L7 – Layered Depth Normal Images

- Introduction
- Related Work
- Structured Point Representation
- Boolean Operations
- Conclusion

Introduction

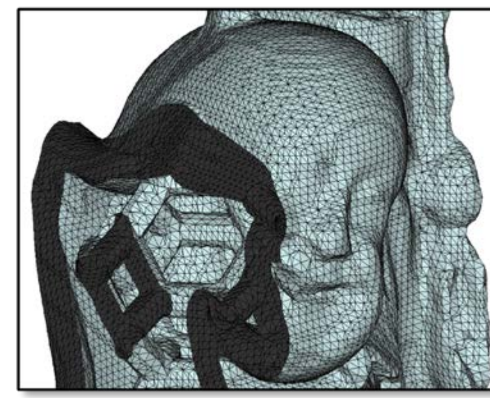
- Purpose: using the computational power on GPU to speed up solid modeling operations
- Models in many applications are with very complex shape and topology
  - virtual sculpting
  - microstructure design
  - rapid prototyping, etc.

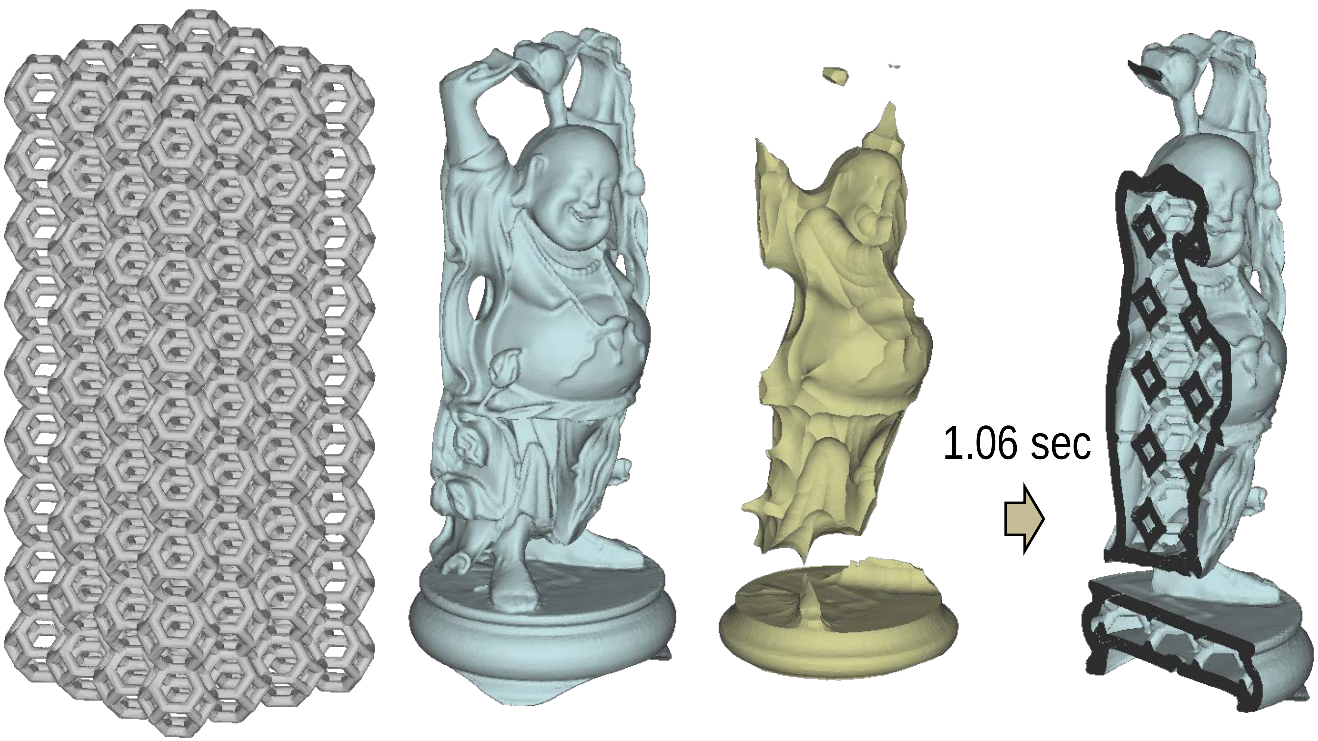
A test part built by SLA

Skull bones in human skeleton

Introduction (cont.)

- Boolean operations on models with massive number of triangles (Wang et al., 2010)





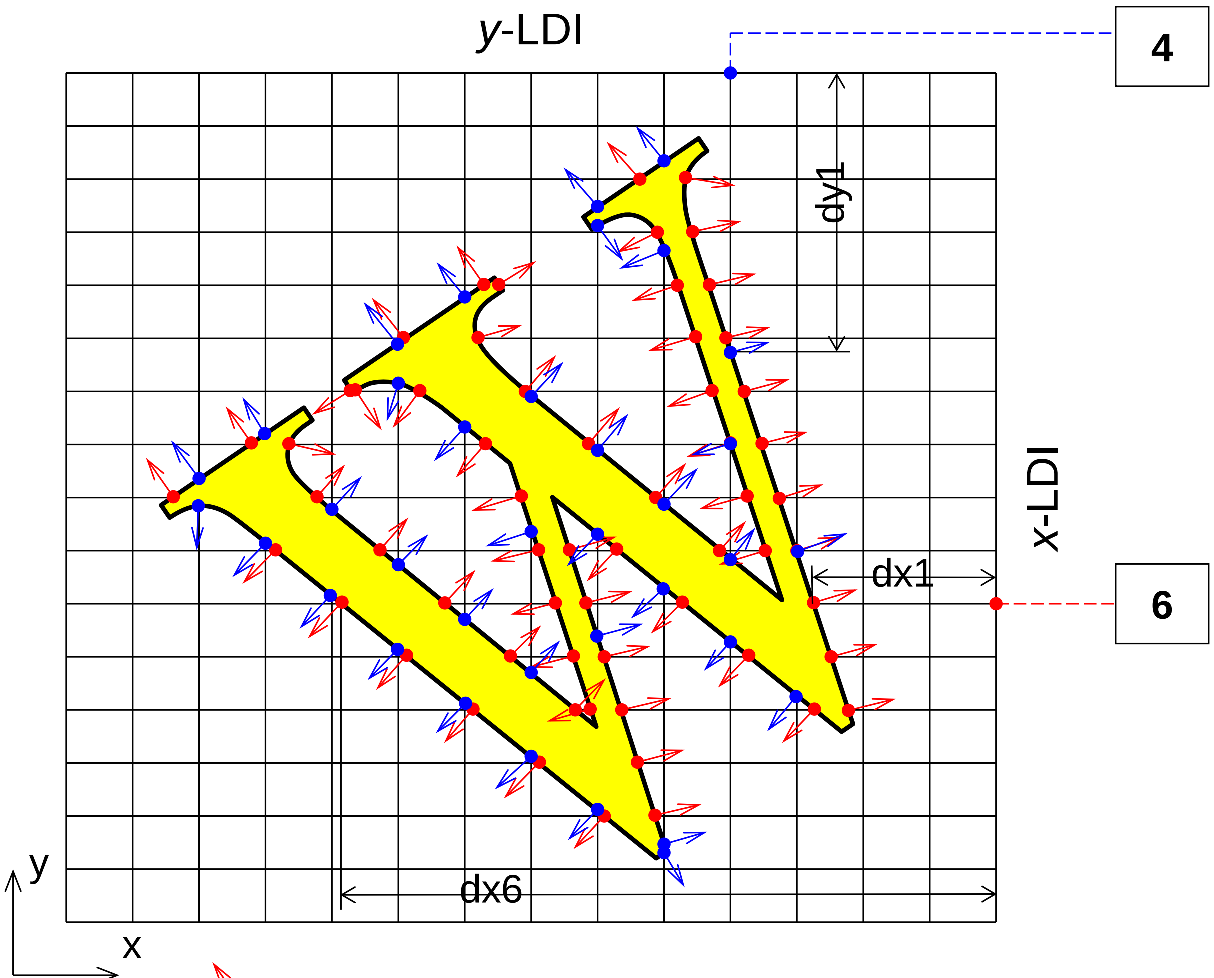
941.9k Faces 497.7k Faces 213.3k Faces 780.4k Faces

On GeForce GTX 580

Introduction (cont.)

- Market available solid modelers: e.g., ACIS using B-rep (speed? and robustness?)
- Existing free academic library: CGAL using complex data structure (speed?)
- Volumetric Representation is a good choice because of robustness
  - How to efficiently convert from and to B-rep?
  - How to effectively map to GPU?
- Our idea: ray-rep by Layered Depth-Normal Images (LDNI) on GPU

Layered Depth-Normal Images



dy1	$n1_x$	$n1_y$	$n1_z$
dy2	$n2_x$	$n2_y$	$n2_z$
dy3	$n3_x$	$n3_y$	$n3_z$
dy4	$n4_x$	$n4_y$	$n4_z$

dx1	$n1_x$	$n1_y$	$n1_z$
dx2	$n2_x$	$n2_y$	$n2_z$
dx3	$n3_x$	$n3_y$	$n3_z$
dx4	$n4_x$	$n4_y$	$n4_z$
dx5	$n5_x$	$n5_y$	$n5_z$
dx6	$n6_x$	$n6_y$	$n6_z$

Sampled depth on *x*-LDNI coupled with surface normal

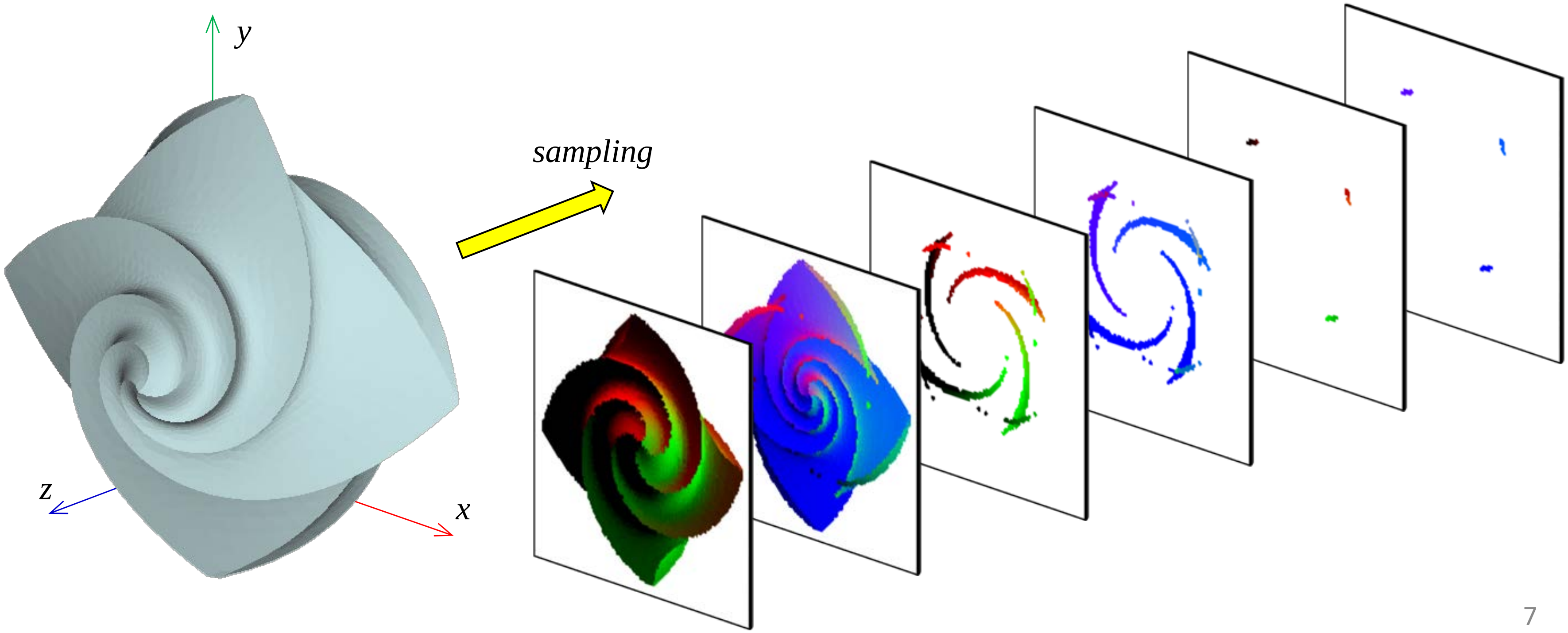
Sampled depth on *y*-LDNI coupled with surface normal

LDNI: a semi-implicit rep.

- A structure of three LDNIs sampled with rays along *x*-, *y*- and *z*-axes
- All with w x w pixels – the same resolution
- Selecting origin carefully – form sampling grids with w x w x w nodes
- Semi-implicit representation – easily detect whether a point is *inside* / *outside* a solid

LDNI: Data Structure on GPU

- Stored as a list of 2D textures
- Maximum number of layers: n_{max}
- Special value M (e.g., ∞) – the white ones below

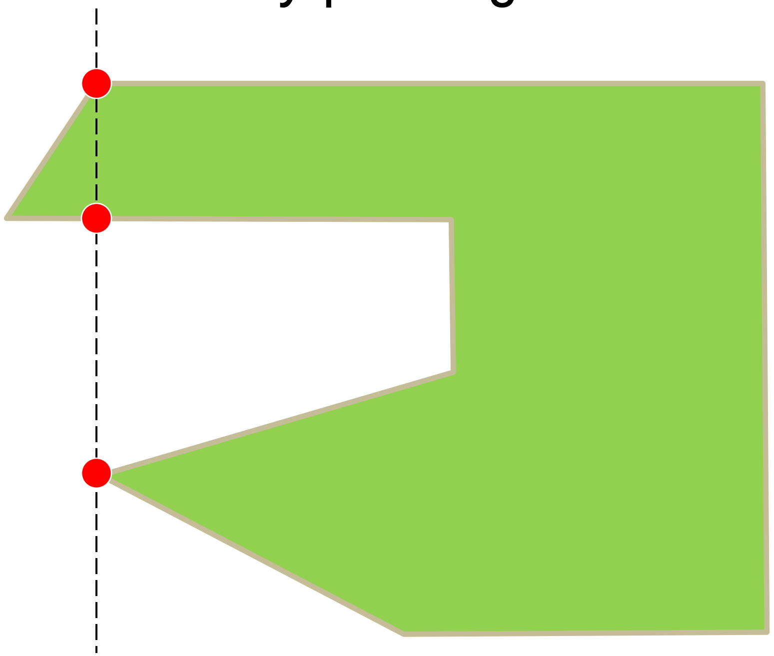


Sampling B-rep into LDNI

- *Input*: 2-manifold mesh surface of a solid model's boundary
- *Output*: 2D textures for LDNI rep on GPU
- Similar to scan-conversion
- Accelerated on the GPU
- Two possible strategies:
  - Depth-peeling using depth-buffer only
  - Using stencil buffer
  - Which one? Why?

Sampling B-rep into LDNI (cont.)

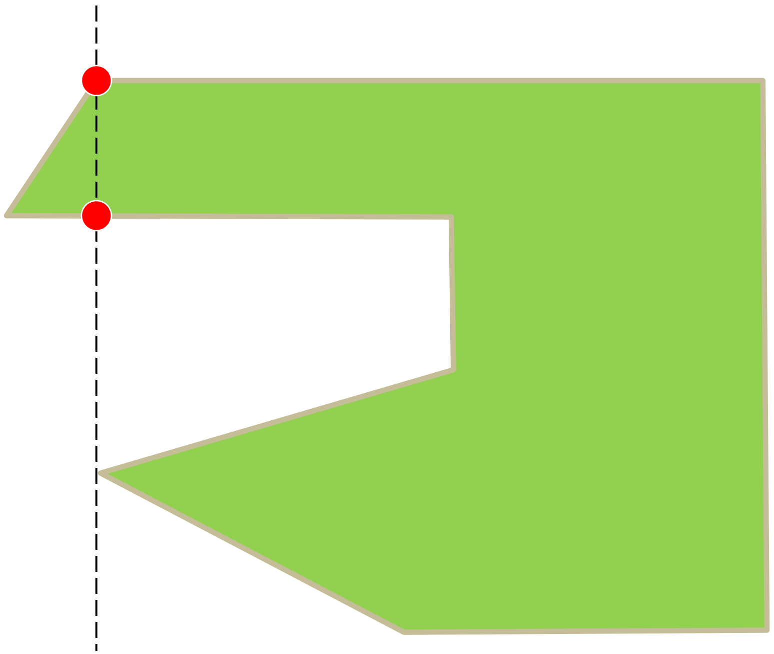
- Why not depth-peeling?
  - Based on the comparison of depth values
  - Only one sample is collected when the ray passing silhouette edge
  - Lead to the ambiguous of *inside* / *outside* detection
  - Although the samples have been sorted
  - Such ambiguity can hardly be recovered



Odd number of samples are reported

Sampling B-rep into LDNI (cont.)

- Problem can be solved by using stencil buffer
  - Multiple rendering (n_{max})
  - Only allow *k*th fragment pass
  - $k = 1, \ldots, n_{max}$
- Limitation
  - Stencil buffer – only 256
  - Solution: volume tiling
- Not only depth value
- But also normal
  - Reason why called LDNI



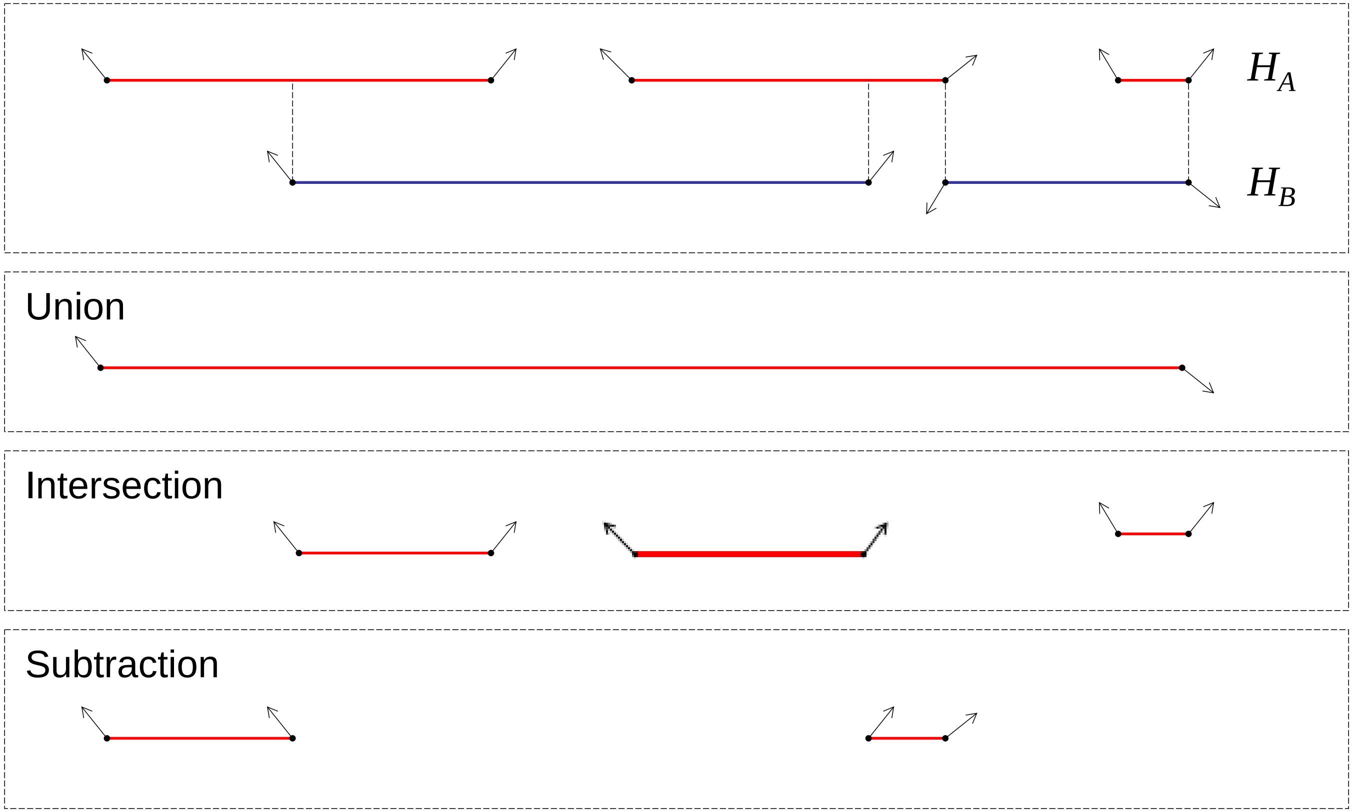
Even number of samples are reported

Sampling B-rep into LDNI (cont.)

- For a model with *m* triangles, the amount of data communication (the bottleneck of GPU-CPU computing)
- Without Shader Program
  - 3*m* vertices – 9*m* x 4 bytes for position
  - *m* normal vectors – 3*m* x 4 bytes
  - Total 48*m* bytes
- With Shader Program (speed up >5 times)
  - *n* vertices – 3n x 4 bytes for position (with $n \approx 0.5m$)
  - *m* indices – 3m x 4 bytes
  - Total 18*m* bytes

Boolean Operations on LDNI

- Inherit the simplicity of Boolean on ray-rep
- Highly parallel – computing on rays of LDNI

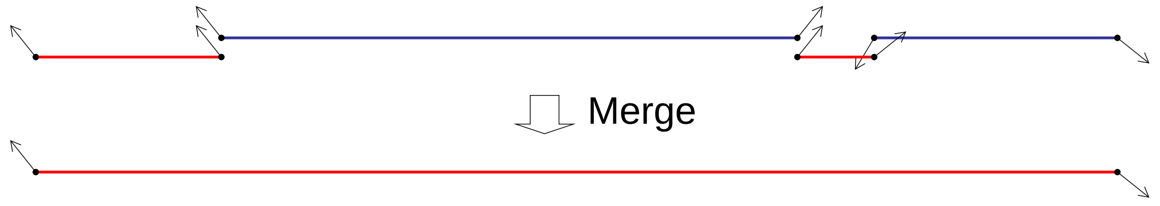


Boolean Operations on GPU

- On each ray, go through the samples on H_A and H_B by their depths (in *parallel*)
- nVIDIA CUDA is selected for the implementation
- To ease the implementation, LDNI rep is mapped to a 1D array
  - Instantly by DirectX
  - But takes a relatively long time by OpenGL
- Result in a new 1D array

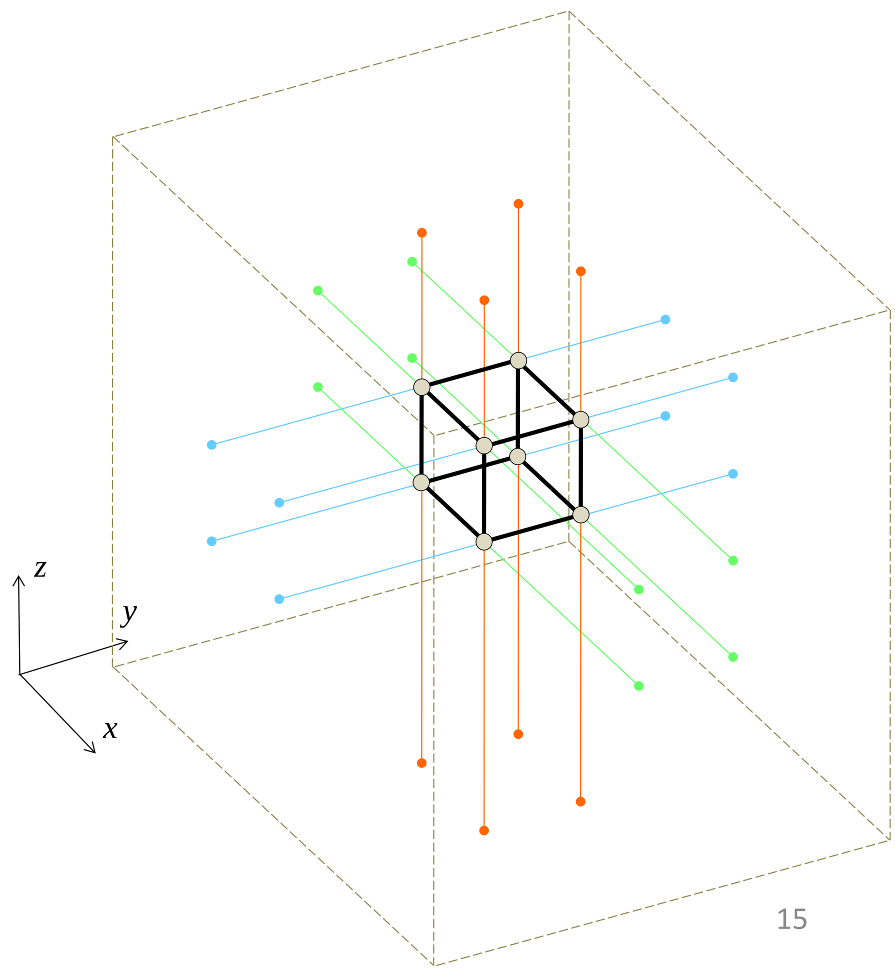
Robustness Enhancement

- A step of small interval removal
  - 1D volume or gap less than ε
  - $\varepsilon = 10^{-5}$ as single precision float is sampled for depth
  - 10^{-7} is almost the smallest number that can be exactly represented by single precision float
- The step of small interval removal can be incorporated into the Boolean algorithm
- Tangential-contact can be well processed



Contouring LDNI Solid to B-rep

- Cells are formed by the rays
  - We do not explicitly construct
  - *Inside* / *outside* of nodes are detected on-site
  - Inconsistency: overcome by majority vote
- An algorithm with two-passes



Contouring LDNI Solid to B-rep (cont.)

- First Pass: construct vertex table
  - Vertices are constructed in the *boundary cells*
  - A vertex in the cell [i, j, k] is given a unique ID

$$ID = (i\,(w - 2)^2 + j\,(w - 2) + k)$$

  - Position: determined by a position minimizing *QEF*
    - Therefore, sharp features can be reconstructed
- Second Pass: construct face table
  - Check the edge of cells – if there is an *inside/outside* change
  - A quadrilateral face by linking vertices in its *four* neighboring cells – by outputting the vertex IDs

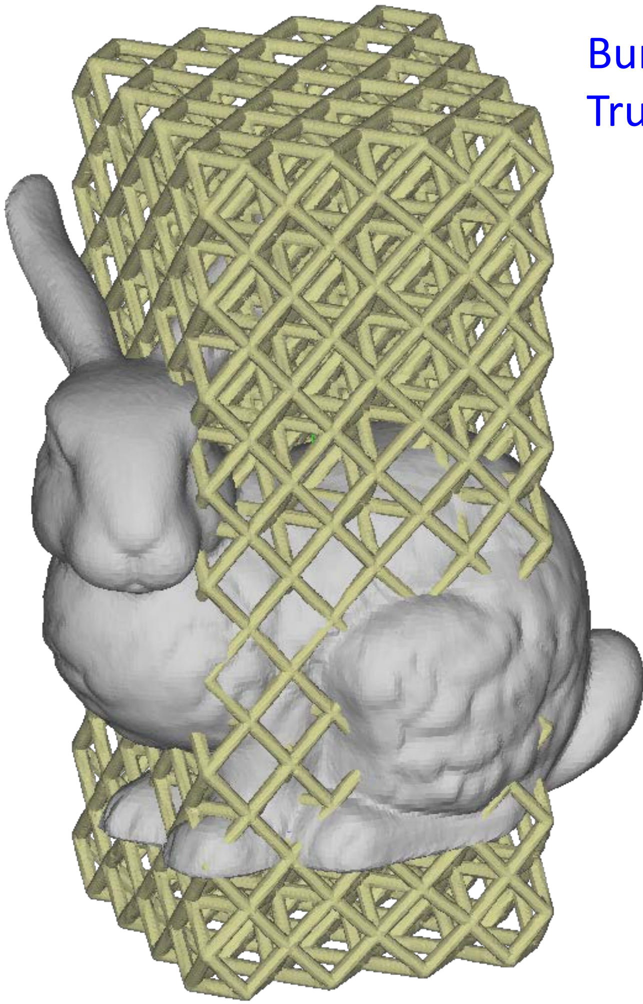
Experimental Results

- Statistics of sampling and memory usage

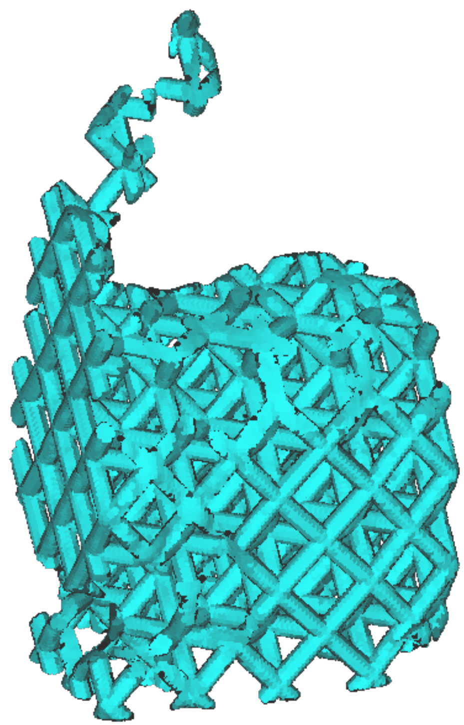
Model	Faces	Vertices	Sampling	Memory
Buddha	498k	249k	0.484s	42MB
Truss	942k	467k	1.015s	146MB
Bunny	70k	35k	0.094s	32MB
Dragon	277k	128k	0.295s	36MB
Truss2	1,026k	510k	1.059s	118MB

  – The tests are conducted at the resolution of 256 x 256
  – On a consumer level PC with Intel Core 2 Quad CPU Q6600 2.4GHz + 4GB RAM and GeForce GTX295

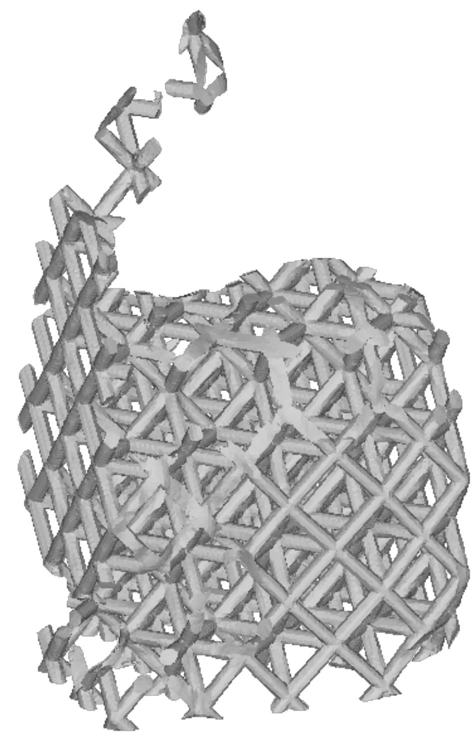
Experimental Results (cont.)



Bunny: 70k faces
Truss2: 1,026k faces



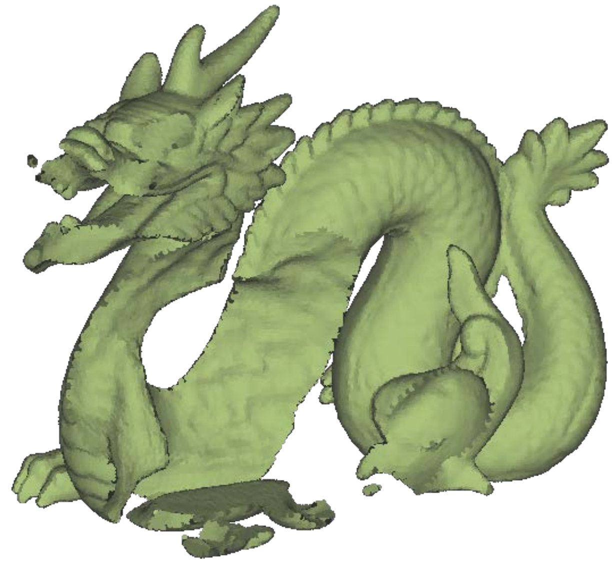
Intersection: 0.077s



Contouring: 0.625s

Experimental Results (cont.)

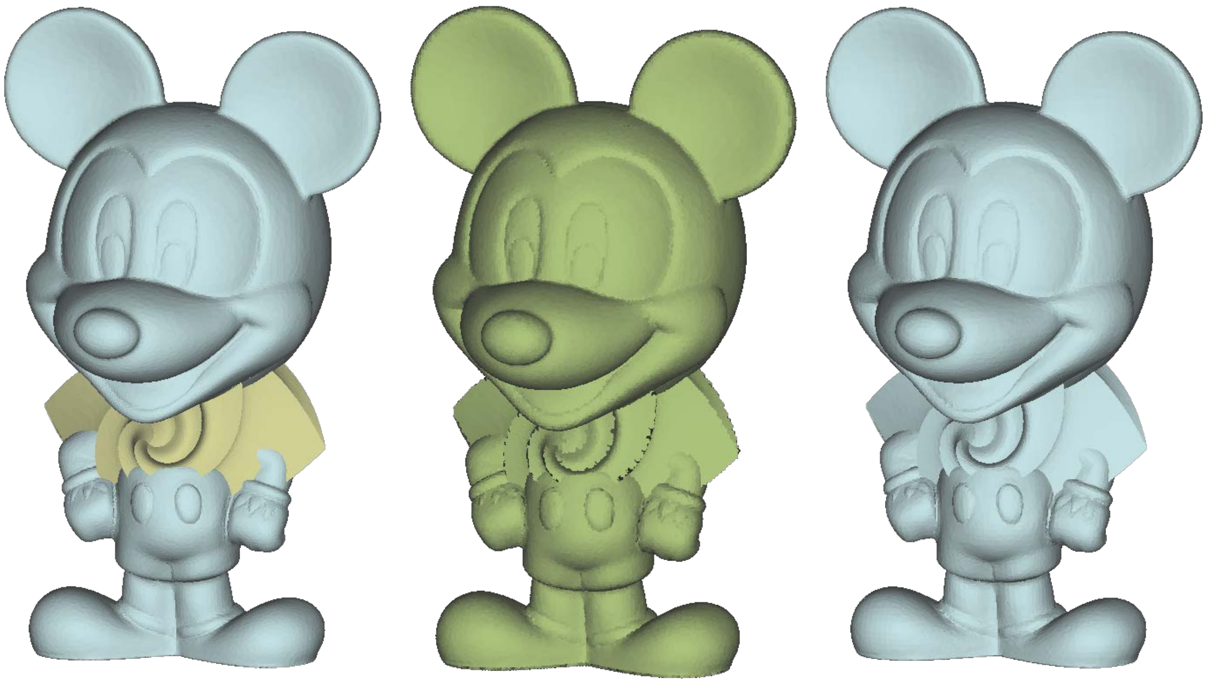
Dragon: 277k faces
Bunny: 70k faces



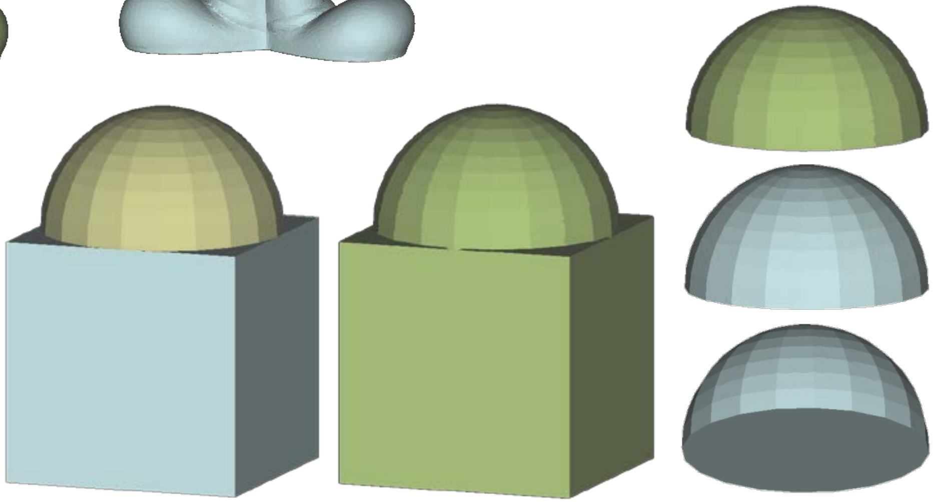
Subtraction: 0.030s

Contouring: 1.216 sec

Experimental Results (cont.)



Mickey: 42.9k faces
Octa-flower: 15.8k faces
Union: 0.016 sec
Contouring: 0.686 sec



Success in Tangential
Contact Case

Testing on ACIS and CGAL

- For comparison
  - An implementation using ACIS R15
  - An implementation using CGAL ver 3.4

Example	ACIS	CGAL	GPU Sampling	GPU Boolean	GPU Contouring
Mickey & Octa-flower	66.409 sec	Fail	0.422 sec	0.030 sec	1.216 sec
Box & Sphere	43.388 sec	0.864 sec	0.125 sec	0.016 sec	0.484 sec
Others	Fail	Fail	< 2 sec	< 0.2 sec	< 1.5 sec

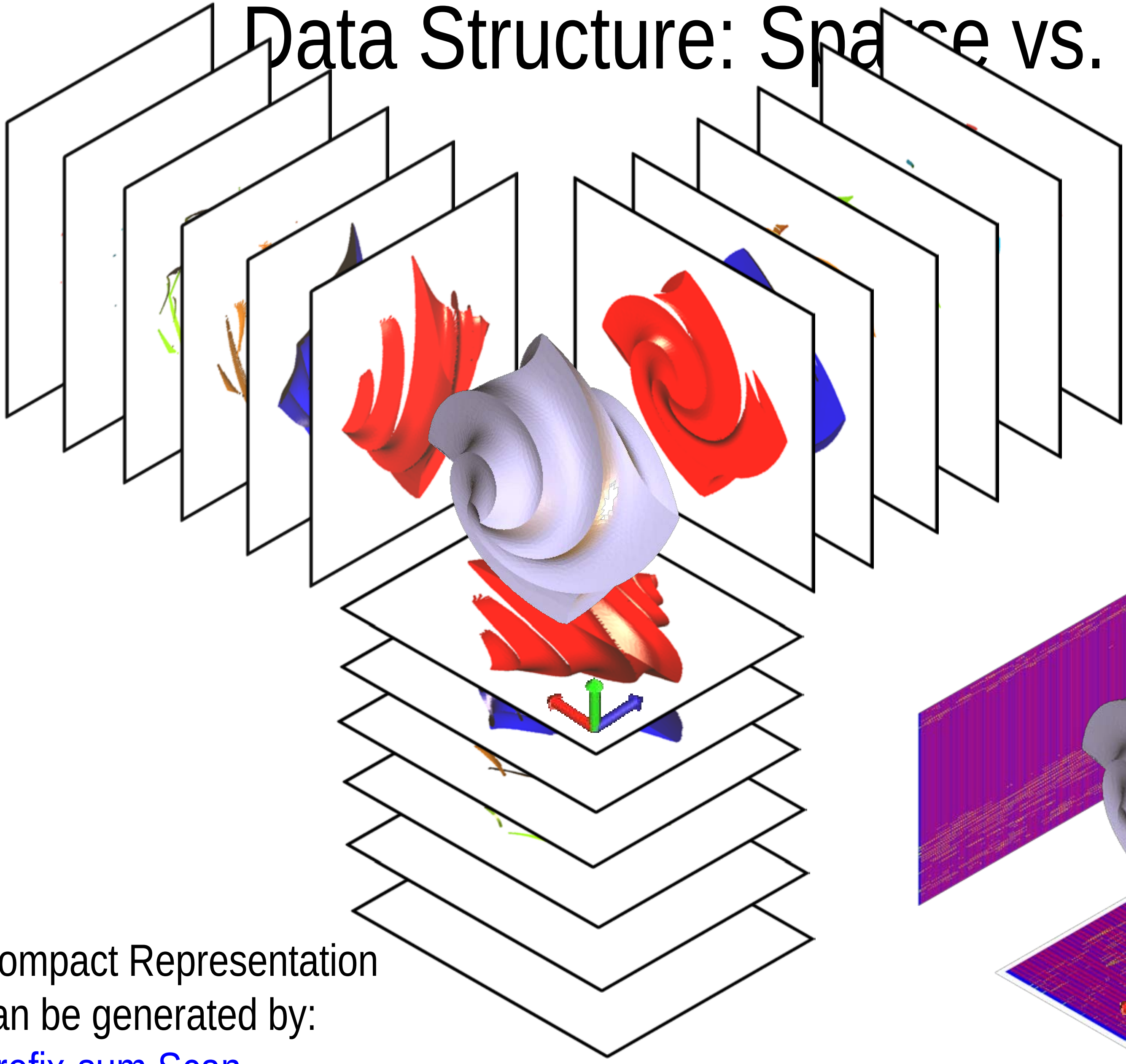
Charlie C.L. Wang, Yuen-Shan Leung, and Yong Chen, "Solid modeling of polyhedral objects by Layered Depth-Normal Images on the GPU", Computer-Aided Design, vol.42, no.6, pp.535-544, June 2010.

[Video]

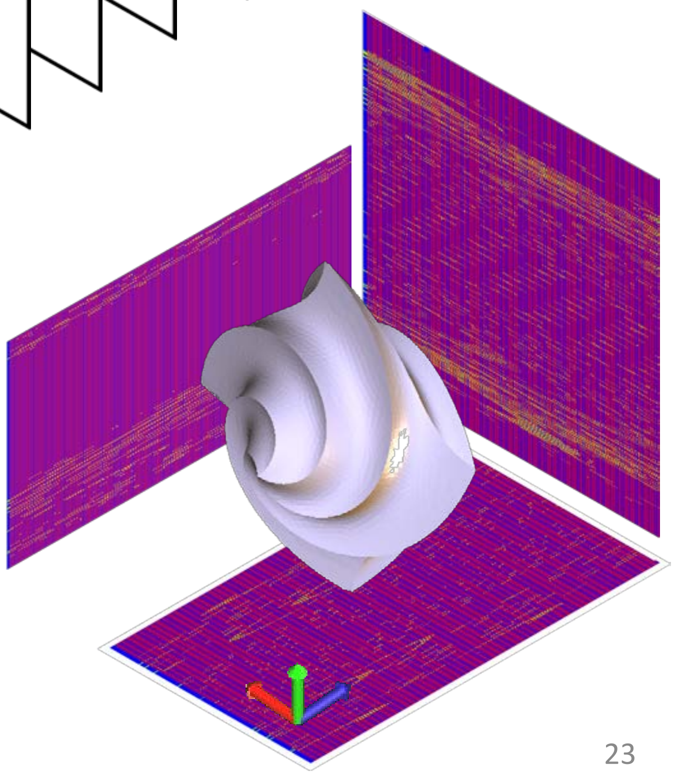
Limitation on Current Implementation

- Memory Usage
  - Processing a dense manner
  - LDNI is actually sparse (could be improved)
- Rotation sensitive
  - Need a continuous representation
- Lack of other solid modeling operations

Data Structure: Sparse vs. Compact

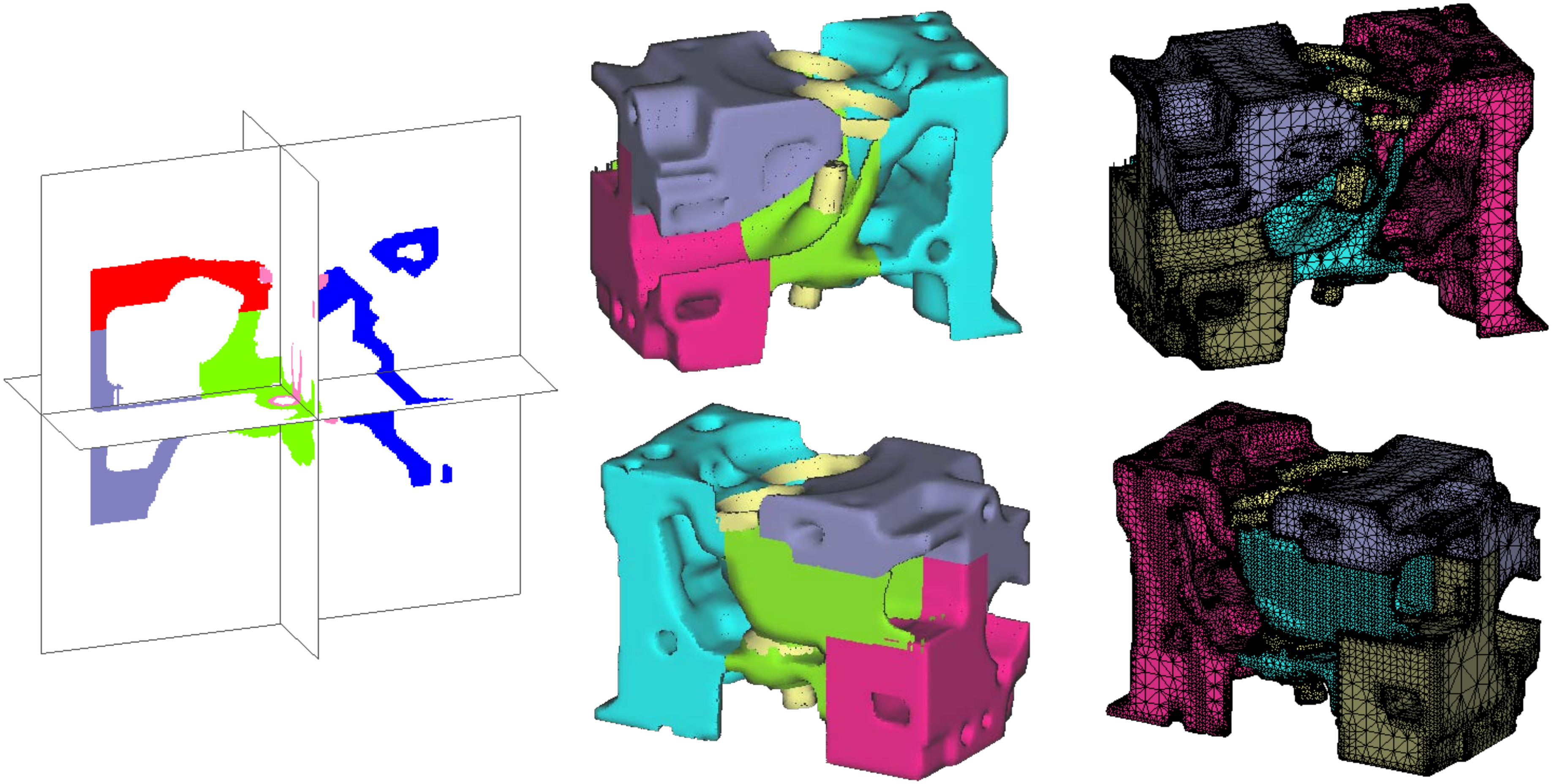


Two Arrays:
1) 2D Index Array
2) 1D Data Array



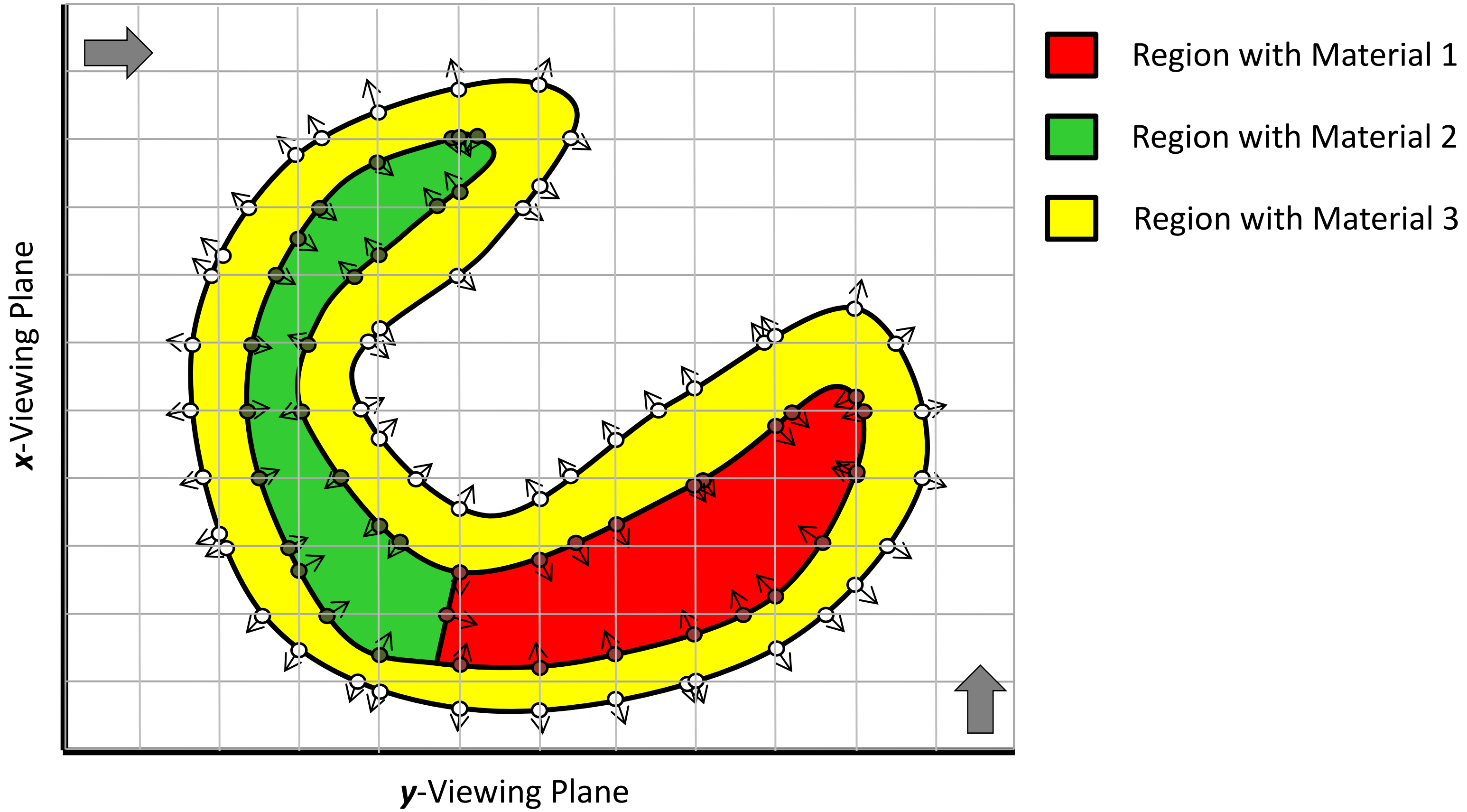
Compact Representation can be generated by: Prefix-sum Scan

Surface Modeling from Multi-Material Volumetric Data

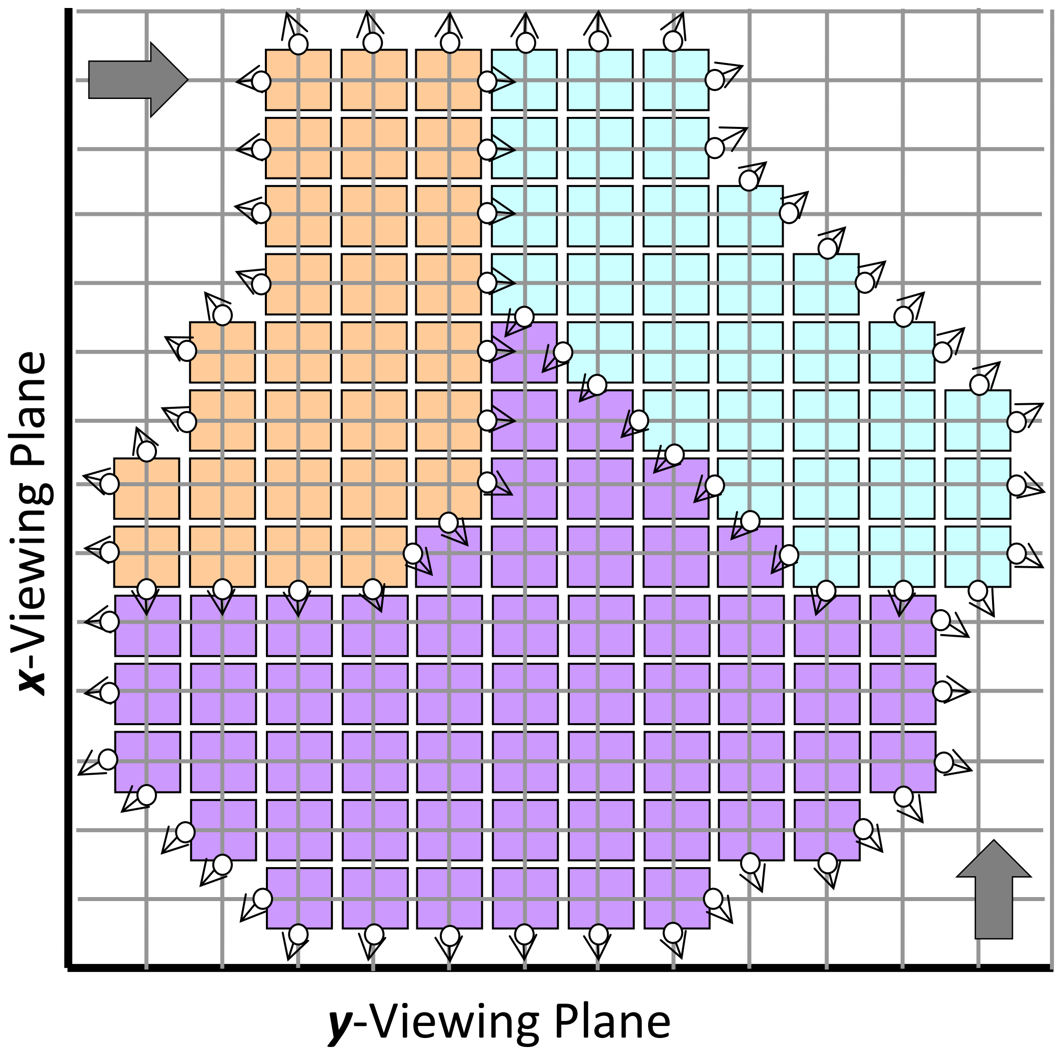


hRay-rep: Extended Ray-rep for Heterogeneous Solids

- Regions with different materials are presented in different colors

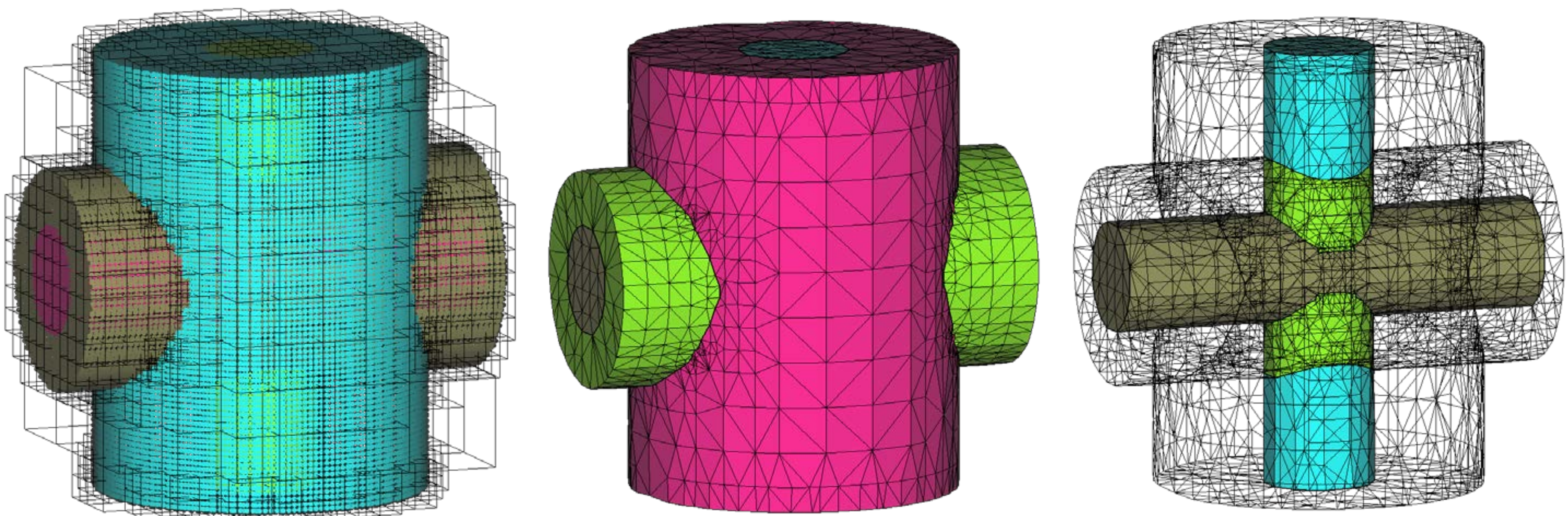


Converting Multi-Material Volumetric Data into a hRay-rep

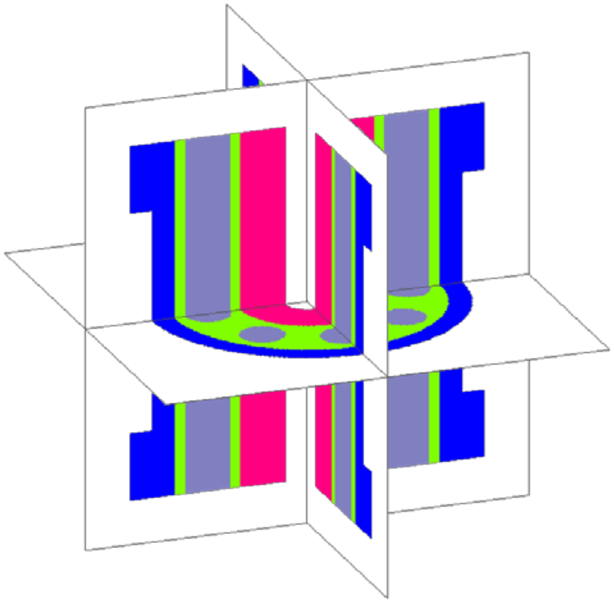
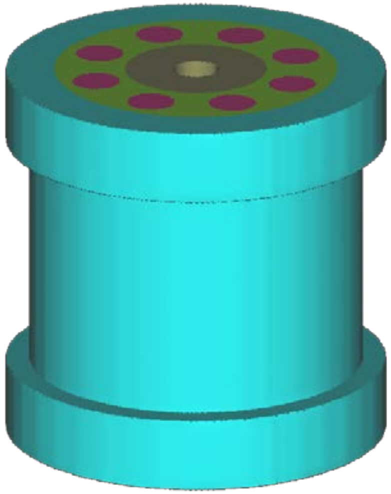
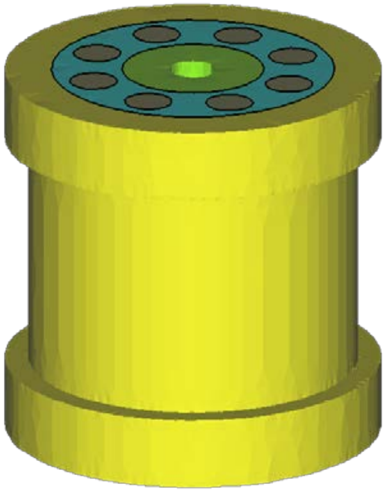
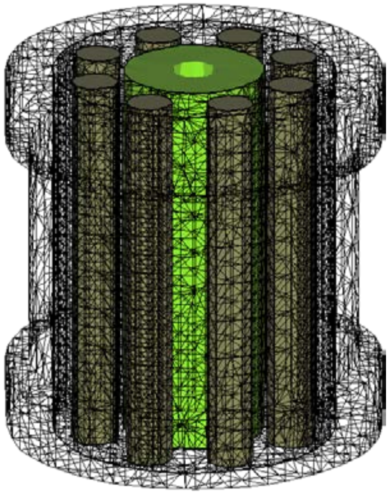
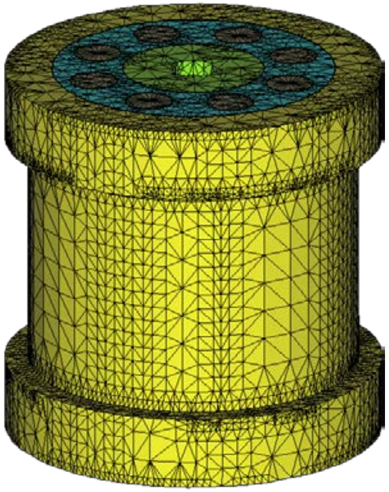
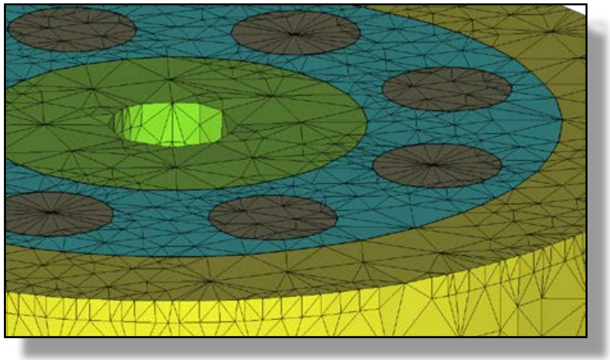


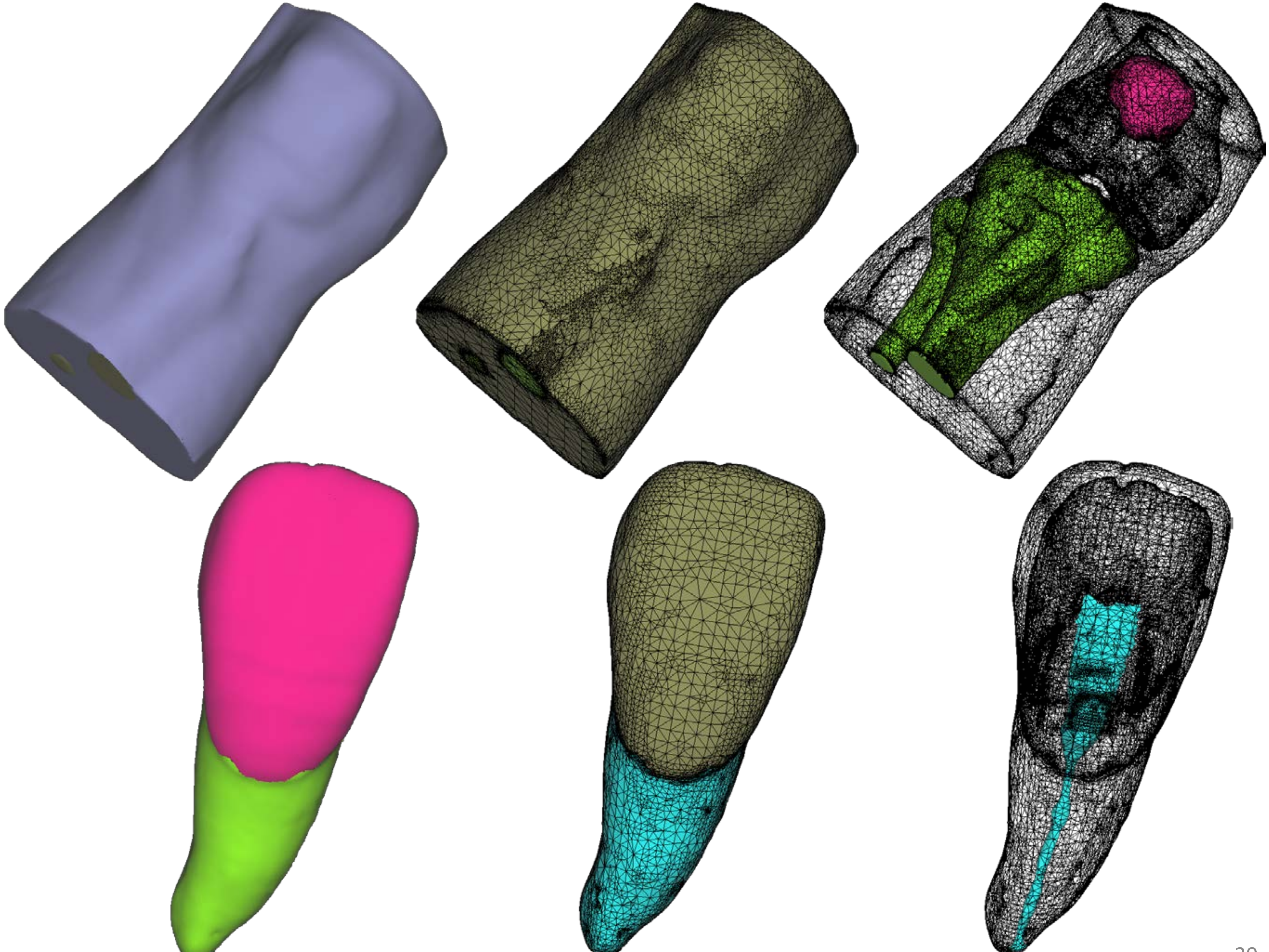
Mesh Generation on hRay-rep of Heterogeneous Solid Using Octree

- The step of octree construction takes the majority of computing time, which however can be processed in parallel easily.

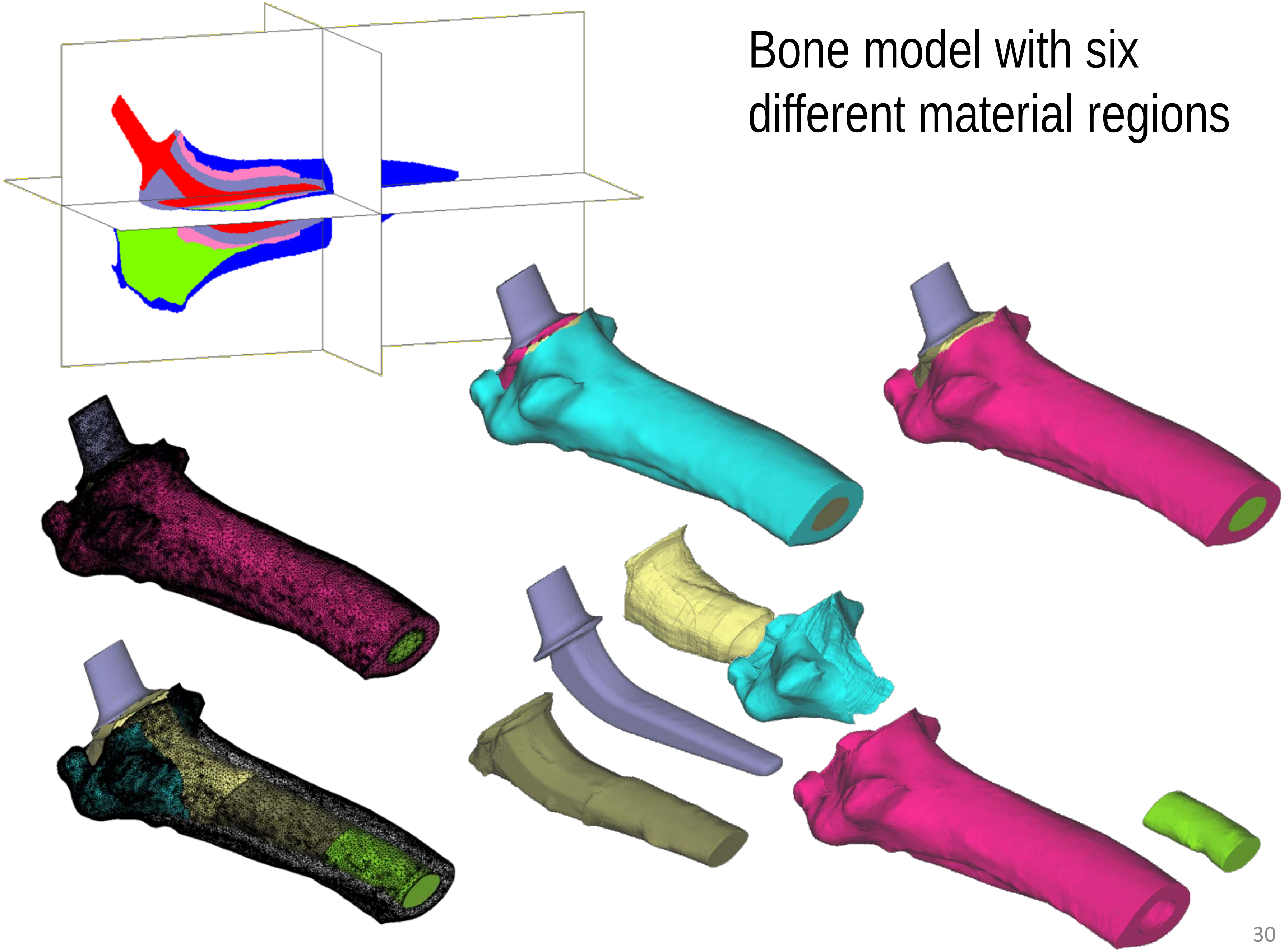


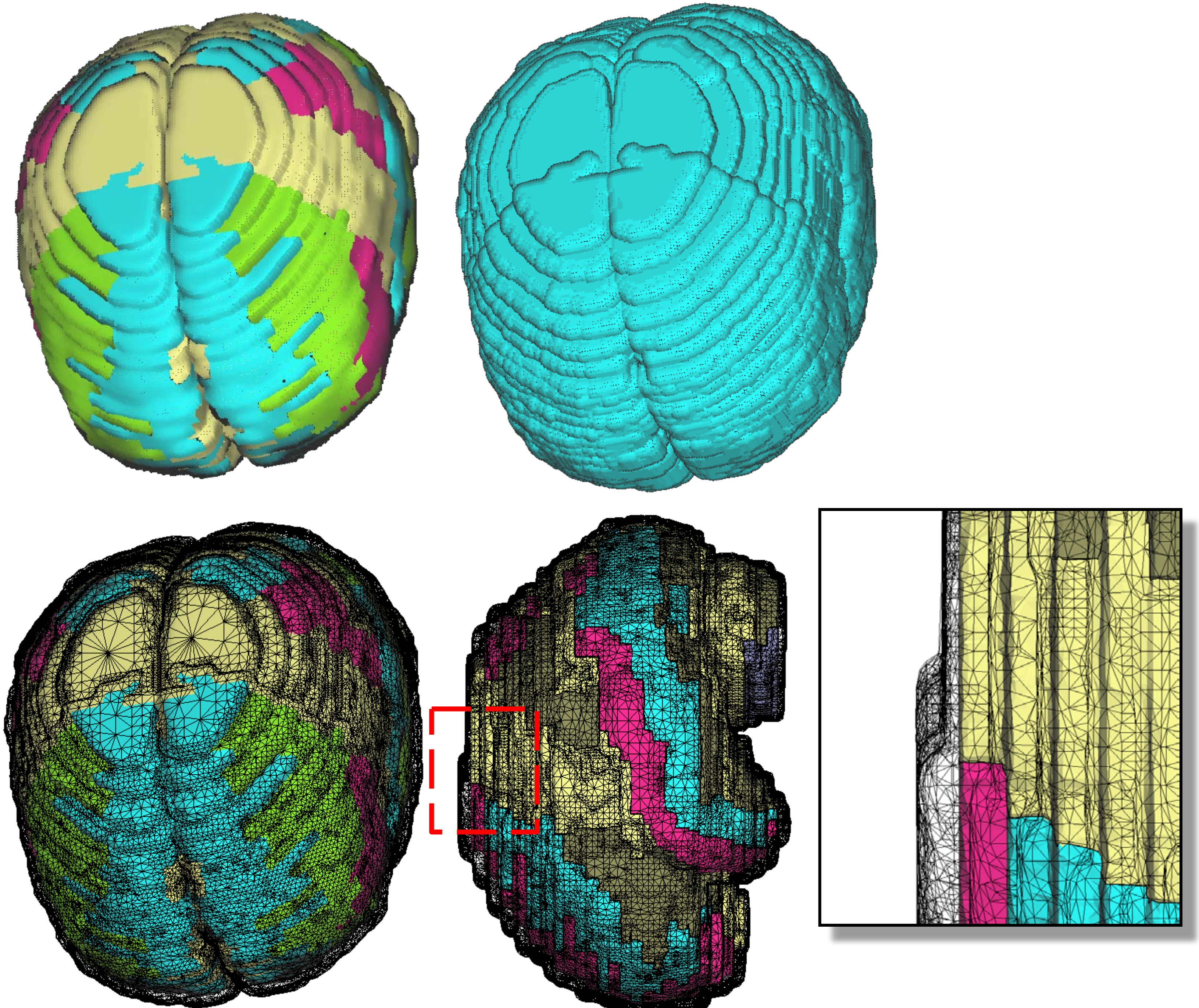
Other Results

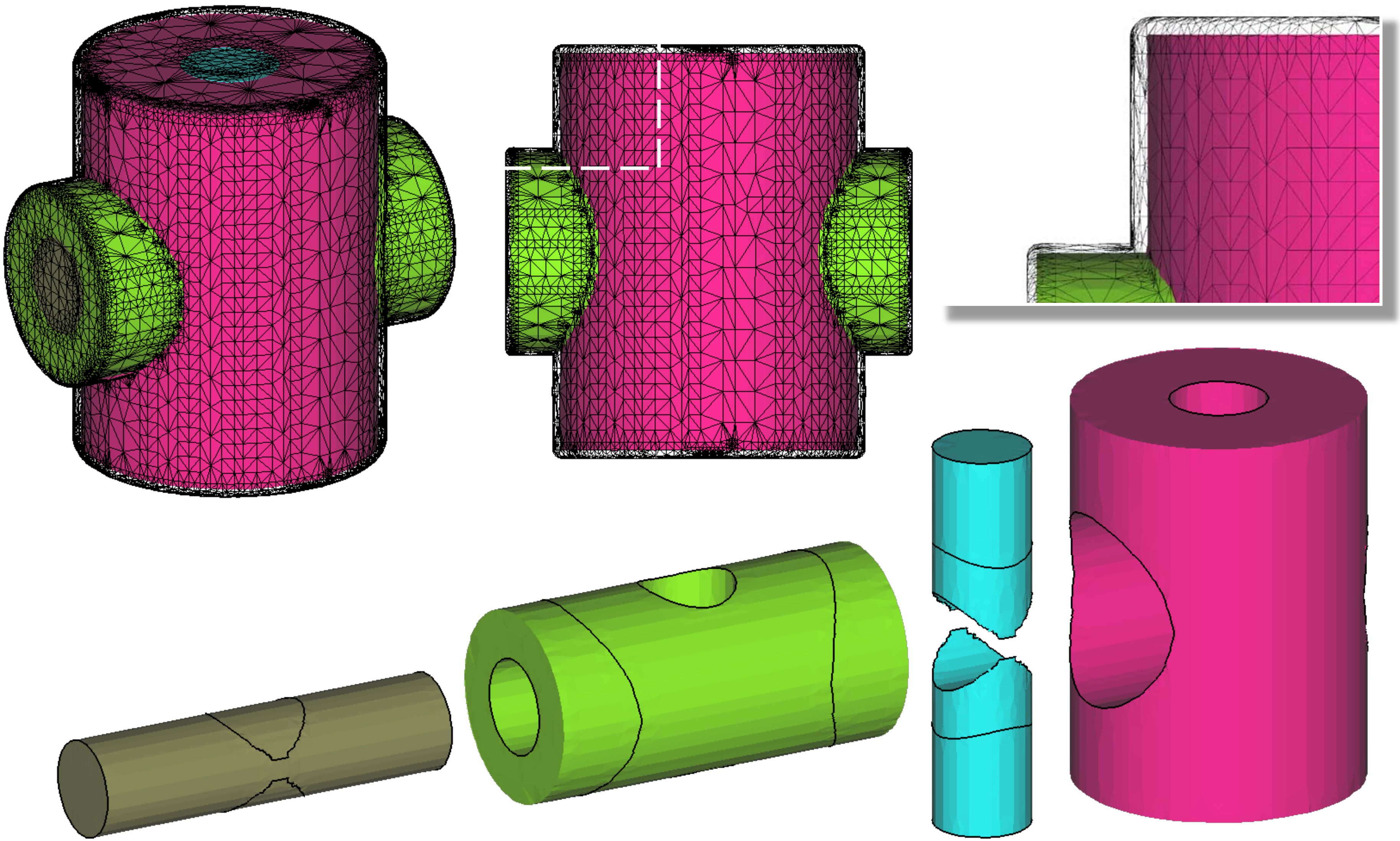









Bone model with six different material regions

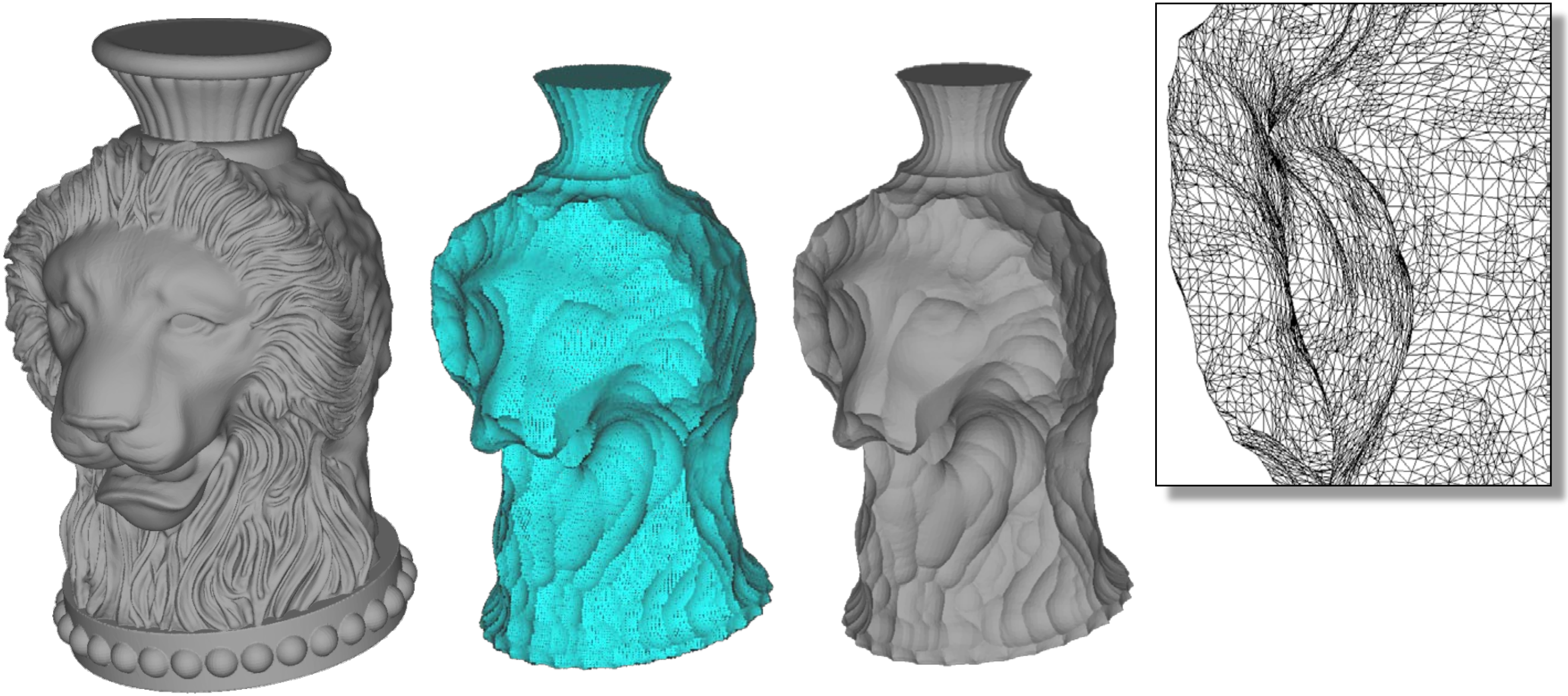






Other Solid Modeling Operations

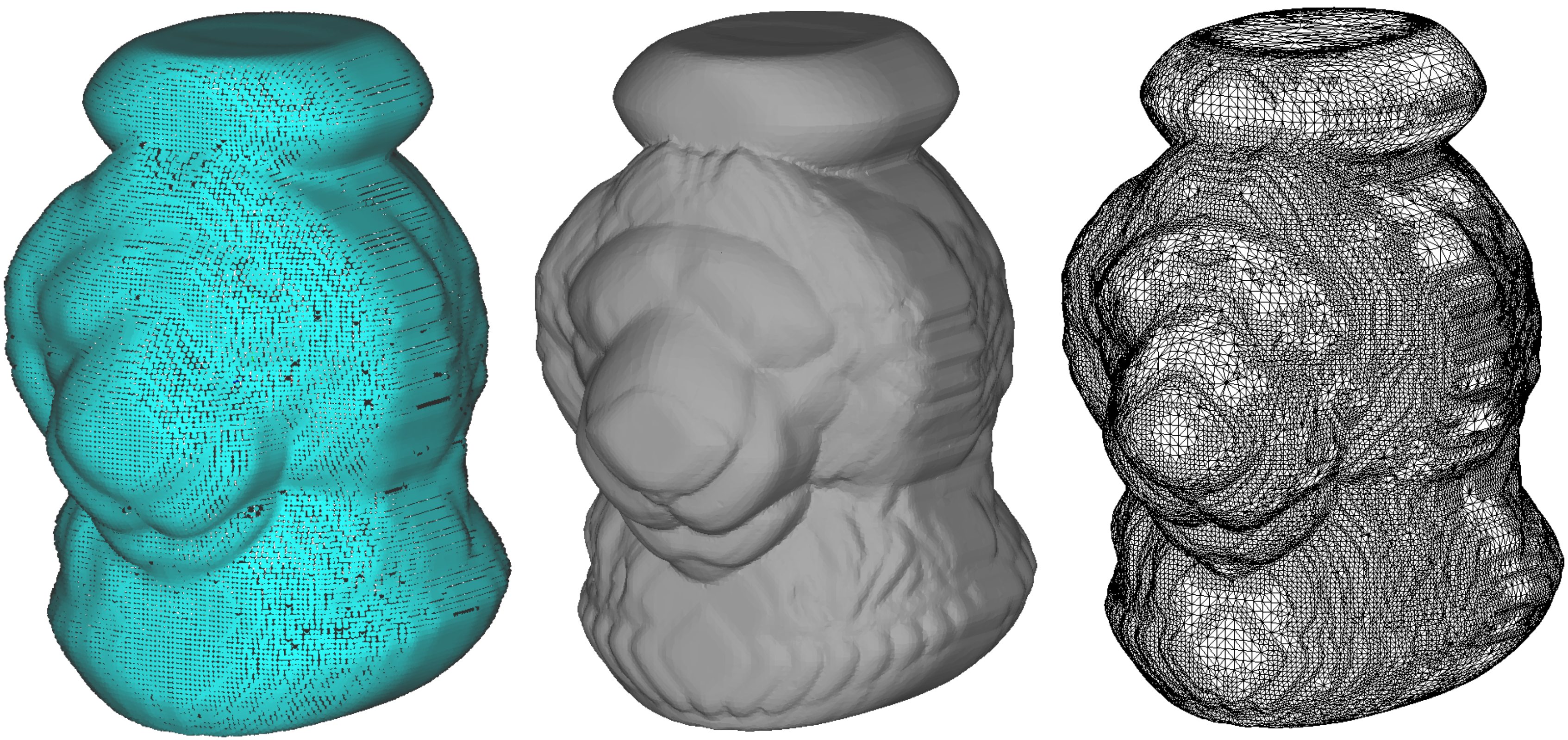
- Offsetting: parallel implementation on CPU with multiple cores (6.35 sec on 8-cores)



400k faces

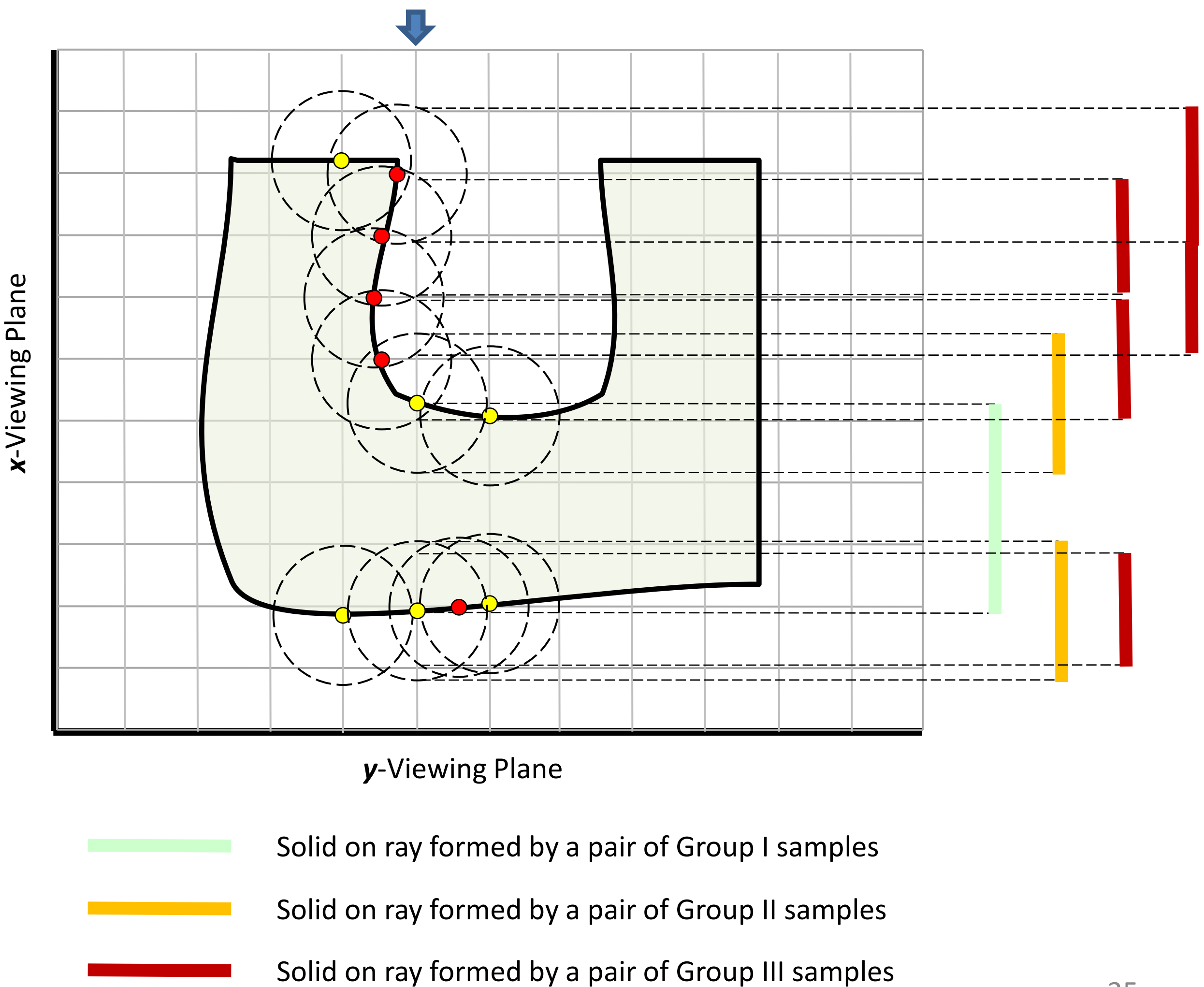
Other Solid Modeling Operations

- Minkowski Sum: parallel implementation on CPU with multiple cores (14.46 sec on 8-cores)



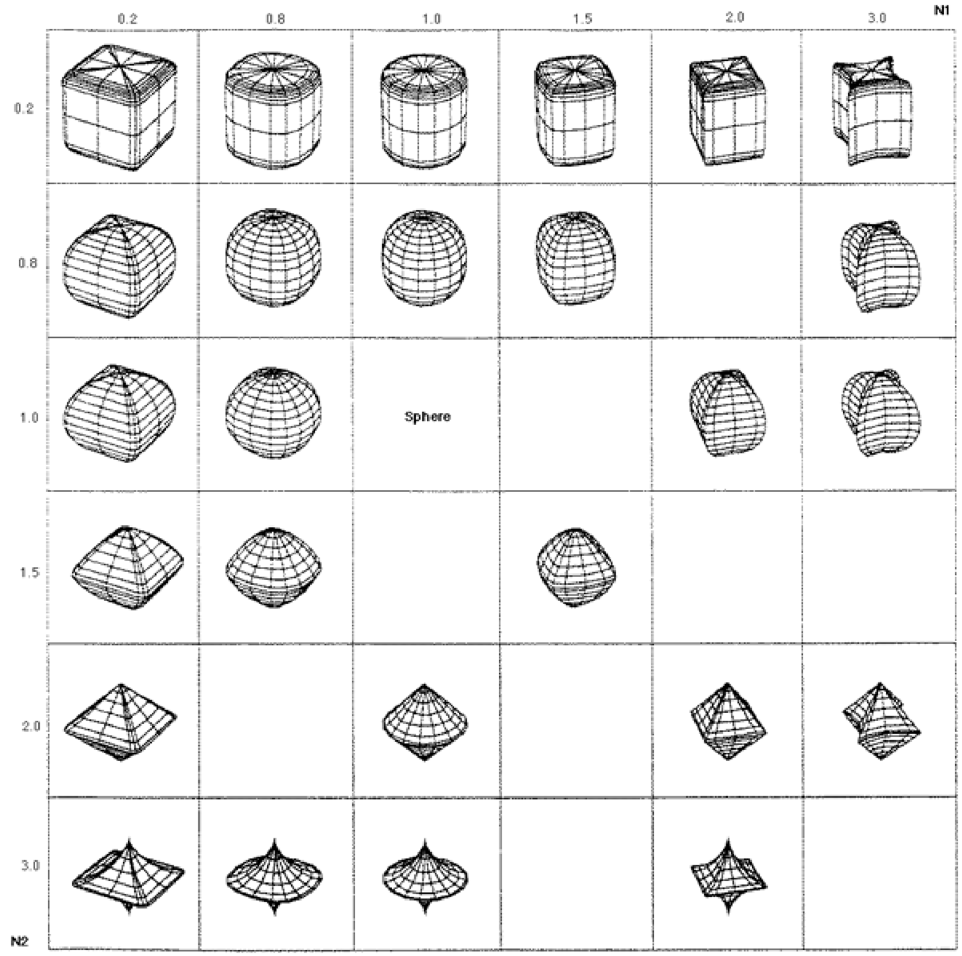
Parallel Computing of General Convolution Surface

- Samples are from three groups



Super-Ellipsoid

- Analytically evaluated
- Covering many shapes



$$\left(\left|\frac{x}{r_x}\right|^{\frac{2}{n_2}}+\left|\frac{y}{r_y}\right|^{\frac{2}{n_2}}\right)^{\frac{n_2}{n_1}}+\left|\frac{z}{r_z}\right|^{\frac{2}{n_1}}=1$$

Charlie C.L. Wang, "Computing on rays: a parallel approach for surface mesh modeling from multi-material volumetric data", Computers in Industry, vol.62, no.7, pp.660-671, September 2011.