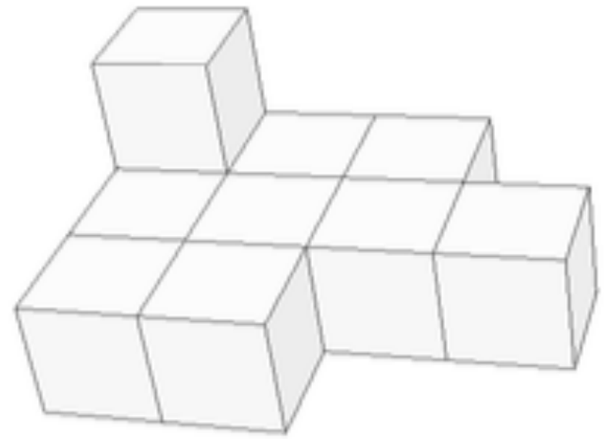
Poly-cube mapping

What are Poly-cubes?

[Tarini et al. 2004]

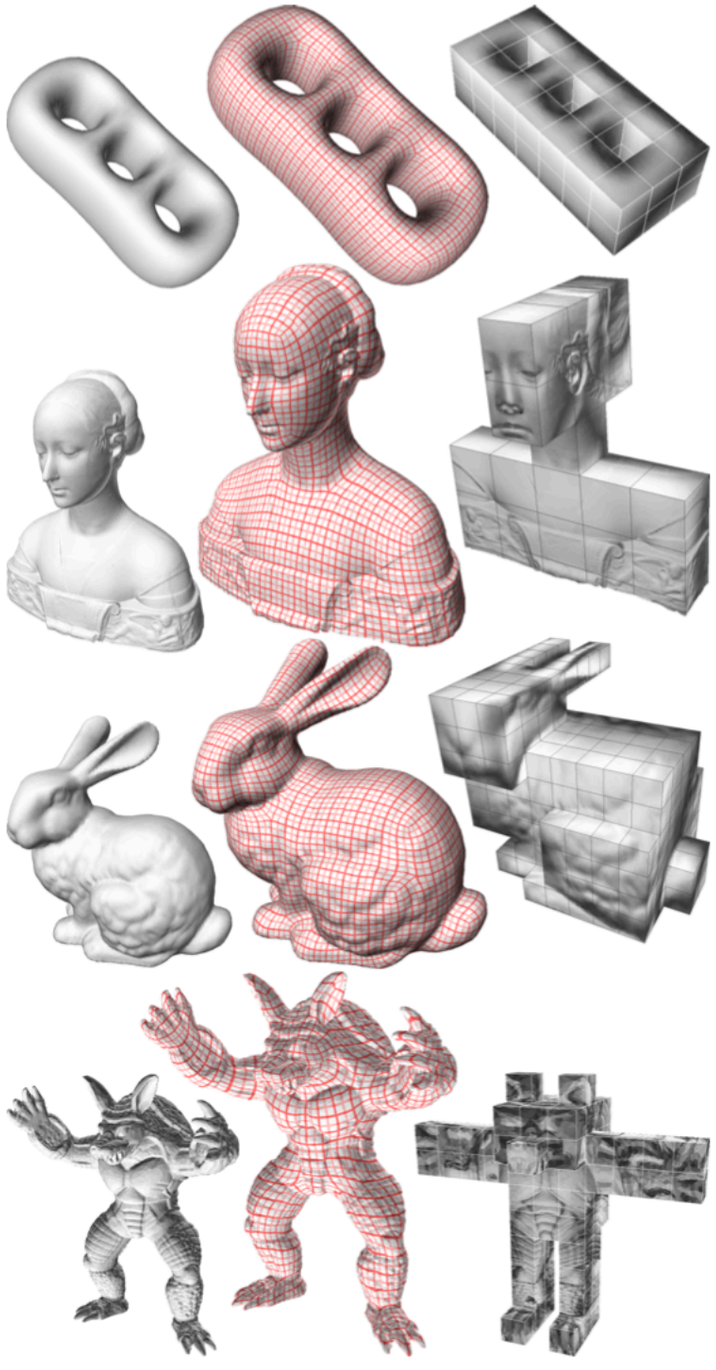
A poly-cube is a 3d shape composed of axis aligned unit cubes that are attached face to face



Poly-cube maps

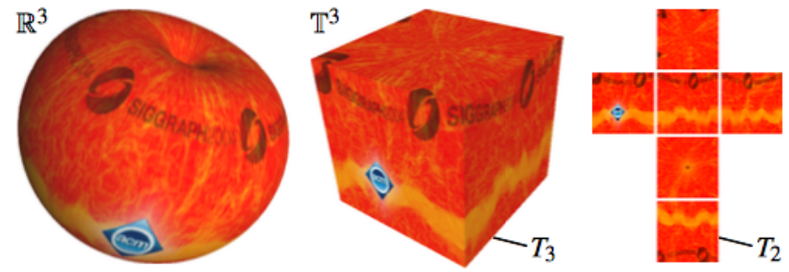
A poly-cube parameterization is a bijective map between a 3D mesh model *M* and a poly-cube domain *P* such that:

- *P* shares the same topology with *M*
- *P* approximates the geometry of *M*
- *P* is simple (i.e. not too many corners)

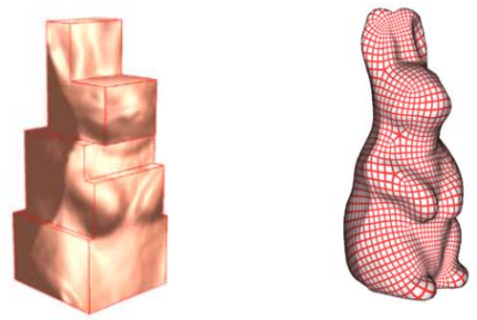


Applications

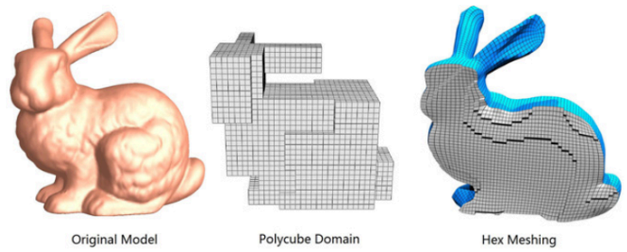
- Efficient texturing [Tarini et al. 2004]



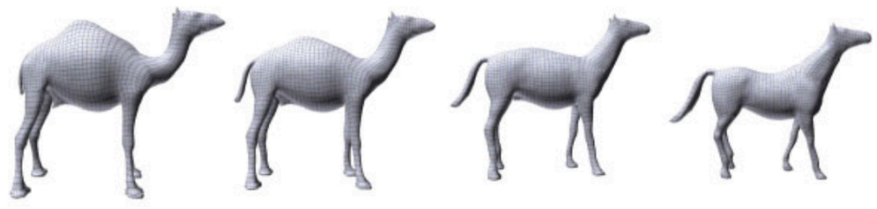
- Quad-remeshing (surface) [He et al. 2009]



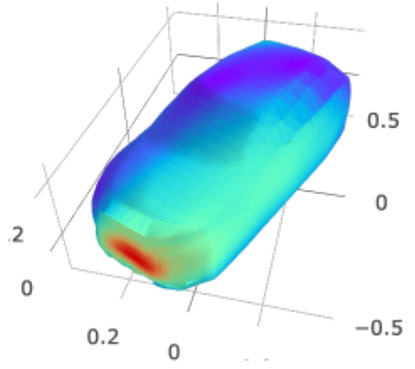
- Hex-meshing (volume) [Yu et al. 2014]



- Shape interpolation [Fan et al. 2005]



- Deep learning on surface meshes?

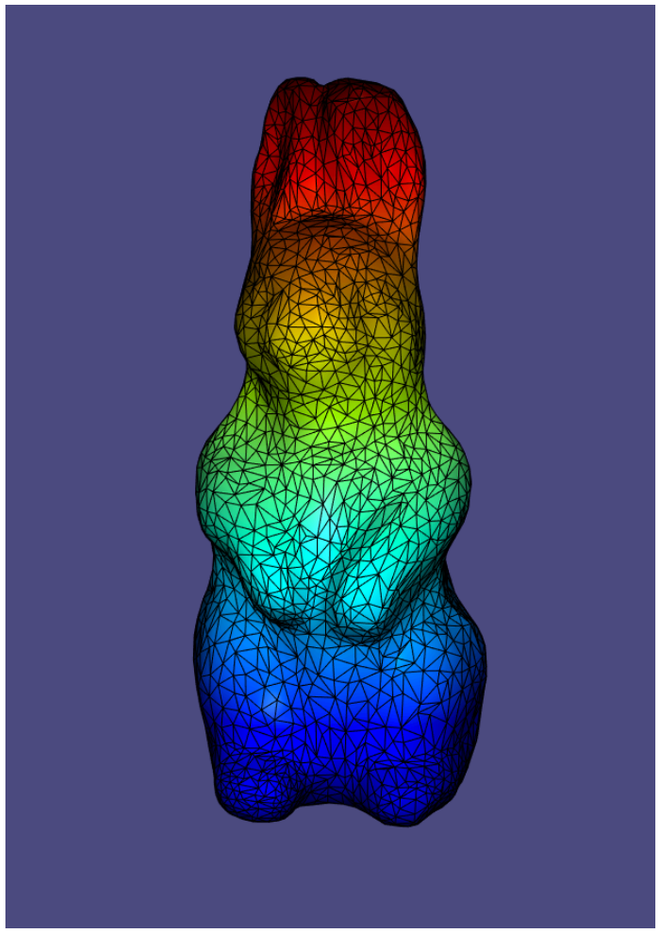


State of the art timeline

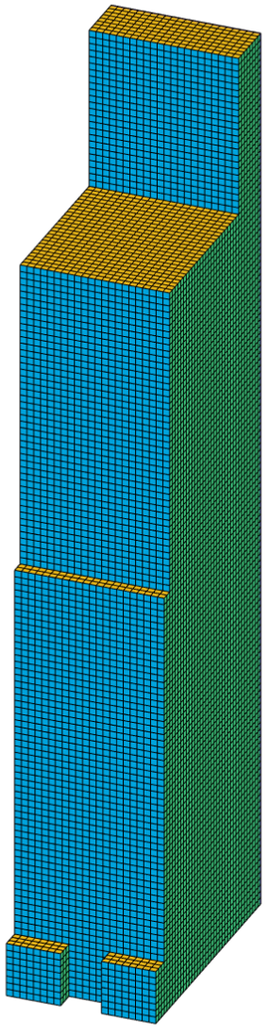
- Manual poly-cube mapping [Tarini et al. 2004]
- …
- Semi-automatic poly-cube mapping [Xia et al. 2010]
- …
- Automatic poly-cube mapping for simple geometries [Gregson et al. 2011]
- ….
- Automatic poly-cube mapping [Yu et al. 2014]

Algorithm produces valid over-complicated poly-cube via voxelization and simplifies it via heuristic local optimization

Robust poly-cube mapping algorithm

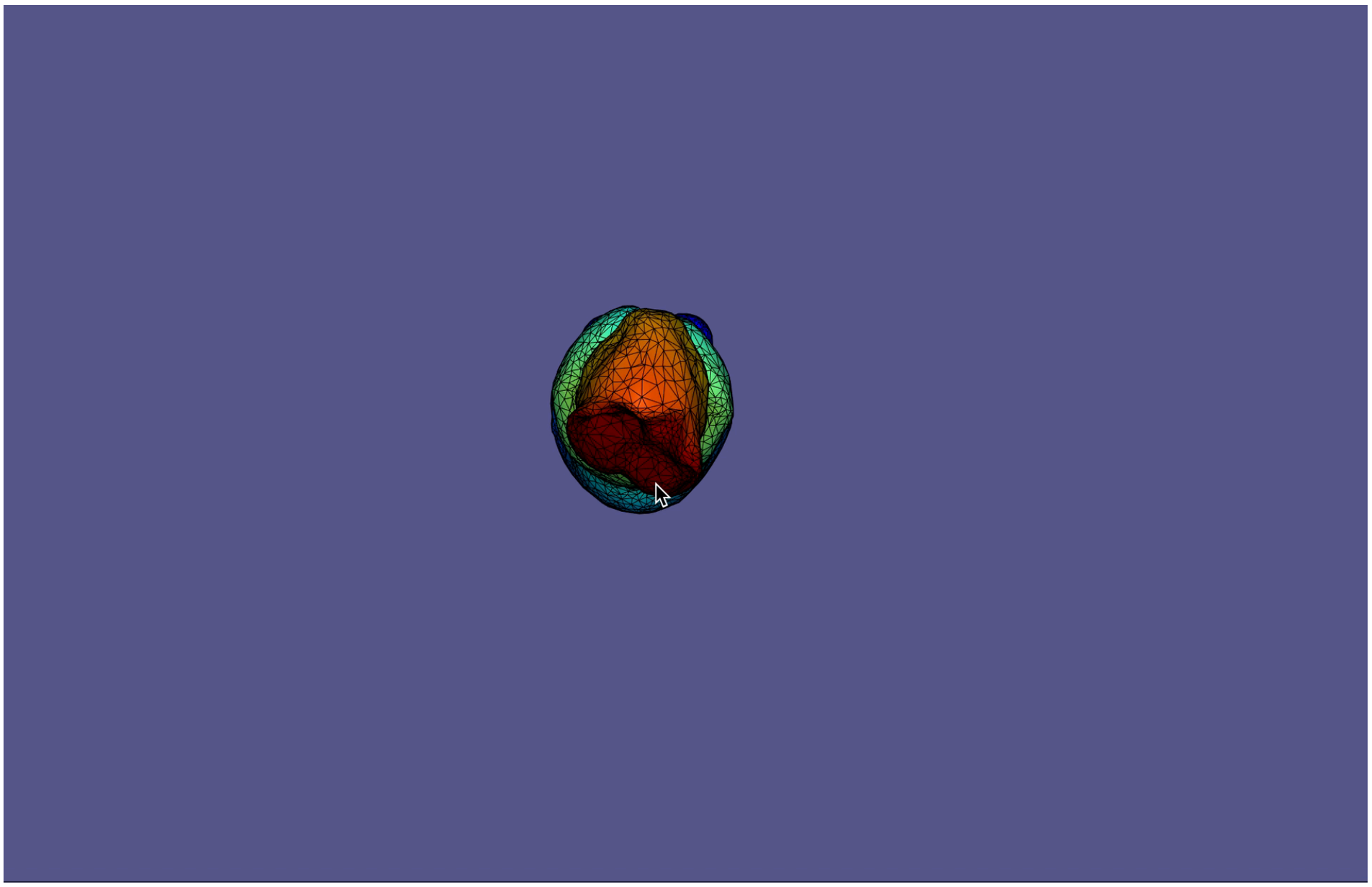


- Axis-aligning mesh deformation
- Full volumetric Voxelization
- Cube extraction via volumetric Hough Transform



1: Axis aligning deformation

Iteratively deform a geometric model into a geometrically regular axis aligned shape



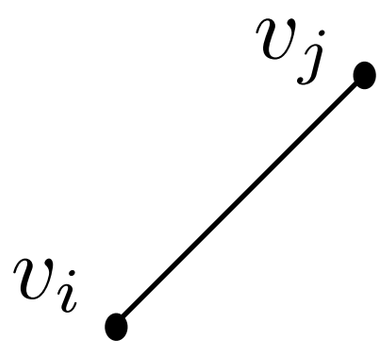
Details…

(1) while not converged:

  (1) Cluster the surface mesh into 6 different regions (±X, ±Y, ±Z) based on vertex normal orientation.

  (2) For every vertex compute the minimal rotation R aligning normal to cluster axis

  (3) Deform surface so that clustered normals align to corresponding nearest axes. Work on edges to balance alignment constraints with volumetric distortion.



for all edges

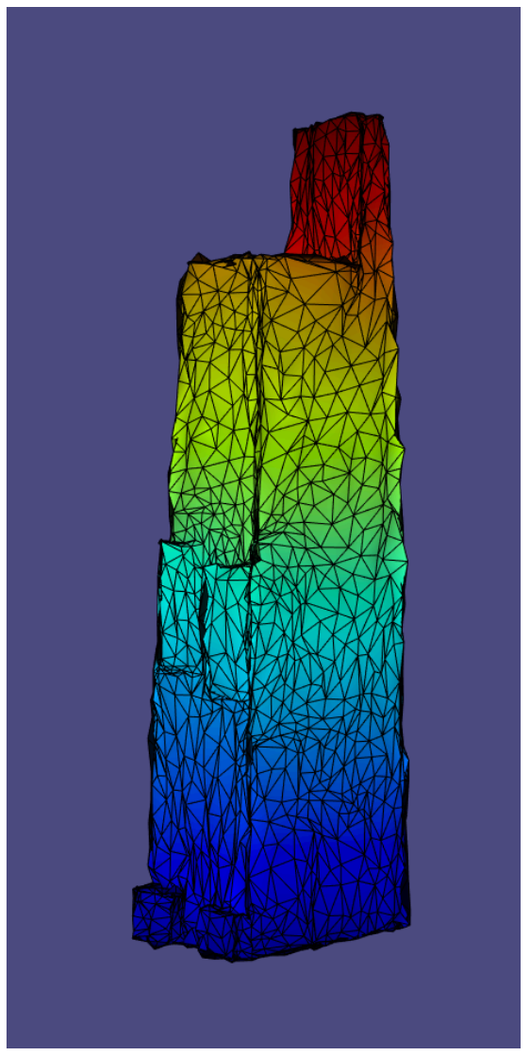
Poisson equation over each vertex

$$v_i - \frac{1}{N_i}\sum v_j = \frac{1}{N_i}\sum \frac{R_i + R_j}{2}(\tilde{v}_i - \tilde{v}_j)$$

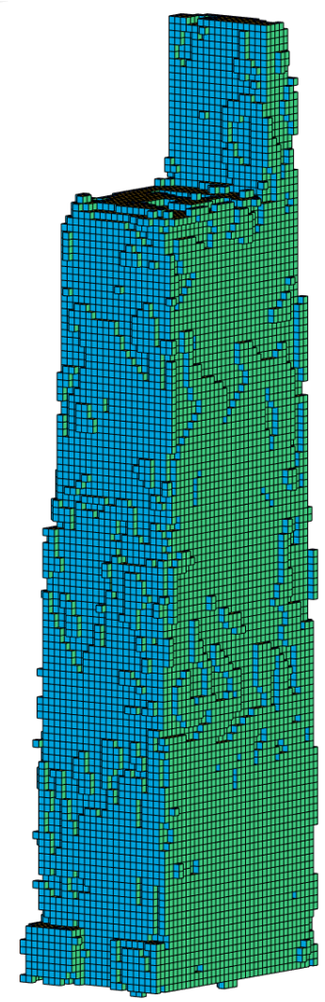
$$v_i - v_j = \frac{1}{2}(R_i + R_j)(\tilde{v}_i - \tilde{v}_j)$$

2: Volumetric Voxelization

Voxelize the axis aligned model by sampling the model signed distance function on a regular grid defined over the model's bounding box

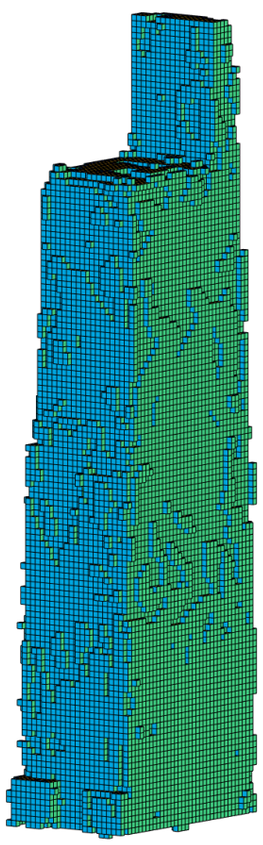


voxelization

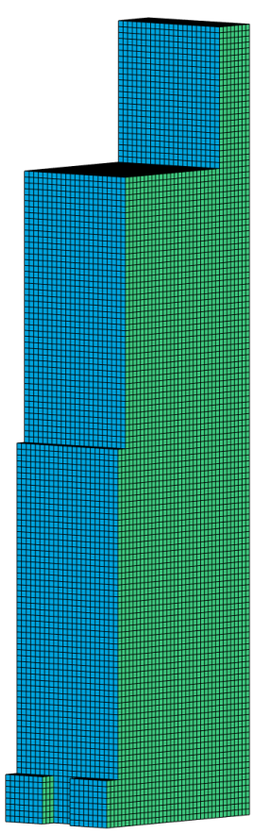


3: Corner detecting 3D Hough Transform

Simplifiy poly-cube structure by applying a corner detecting 3d Hough transform



hough transform



Details…

- Create the accumulator space, which is made up of a cell for each voxel, and set each cell to zero.
- voting: for each boundary voxel (i,j,k) increment by 1 all cells that belong to the axis aligned planes centered in voxel (i,j,k)
- Search for local maxima in the accumulator space, these voxels represent the corners in the binary volume detected by the algorithm
- Find poly cubes in the domain by finding matching corners

Input binary image

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

voting

0	1	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

loop…

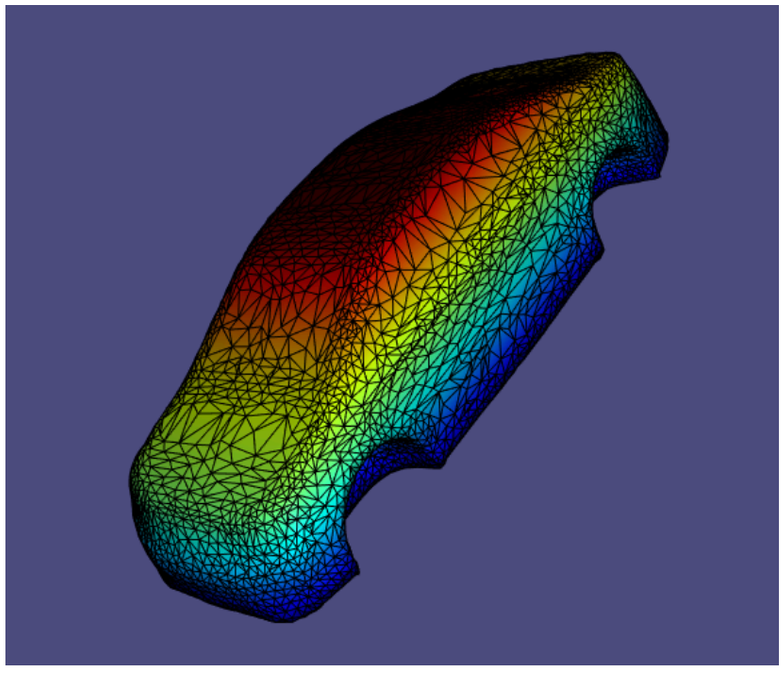
0	7	2	2	5	2	2	4	0	0
4	11	6	6	9	6	6	8	4	4
2	9	4	4	7	4	4	6	2	2
2	9	4	4	7	4	4	6	2	2
5	12	7	7	10	7	7	9	5	5
2	9	4	4	7	4	4	6	2	2
2	9	4	4	7	4	4	6	2	2
7	14	9	9	12	9	9	11	7	7
0	7	2	2	5	2	2	4	0	0
0	7	2	2	5	2	2	4	0	0

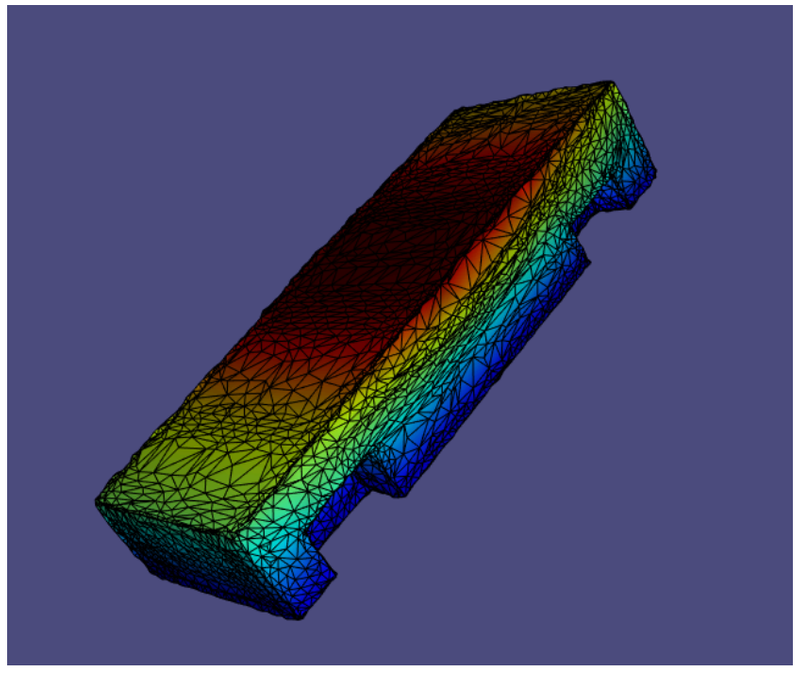
find admissible matching corners

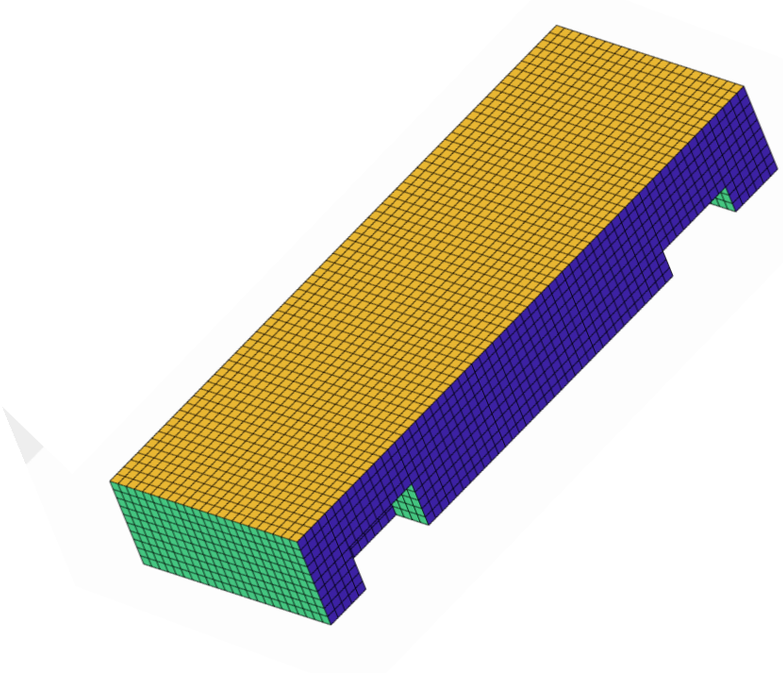
Poly-square representation

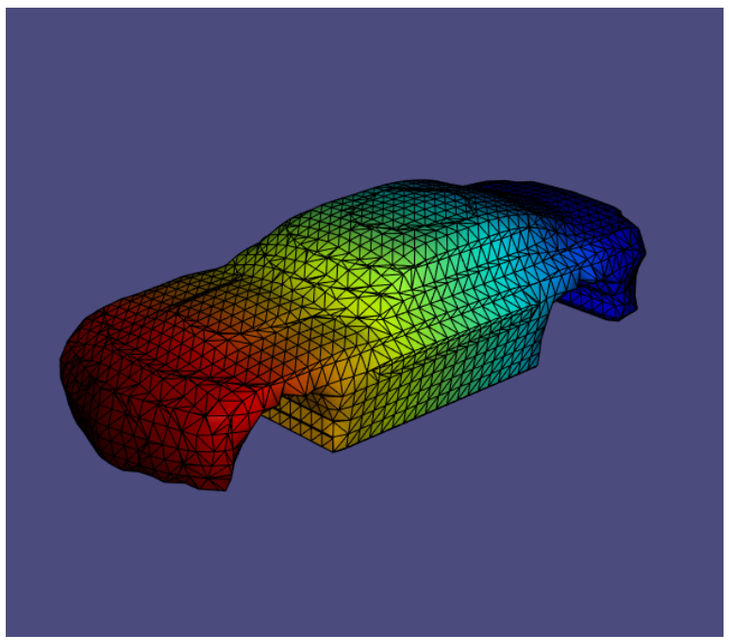
0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

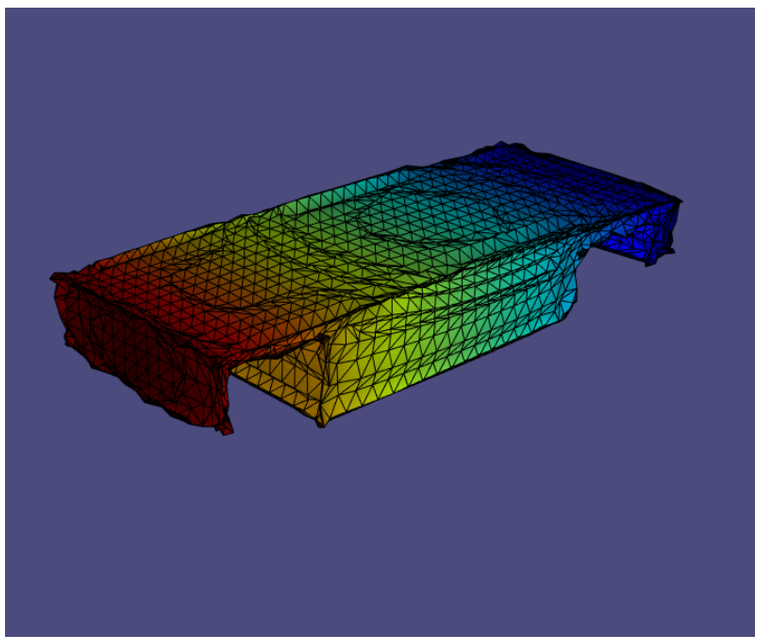
Results: Cars dataset

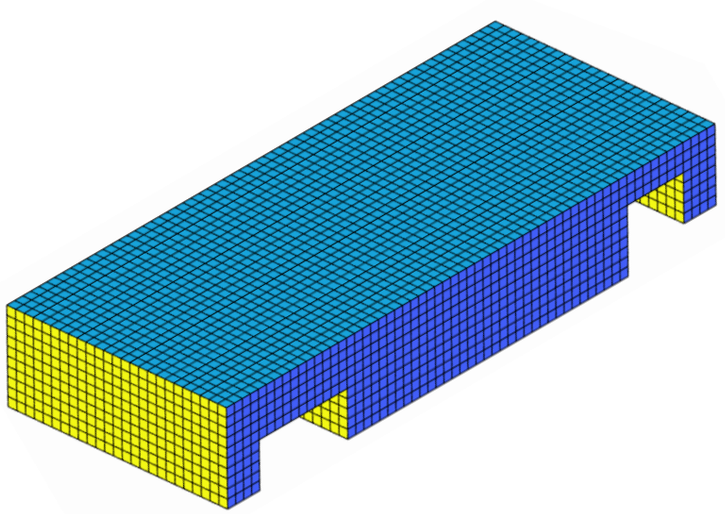




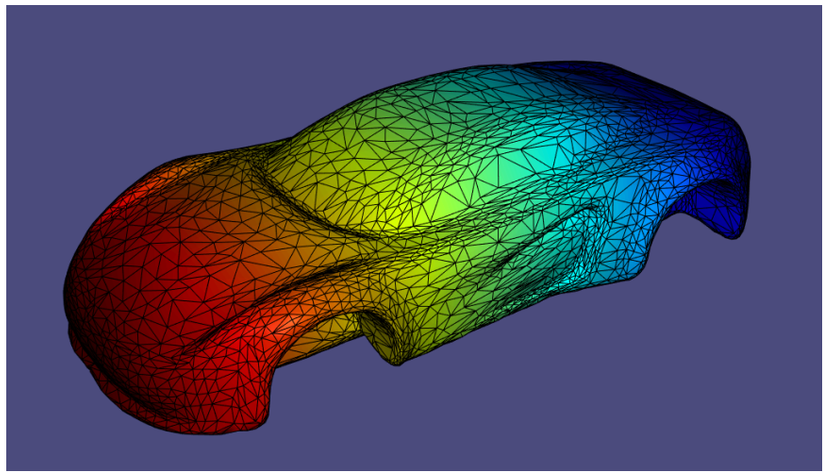


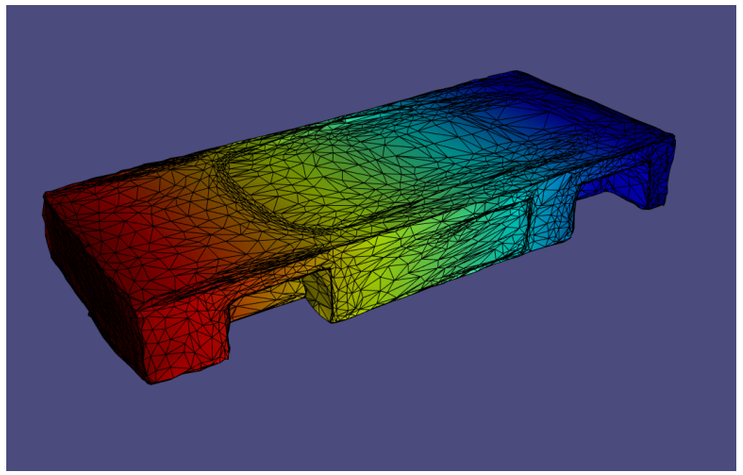


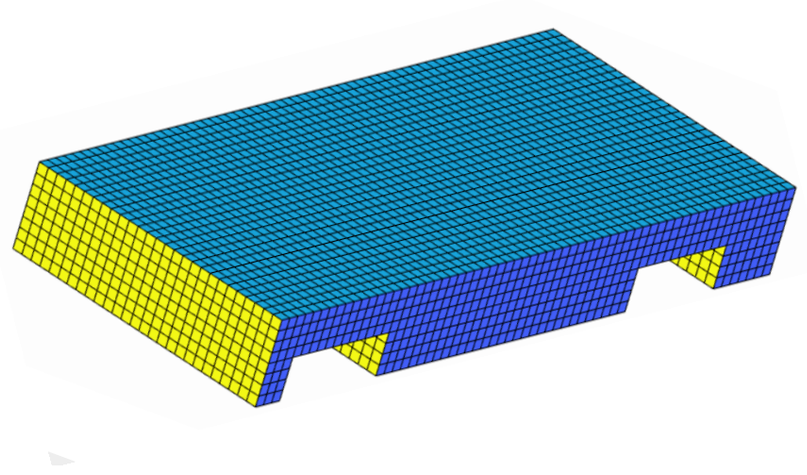


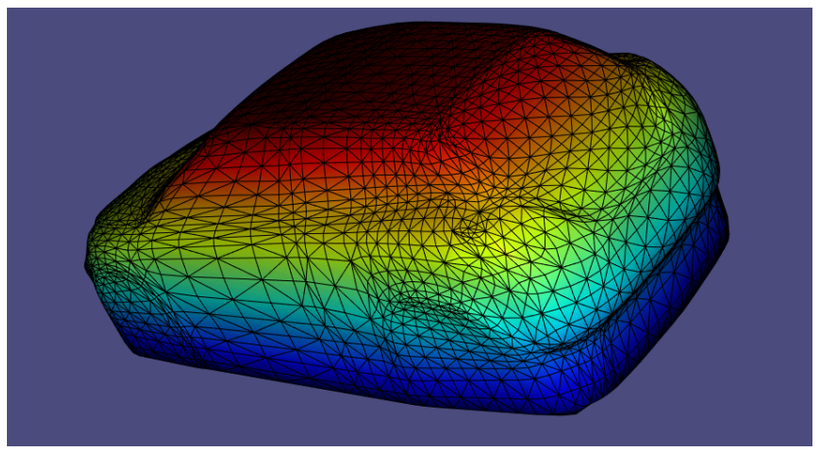


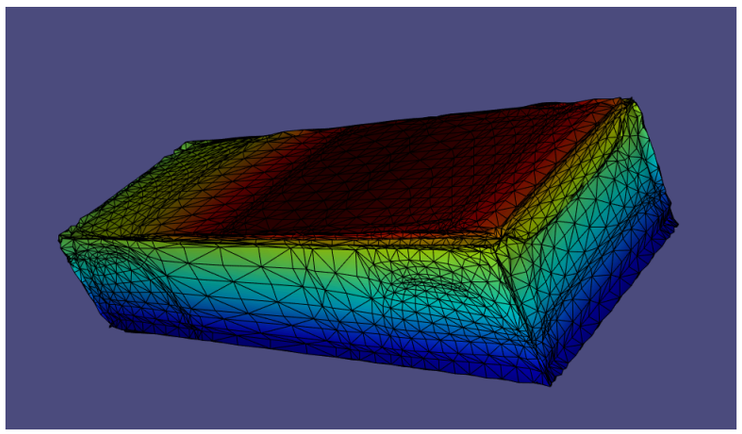
Results: Cars dataset

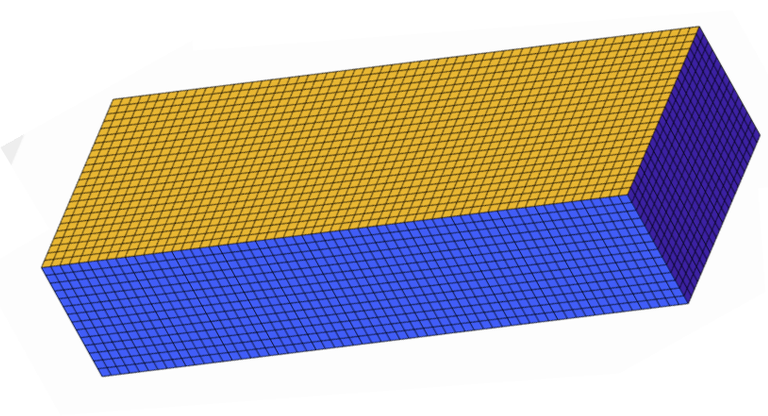




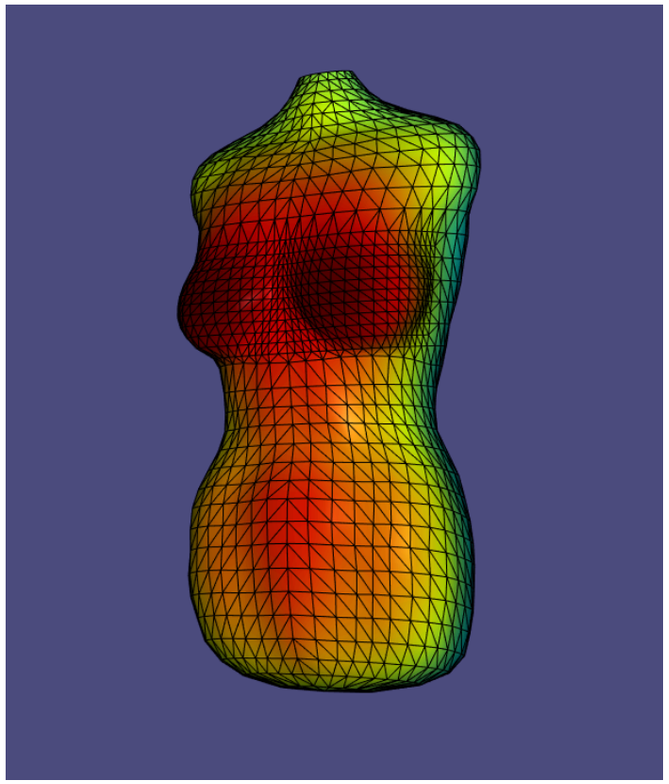


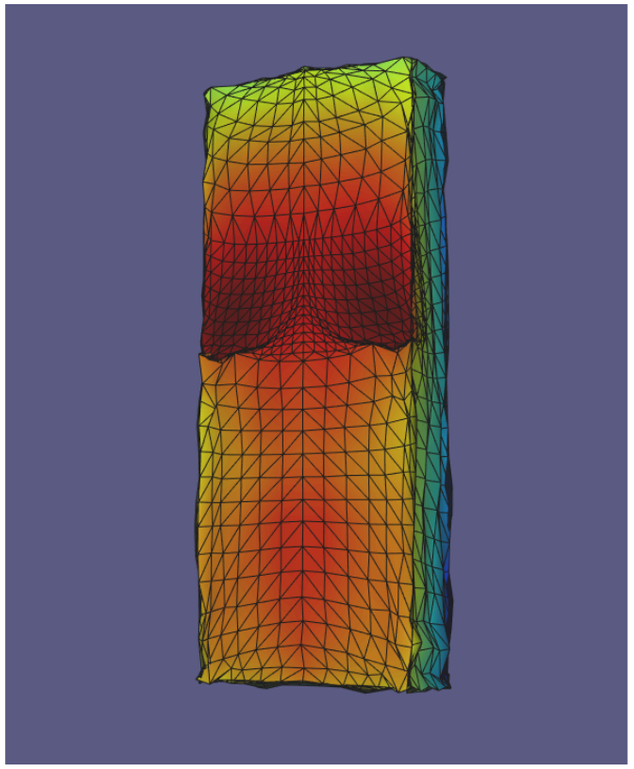


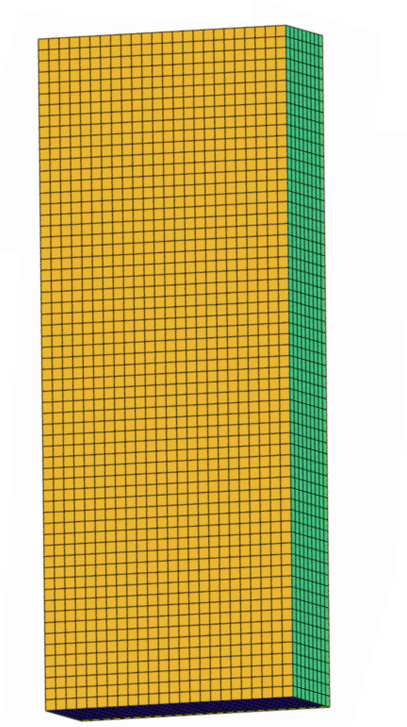


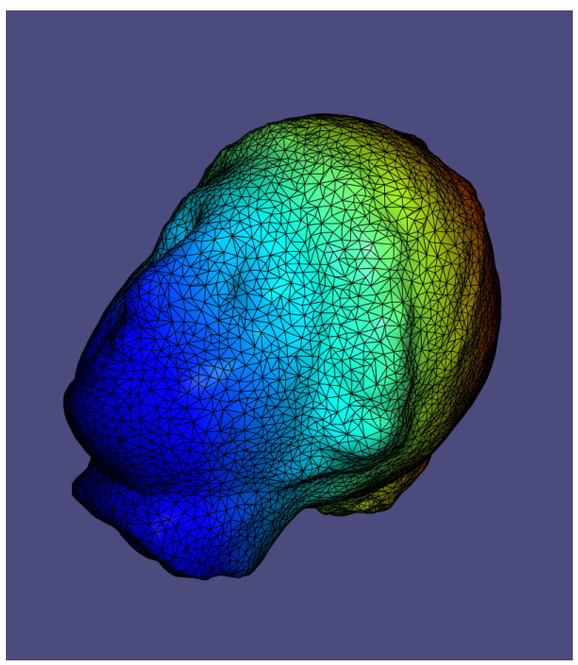


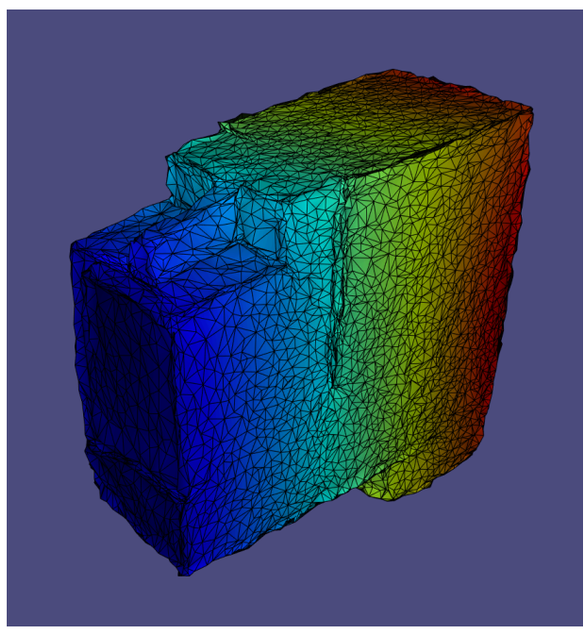
Results: Body and faces

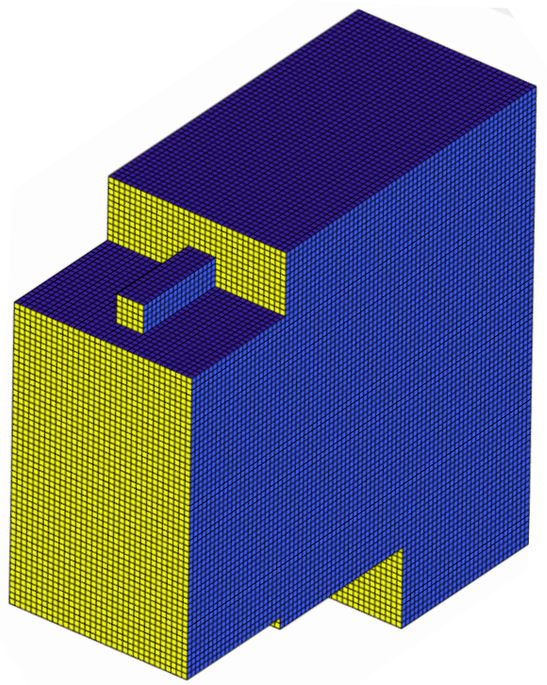












Conclusions and next steps

- Robust poly-cube mapping algorithm
- Naive MATLAB implementation ~ 50 times faster than state of the art
- How do we evaluate the quality of mapping?
- Can we learn poly-cube mapping…
  - from voxelized meshes?
  - from RGB/RGBD images?
- How to implement efficiently convolutions on poly-cube domains?