SnappyHexMesh in Practice

Tuomo.Keskitalo@iki.fi

Finnish OpenFOAM User Day 2025

What is a good CFD mesh?

- No general agreement about the definition of mesh goodness exists
- My practical definition for a good mesh:

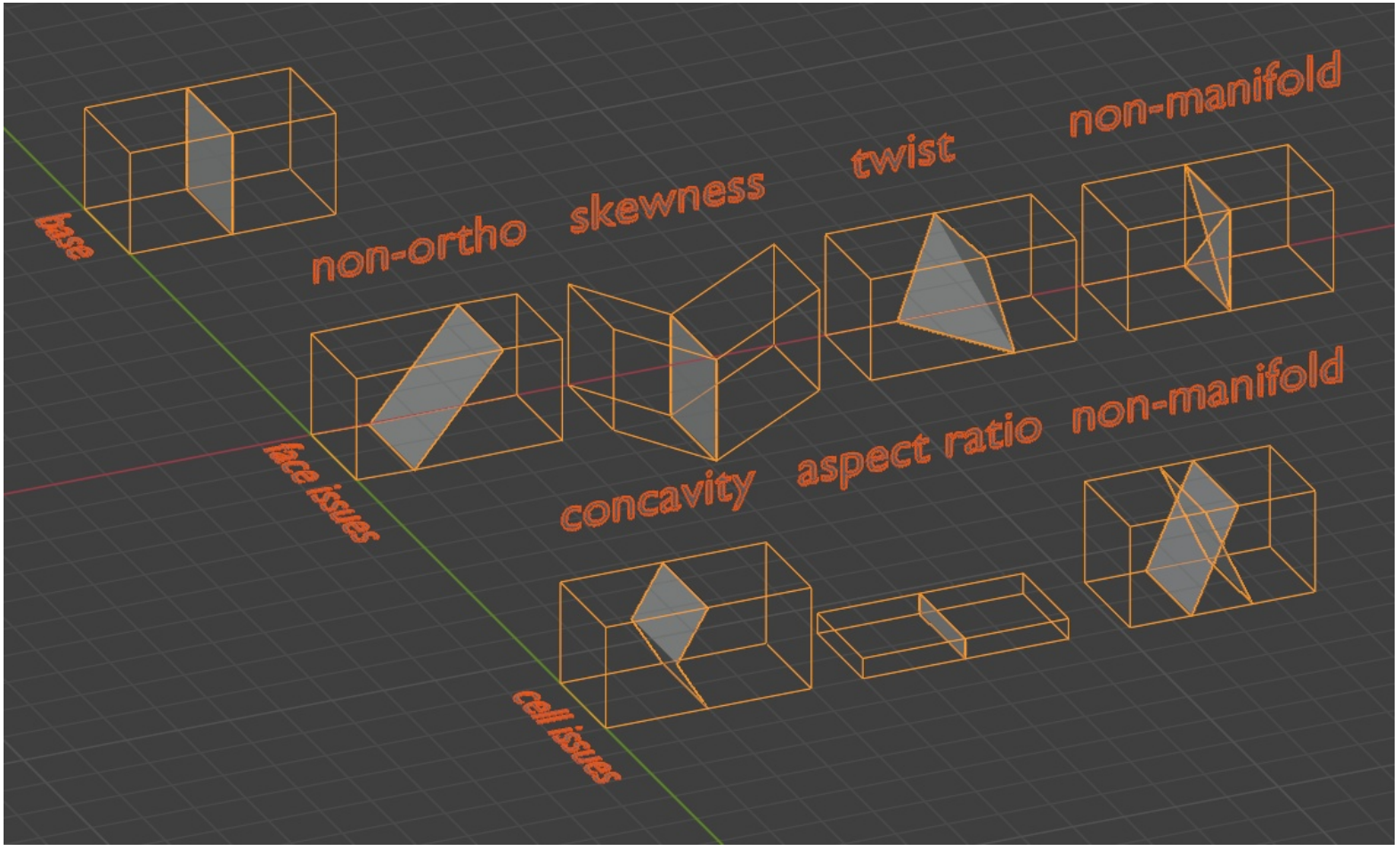
  1.) It ***works*** with your solver (does not cause crashing)

  2.) It captures ***well enough*** the relevant physics that you seek to simulate

  3.) Mesh creation ***can be done fast enough,*** even for complex geometry

- Automatic mesh generation is the practical way to go, and SnappyHexMesh is probably the best open source alternative out there

Mesh issues


base
non-ortho
skewness
twist
non-manifold
face issues
concavity
aspect ratio
non-manifold
cell issues

Quick Intro to Snappy

- Inputs:
  - Background mesh (preferably made of cubes with blockMesh)
  - Surfaces of geometry (e.g. STL files)
  - Algorithm related settings (snappyHexMeshDict)
  - Mesh quality criteria (snappyHexMeshDict or separate meshQualityDict)
- Algorithm:
  1) Castellation iterations (includes mesh refinement iterations) (fast step)
  2) Snapping iterations (medium fast step)
  3) Layer addition iterations (very slow step)
- Openfoam.com version has more features than openfoam.org, use latest stable versions
- More information:
  - Engys presentation slides from OFW7 and OFW11
  - Annotated case dicts: openfoam.com, openfoam.org
  - Openfoamwiki.net SnappyHexMesh page

Snappy algorithm pros and cons

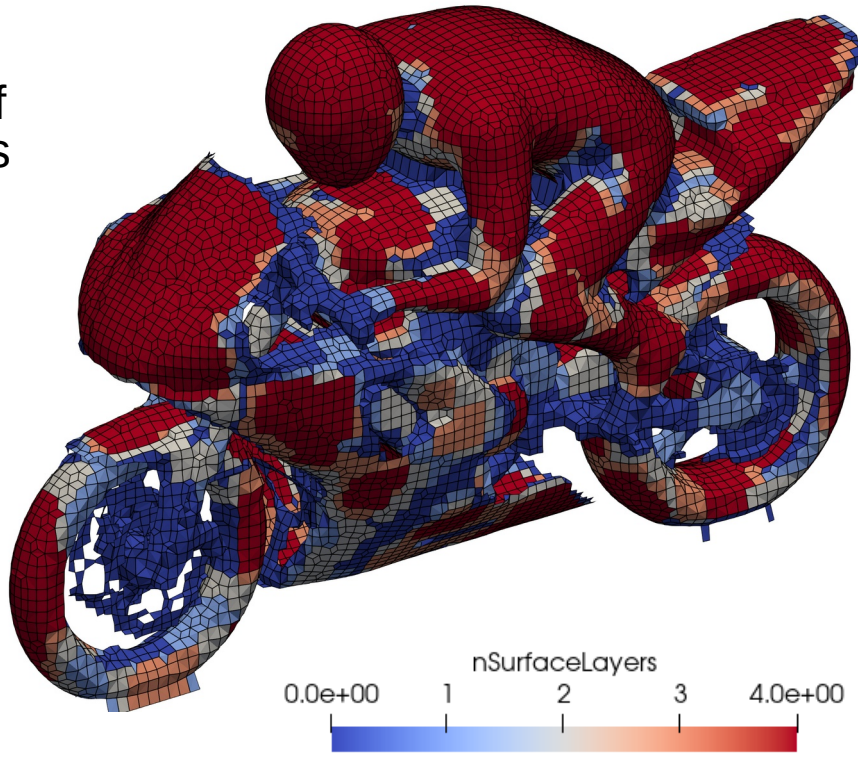
- A core feature in Snappy is that ***mesh quality is checked*** after every mesh modification iteration, and ***the mesh is <u>partially</u> reverted*** to previous stage if quality is compromised (undo iterations)
  - Pro: Quality is maintained all the time, bad cells are handled early
  - Con: Snappy meshing is a stochastic ”noisy” process.
    Small changes in settings may cause changes throughout the mesh.
  - Pro/Con: Algorithms are not required to always make sane changes to the mesh, as the undo iterations tend to fix the problems.
- Pro: **Castellation and snapping phases are fast** (cartesian background mesh generation is fast) and **majority of the cells are high quality**
- Con: **Layer addition phase is slow** and it’s hard to get good results (mesh shrinking creates squished cells, good quality is hard to obtain for extruded layer cells)

Why is Snappy so hard to use?

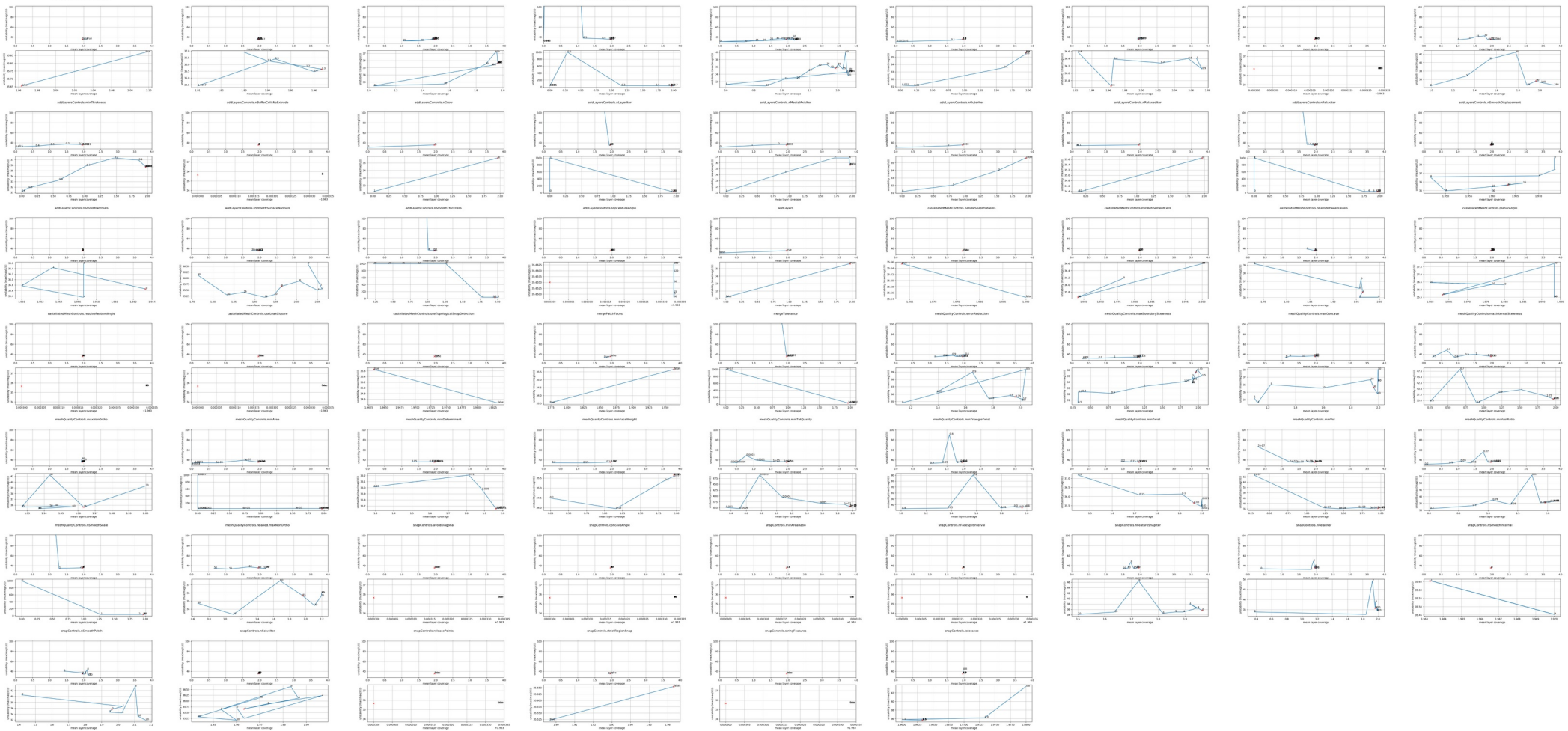
- Because
  - automatic mesh generation is a complex task
    → Snappy is a complex tool (>50 non-geometry parameters!)
  - no-one has taken the effort or provided the funding to make it easy to use
    - Commercial meshing software is available
    - Non-commercial developers can spare only a limited amount of time

Experiments with Snappy Settings

- Original ”snappyLayerTests” setup by Ben Malin which I developed further
- Idea: automatic variation, testing and evaluation of the effect of various Snappy settings on the results including:
  - Mesh quality checks from checkMesh
  - Visual images of the mesh
  - Stability testing of the mesh with a solver (simpleFoam, 200 iterations)
- Test case: The OF motorBike geometry (modified) with a coarse mesh (~100 000 cells)
  - Intentionally tested with a coarse mesh, because you can never afford to have ”enough” mesh refinement in practice for cases like this



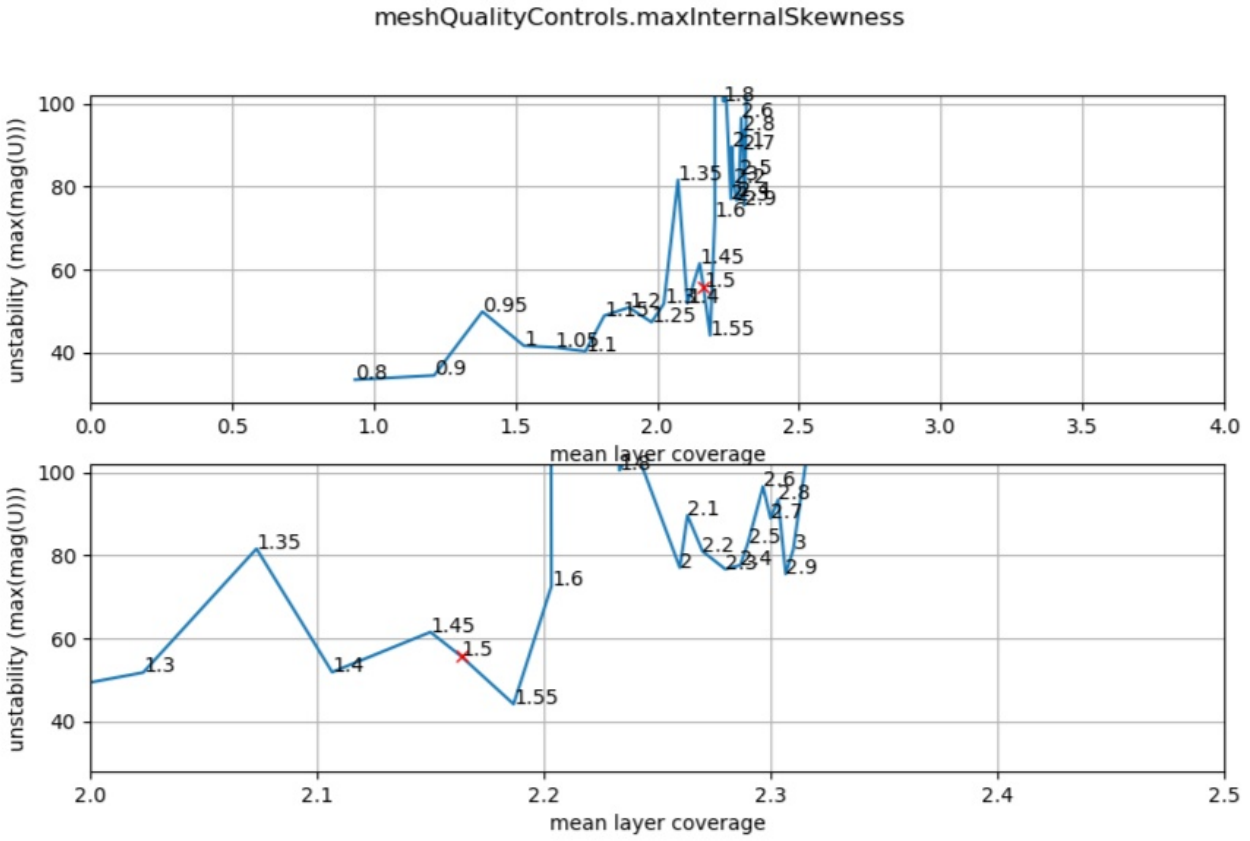
Sweep of Snappy Parameters



Tuning of Snappy Settings

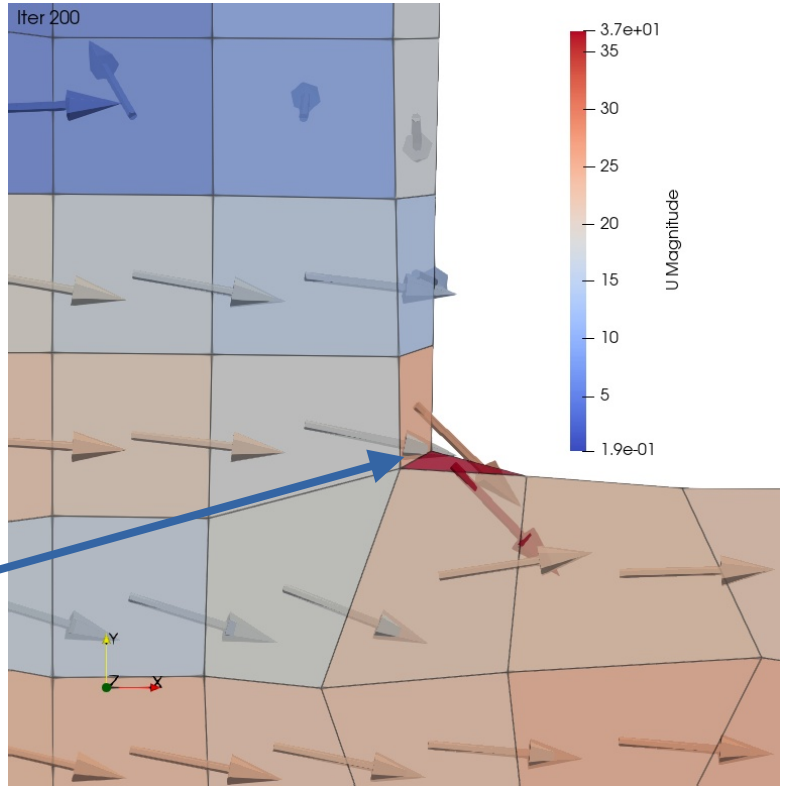
Heuristic tuning of Snappy parameters was done to:

- Maximize layer coverage
- Minimine max(mag(U)) of simpleFoam
- Try to find an acceptable tradeoff



What I learned from Snappy Experiments

- Automatic mesh generation is not trivial
- There is tradeoff between mesh quality and snapping quality / layer coverage
- Improving snapping or increasing coverage by relaxing quality criteria will eventually cause solver to crash due to numerical issues caused by the mesh
- Visually bad/good is not necessarily numerically bad/good
- Not all checkMesh errors are fatal.
  A mesh with some errors can work just fine.
- Collapsing layers on concave mesh features can help with solver stability (layerTerminationAngle setting in openfoam.com version)

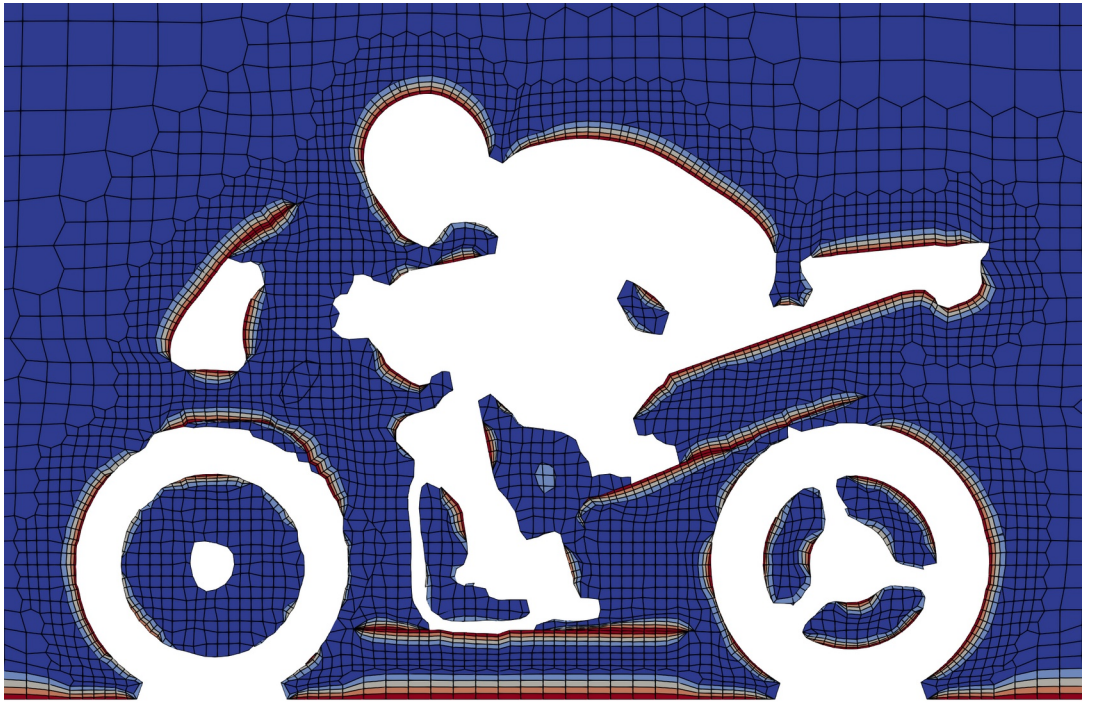


Tips to get best out of Snappy

- If you don’t know Snappy, use the SnappyHexMesh GUI Blender add-on
- Don’t be fooled by Paraview visual illusions (e.g. don’t use ”Decompose polyhedra”!)
- STLs must be ”clean” and ”water tight” for best results
- Background mesh is ideally made of perfect cubes which coincide with STL surfaces (to get good snapping)
- Feature edge detection must be correct (check the eMesh visually!), otherwise you will get bad edge snapping
- Use good Snappy settings (e.g. from SnappyHexMesh GUI), don’t just copy from a random tutorial
- Check the mesh from all stages visually (castellated mesh, snapped mesh, layered mesh)
- Use the Iterative Snappy Workflow
  - Start with very coarse mesh and a simple geometry, and increase mesh size and geometric complexity step by step to catch setup errors early
  - Workflow: modify geometry, clean case folder, export STLs, regenerate mesh, check the results
  - Add final refinements and layers as final steps to make iterations fast
- More tips in SnappyHexMesh GUI docs

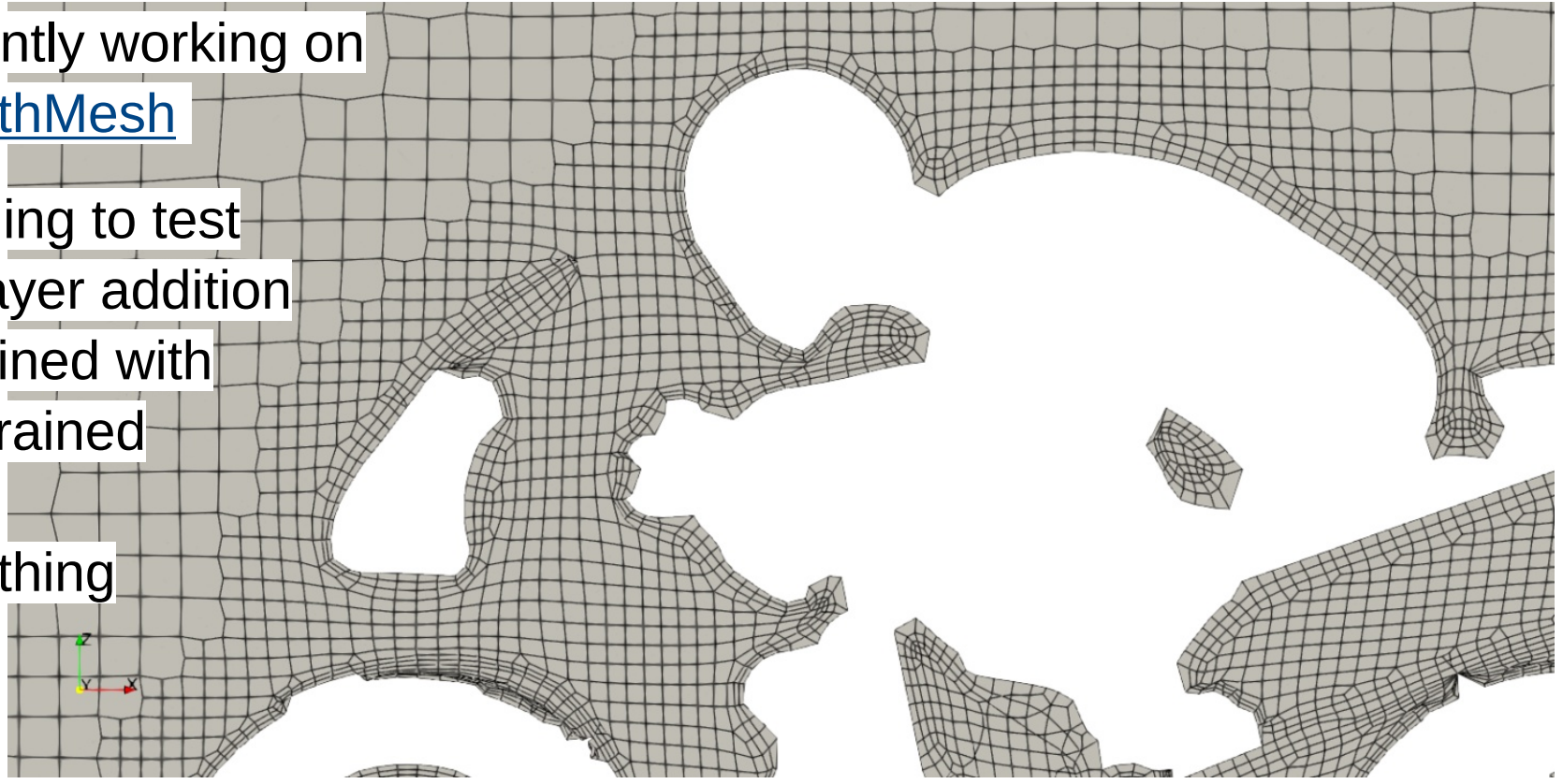
Why good boundary layer coverage is so hard to obtain?

- Because of mesh quality criteria
  - Snappy collapses edges away aggressively to improve quality → layer coverage tends to decrease recursively during layer iterations
  - Twisted boundary faces create bad quality layer cells → would need to split twisted faces prior to layer addition (this is not supported now)



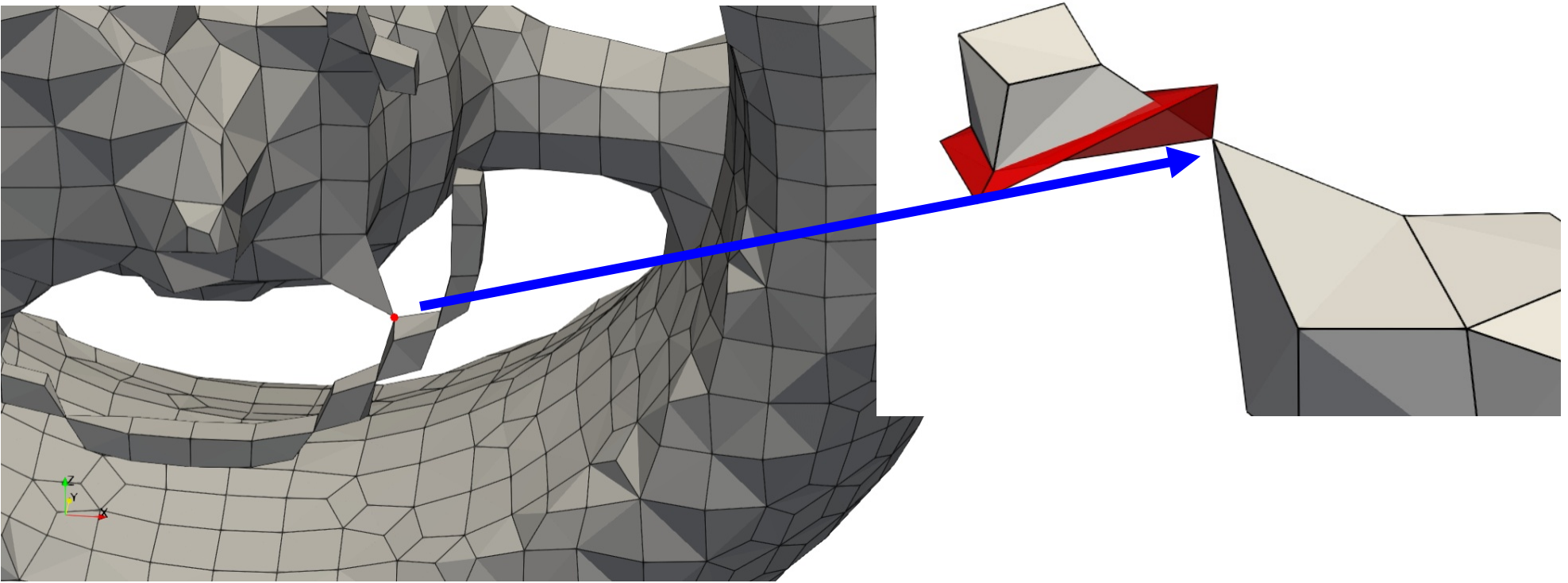
Work in Progress: Mesh smoothing

- Currently working on smoothMesh
- Planning to test thin layer addition combined with constrained mesh smoothing



Work in Progress: Learning and Debugging Snappy

- Non-manifold point issue in layer addition



Thanks!