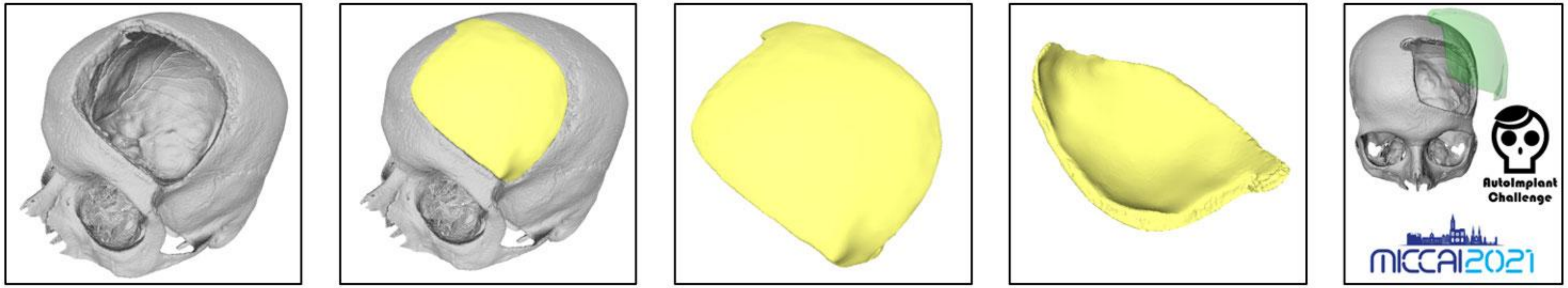


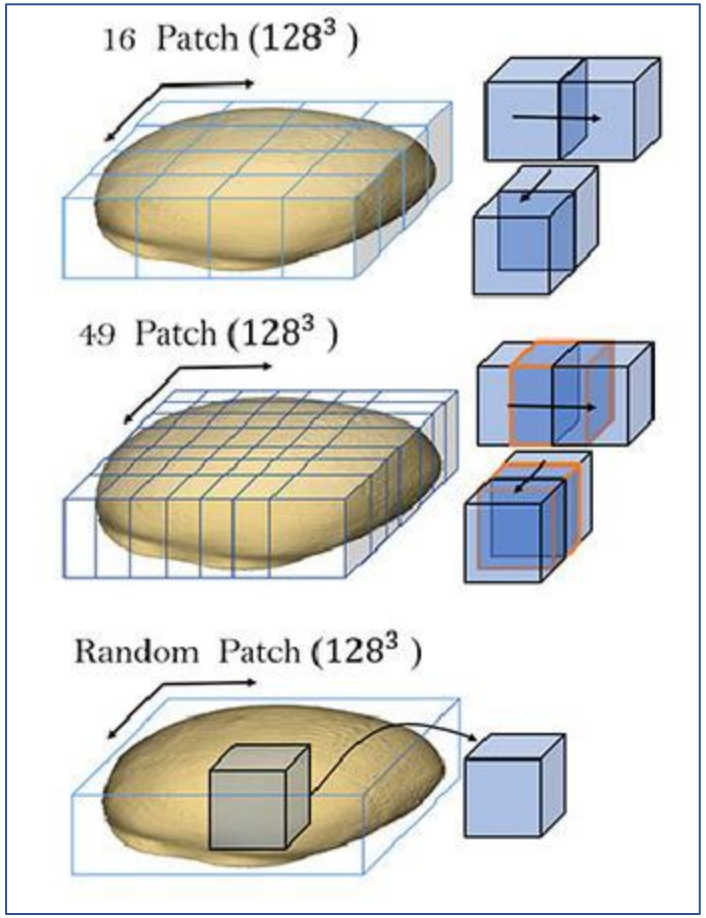
Learning to Rearrange Voxels in Binary Segmentation Masks for Smooth Manifold Triangulation

Jianning Li [1, 2], Antonio Pepe [1], Christina Gsaxner [1],
Yuan Jin[1] , Jan Egger [1, 2]

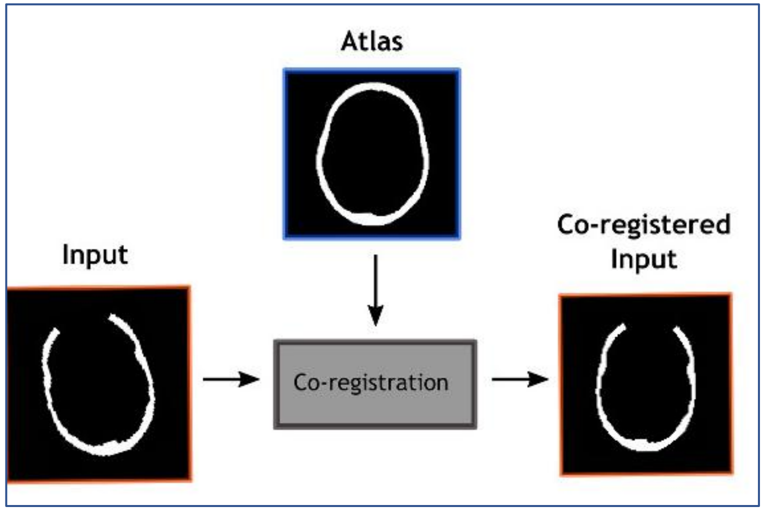
1 TU Graz

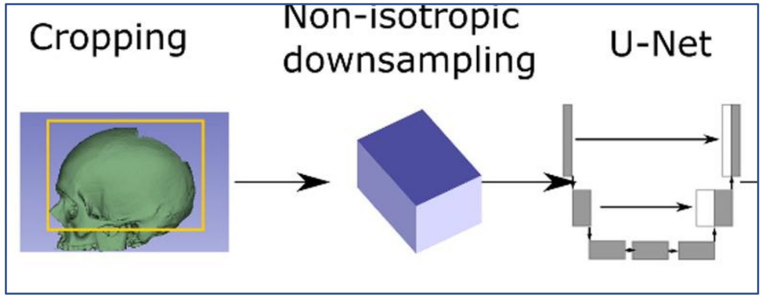
2 Universitätsmedizin Essen
Institut für KI in der Medizin (IKIM)

Problem to address: Neural Nets + Limited (GPU) Capacity + Large Medical Image

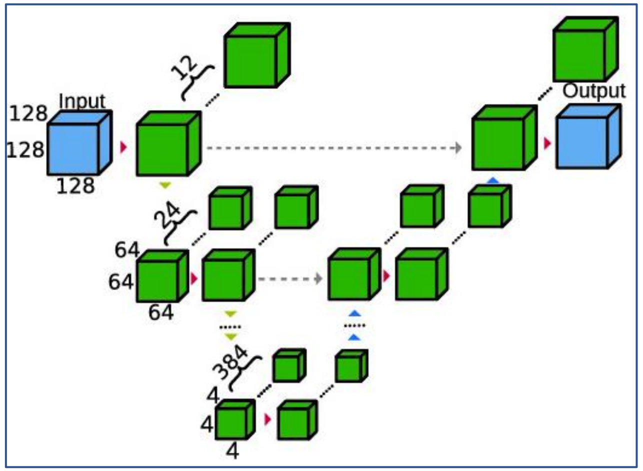


Patch
[Li, J. et al, 2021]





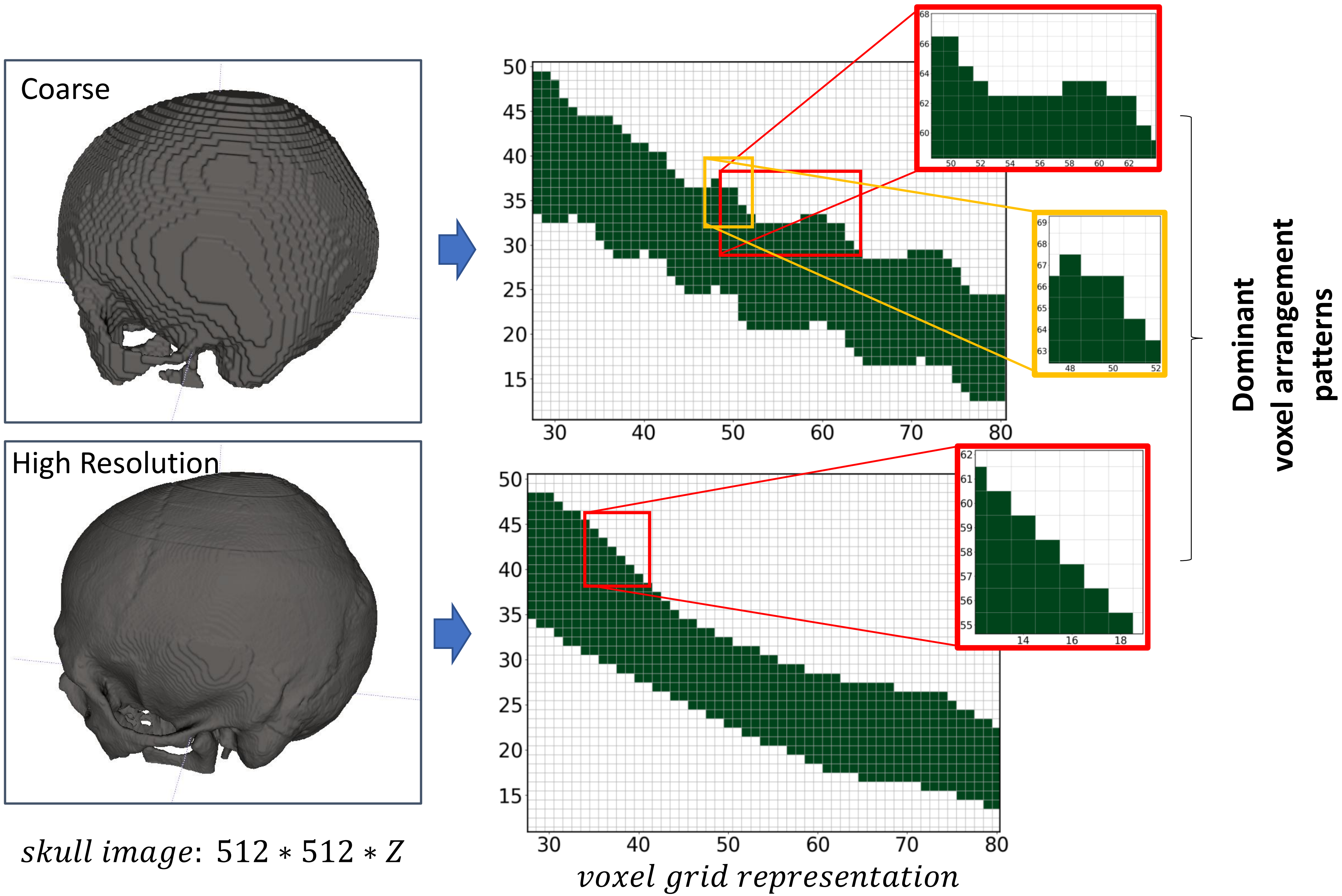
Resample
[Matzkin, F. et al, 2020]
or Downsampling
[Mainprize, J.G, et al, 2020]



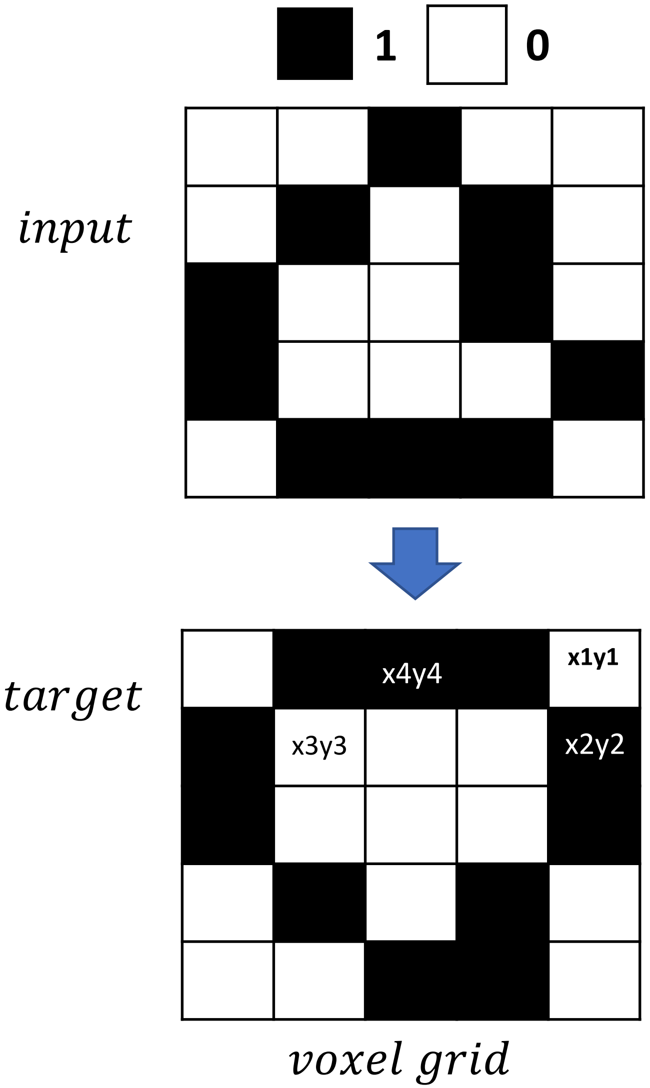
Cascaded CNN
[Kodym, O. et al, 2020]

Sparse CNN
[Artem, K., Li, J. et al, 2021]

Voxel Rearrangement = **High-resolution Output** + Low Memory Consumption (~6GB)



How - Voxel Rearrangement: Conversion between 0s and 1s for binary images

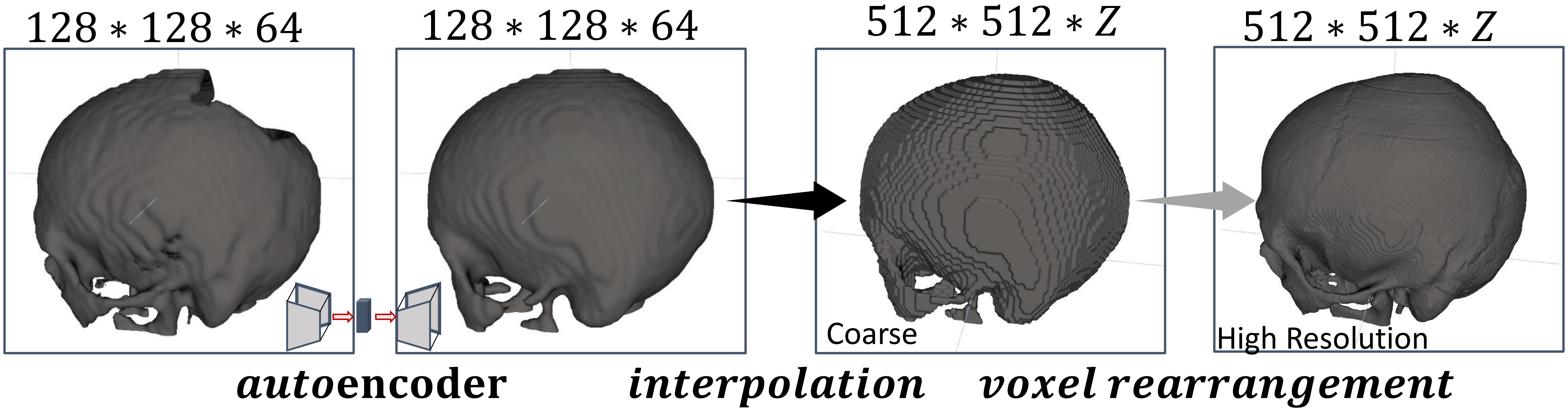


$$\begin{cases} F(0) = 0, x1y1 \\ F(0) = 1, x2y2 \\ F(1) = 0, x3y3 \\ F(1) = 1, x4y4 \end{cases}$$

$$F?$$

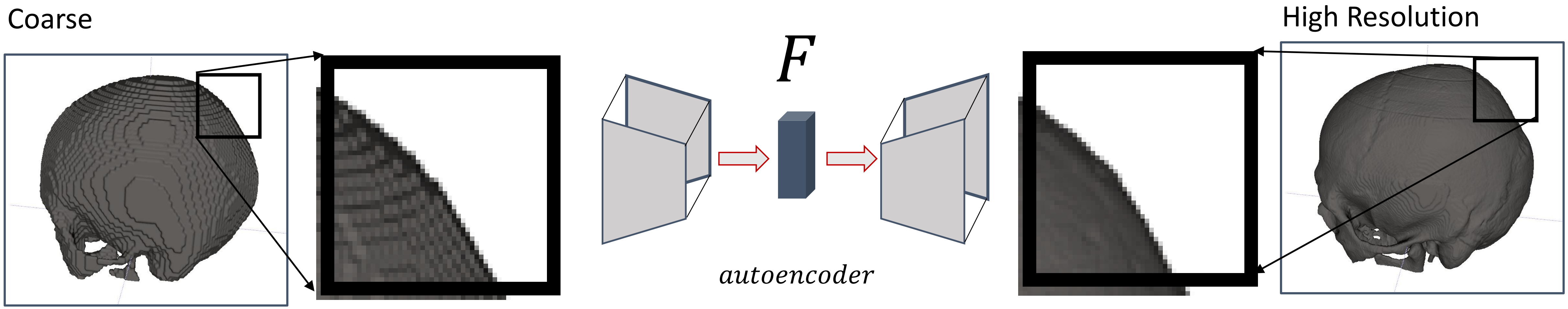
- can be learned: CNN
- updating voxels based on a high-resolution template image

Voxel Rearrangement = High Resolution Output + **Low Memory Consumption (~6GB)**



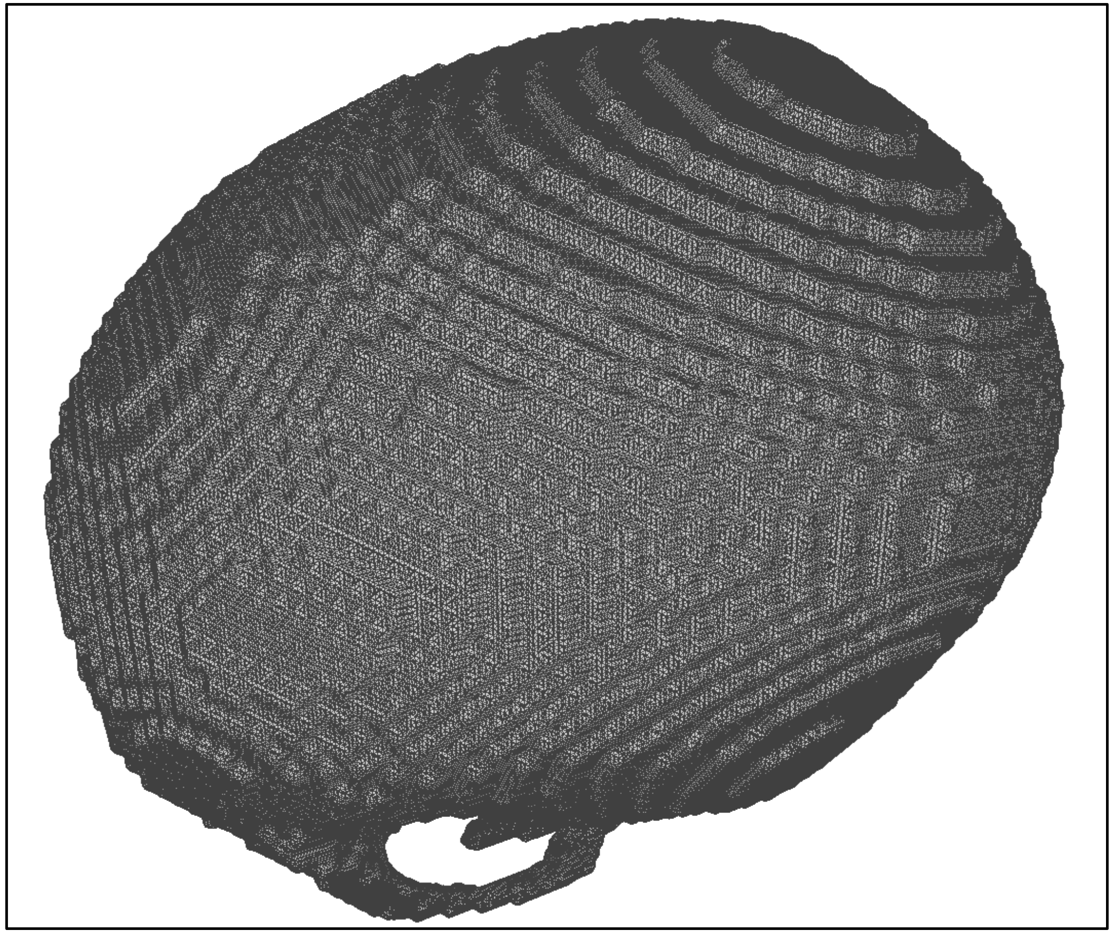
- A **coarse-to-fine (C2F)** Framework: the memory consumption is equivalent to processing low-resolution ($128 * 128 * 64$) images.
- High-resolution outcome can be obtained from the coarse output, using only roughly 6GB GPU memory.

Learned Voxel Rearrangement



Learned Voxel Rearrangement

Resulting Skull Voxel Grid:

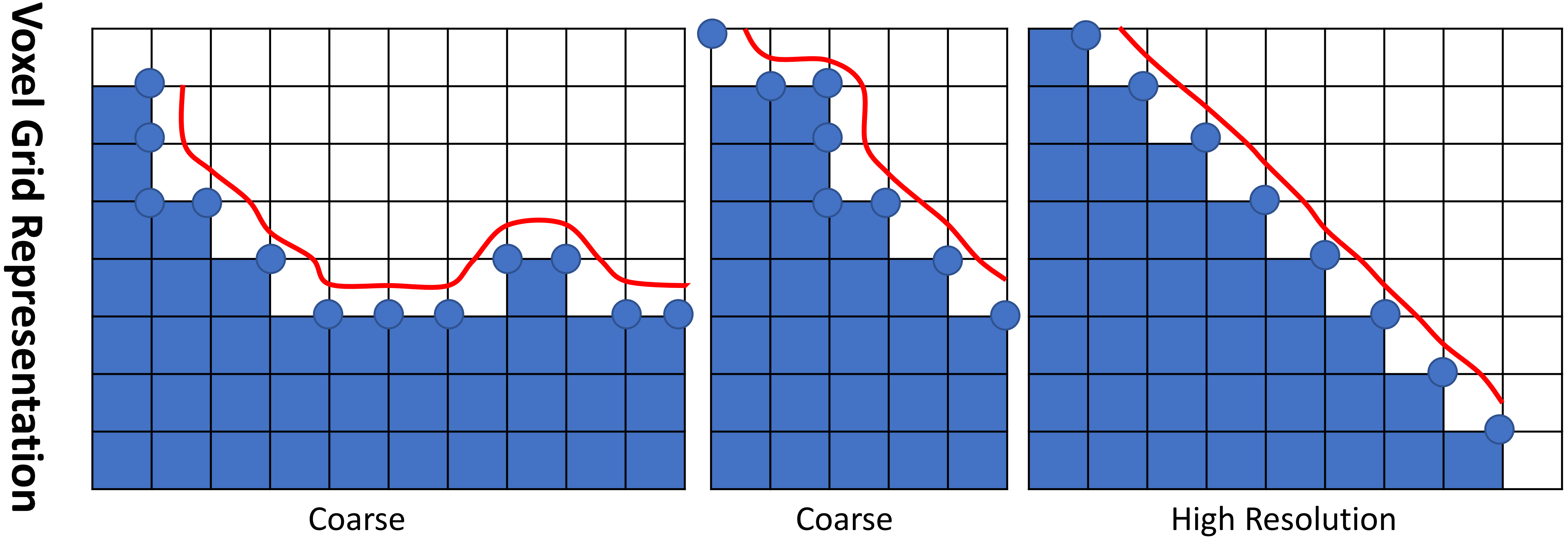


$(\boldsymbol{a})$ Coarse skull

$(\boldsymbol{b})$ Reconstructed skull through voxel rearrangement of the coarse skull

Learned Voxel Rearrangement

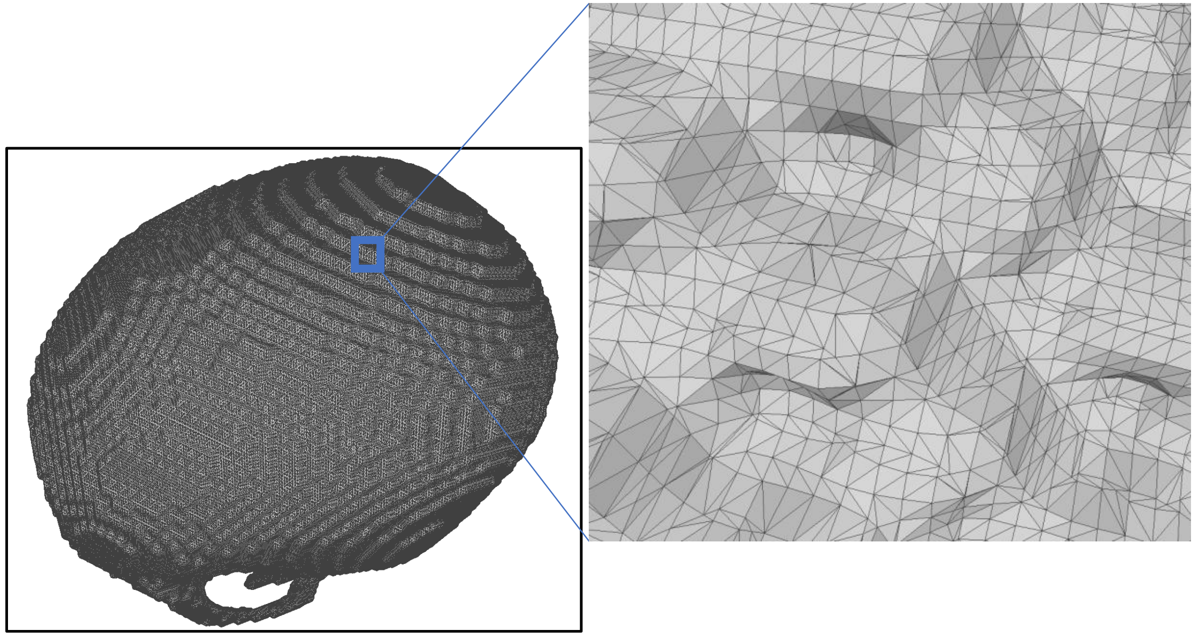
3D printing - Voxel grid to Mesh: Marching Cube



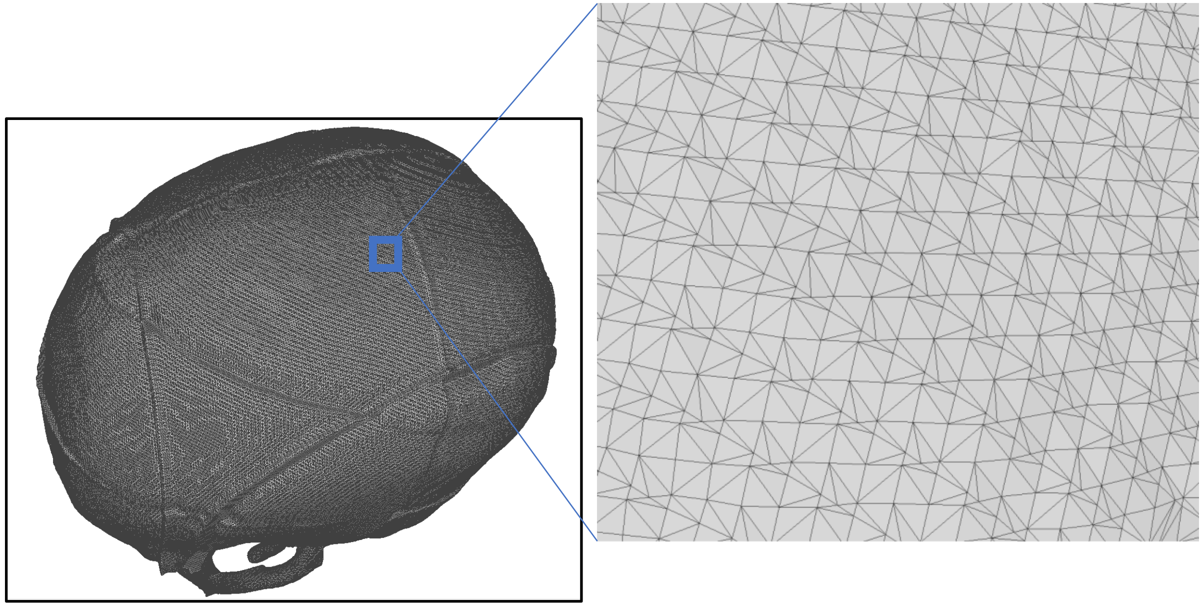
Red line: extracted isosurface from the corresponding voxel grid

Learned Voxel Rearrangement

Resulting Skull Triangular Mesh:



(a) Coarse skull



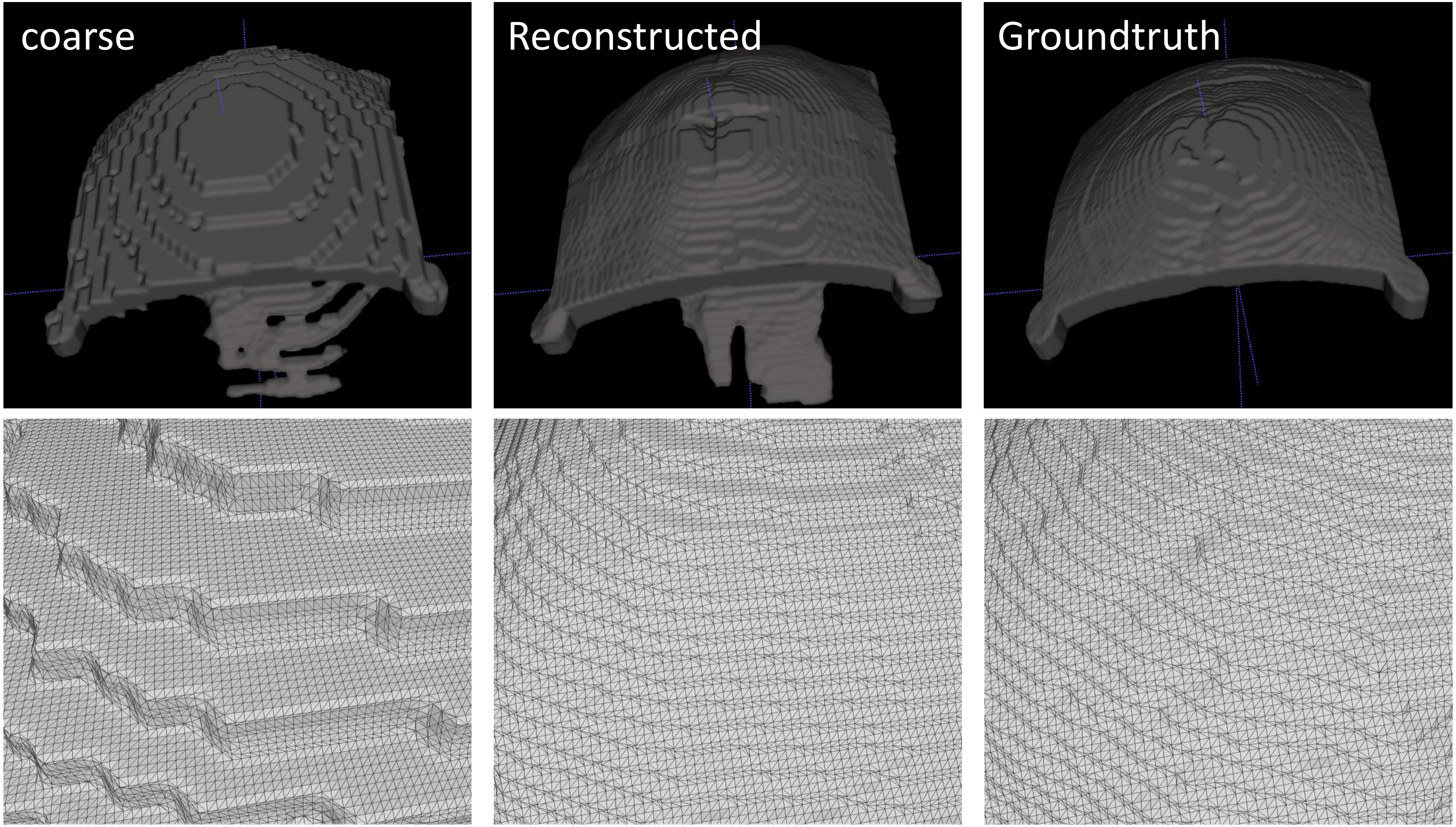
(b) Reconstructed skull through voxel rearrangement of the coarse skull

Learned Voxel Rearrangement

3D Printing of Cranial Implant

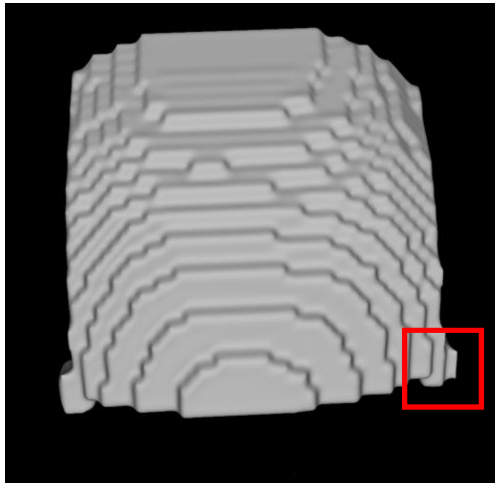
Voxel grid

Mesh



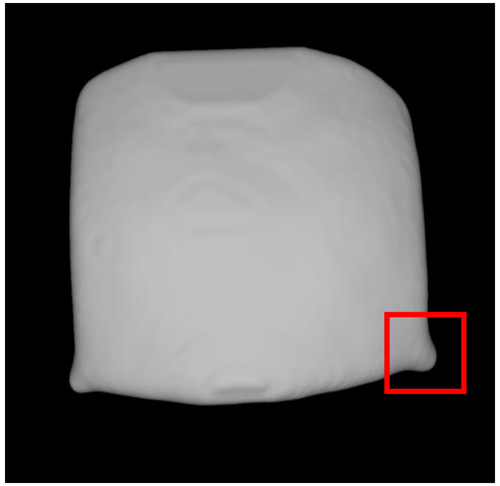
Learned Voxel Rearrangement

Smoothing Filters

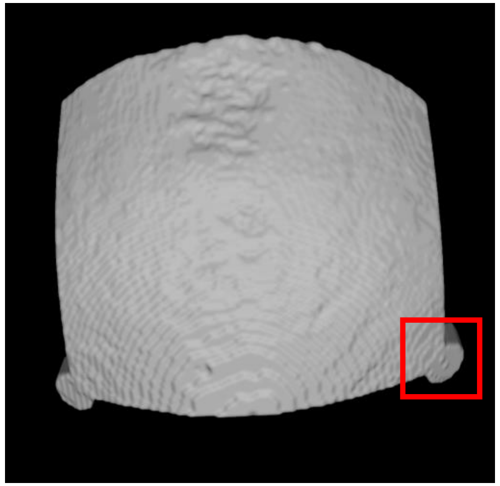


coarse

Median filter:
15*15*11 kernel



Gaussian filter:3mm std



Ground truth

Smoothing Filters:

- Non-detail preserving
- Substantially erasing voxels non-discriminatively

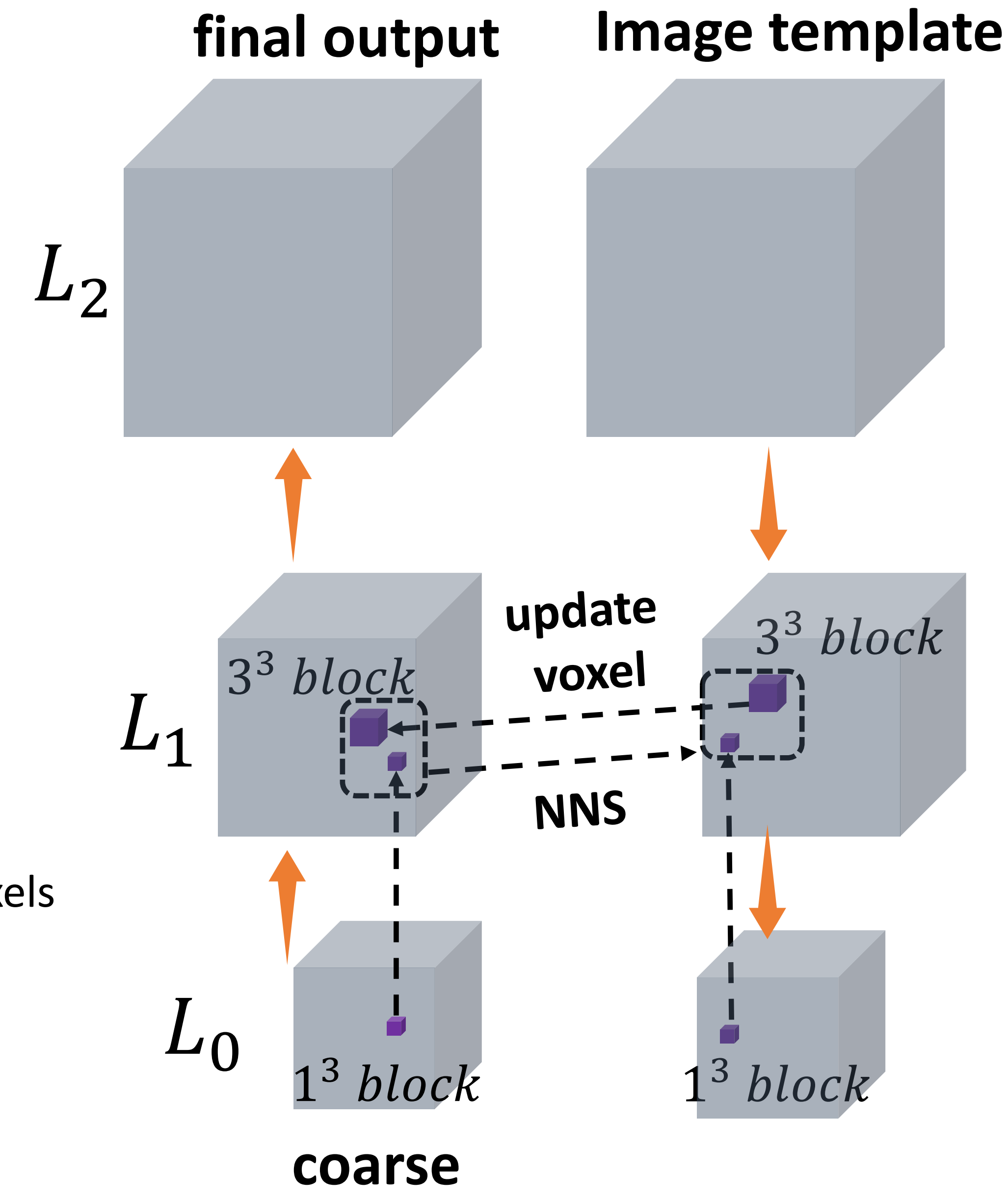
Voxel Updating based on a High-resolution Template Skull

Workflow:

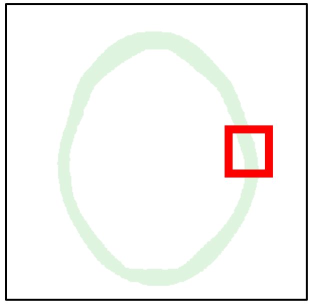
- Given a coarse input, preselect a high-resolution skull image as a template
- Create an (three-level) image pyramid for the coarse and template image
- Starting from the second level (L_1), update each voxel (0,1 conversion) based on its closest voxel in the template image (NNS)
- Repeat until all voxels in L_1 and L_2 are updated

Challenges:

- too many voxels to update: L_2 alone has over 60 M voxels
- linear search too slow for 60 M x 60 M searches
- memory-consuming



Voxel Updating based on a High-resolution Template Skull



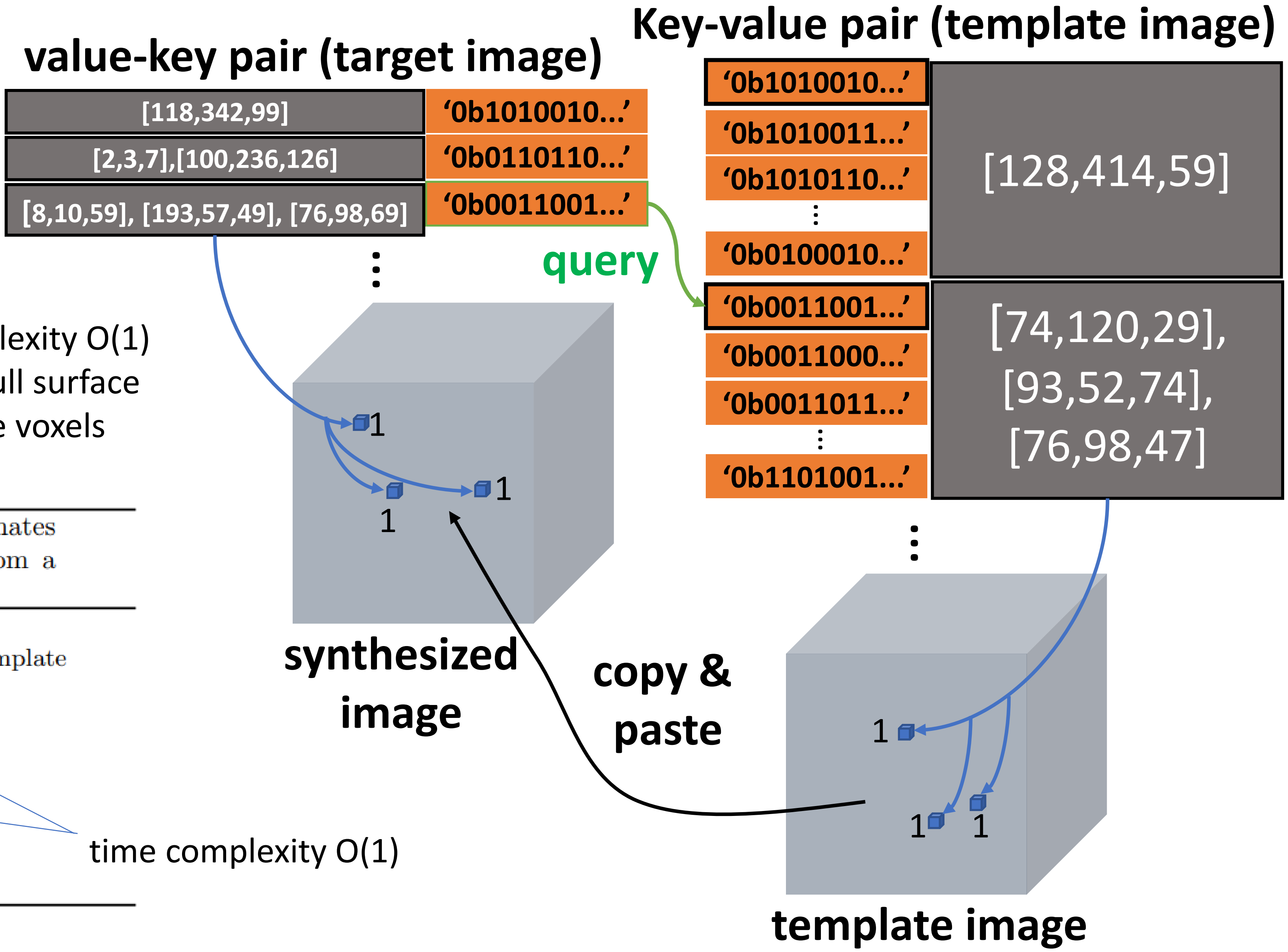
Solutions:

- Hash table-based NNS: time complexity O(1)
- Sparsity: only update voxels on skull surface
- Binariness: using bit-string to store voxels

Algorithm 1: Retrieving the coordinates corresponding to an entry (bit string) from a hash table

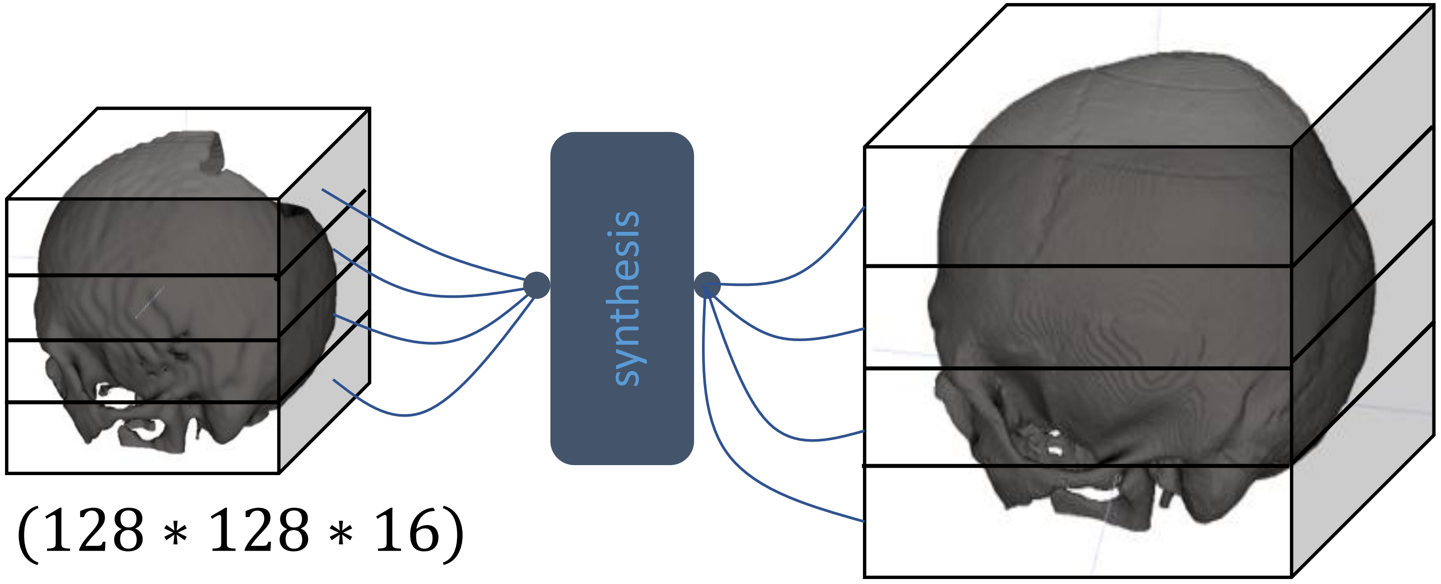
Input: a *key* K_c from the coarse pyramid ;
Output: coordinate(s) (x, y, z) from the template pyramid ;
if K_c in S_{ta} **then**
  coordinates=S_{ta}.get_value(K_c) ;
else if K_c in S_{tn} **then**
  coordinates=S_{tn}.get_value(K_c) ;
else
  assign 0 or 1 to the voxels ;

time complexity O(1)



Voxel Updating based on a High-resolution Template Skull

Further acceleration: Data Parallelism by Multi-core CPU



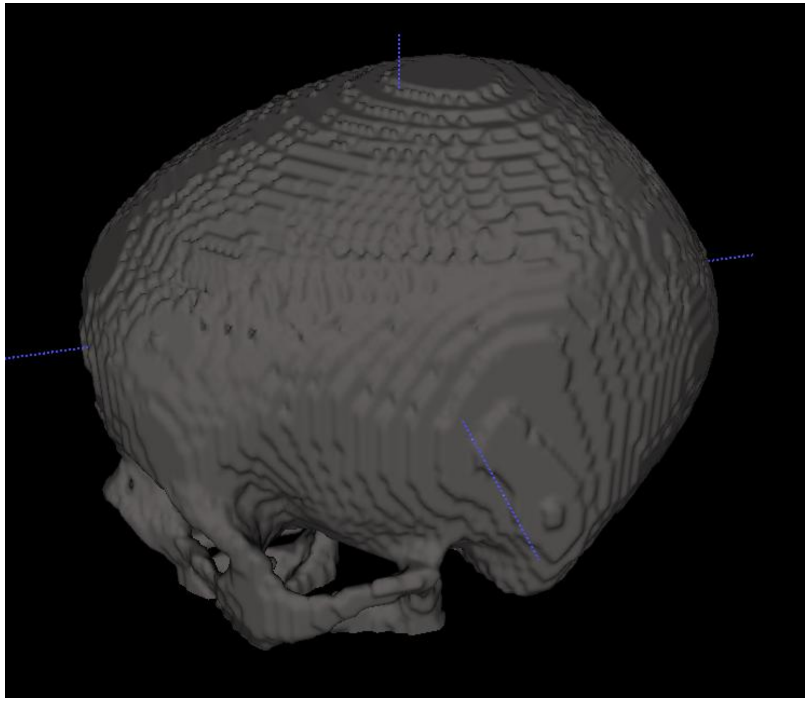
$4 * (128 * 128 * 16)$

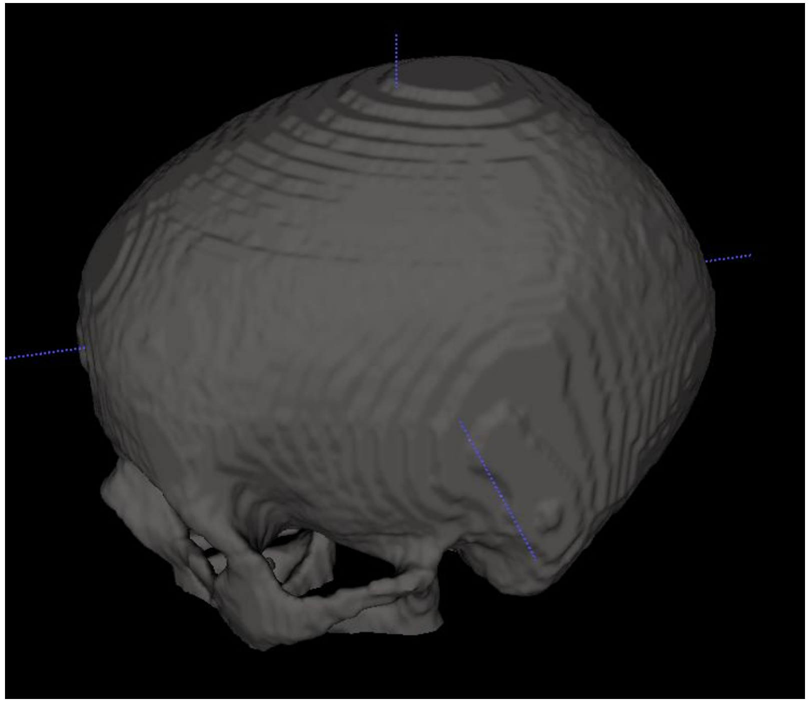
$4 * (512 * 512 * 64)$

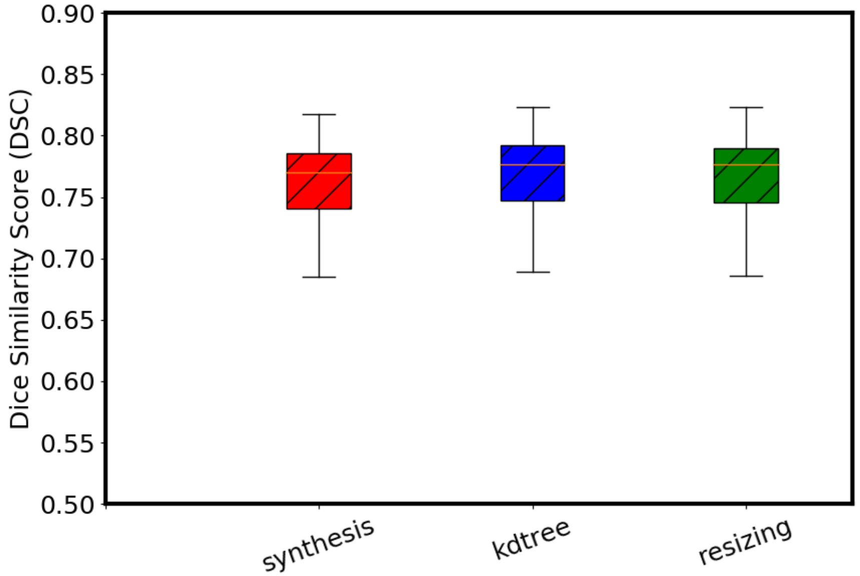
- Divide the voxel grid to several smaller sections (number depends on the number of cores, e.g., 4)
- NNS is performs only the the corresponding sections
- Combining the resulting sections yields the final skull

Voxel Updating based on a High-resolution Template Skull

Hash table vs **KD-tree**





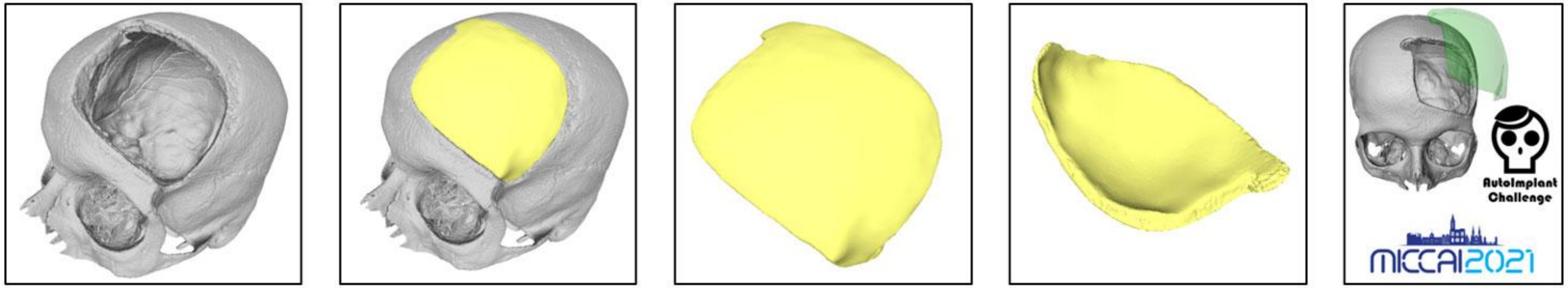


- KD-tree as another alternative data structure for fast NNS, besides hash table
- KD-tree requires reduction of feature dimension (from 27 to 20, using PCA) for fast search
- KD-tree yields better results but comsumes more memory than the hash table based search

Conclusion

- The difference between high-resolution and coarse (skull) voxel grids is their voxel arrangement.
- By exploiting the spatial sparsity and binariness of the skull images, the reconstruction time and memory consumptation can be effectively reduced.

Dataset: https://autoimplant2021.grand-challenge.org/
Codes: https://github.com/Jianningli/voxel_rearrangement



Learning to Rearrange Voxels in Binary Segmentation Masks for Smooth Manifold Triangulation

Jianning Li [1,2], Antonio Pepe [1], Christina Gsaxner [1],
Yuan Jin[1] , Jan Egger [1,2]

1 TU Graz

2 Universitätsmedizin Essen
Institut für KI in der Medizin (IKIM)