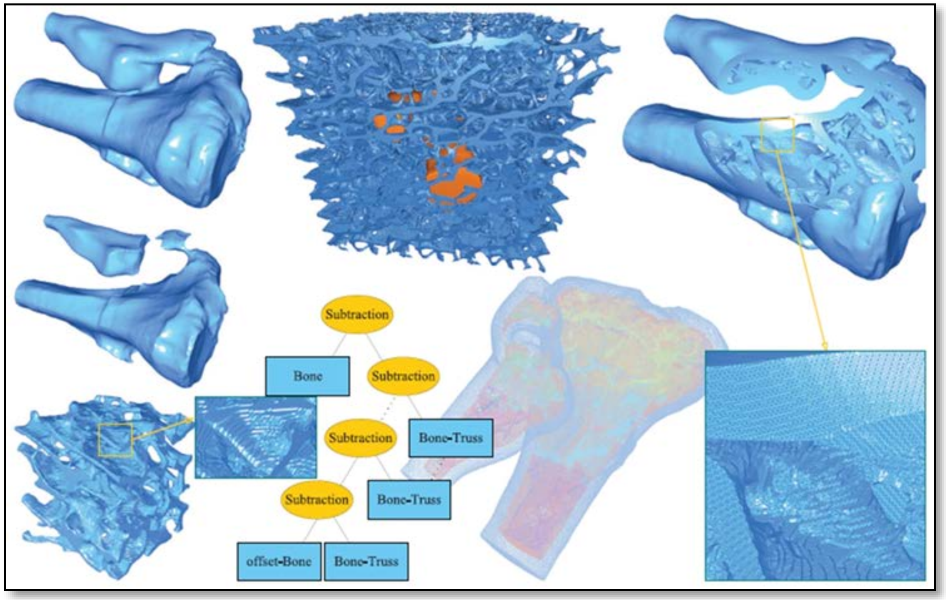
L8 – GPU-based Solid Modeling for AM

- Introduction
- Related Work
- Structured Point Representation
- Boolean Operations
- Offsetting by Super-Union of Balls
- Solid Modeling for Rapid Fabrication
- Conclusion

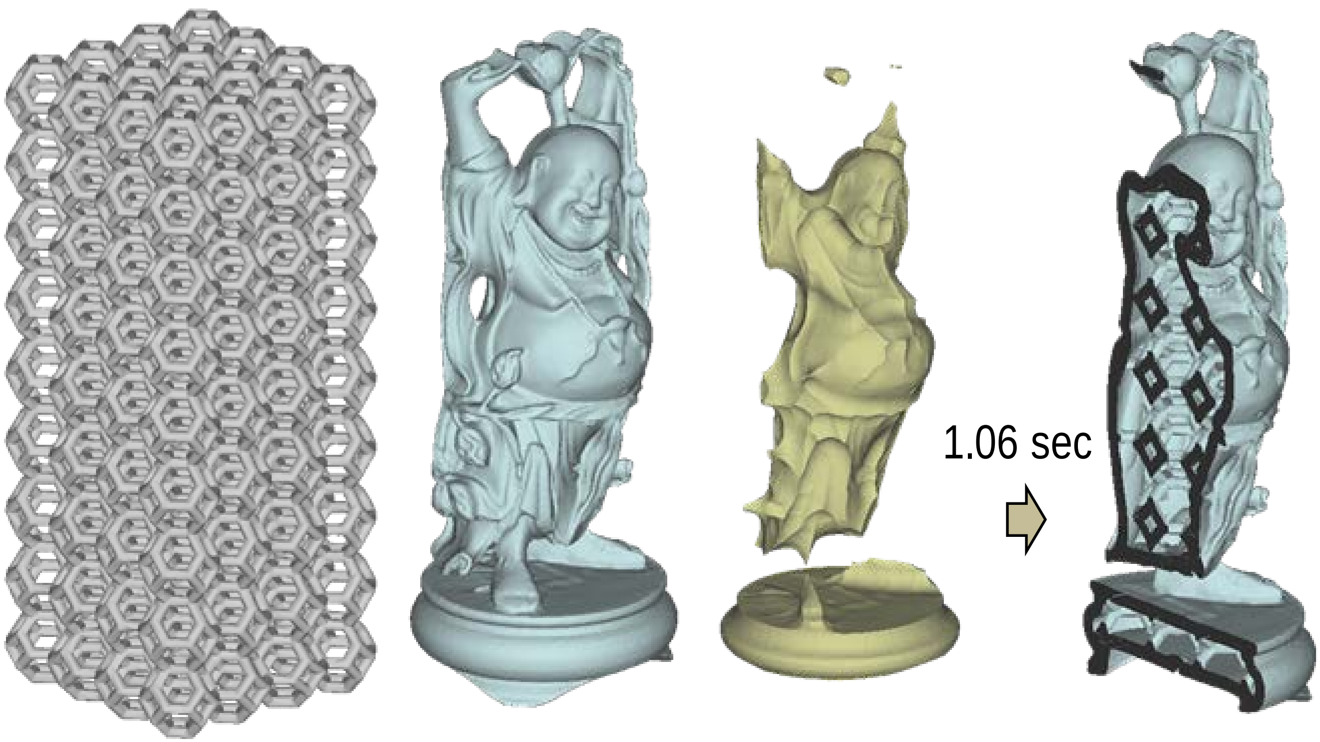
Solid Modeling for Fabrication

- **Framework:** *GPU-based Solid Modeler for Complex Objects*
- **Purpose:** using the computational power on GPU to speed up solid modeling operations in Layered Depth-Normal Images (LDNI) rep.
- Models in many applications have very complex shape and topology (e.g., microstructure design, rapid prototyping, etc.)

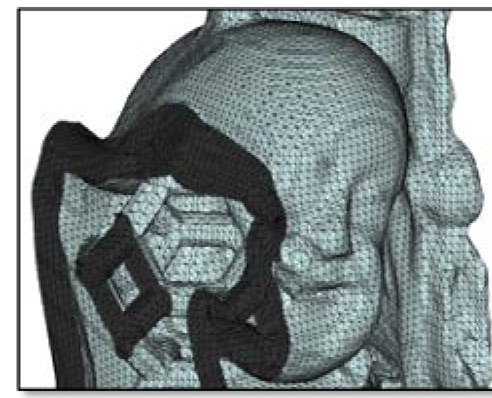


Boolean Operations

- 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



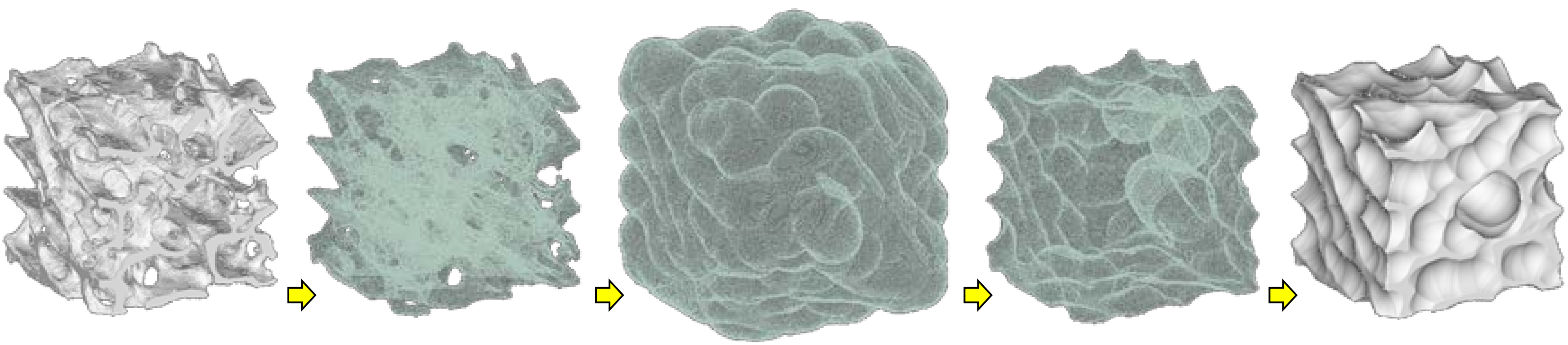
On GeForce GTX 580

Offsetting

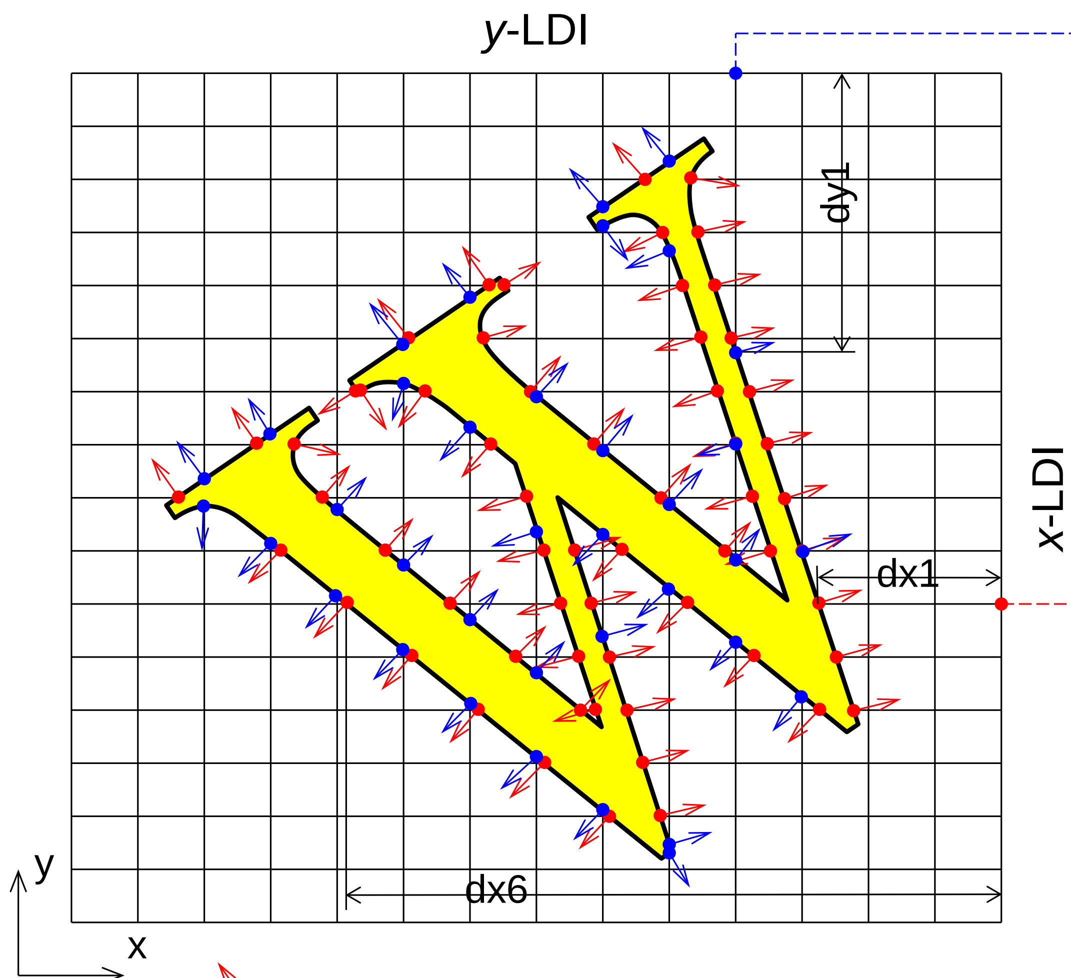
- **Problem Definition:**

  Given a solid model H with its boundary surface approximated by the set P_H of sample points in LDNI-representation, we compute the boundary surface of exterior offset (or interior offset) and represent it by a point set in LDNI-representation.

- **Our Approach:**
  - Fast Approximate Offsetting
  - Directly offsetting solid models in LDNI representation
  - Highly parallel algorithm



Structured Point Representation - LDI



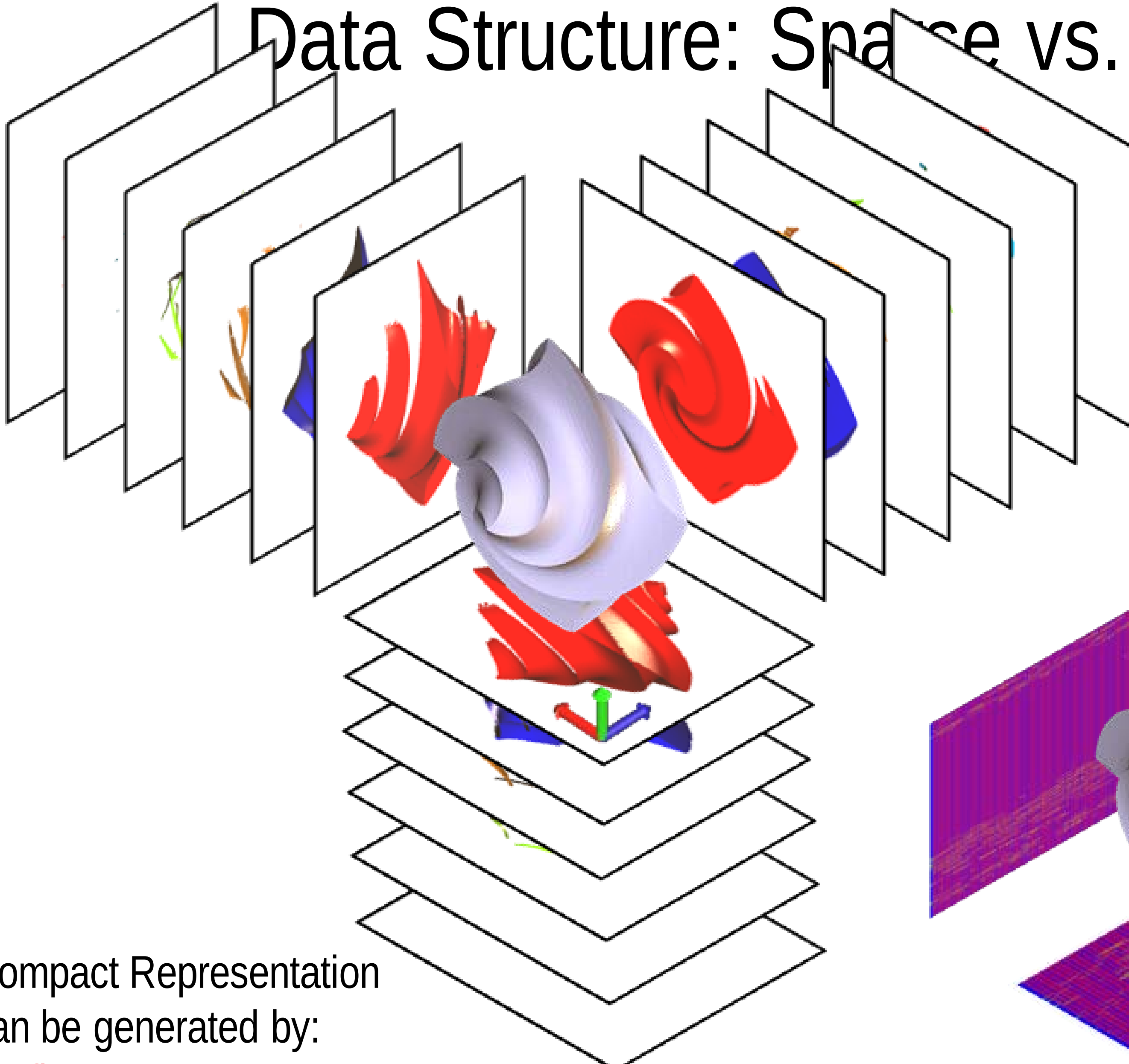
Sampled depth on *x*-LDNI coupled with surface normal

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

4			
dy1	$n1_x$	$n1_y$	$n1_z$
dy2	$n2_x$	$n2_y$	$n2_z$
dy3	$n3_x$	$n3_y$	$n3_z$
dy4	$n4_x$	$n4_y$	$n4_z$

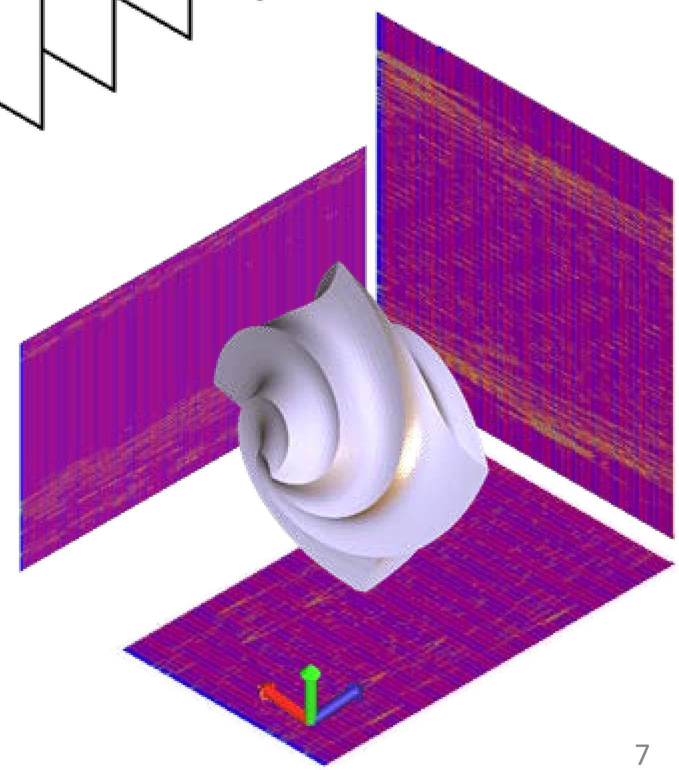
6			
dx1	$n1_x$	$n1_y$	$n1_z$
dx2	$n2_x$	$n2_y$	$n2_z$
dx3	$n3_x$	$n3_y$	$n3_z$
dx4	$n4_x$	$n4_y$	$n4_z$
dx5	$n5_x$	$n5_y$	$n5_z$
dx6	$n6_x$	$n6_y$	$n6_z$

Data Structure: Sparse vs. Compact



Two Arrays:

1) 2D Index Array
2) 1D Data Array

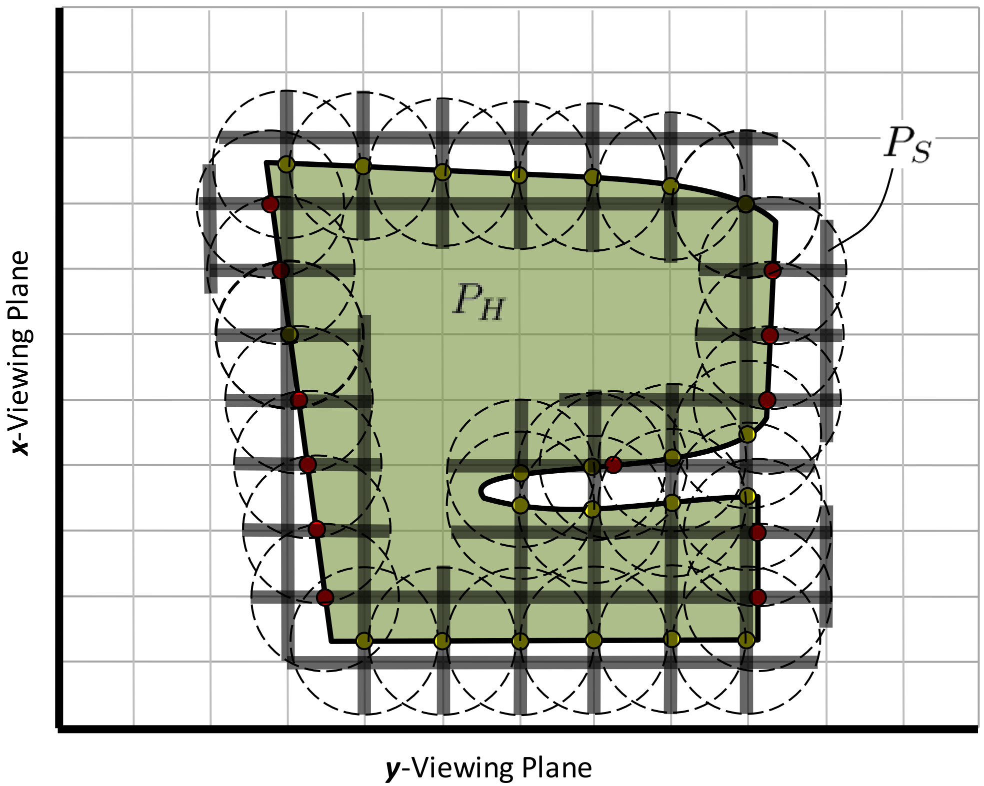


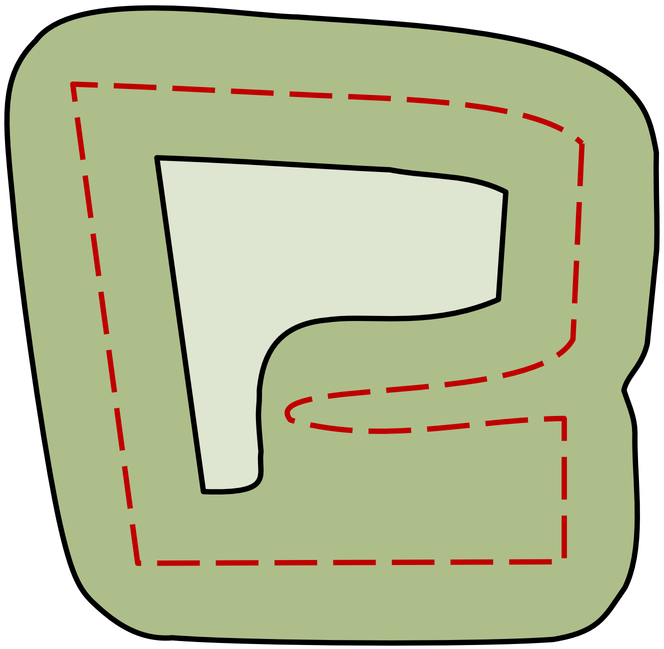
Compact Representation can be generated by: Prefix-sum Scan

Offsetting by Super-Union of Balls

- Boolean operations on LDNI solids – Boolean on 1D rays
  - Highly parallel and robust
- Offsetting shell P_S – by union many balls

Offsetting Result:
Exterior Offset = $P_H \cup P_S$
Interior Offset = $P_H - P_S$





Offsetting by Super-Union (Cont.)

- Ray-based Computation
- Two Groups

I) By rays in the same view

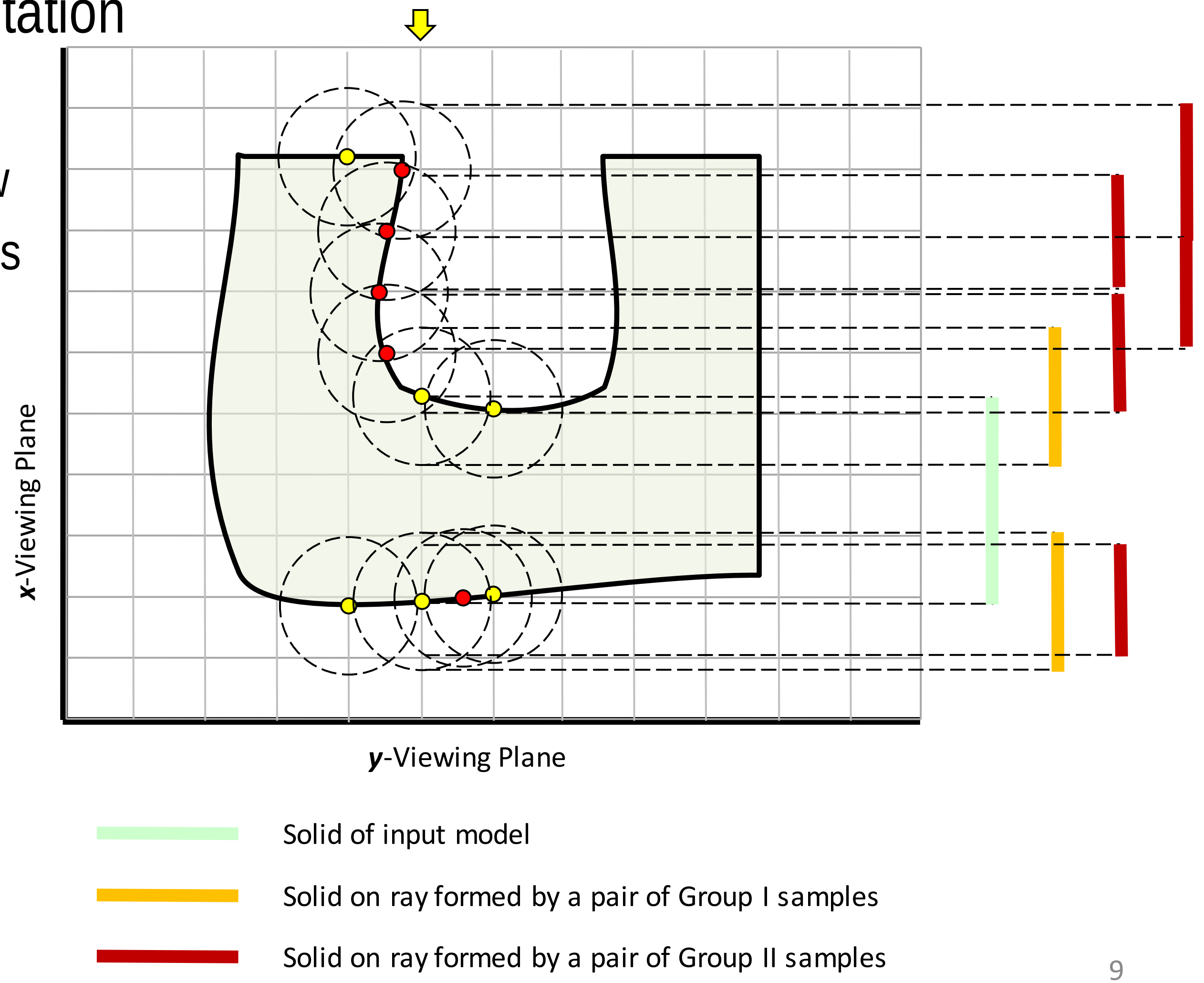
II) By rays in different views

- Super-union

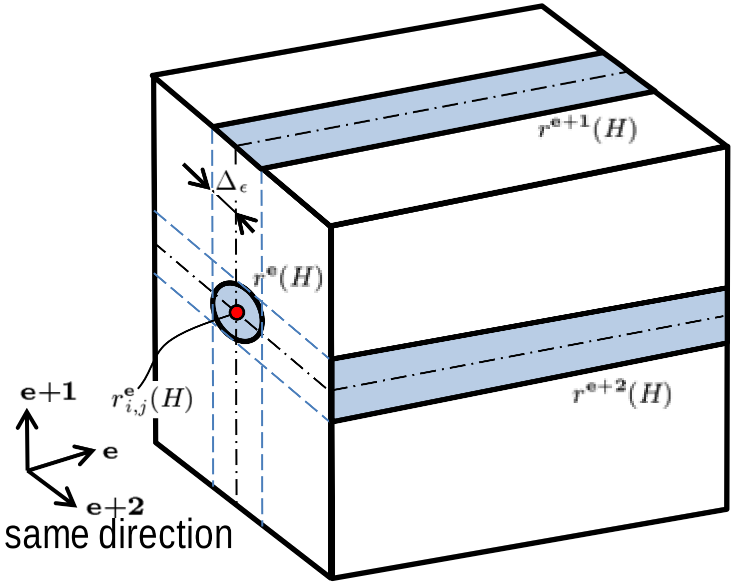
Status update by entering / leaving samples

- Problem:

Efficient ray-sphere intersection detection



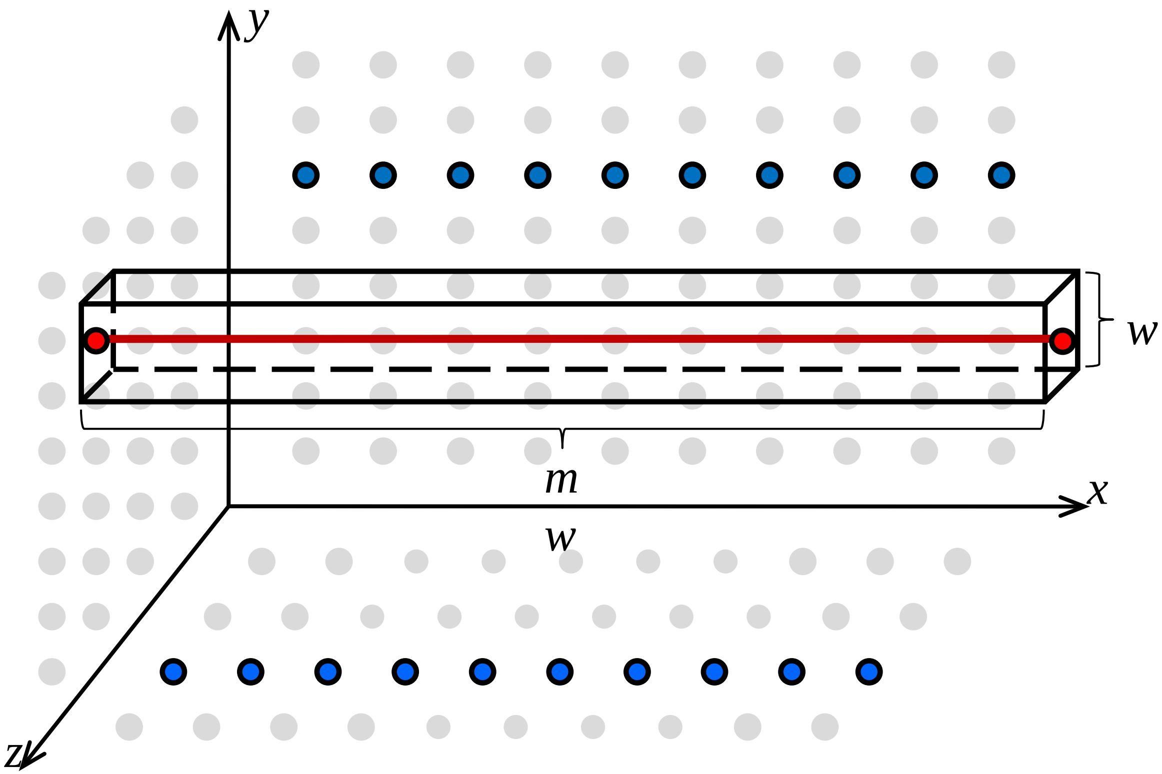
Primary GPU Scheme



- Three steps algorithm:
  1) For each ray in one direction in parallel
     - Search the intersections between this rays and spheres generated by samples on the rays in the same direction
     - Merging intersected 1D solids
     - Storing the result in a global data buffer array
  2) For each ray in one direction in parallel
     - Search the intersections between this ray and spheres centered at the rays in other directions
     - Merging intersected 1D solids into the existing 1D solid on this ray
  3) Rebuild the index array and the resultant data array (by Prefix-sum Scan)
- Reconstruct normal vectors on the resultant samples
  - Orientation-aware *Principal Component Analysis* (PCA)
  - Carry on the neighborhoods of a sample

GPU-based Algorithm: Spatial Hashing

- Bottleneck of primary GPU algorithm – Step 2) taking 80%-85% time
- Searching too many rays in other directions: (2m) x (2r / w)
- Redundancy: not every ray has sample fall in the range



Solution:
- Sorting samples from other rays by their coordinate in the yoz plane
- Building spatial hashing bins around ray in x direction
- Step 2) can be conducted by only searching samples in these bins

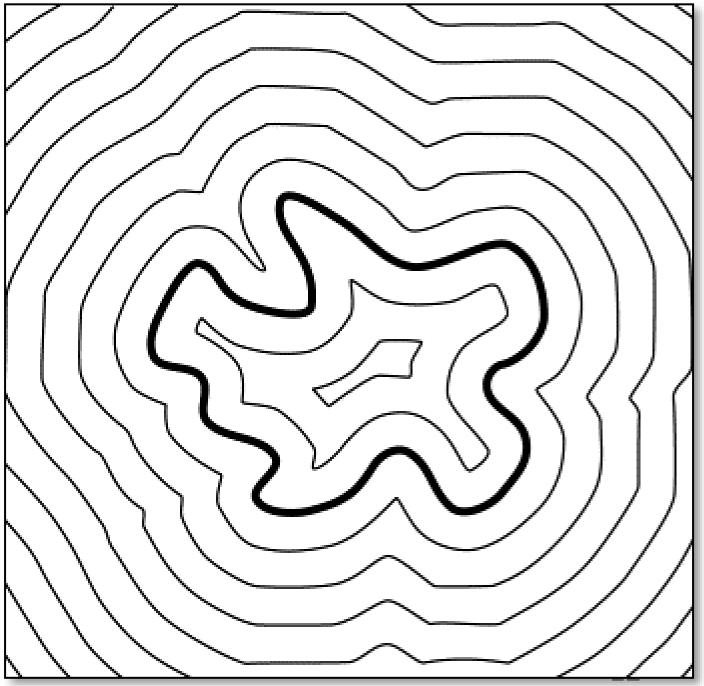
Result: search only (2r / w) x (2r / w) bins

Successive Offsetting for Large Offset

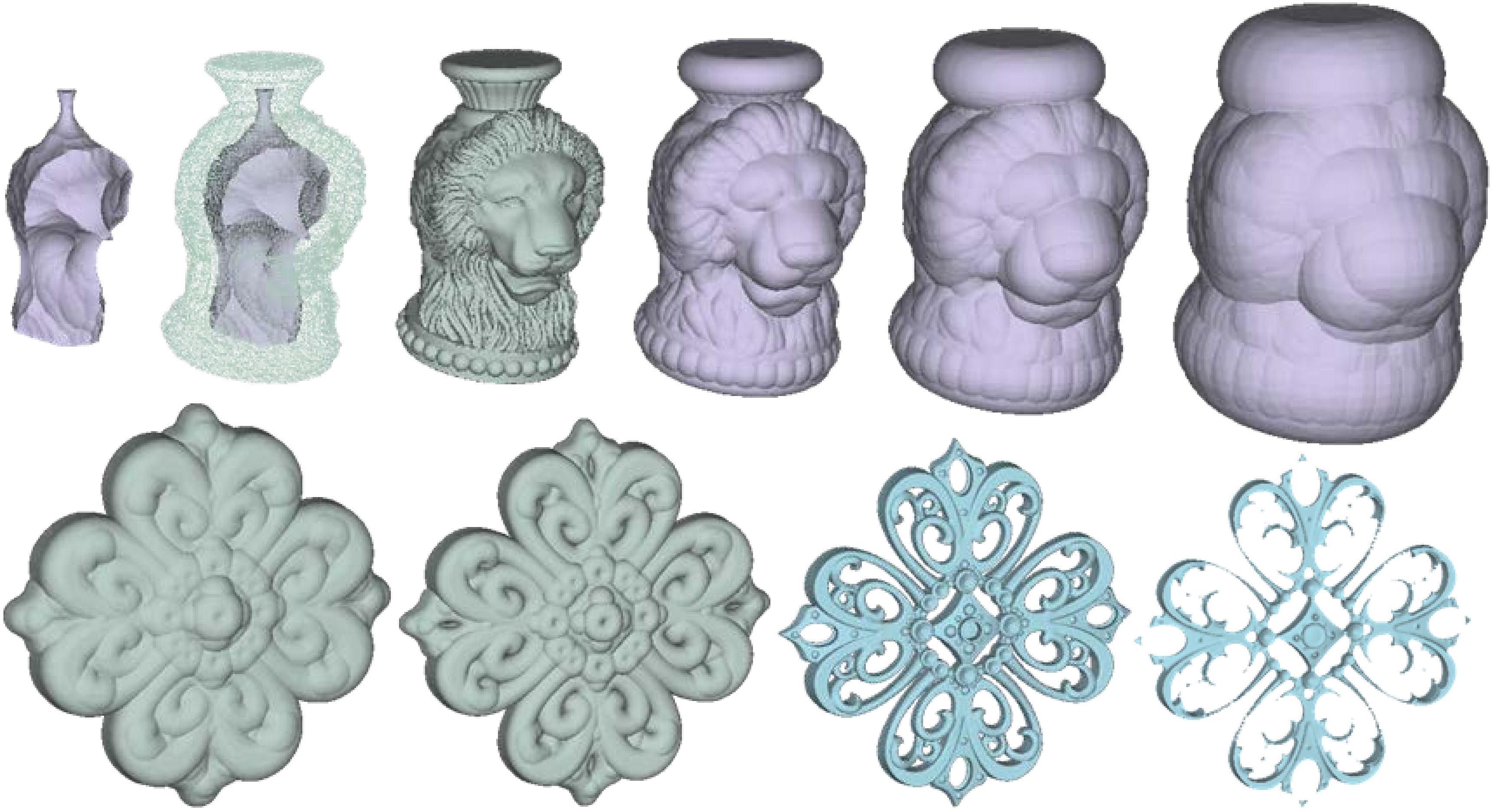
- Computation cost on each ray: $O((2r/w)^2)$ – the search range
- Slow, when r is very large
- Offsetting with large distance r can be decomposed into n successive offsetting with smaller distance r_i where $r = nr_i$ (Rossignac & Requicha, 1986)
- Computational cost is reduced to $1/n$

  $O(n(2r_i/w)^2) = O((2r/w)^2)/n$
- At the downside, performing offsets too many times in succession

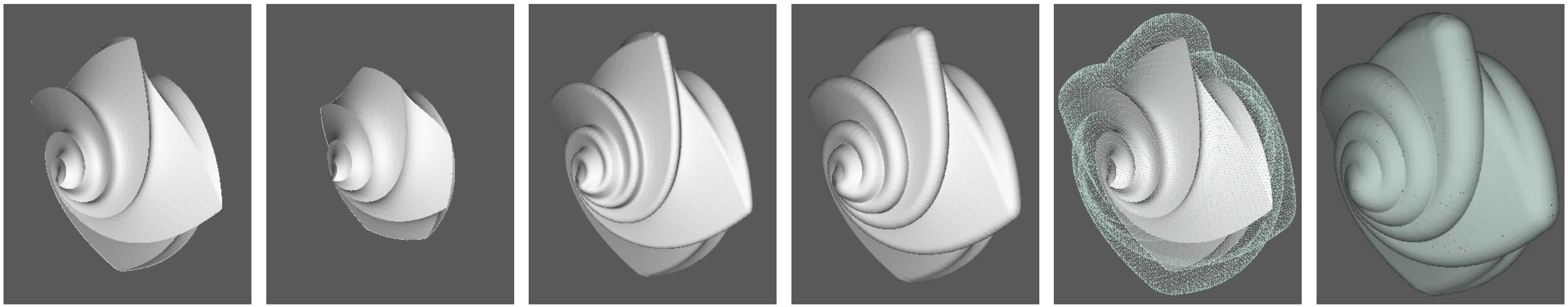
  => Large approximation error

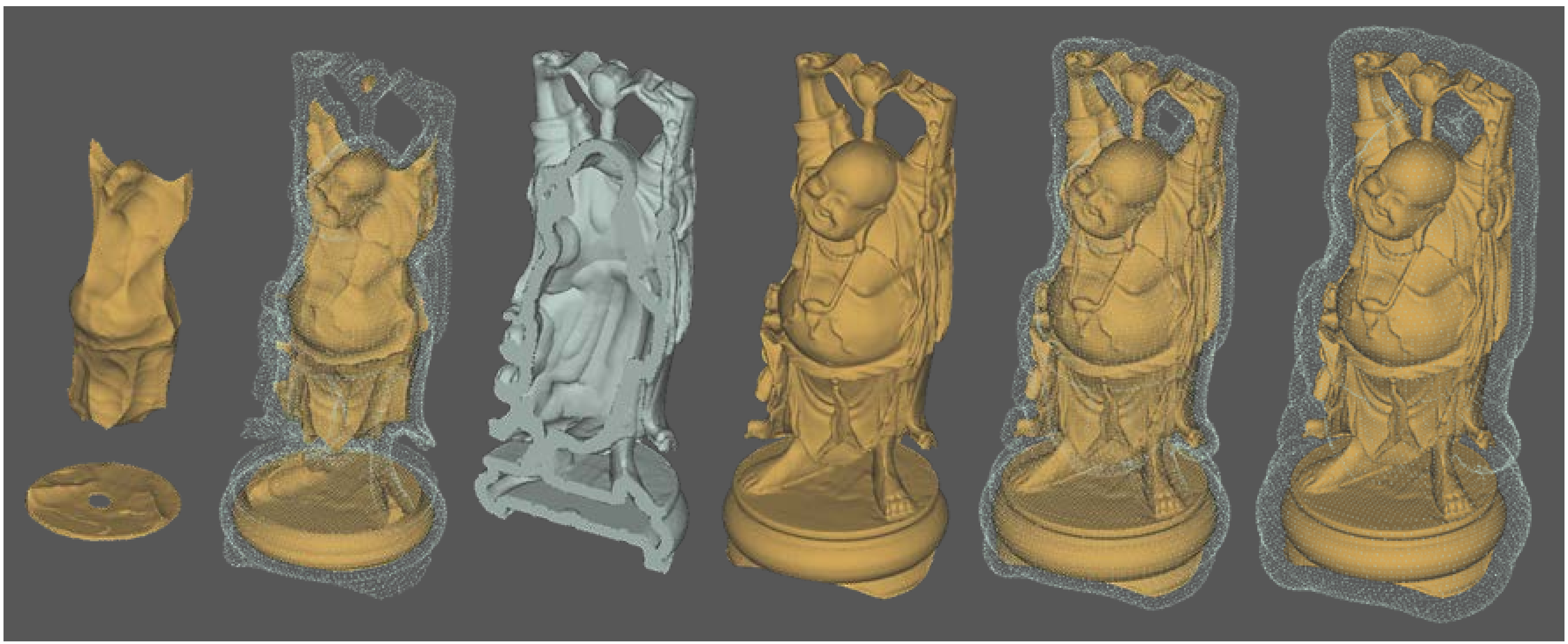


Offsetting Results



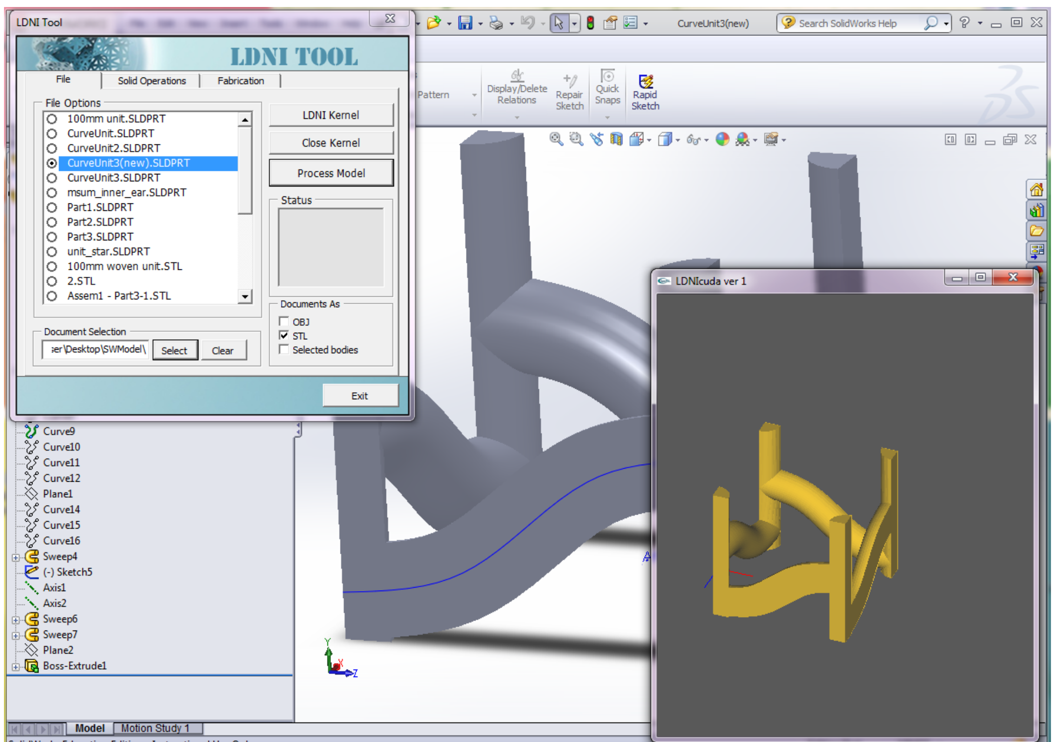
Offsetting on Different Models

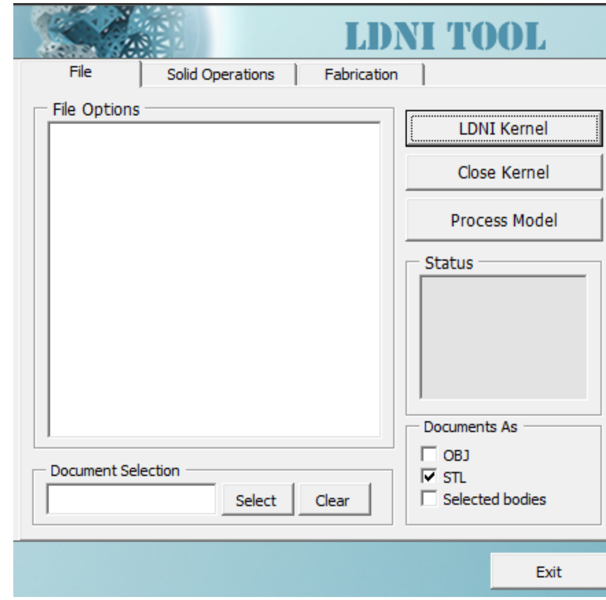


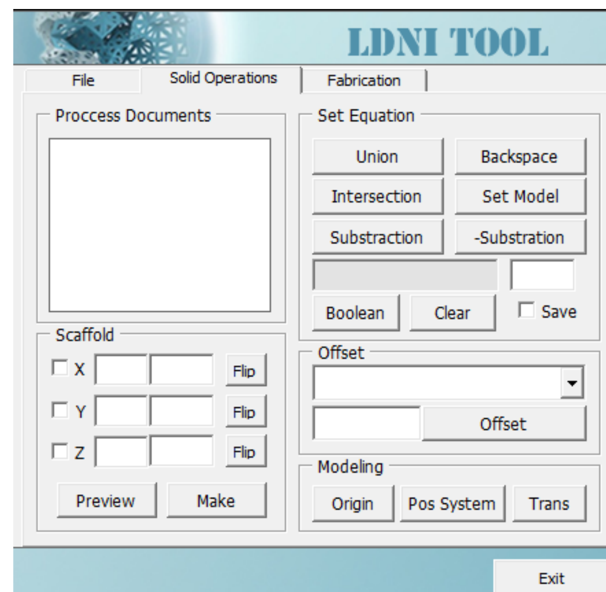


Current Development

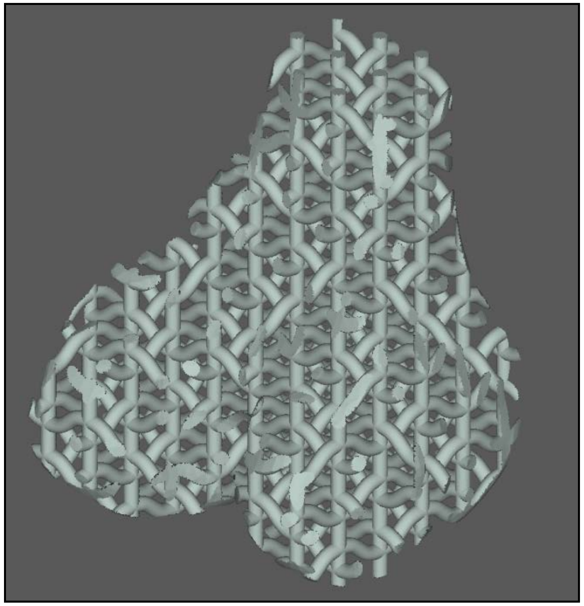
- Not only the framework of our kernel, we also develop an interface for users to interact between the SolidWorks (a commercial CAD tool) and our framework
- Increase the utility of our work

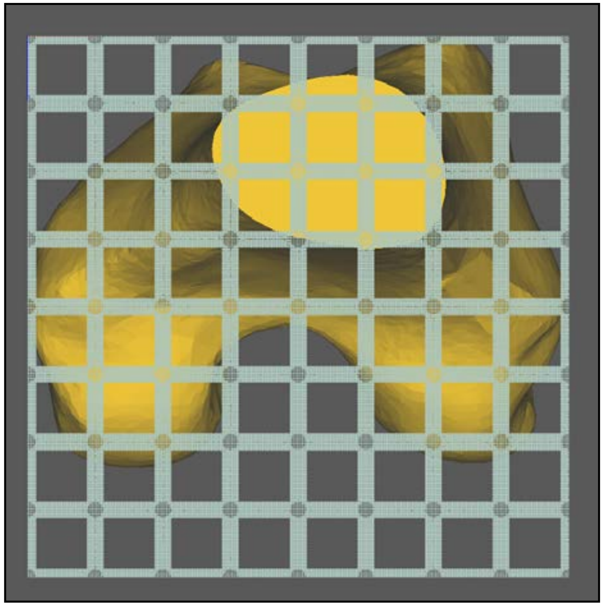


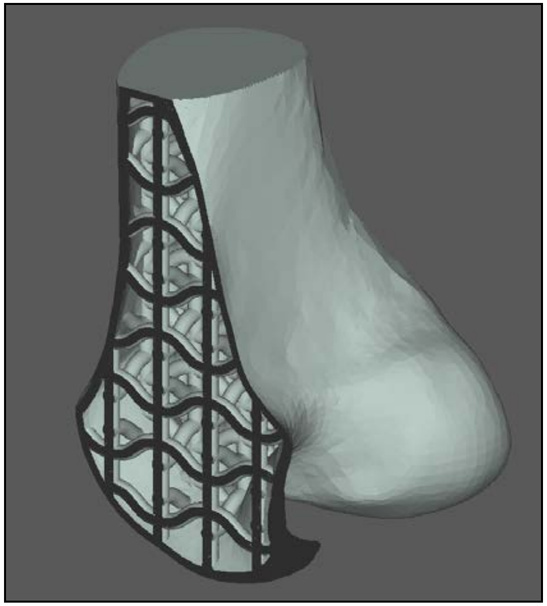




Boolean Operations







LDNI Resolution	1024 × 1024
Models	Femur ∩ Scaffold (832 cells)
GPU Memory Usage	27.3MB
*Operation Time (sec)	3.71s

LDNI Resolution	1024 × 1024
Models	Femur / (FemurOff / Scaffold)
GPU Memory Usage	49.6MB
*Operations Time (sec)	4.07s

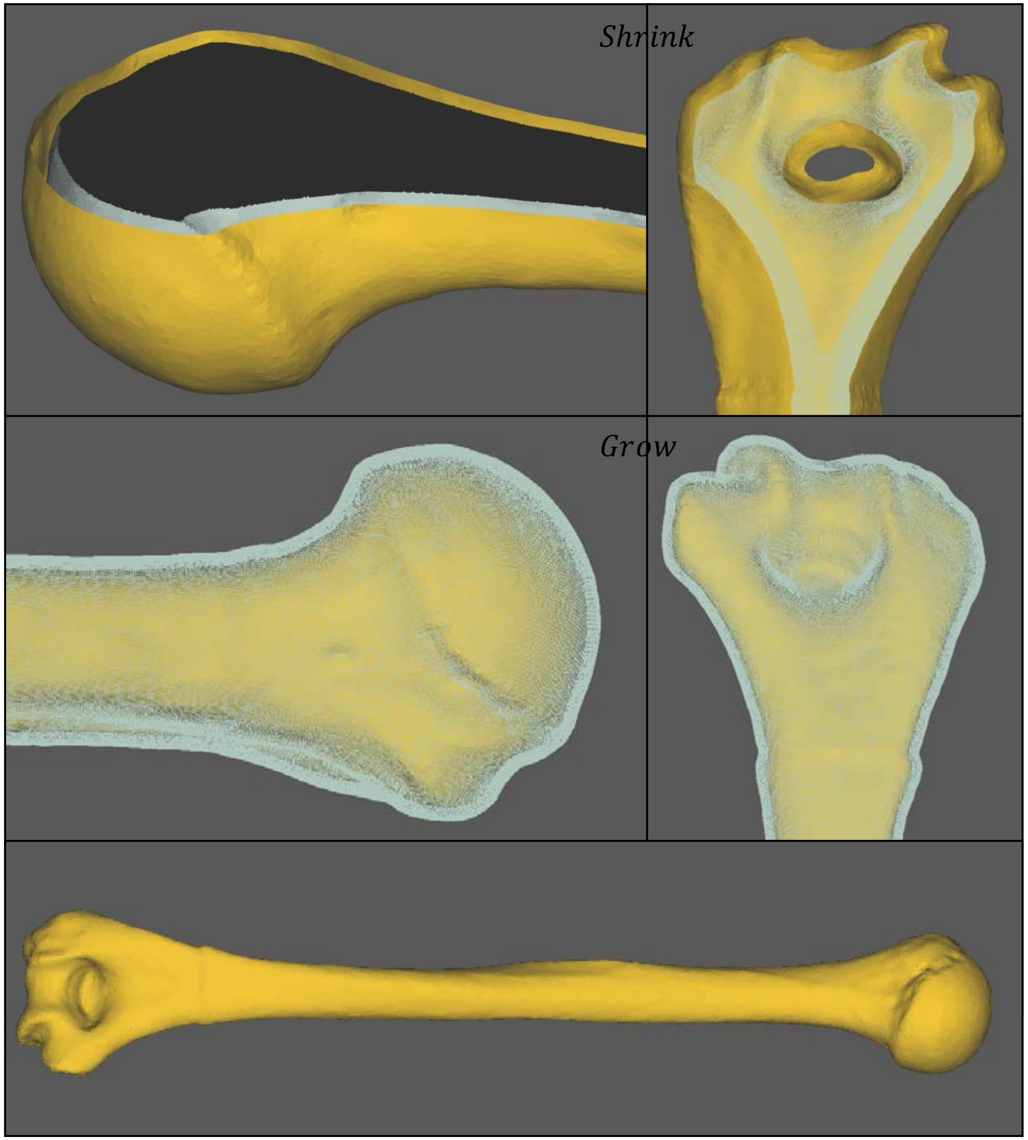
*Included scaffolding and sampling time

Offsetting

Shrinking Offset	
LDNI Resolution	2048×2048
Offset value	$-15 \times \varepsilon$
Face Num.	70K
GPU Memory Usage	84.1 MB
Operation Time (sec)	8.14s

Growing Offset	
LDNI Res	2048×2048
Offset value	$10 \times \varepsilon$
Face Num.	70K
GPU Memory Usage	32.8 MB
Operation Time (sec)	4.602s

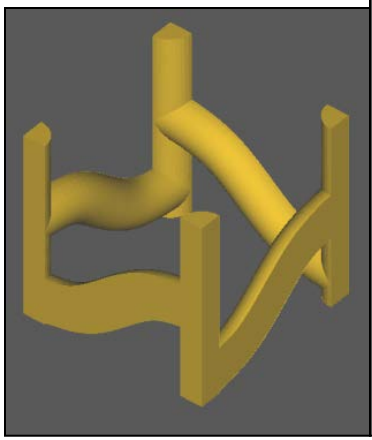
$^{*}\varepsilon$ = *sampling distance width*

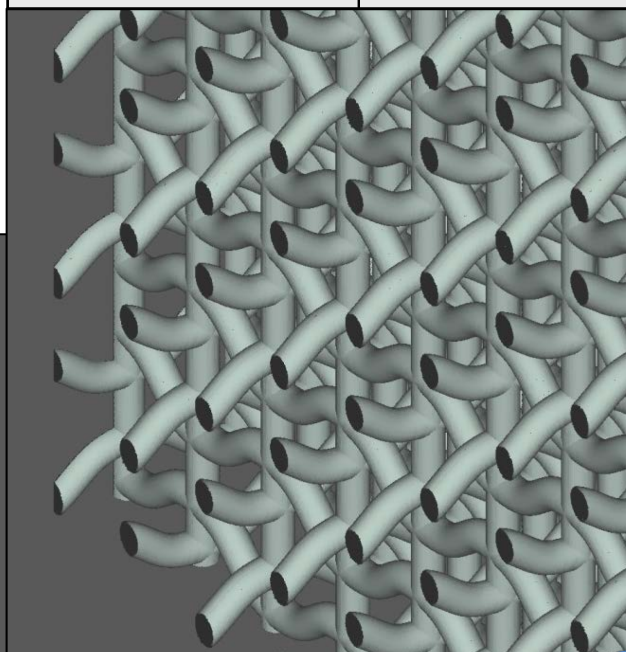


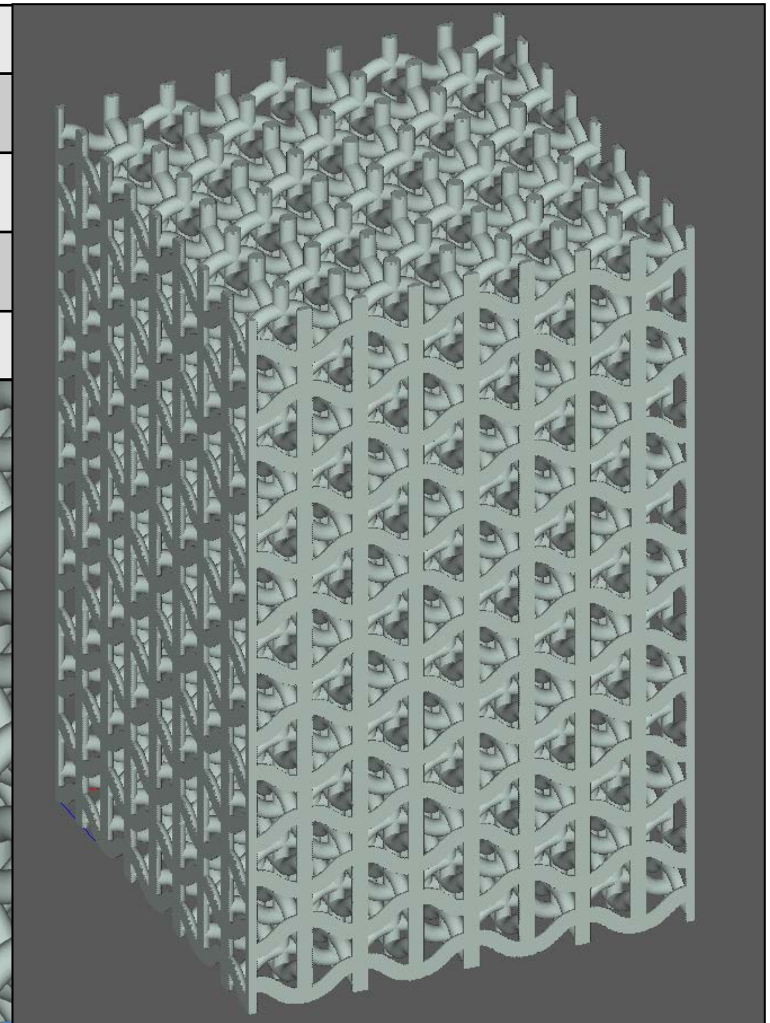
Scaffolding

- Union operations applied on instances of a model at the same time

LDNI Res	1024 × 1024
Cell Num.	8 x 13 x 8 (832)
Face Num.	7.6K/per cell
GPU Memory Usage	151 MB
Operation Time (sec)	3.46s



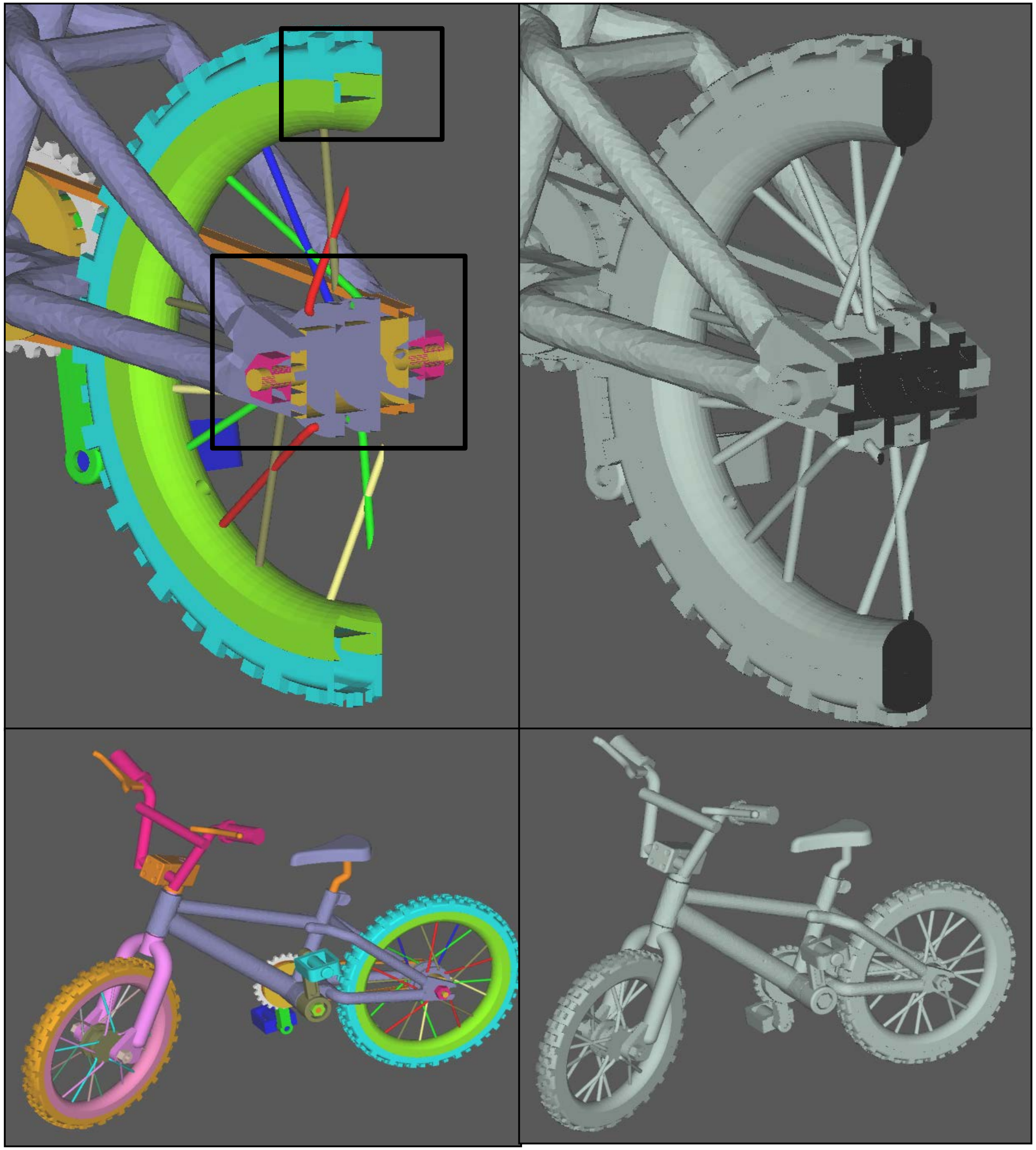




Super-Union

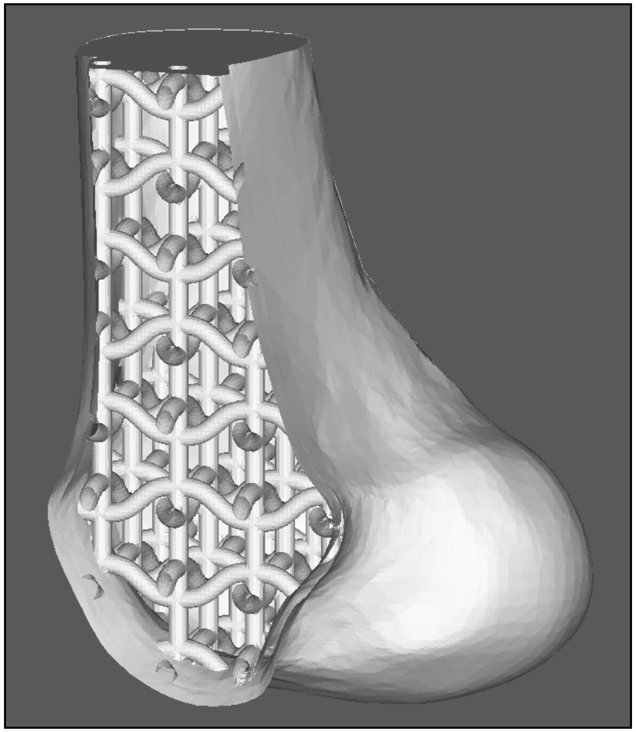
- Union operations applied on multiple different models at the same time
- Overlapped or intersected objects can be converted into one solid

Components	32
Total Face Num.	161K
Resolution	4096 × 4096
GPU Memory Usage	232.6MB
Operation Time (sec)	1.88s

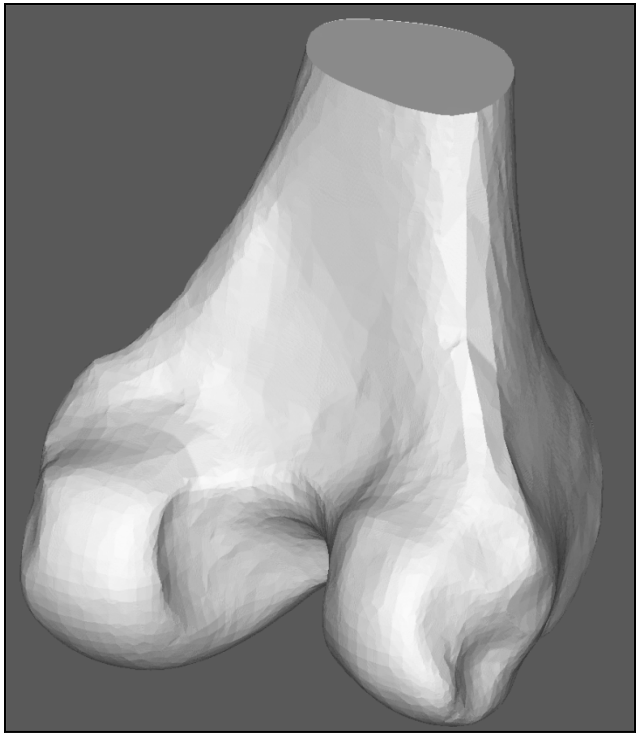


Contouring

- Convert LDNI back to B-rep representation
- For further operations that require boundary information



Boolean of LDNI →Mesh	
LDNI Res	1024 × 1024
Face Num.	513K
Time (sec)	0.23s

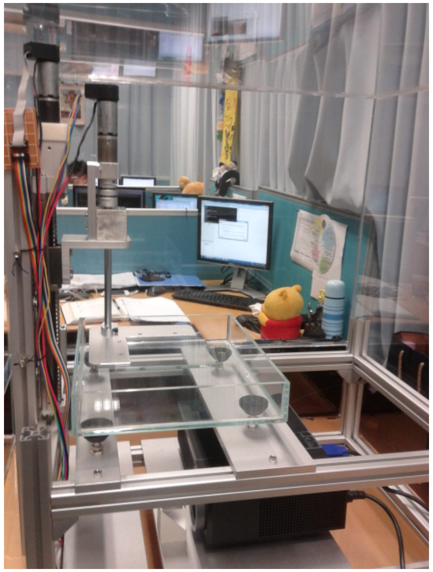


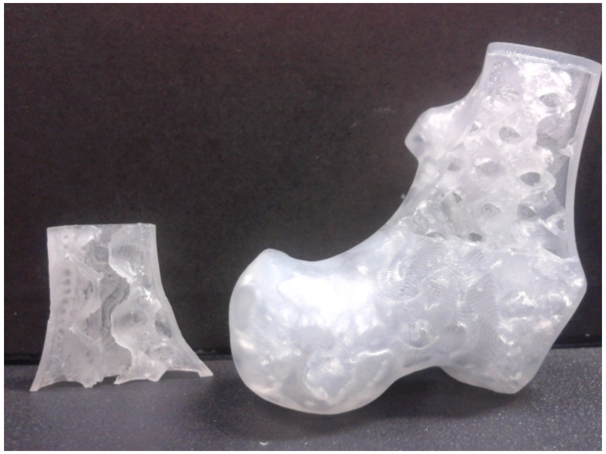
Offset in LDNI → Mesh	
LDNI Res	1024 × 1024
Face Num.	338K
Time (sec)	0.11s

[Link of source code]

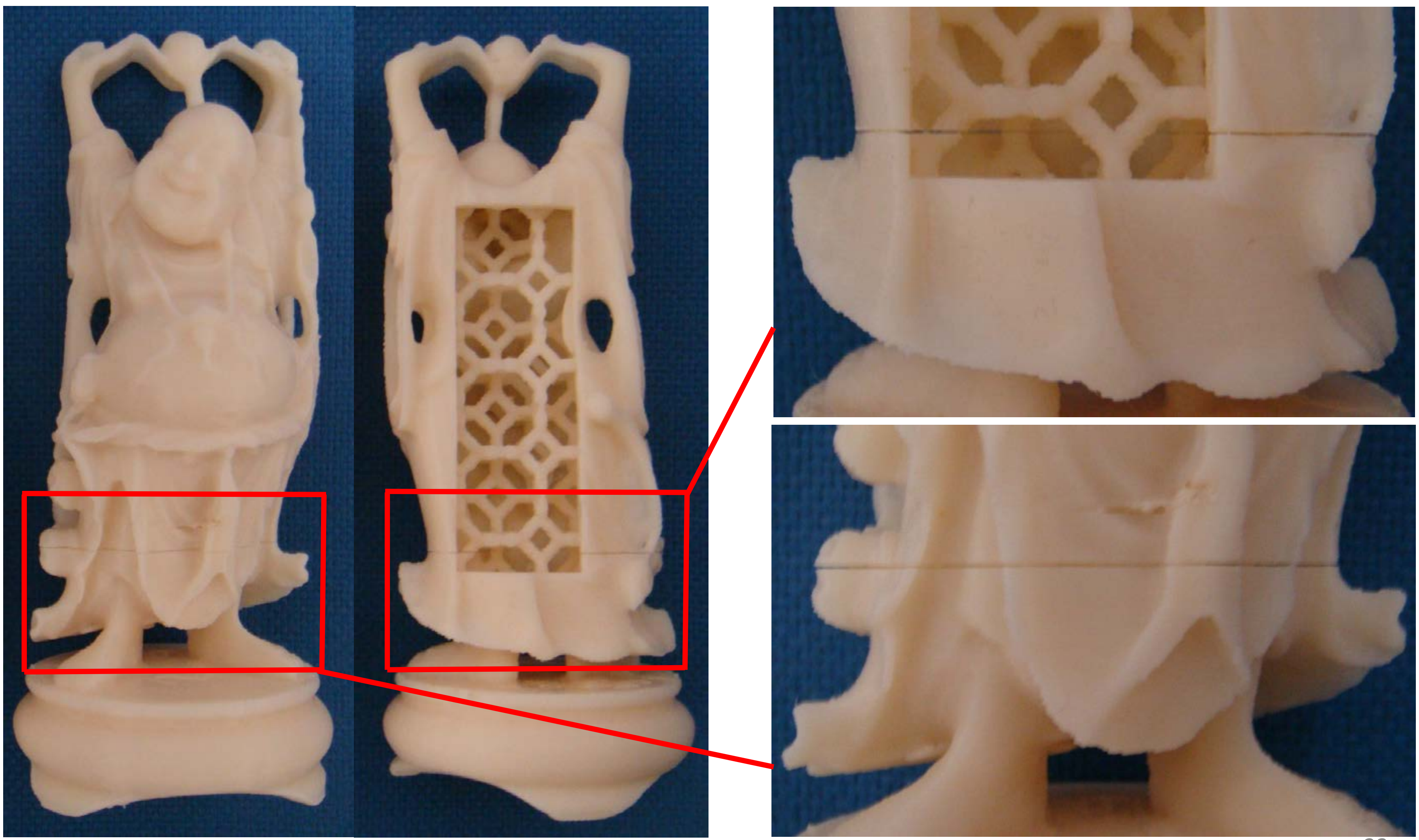
Downstream Apps

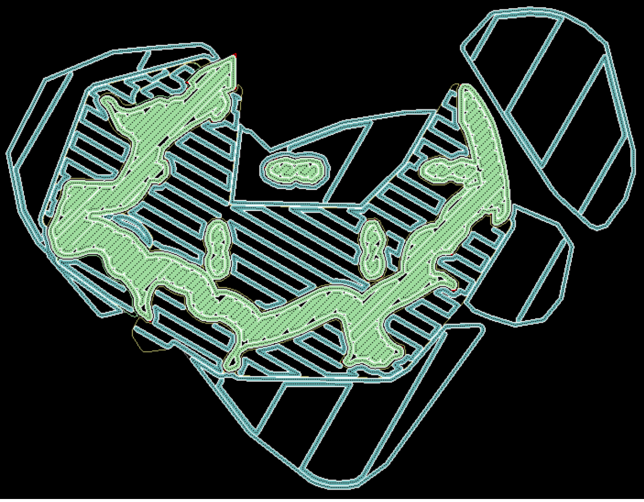
- Fused Deposition Modeling (FDM)
- StereoLithography Apparatus (SLA)
  - Contours are needed
- Mask-projection SLA
  - Direct binary image projection



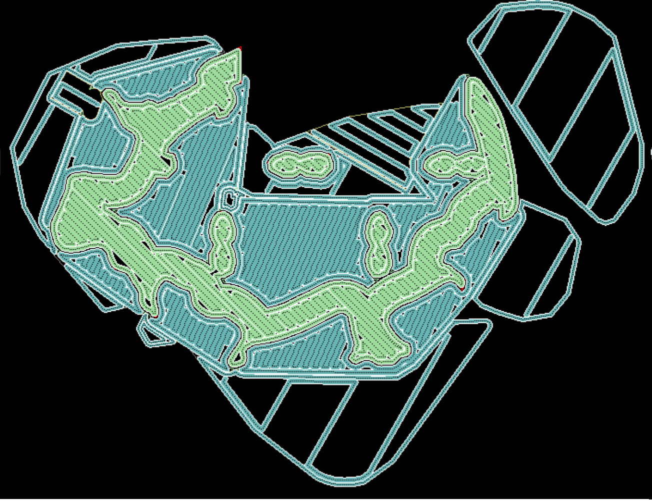


Problem with Existing Approaches (by B-rep)

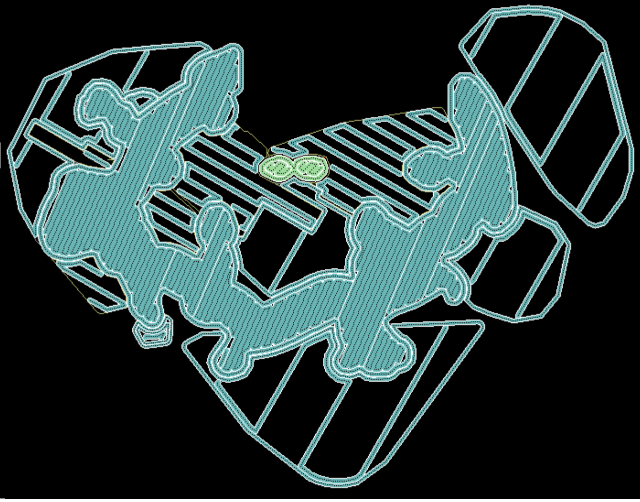




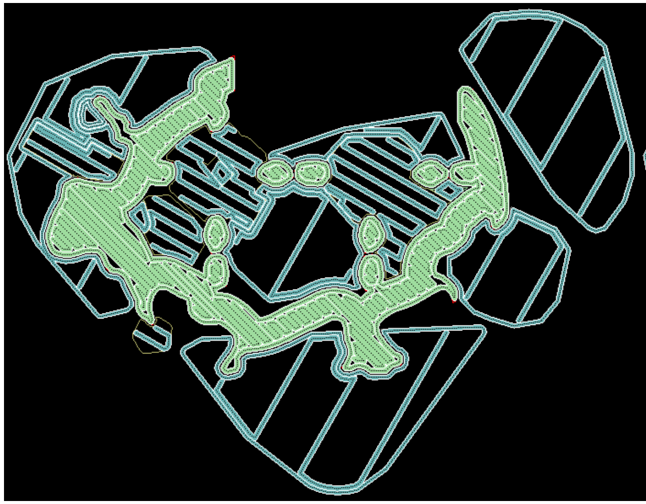
Height = 1.77 inch



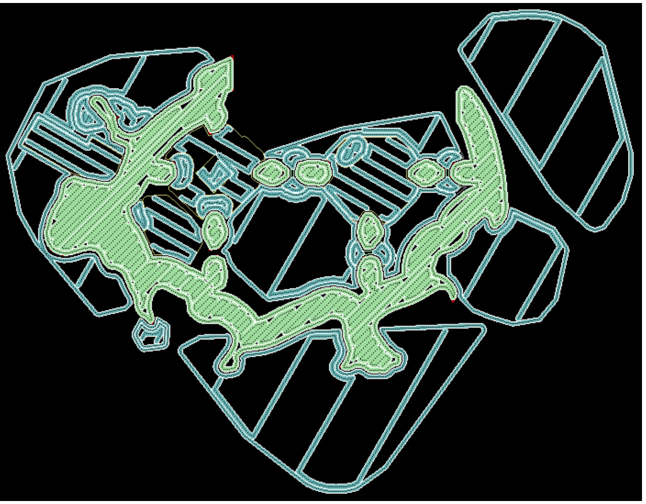
Height = 1.78 inch



Height = 1.79 inch



Height = 1.80 inch

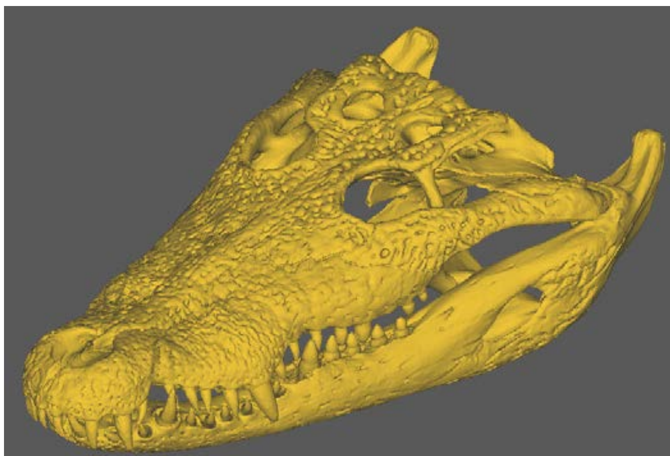


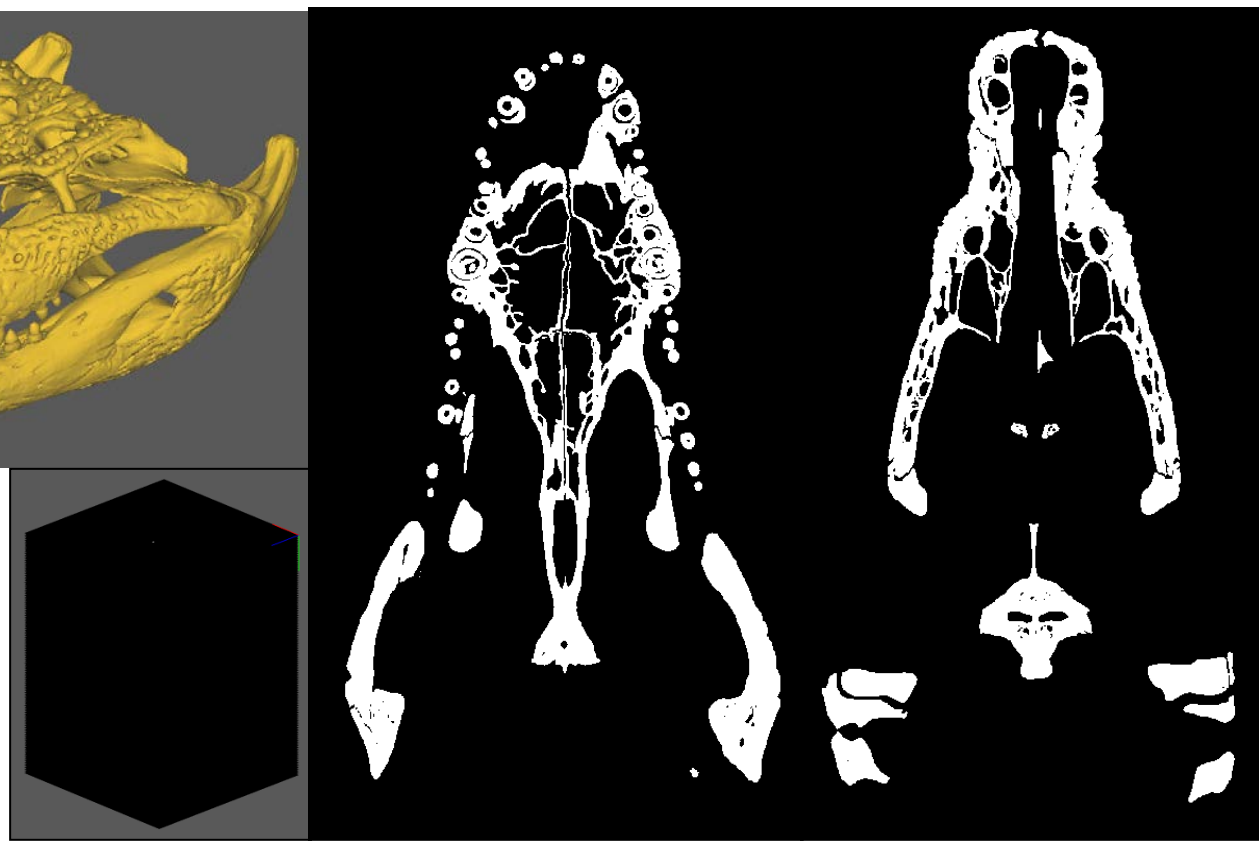
Height = 1.81 inch

Generated by Commercial Software for FDM

How to provide reliable information for fast fabrication?

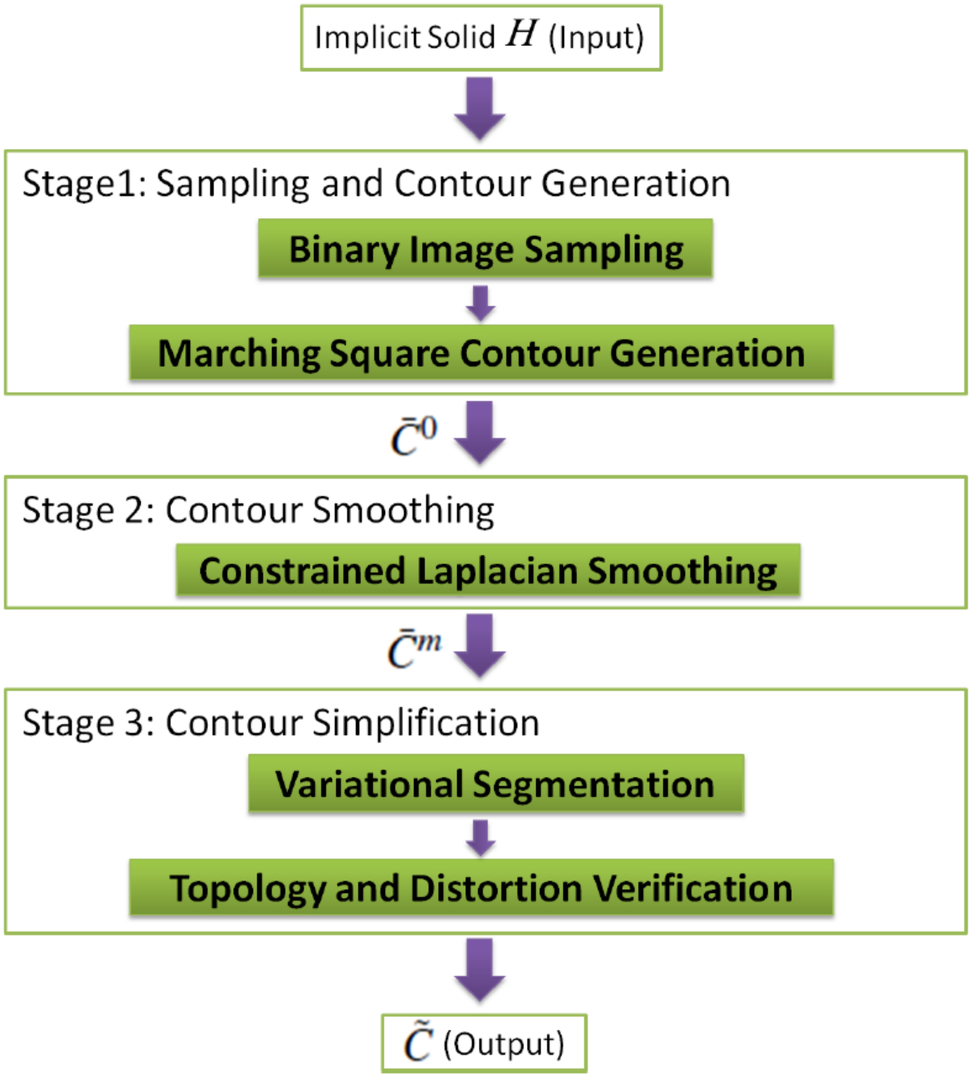
- Slicing or Modeling (by LDNI-rep) in image space
- Fabrication in image space – Mask-Projection based SLA





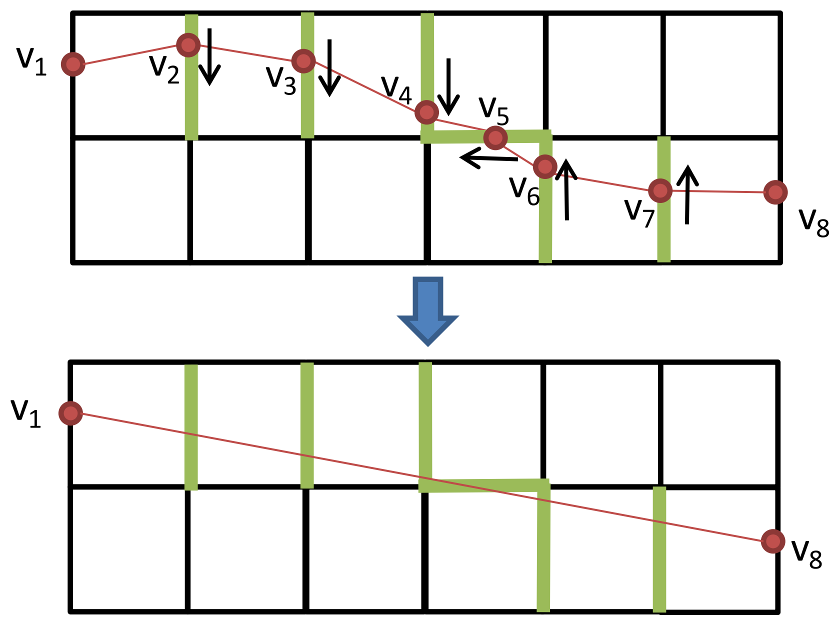
LDNI Res	1024 × 1024
Image Size	2000 x 132 x 2000
Time (sec)	9.13s

Reliable Slicing in Image Space



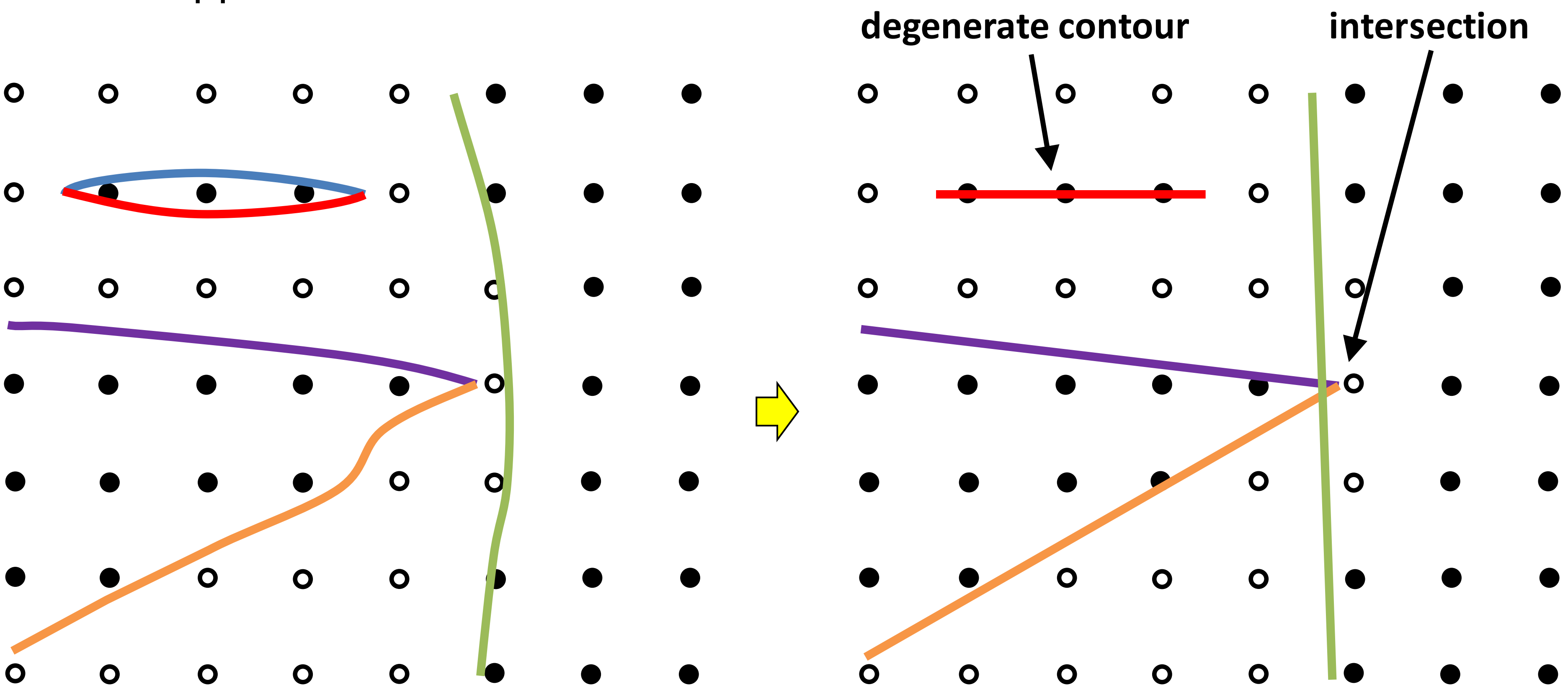
Binary Image Sampling by using the concept of r-regular to guarantee the topological faithful

In the Stages 2 and 3, the self-intersection must be prevented by the stick-concept when sliding on the edges



Self-intersection-free Contours

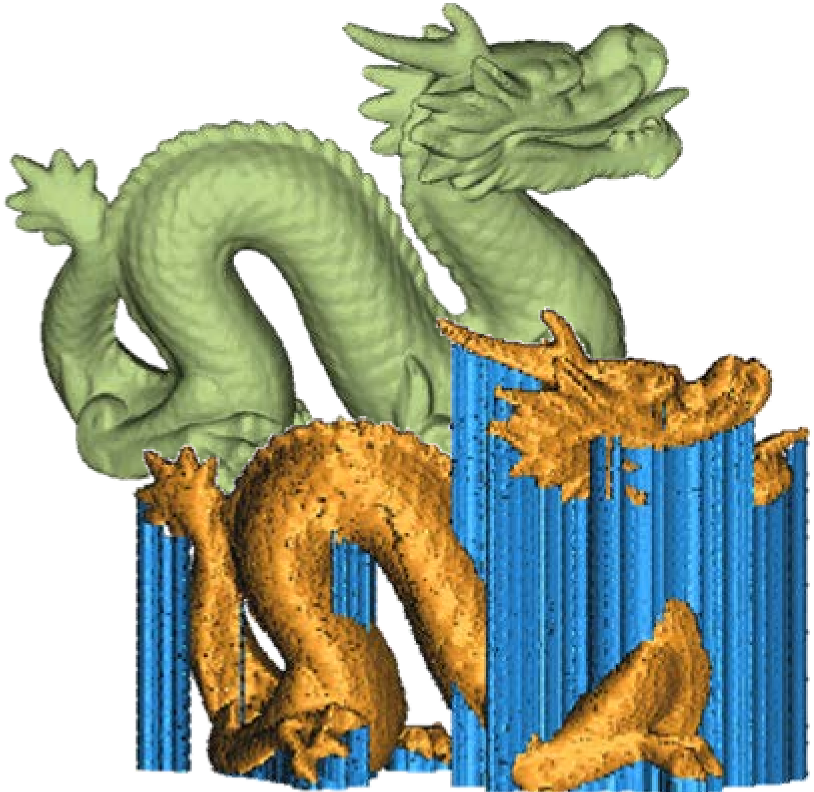
- Without snapping the contours on the edge-sticks, self-intersection happens



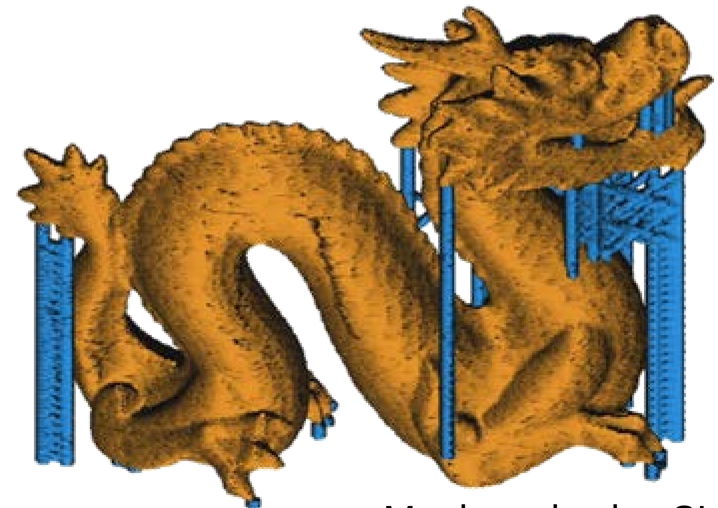
Topological Faithful Contouring Result

Pu Huang, Charlie C.L. Wang, and Yong Chen, "Intersection-free and topologically faithful slicing of implicit solid", ASME Journal of Computing and Information Science in Engineering, vol.13, no.2, 021009 (13 pages), June 2013.

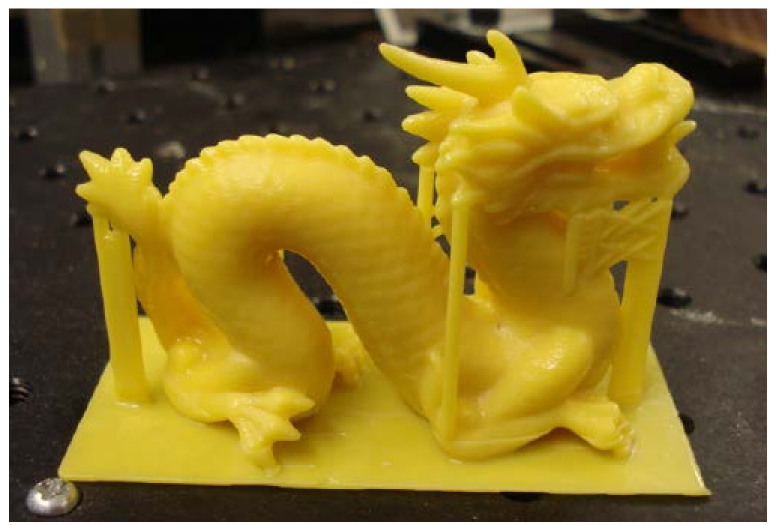
Supporting Structure?



FDM

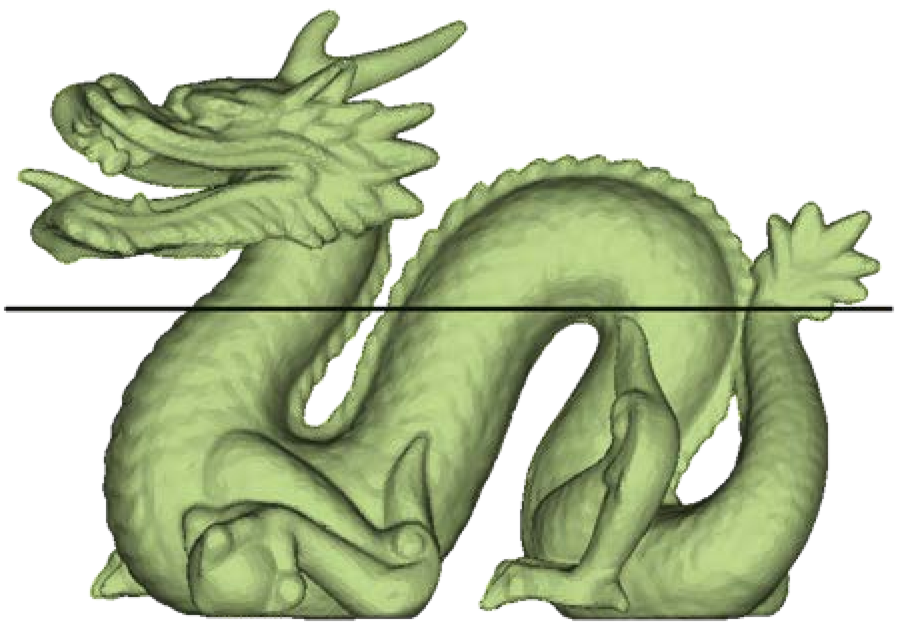


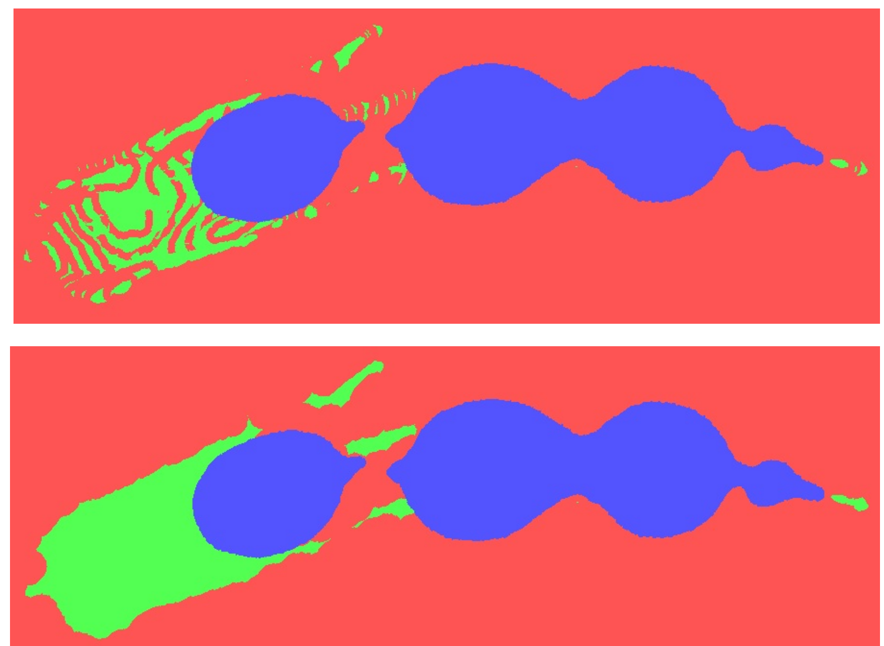
Mask-projection SLA



Algorithms for Generating Supporter

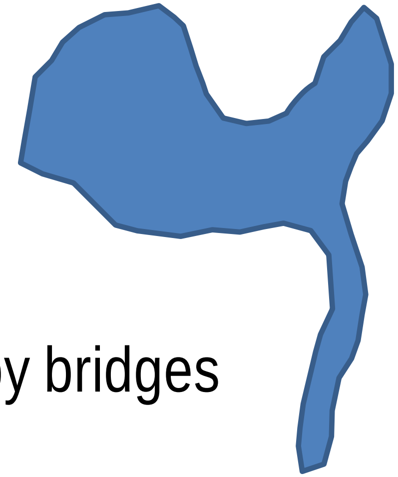
- FDM's supporter is based on *Reliable & Robust Region Subtraction*
- Dilation and erosion must be applied to remove those self-supported regions
- Numerical pruning as a post-processing step is needed

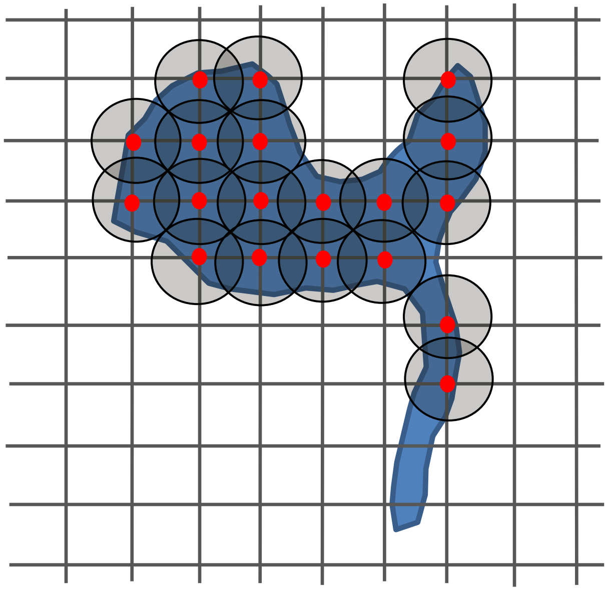


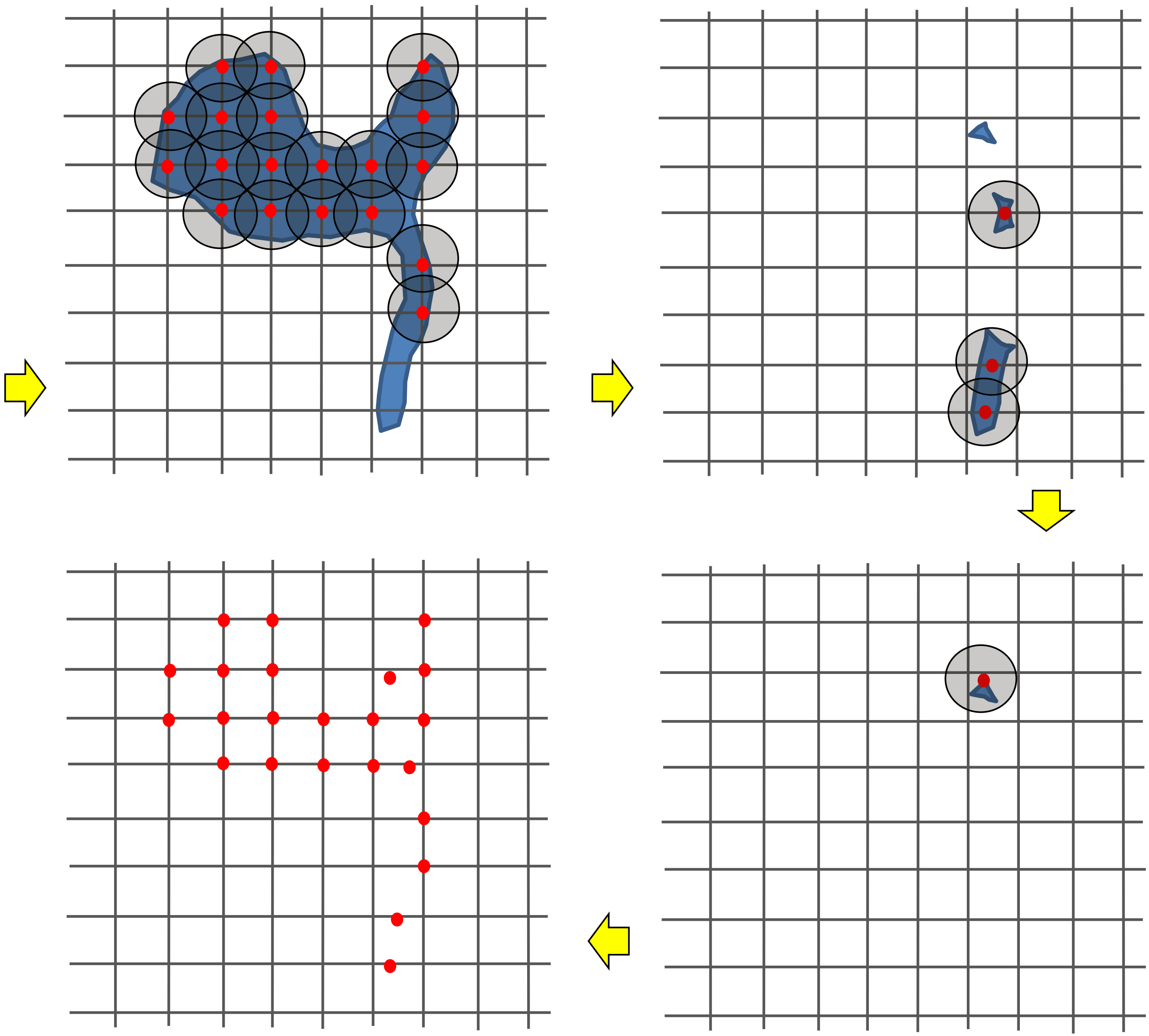


Algorithms for Generating Supporter

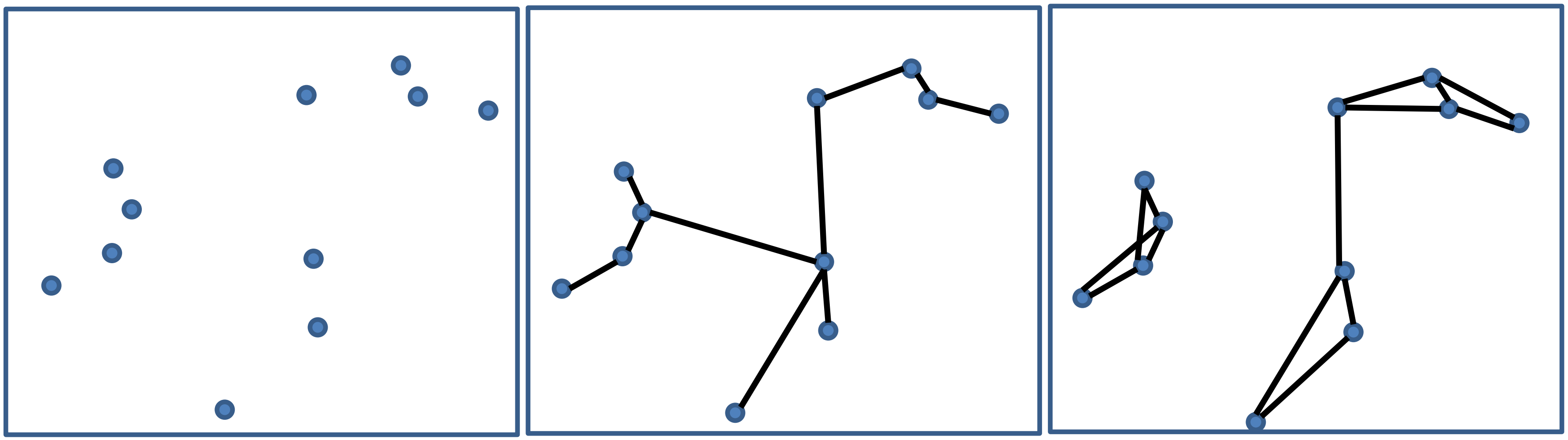
- SLA is based on *Region Subtraction* but using *Anchor Maps*
- Anchor maps are used to represent regions and also take the region subtraction
- Scanning orders:
  - Grid Nodes
  - Grid edges
  - Remaining region
- Linking anchor points by bridges







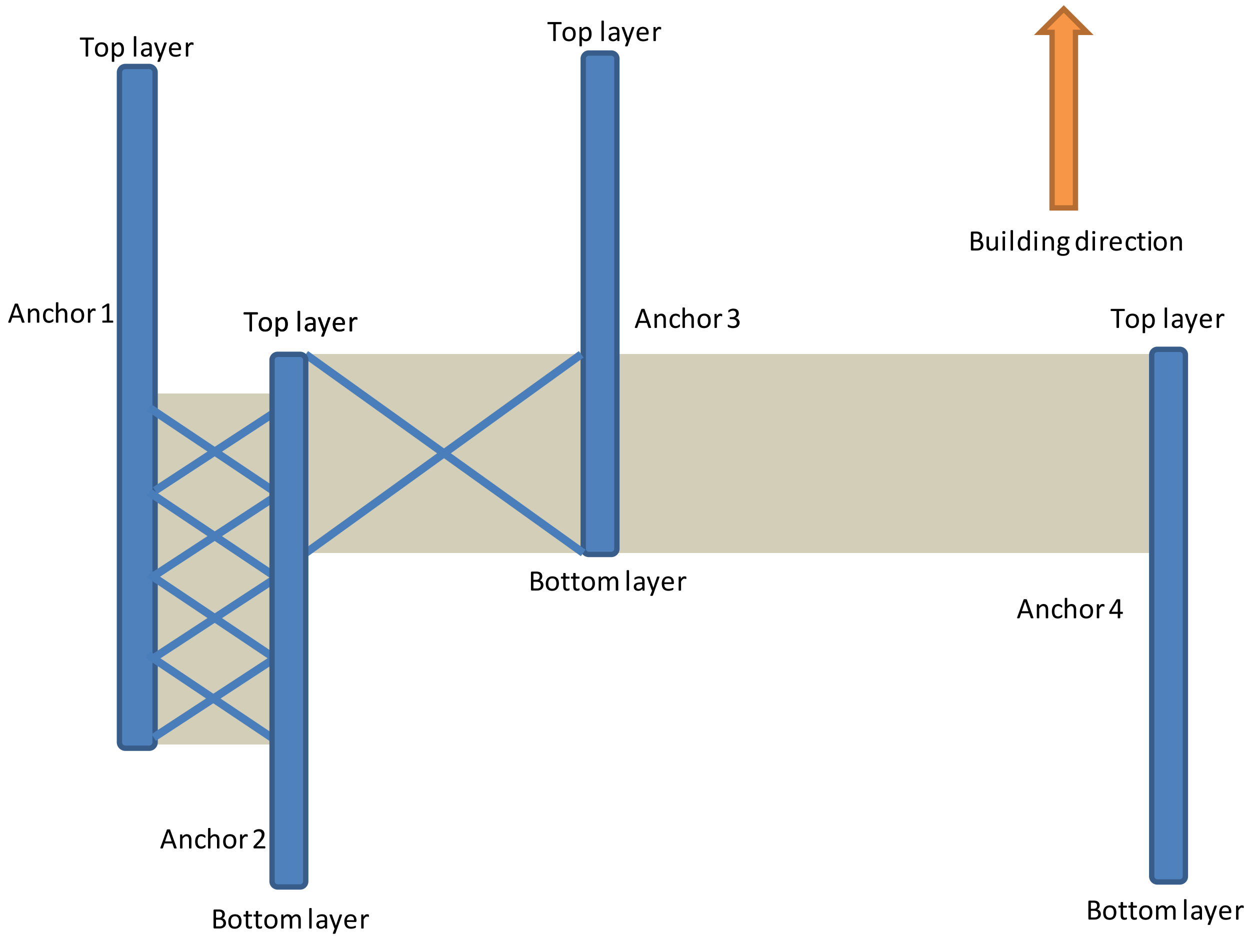
Linking Anchors by Bridges

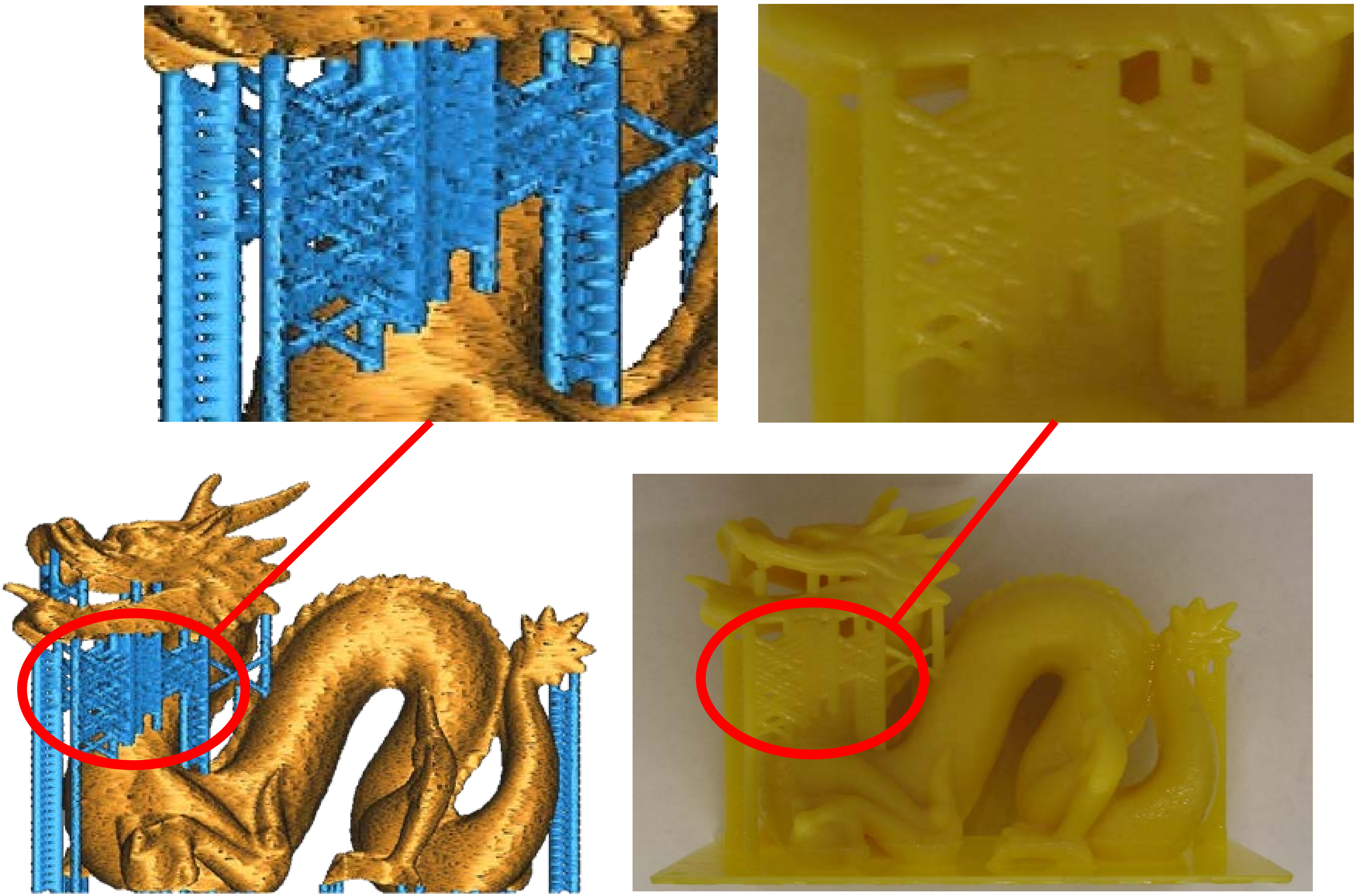


- One approach is based on *Minimal Spanning Tree* (MST)
- Another is based on closest neighbor search
- Which is better? The latter one.
- For building a long bridge, the mechanical stiffness is not good.

Linking Anchors by Bridges (cont.)

- Anchors are located in different heights





Pu Huang, Charlie C.L. Wang, and Yong Chen, "Algorithms for layered manufacturing in image space", Book Chapter, ASME Advances in Computers and Information in Engineering Research, 2014.

Conclusion

- Solid modeling on complex models is very important for additive manufacturing
- Reliable and efficient approaches have been developed in the image space by borrowing the computational power from GPU
- Techniques developed include:
  - Boolean operations
  - Offsetting for hollowing, erosion and dilation, etc.
  - Super-union for meso-structure building
  - Topological faithful contouring
  - Supporter generation