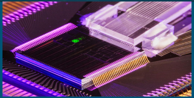
Sandia National Laboratories

Fast and automatic mesh generation from segmentation



SMART Exchange 2025 – Livermore, California, USA

Presented by:

Michael R. Buche

Michael R. Buche[1,*] and Chad B. Hovey[1]

[1]Sandia National Laboratories

*mrbuche@sandia.gov

4/23/2025

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

SAND NO. 2025-04175C

Abstract

Creating finite element meshes from image stacks is a bottleneck in analyst workflows.

- Difficult to recover a high-quality, physically-representative, tractable mesh.
- Often times one (or even two) of these three goals must be sacrificed.

Solutions are sometimes inflexible, slow, or unavailable (closed-source or non-existent).

- Stringing together tools to fill capability gaps complicates and threatens workflows.
- Closed-source tools inhibit collaboration and fall behind cutting-edge methodology.

`automesh` [1], an open-source automatic mesh generation tool written in Rust (CLI+Python).

- Robust automation of many features (meshing, surface reconstruction, etc.).
- Testing of completed features is showing significantly faster time-to-solution.
- Capabilities are comparable to Sculpt [2], providing a useful measuring stick.

Segmentation

Images become segmentations through categories.

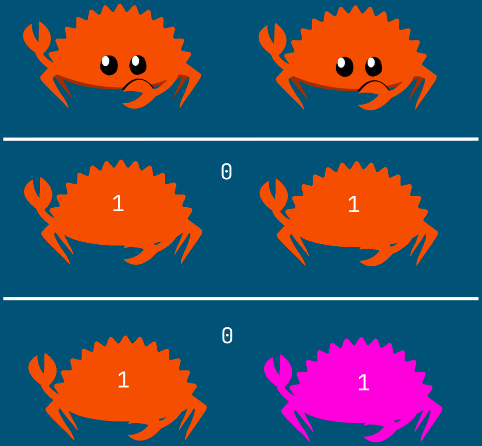
- Image stacks obtained from scans (e.g., CT).
- Each pixel in an image is assigned a class.
- Semantic segmentations aggregate objects.
- Instance segmentations differentiate objects.

Stacks of categorized pixels create a set of voxels.

- Simply an ordered list of unsigned integers.
- Storage cost is typically small (NPY, SPN).
- This is the starting point for `automesh` users.

Many applications for image-to-mesh workflows:

- Traumatic brain injury modeling (ONR).
- Defects, microstructures, surveillance, etc.



Meshing

Direct voxel-to-hex meshing:

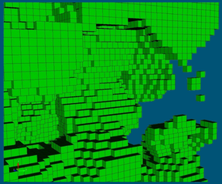
- Simple, robust, perfect quality [2].
- Often intractable element count.
- Poor representation of internal surfaces.
- Past focus of `automesh` (completed).

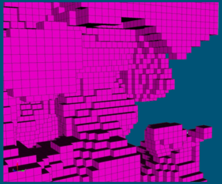
Internal surface reconstruction:

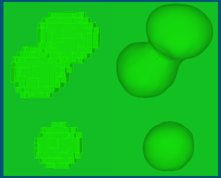
- Marching cubes, dual contouring, etc.
- Smooth, but preserve volume and manifold.
- Current focus of `automesh` (nearly done).

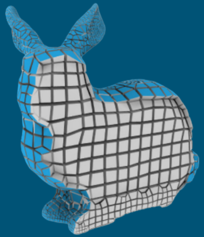
Adaptive hexmeshing [3] or tetmeshing:

- With or without surface reconstruction.
- Secondary focus of `automesh` (for now).









Performance

Measure time for direct voxels-to-hexes meshing.

- Perfect cube of a single material type.
- Optionally† also an embedded sphere.

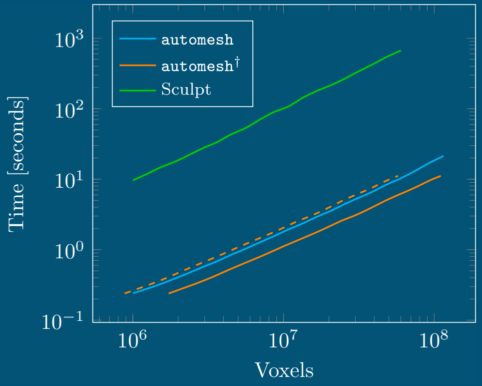
Ideal $O(N)$ scaling on one processor, rates are:

- `automesh` (5.7 million voxels/second)
- `automesh`† (9.9 million voxels/second)
- Sculpt (0.1 million voxels/second)

† Rate is approximately scaled by the fraction of retained voxels, i.e., it is about 89% of the original rate when including removed voxels.

Memory not studied as closely yet, but with 125 GB:

- `automesh` could do about 1 billion voxels.
- Sculpt could do about 100 million voxels.



Smoothing

Laplace smoothing:

- Moves nodes towards average of neighbors.

$$\Delta \mathbf{x}_a = \lambda \left(\frac{1}{n} \sum_{b=1}^{n} \mathbf{x}_b - \mathbf{x}_a \right)$$

- Drastically reduces volume (-16% in 10 steps).

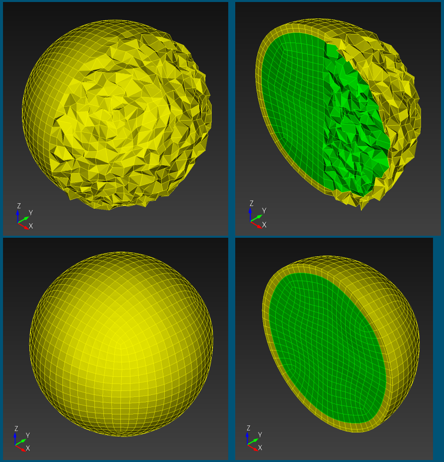
Taubin [4] smoothing:

- Alternates deflation and inflation ($\lambda < -\mu$).
- Nearly preserves volume (+1% in 200 steps).
- Acts like a low-pass filter, $k = \lambda^{-1} + \mu^{-1}$.

$$\lambda = 0.63,\ k = 0.1,\ \mu = \left(k - \lambda^{-1}\right)^{-1} = -0.67$$

Optional [5] hierarchical control:

- "Separately" smooth surfaces and volumes.



Octree

Efficient representation using an octree [6, 7].

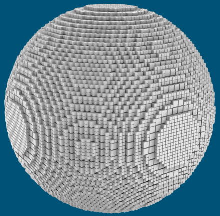
- Recursive subdivision of space into 8 octants.
- Rules for subdivision based on materials.
- Additional balancing and pairing rules.

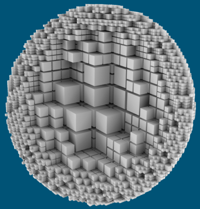
An octree as an intermediate workflow step.

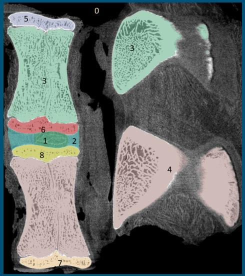
- Creates an "adaptive" segmentation.
- Accelerates methods like defeaturing.
- Enables adaptive finite element meshes.

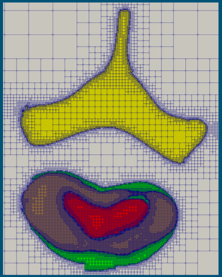
Micro-CT of a spinal unit from SwRI [8].

- 1 billion voxels become 10 million cells.
- 5 million cells are removable void.
- 200x reduction, and in only 36 seconds.









Reconstruction

Smooth reconstruction of internal surfaces.

- More representative of physical features.
- Output (STL) facilitates volume meshing.

`automesh` simply facets the material boundaries.

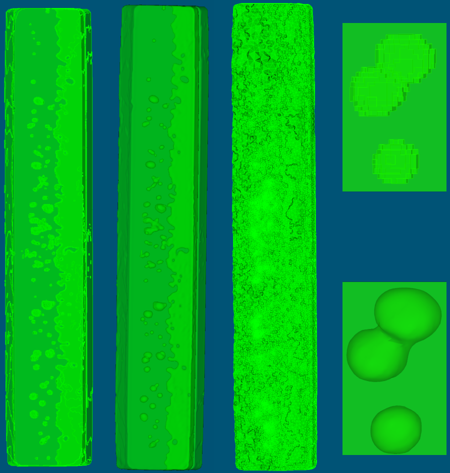
- Volume-preserving and free of holes.
- Rectify non-manifold edges and vertices.
- Defeaturing and Taubin smoothing are key.

Micro-CT scan of a laser weld section [9].

- Input: 2 materials, 6,748,800 voxels.
- Read, defeature†, mesh‡, smooth, write.
- Output: STL with 762,396 facets, 33 seconds.

† 13.5 seconds, compared to 36 minutes for Sculpt (160x slower).

‡ 18.2 seconds; with defeaturing, is about 95% of the total time.



Adaptivity

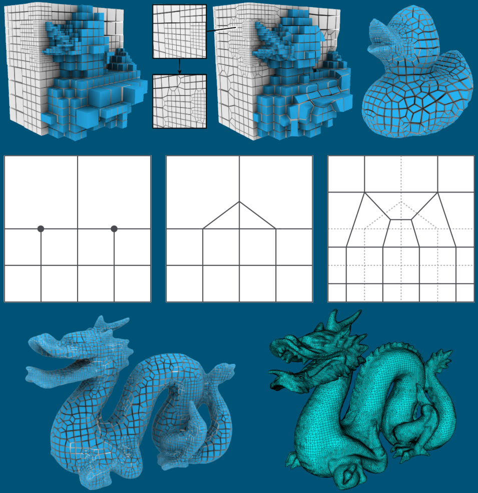
Adaptive hexmeshing using dualization [3].

- A (weakly) balanced and paired octree.
- Hanging nodes connect for a polygonal mesh.
- Centroids connect for guaranteed hexes.
- Levels of adaptivity are automatic.
- Faster transitions than other methods.

Work-in-progress within `automesh`.

- Strongly balanced (16% average difference).
- Pairing work [10] would be better (50% less).
- Element quality is somewhat low (initially).
- Adaptive tetmeshing also possible [11].

Surface reconstruction typically still necessary.



Flexibility

Numerous capabilities and options to enable users to tackle many problems.

- Conversion, defeaturing, meshing, metrics, reconstruction, removal, scaling, smoothing.
- Growing collection of input and output types (Abaqus, Exodus, CSV, NPY, SPN, STL, VTK).

User-friendly tool with multiple interfaces and documentation for a variety of users.

- CLI, Python, and Rust interfaces, each complete with documentation.
- User guide for getting started and understanding the methods.
- Automated testing, packaging, and deployment using CI/CD.

```
$ automesh mesh -i weld.npy --remove 0 --defeature 64 --surface -o weld.stl smooth
    automesh 0.3.2
     Reading weld.npy
        Done 341.475783ms [2 materials, 6,748,800 voxels]
 Defeaturing clusters of 64 voxels or less
        Done 13.476138903s
     Meshing internal surfaces
        Done 18.16082286s [1 blocks, 762,396 elements, 381,249 nodes]
   Smoothing with 200 iterations of Taubin
        Done 632.348188ms
     Writing weld.stl
        Done 61.946009ms
       Total 33.199842742s
```

Conclusion

`automesh`: Automatic mesh generation.

- Flexible, effective, user-friendly software.
- Significantly faster than a comparable tool.

Rust and the open-source world have been key.

- Rust enables high-performance tools [12].
- Open-source is more dynamic and advanced.

These ingredients continue to fuel rapid progress.

- Tweak reconstruction, reconsider adaptivity.
- Rust for HPC is coming, e.g., `lamellar` [13].

Success is possible when shifting paradigms.

- Massively parallel vs. massive improvement.
- Technical debt vs. investing in starting over.

```
$ automesh --help

     @@@@@@@@@@@@@@@@
      @@@@  @@@@@@@@@@
     @@@@  @@@@@@@@@@@
    @@@@  @@@@@@@@@@@@    automesh: Automatic mesh generation
      @@    @@    @@      Chad B. Hovey <chovey@sandia.gov>
      @@    @@    @@      Michael R. Buche <mrbuche@sandia.gov>
    @@@@@@@@@@@@  @@@
    @@@@@@@@@@@  @@@@     Notes:
    @@@@@@@@@@ @@@@@ @    - Input/output file types are inferred
     @@@@@@@@@@@@@@@@     - Scaling is applied before translation

Usage: automesh [COMMAND]

Commands:
  convert    Converts between mesh or segmentation file types
  defeature  Defeatures and creates a new segmentation
  mesh       Creates a finite element mesh from a segmentation
  metrics    Quality metrics for an existing finite element mesh
  smooth     Applies smoothing to an existing mesh
  help       Print this message or the help of the given subcommand(s)

Options:
  -h, --help     Print help
  -V, --version  Print version

$ automesh smooth --help
Applies smoothing to an existing mesh

Usage: automesh smooth [OPTIONS] --input <FILE> --output <FILE>

Options:
  -c, --hierarchical       Pass to enable hierarchical control
  -i, --input <FILE>       Mesh (inp | stl) input file
  -o, --output <FILE>      Smoothed mesh (exo | inp | mesh | stl | vtk) output file
  -n, --iterations <NUM>   Number of smoothing iterations [default: 20]
  -m, --method <NAME>      Name of the smoothing method [default: Taubin]
  -k, --pass-band <FREQ>   Pass-band frequency for Taubin smoothing [default: 0.1]
  -s, --scale <SCALE>      Scaling parameter for smoothing [default: 0.6307]
      --metrics <FILE>     Name of the quality metrics file (csv | npy)
  -q, --quiet              Pass to quiet the terminal output
  -h, --help               Print help
```

References

[1] C. B. Hovey and M. R. Buche, automesh 0.3.3 (2025).

[2] S. J. Owen, C. E. Ernst, and C. J. Stimpson, Sandia National Laboratories **6412** (2019).

[3] M. Livesu, L. Pitzalis, and G. Cherchi, ACM Transactions on Graphics **41**, 1 (2021).

[4] G. Taubin, Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, 351 (1995).

[5] Y. Chen and M. Ostoja-Starzewski, Acta Mechanica **213**, 155 (2010).

[6] D. J. Meagher, Rensselaer Polytechnic Institute IPL-TR-80-111 (1980).

[7] M. Bracci, M. Tarini, N. Pietroni, M. Livesu, and P. Cignoni, Computer-Aided Design **110**, 24 (2019).

[8] D. P. Nicolella, S. K. Shaffer, L. L. Frazer, A. D. Pant, and V. B. Kote, I-PREDICT, MTEC (2025).

[9] A. T. Polonsky, J. D. Madison, M. Arnhart, H. Jin, K. N. Karlson, A. J. Skulborstad, J. W. Foulk, and S. G. Murawski, Tomography of Materials and Structures **2**, 100006 (2023).

[10] L. Pitzalis, M. Livesu, G. Cherchi, E. Gobbetti, and R. Scateni, ACM Transactions on Graphics **40**, 1 (2021).

[11] X. Liang and Y. Zhang, Engineering with Computers **30**, 211 (2014).

[12] Trifecta Tech Foundation, zlib-rs 0.4.2 (2025).

[13] Pacific Northwest National Laboratory, lamellar 0.7.0 (2024).