest eigenvalue: confidence in normal
•Normal computed for every 3D point
•Structure of range images used for K by K neighbourhood

Neighborhood used for normal computation



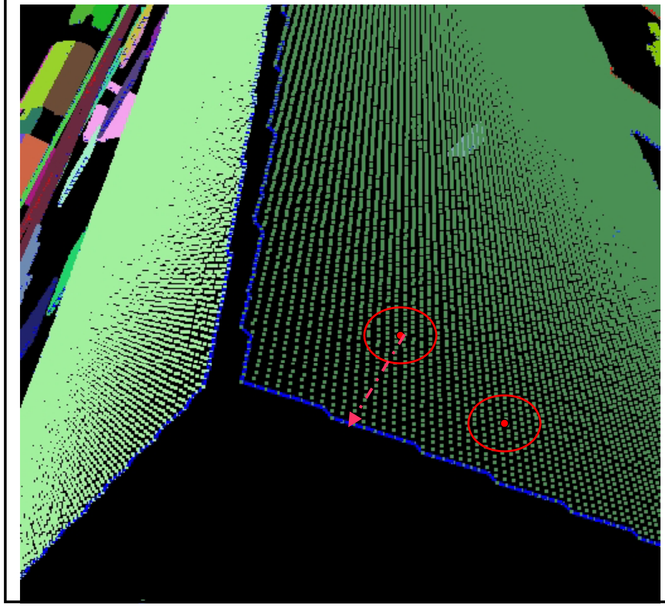
Accurately computed normals in locally planar regions

Neighborhood used for normal computation



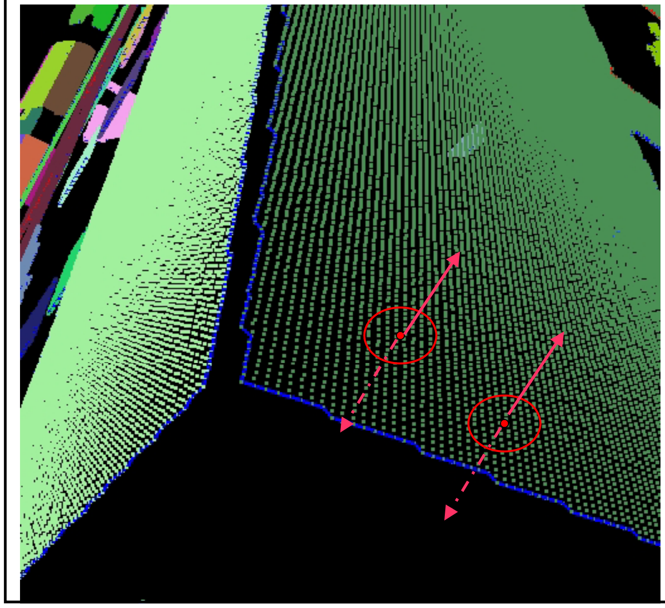
Two possible directions [towards outside]

Neighborhood used for normal computation



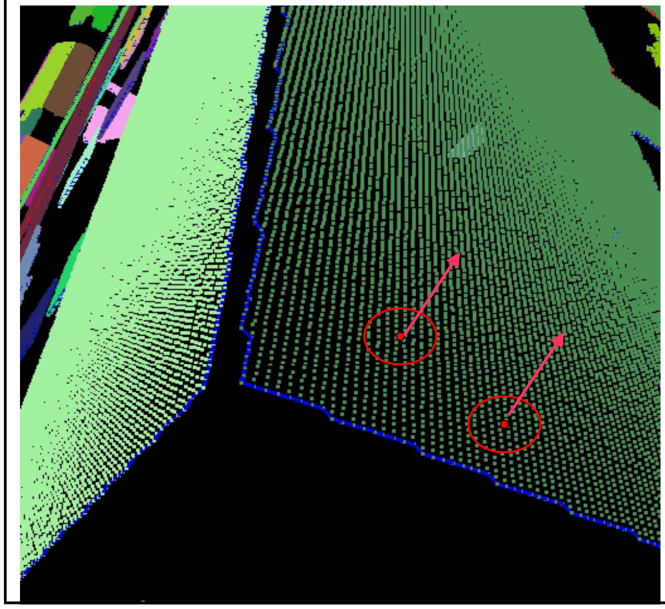
Two possible
directions
[towards inside]

Neighborhood used for normal computation



Consistent normal
direction
for all points

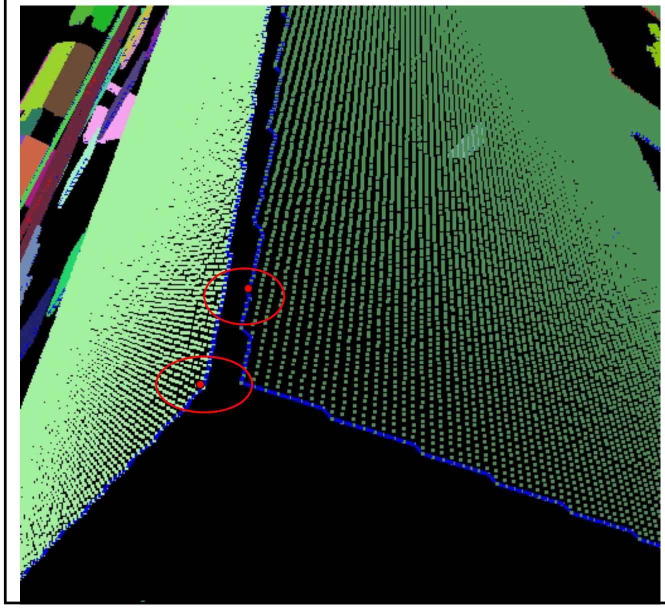
Neighborhood used for normal computation



Consistent normal direction
for all points

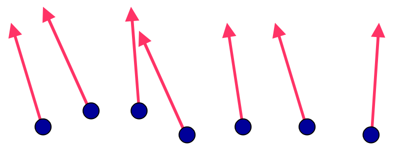
Choose outside orientation

Neighborhood used for normal computation



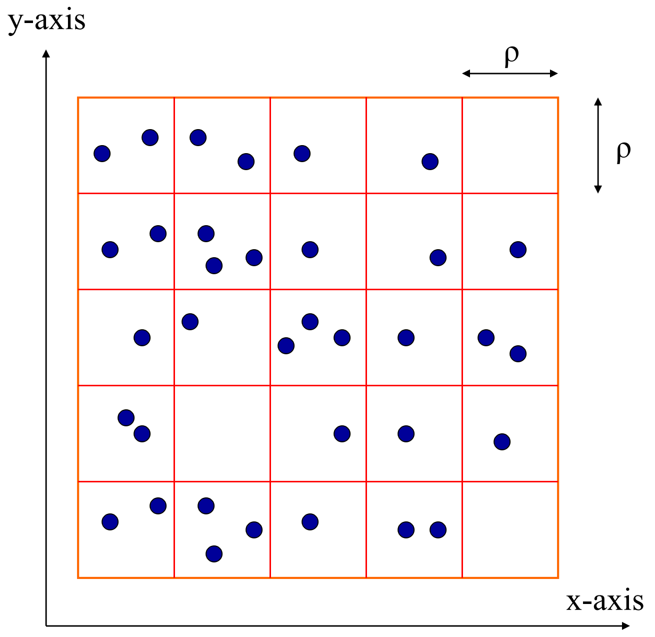
Inaccurately computed normals
close to boundaries

Ball Pivoting Algorithm [Input]



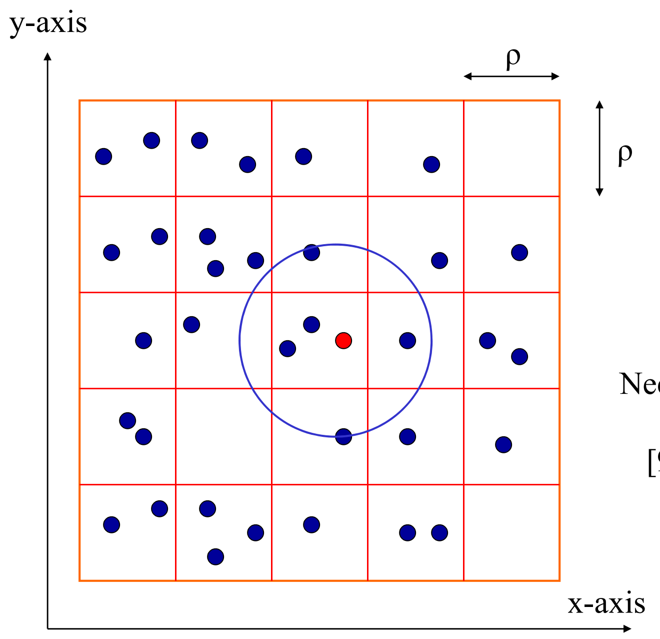
- Points/normals of all range images in data structure
- Connectivity of points in range images not needed anymore
- Radius ρ selected by the user.

Voxel-based data structure



Fast search for points within circle of radius ρ

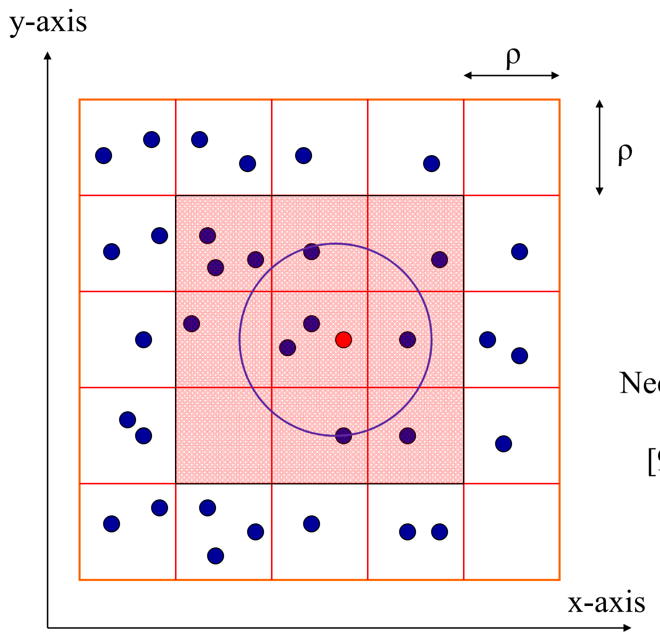
Voxel-based data structure



Fast search for points within circle of radius ρ

Need to search only in adjacent voxels [9 in this picture – 27 in 3D space]

Voxel-based data structure

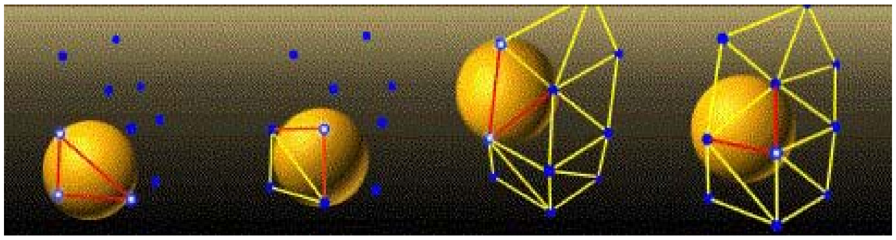


Fast search for points within circle of radius ρ

Need to search only in adjacent voxels [9 in this picture – 27 in 3D space]

Basic Operations

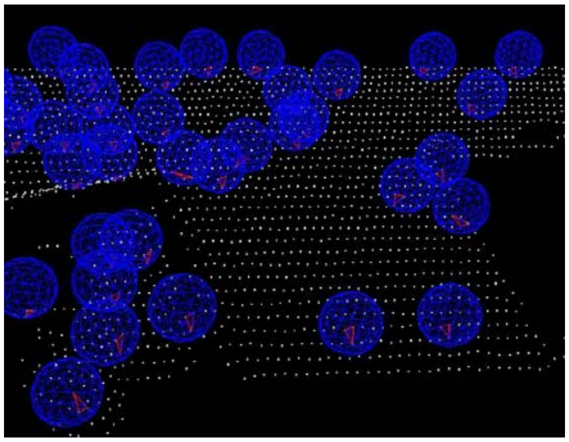
F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin.
The ball-pivoting algorithm for surface reconstruction.
IEEE Trans. on Vis. and Comp. Graph. 5 (4), pages 349-359, October-December 1999.



*A sequence of ball-pivoting operations. From left to right: A **seed triangle** is found; **pivoting** around an edge of the current front adds a new triangles to the mesh; after a number of pivoting operations, the **active front** closes on itself; a final ball-pivoting completes the mesh.*

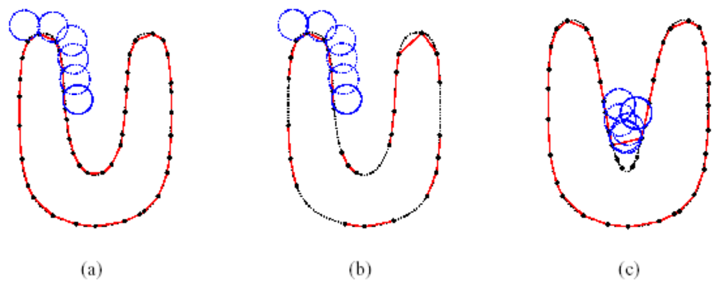
Closely related to alpha-shapes, Edelsbrunner 94

Seed triangles



A number of seed triangles with their associated spheres shown

Pivoting in 2D



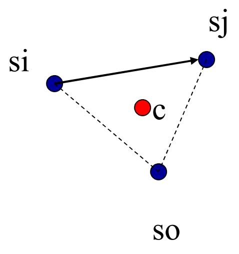
(a) Circle of radius ρ pivots from point to point, connecting them with edges.

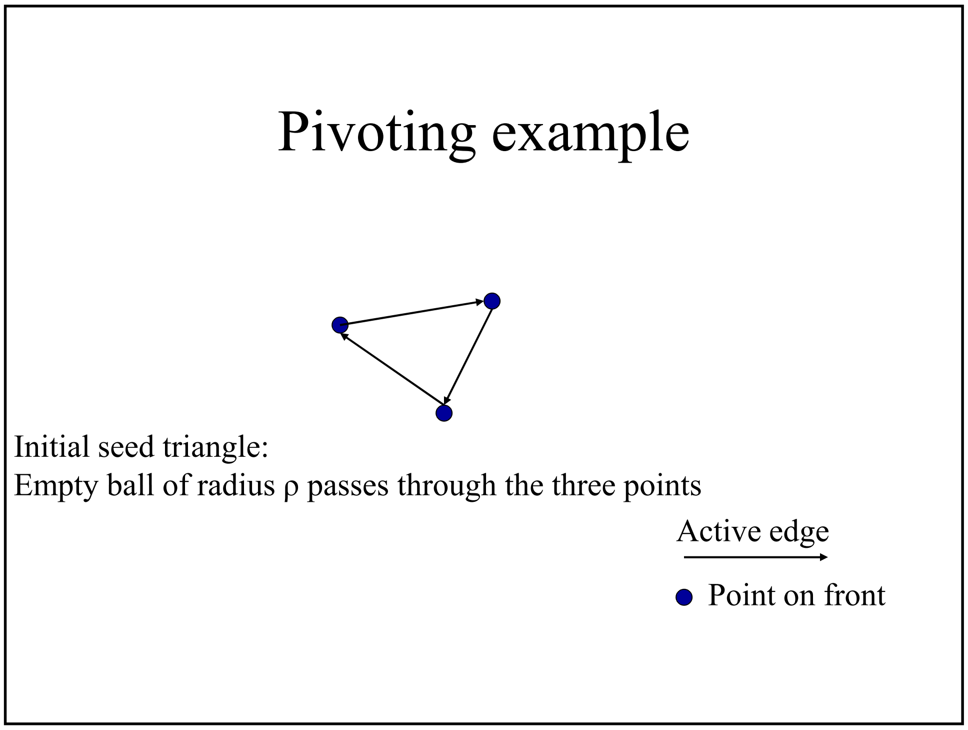
(b) When sampling density is low, some of the edges will not be created, leaving holes.

(c) When the curvature of the manifold is larger than 1/ρ, some of the points will not be reached by the pivoting ball, and features will be missed.

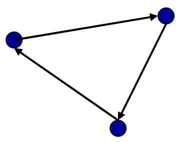
The algorithm [Edge representation]

- Edge (si, sj)
  - Opposite point so, center of empty ball c
  - Edge: “Active”, “Boundary”, or “Frozen”





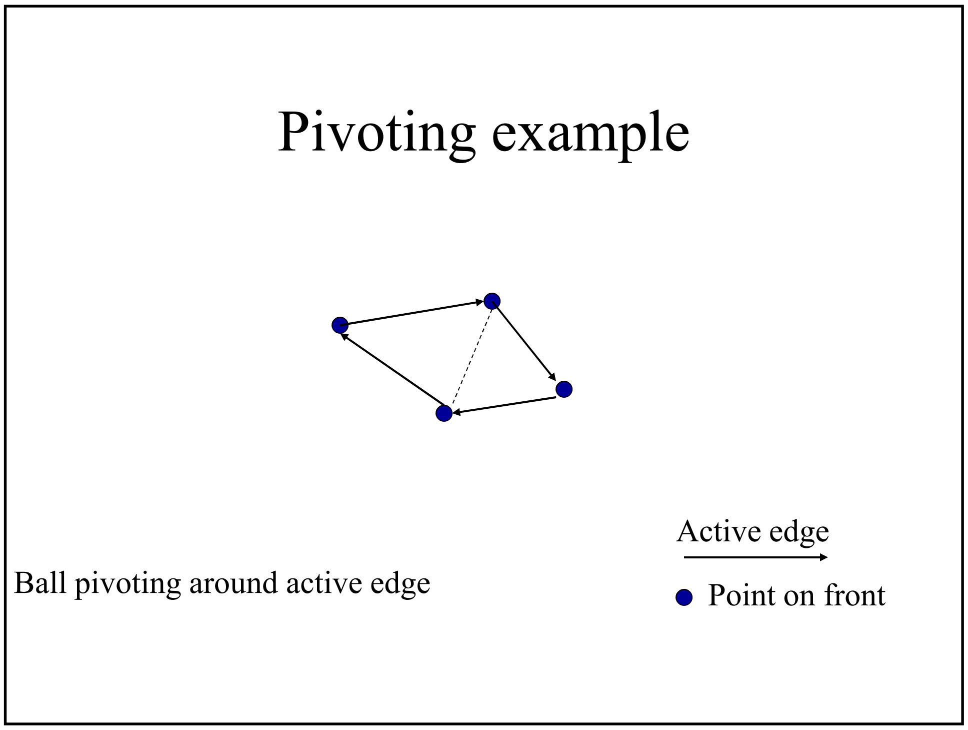
Pivoting example



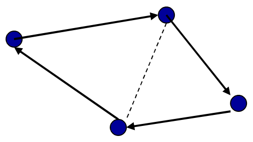
Initial seed triangle:
Empty ball of radius ρ passes through the three points

Active edge
$\longrightarrow$

● Point on front



Pivoting example

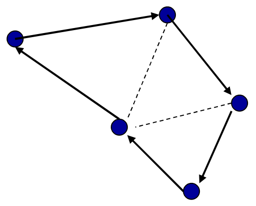


Active edge
$\longrightarrow$

Ball pivoting around active edge

● Point on front

Pivoting example

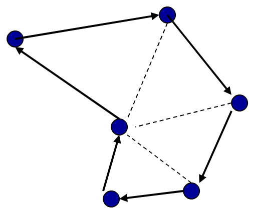


Ball pivoting around active edge

Active edge $\longrightarrow$

● Point on front

Pivoting example

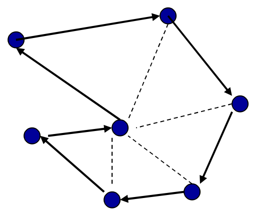


Ball pivoting around active edge

Active edge $\longrightarrow$

● Point on front

Pivoting example



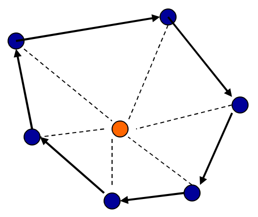
Ball pivoting around active edge

Active edge

$\longrightarrow$

● Point on front

Pivoting example



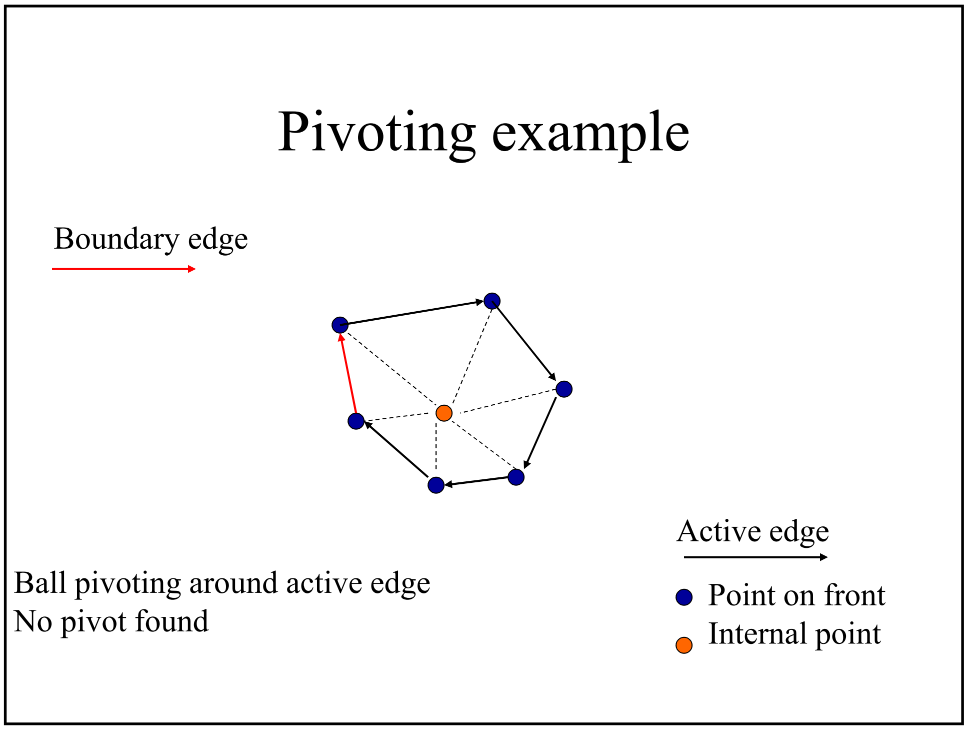
Ball pivoting around active edge

Active edge

$\longrightarrow$

● Point on front

● Internal point



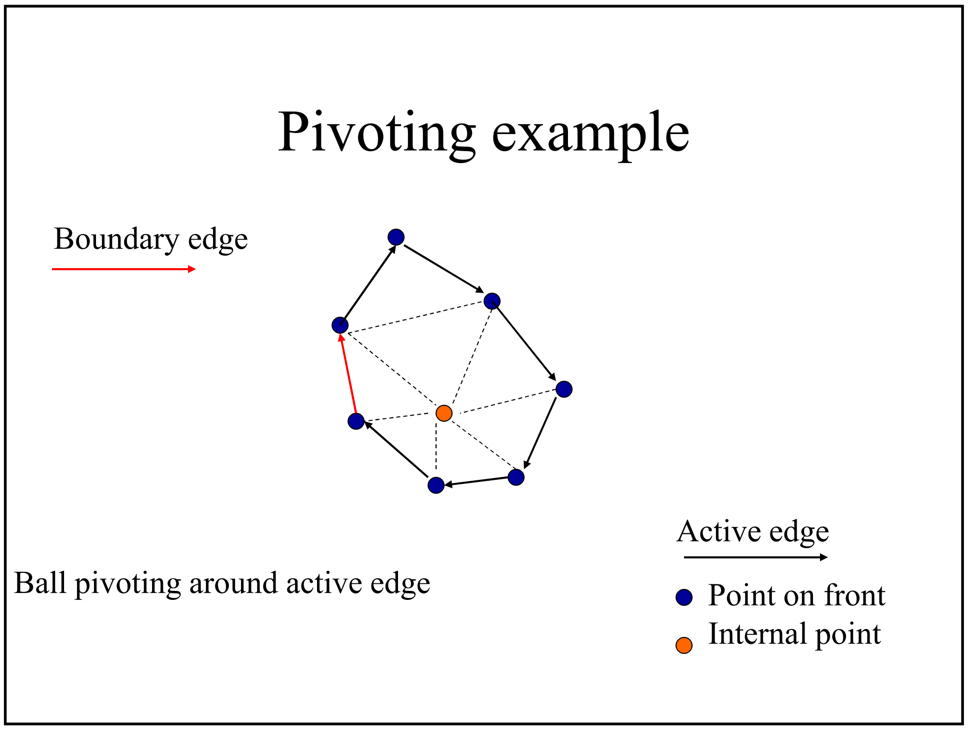
Pivoting example

Boundary edge

Ball pivoting around active edge
No pivot found

Active edge

● Point on front
● Internal point



Pivoting example

Boundary edge

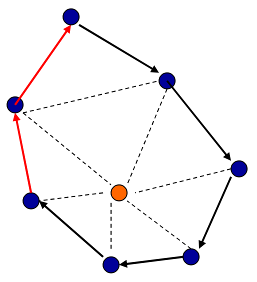
Ball pivoting around active edge

Active edge

● Point on front
● Internal point

Pivoting example

Boundary edge



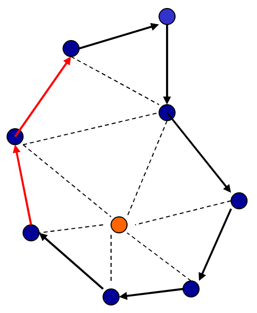
Ball pivoting around active edge
No pivot found

Active edge

● Point on front
● Internal point

Pivoting example

Boundary edge

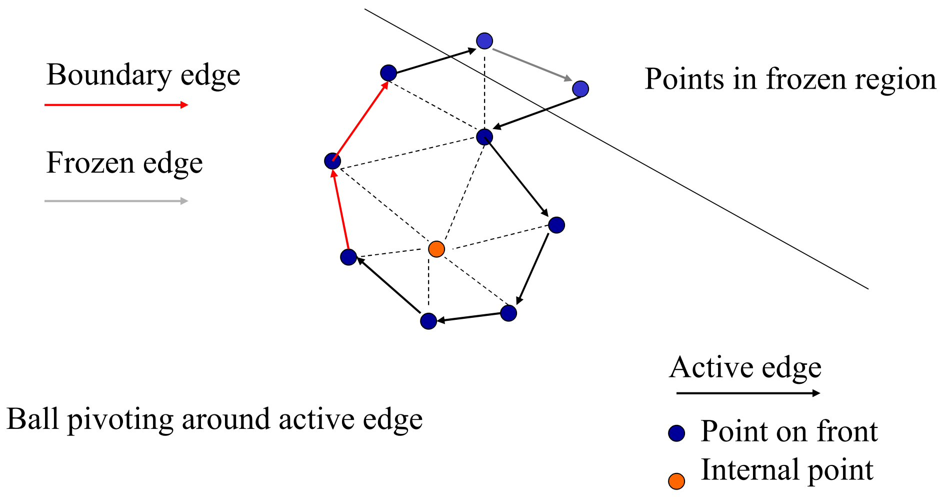


Ball pivoting around active edge

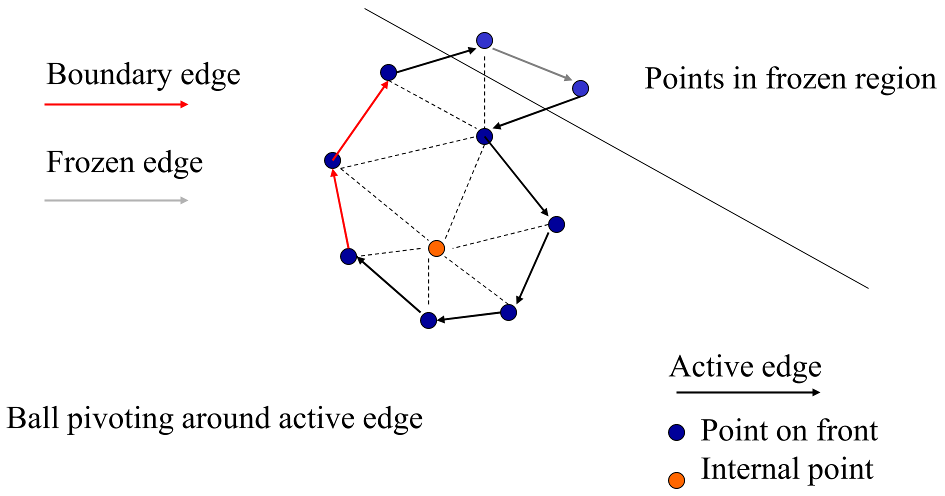
Active edge

● Point on front
● Internal point

Pivoting example



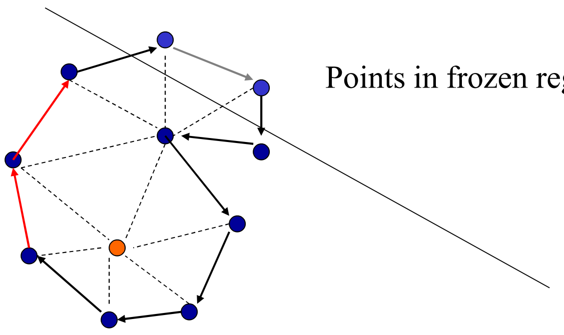
Pivoting example



Pivoting example

Boundary edge

Frozen edge



Points in frozen region

Ball pivoting around active edge

Active edge

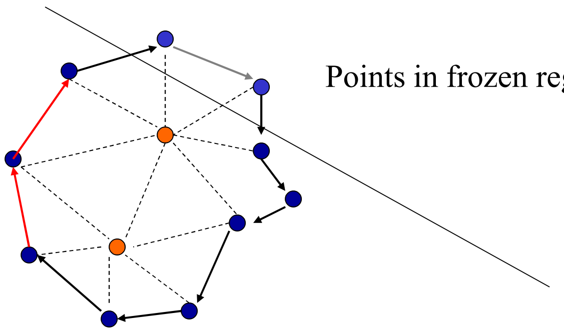
● Point on front

● Internal point

Pivoting example

Boundary edge

Frozen edge



Points in frozen region

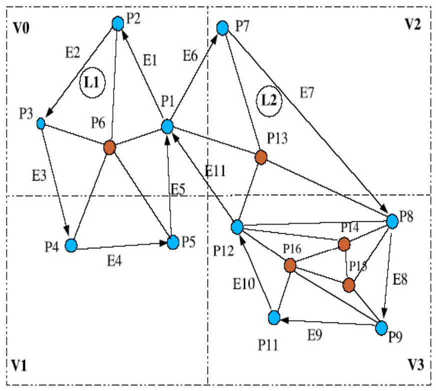
Ball pivoting around active edge

Active edge

● Point on front

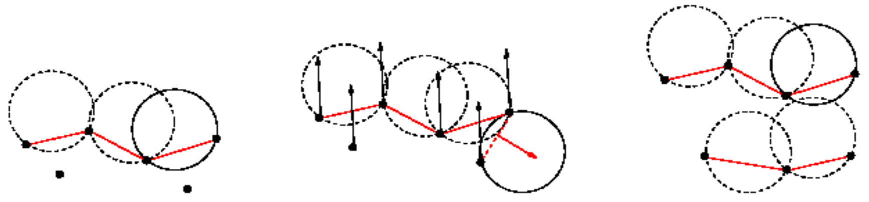
● Internal point

Algorithm Snapshot



- 13 points (P1, …)
- 4 voxels (V0, …)
- 11 edges (E1, …)
- Two loops: L1, L2

Noisy data

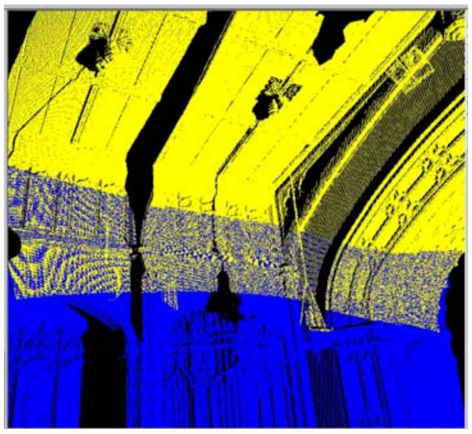

(a) (b) (c)

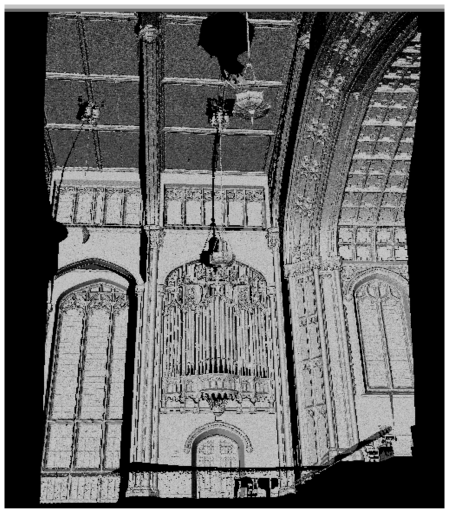
(a) Points "below" surface level are not touched by the pivoting ball and remain isolated (and are discarded by the algorithm).

(b) Due to missing data, the ball pivots around an edge until it touches a sample that belongs to a different part of the surface. By checking that triangle and data point normals are consistently oriented, we avoid generating a triangle in this case.

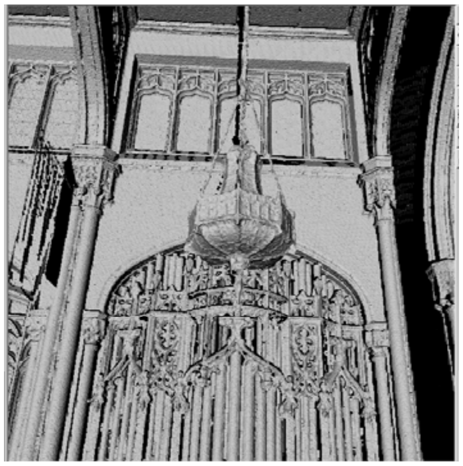
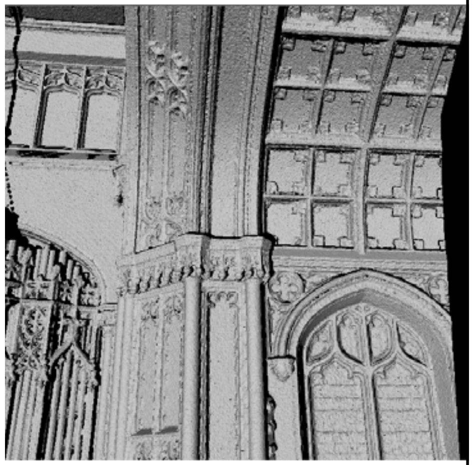
(c) Noisy samples form two layers, distant enough to allow the ball to "walk" on both layers. A spurious small component is created.

Ball Pivoting Algorithm


Overlapping scans


3D mesh

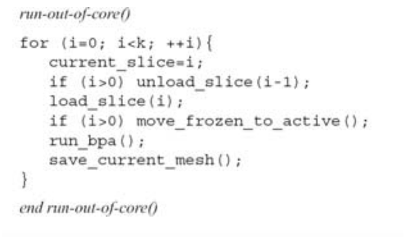
Ball Pivoting Algorithm



3D mesh detail

3D mesh detail

Out of core implementation

- Process data in slices that fit in memory
- No limit in size of input
- BPA’s active front provides natural implementation
- Of major importance for large scale scenes

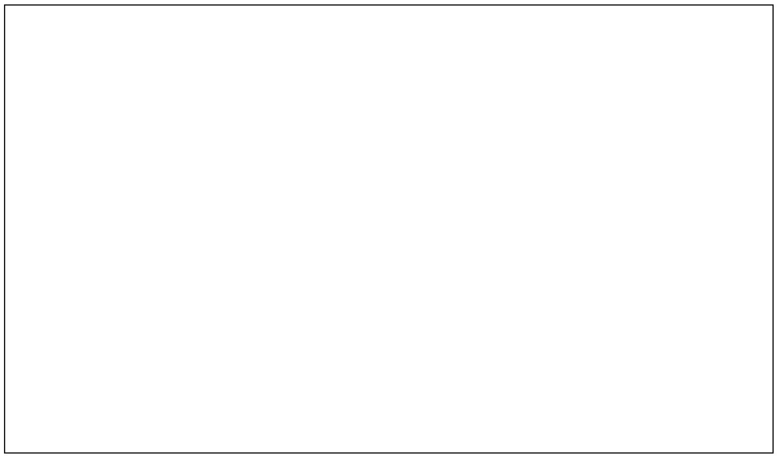


run-out-of-core()

```
for (i=0; i<k; ++i){
   current_slice=i;
   if (i>0) unload_slice(i-1);
   load_slice(i);
   if (i>0) move_frozen_to_active();
   run_bpa();
   save_current_mesh();
}
```

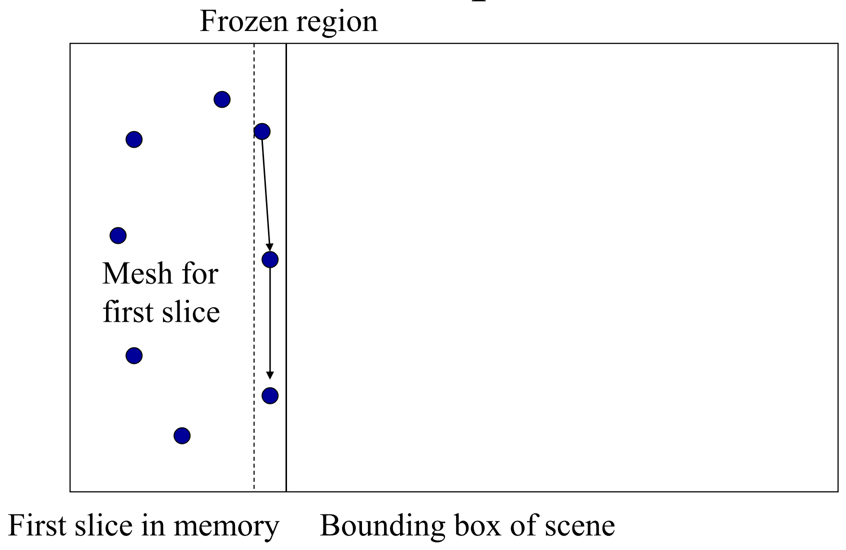
end run-out-of-core()

Out of core implementation

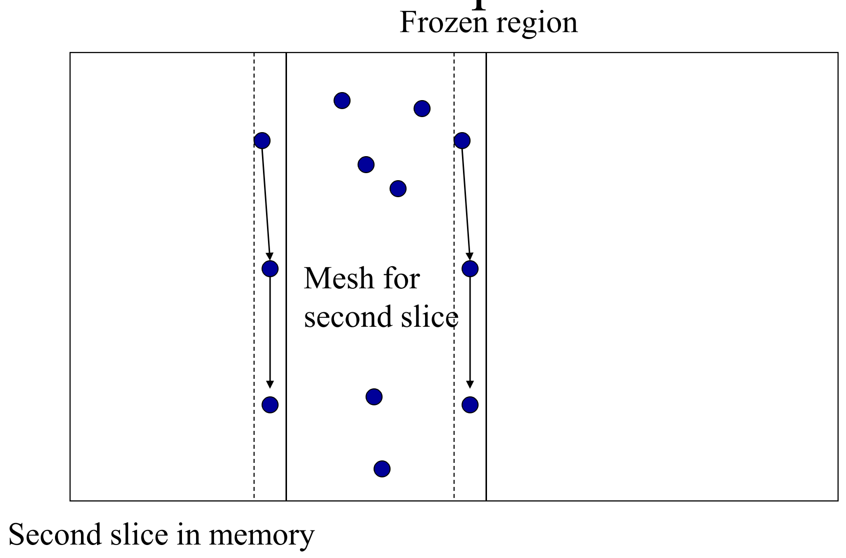


Bounding box of scene

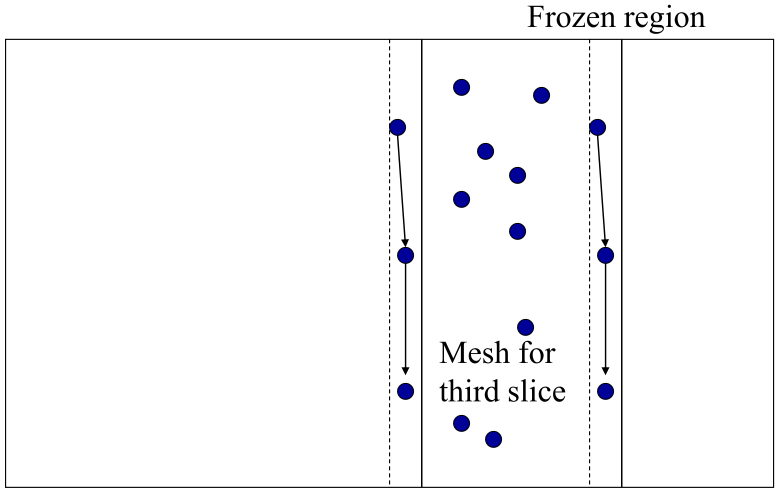
Out of core implementation



Out of core implementation

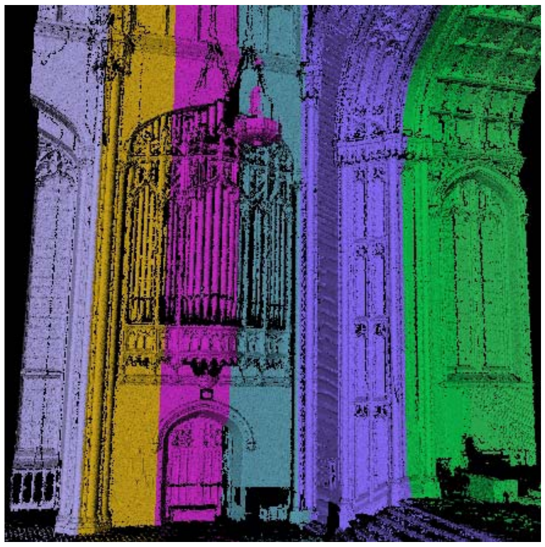


Out of core implementation



Third slice in memory, etc.

Out of core results



Part of the Great Hall mesh. The different colors correspond to meshes of different slices produced by the out-of-core implementation.

Results

- Grand Central Station
  - Leica 4500 (phase-based)
    - 27 input scans ~ 40 million points
    - Split into 30 slices
    - Preprocessing ~ 40 minutes
    - Split operation ~ 37 minutes
    - BPA out-of-core ~ 22 hours

    [Intel Xeon Processor at 2GHz, 2Gb of RAM]
    - Output mesh ~ 7.95 million triangles / 4.19 million vertices
    - Ball radius ρ = 0.2 meters (~ 7.87 inches)

Results

- Grand Central Station
  - Cyrax 2500 (time of flight)
    - 45 input scans ~ 45 million points
    - Split into 50 slices
    - Normal computation ~ 38 mins
    - Split operation ~ 65 mins
    - BPA out-of-core ~ 18 hours

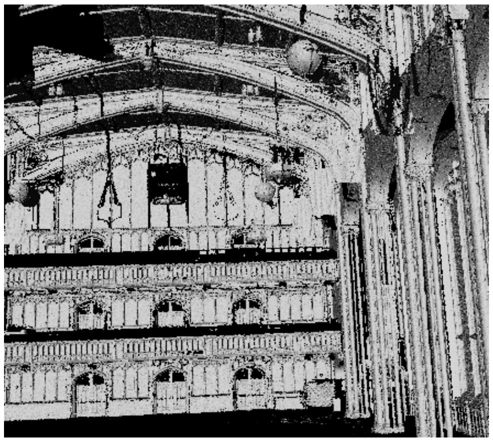
    [Intel Xeon Processor at 2GHz, 2Gb of RAM]
    - Output mesh ~7.74 million triangles / 3.98 million vertices
    - Ball radius ρ = 0.1 meters (~ 3.93 inches)

Results

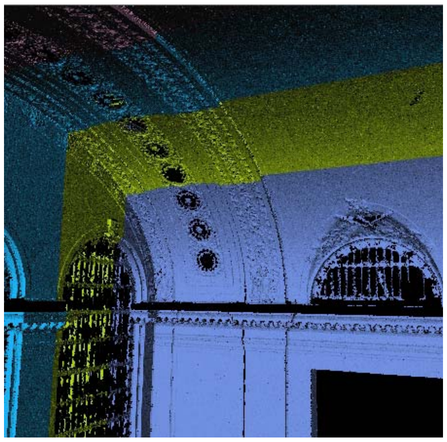
- Great Hall
  - Cyrax 2500 (time of flight)
    - 27 input scans ~ 27 million points
    - Split into 30 slices
    - Normal computation ~ 20 mins
    - Split operation ~ 27 mins
    - BPA out-of-core ~ 3 hours

    [Intel Xeon Processor at 2GHz, 2Gb of RAM]
    - Output mesh ~ 20.40 million triangles / 11 million vertices
    - Ball radius ρ = 0.03 meters (~ 1.18 inches)

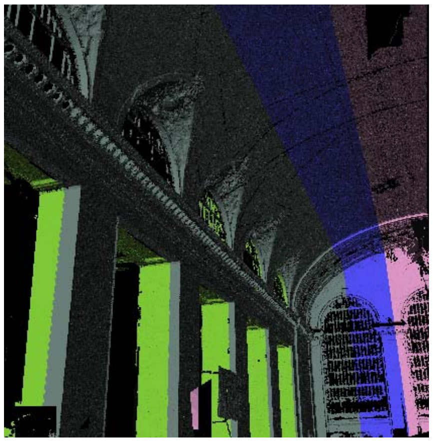
Out of core result [Great Hall]



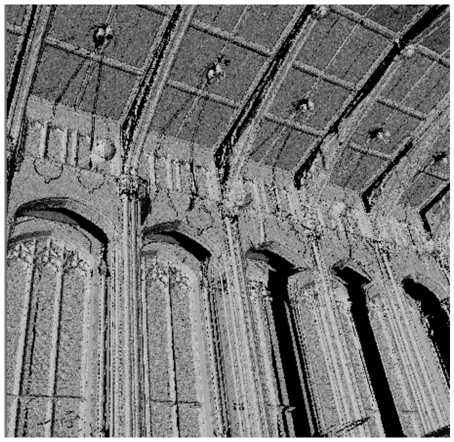
Out of core result [Grand Central]



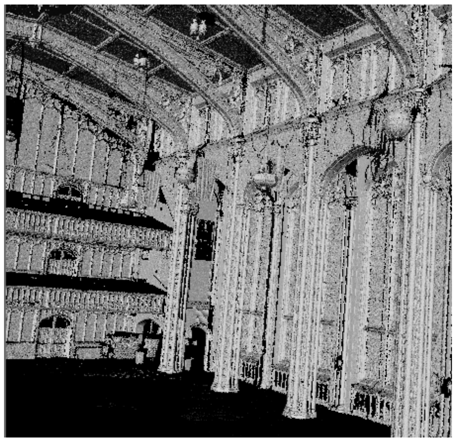
Out of core result [Grand Central]



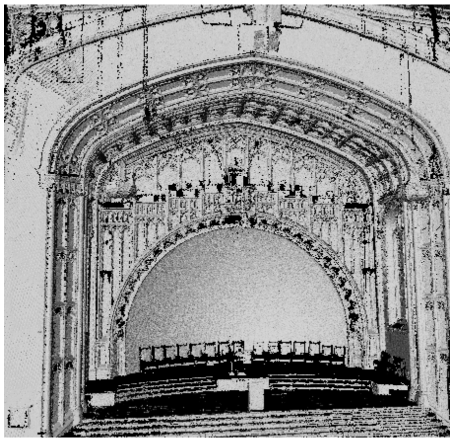
Out of core result [Great Hall]



Out of core result [Great Hall]



Out of core result [Great Hall]

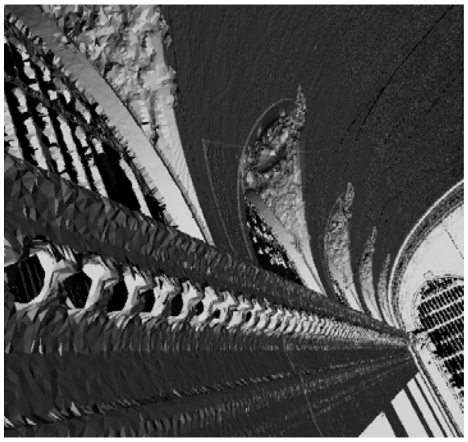


Out of core result [Great Hall]

Out of core result [Grand Central]

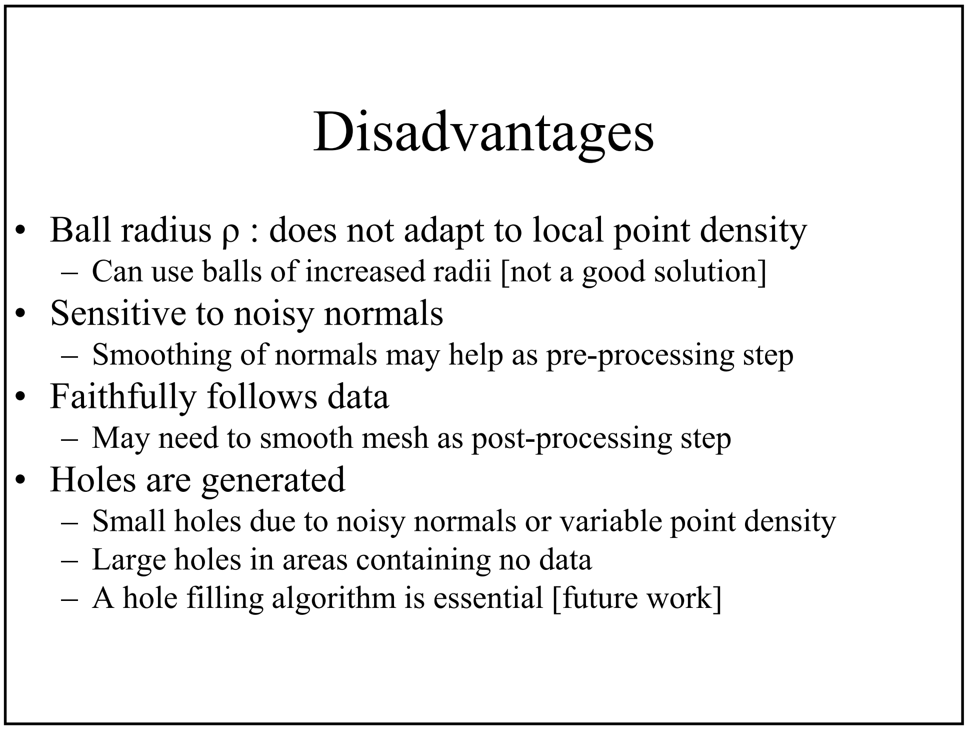
Out of core result [Grand Central]

Out of core result [Grand Central]



Advantages

- Efficient and conceptually natural algorithm
- Provides topologically correct mesh
- Faithfully follows data
- Models scenes of any geometric type
- Models scenes of any size (out of core)
- Larger ball : cruder / smaller model
- Smaller ball: sharper / larger model [captures detail]



Disadvantages

- Ball radius ρ : does not adapt to local point density
  - Can use balls of increased radii [not a good solution]
- Sensitive to noisy normals
  - Smoothing of normals may help as pre-processing step
- Faithfully follows data
  - May need to smooth mesh as post-processing step
- Holes are generated
  - Small holes due to noisy normals or variable point density
  - Large holes in areas containing no data
  - A hole filling algorithm is essential [future work]