Cleaver

Release v2.4

Jonathan Branson, Brig Bagley, Jess Tate, Ally Warner, Dan White

May 14, 2023

CONTENTS

Cleaver is an open-source multimaterial tetrahedral meshing tool developed by the NIH Center for Integrative Biomedical Computing at the University of Utah Scientific Computing and Imaging (SCI) Institute.

CHAPTER

ONE

ABOUT

1.1 Overview

Cleaver is an open-source multi-material tetrahedral meshing tool that creates conforming tetrahedral meshes for multimaterial or multiphase volumetric data. These meshes ensure both geometric accuracy and bounded element quality using the Lattice Cleaving algorithm.

1.2 Method

The Cleaver Library is based on the `Lattice Cleaving` algorithm.

The method is a stencil-based approach, and relies on an octree structure to provide a coarse level of grading in regions of homogeneity. The cleaving algorithm works by utilizing indicator functions, which indicate the strength or relative presence of a particular material. At each point, only the material with the largest indicator value is considered present.

The method is theoretically guaranteed to produce valid meshes with bounded dihedral angles, while still conforming to multimaterial material sur- faces. Empirically these bounds have been shown to be well within useful ranges, thus creating efficient meshes for analysis, simulation, and visualization.

Reference:

> Bronson J., Levine, J., Whitaker R., "Lattice Cleaving: Conforming Tetrahedral Meshes of Multimaterial Domains with Bounded Quality". Proceedings of the 21st International Meshing Roundtable (San Jose, CA, Oct 7-10, 2012)
>
> See https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4190882/

1.3 Authors

Cleaver is an open-source project with a growing community of contributors. The software was initially developed by the NIH Center for Integrative Biomedical Computing at the University of Utah Scientific Computing and Imaging (SCI) Institute.

Many Cleaver contributors are listed in the Contributors Graph. However, the following authors have made significant contributions to the conception, design, or implementation of the software and are considered "The Cleaver Developers":

- Jonathan Bronson
- Brig Bagley
- Jess Tate

- Ally Warner
- Dan White
- Ross Whitaker

1.4 Acknowledgement

This project was supported by the National Institute of General Medical Sciences of the National Institutes of Health under grant numbers P41 GM103545 and R24 GM136986.

1.5 Citing Cleaver

When citing Cleaver in your scientific research, please mention the following work to support increased visibility and dissemination of our software:

> Cleaver: A MultiMaterial Tetrahedral Meshing Library and Application. Scientific Computing and Imaging Institute (SCI), Download from: http://www.sci.utah.edu/software.html, 2015.

For your convenience, you may use the following BibTex entry:

```
@Misc{SCI:Cleaver,
  author =    "CIBC",
  year =      "2015",
  note =      "Cleaver: A MultiMaterial Tetrahedral Meshing
              Library and Application. Scientific Computing and
              Imaging Institute (SCI), Download from:
              http://www.sci.utah.edu/software.html",
  keywords =  "Cleaver, CIBC",
}
```

1.6 Bibliography

Below is a list of publications that reference Cleaver.

Note: Please note that this list only includes citations from publications published after 2017 and not involving researchers or developers from the SCI Institute.

1. Frank Abdi, Harsh Baid, Rashid Miraj, Beth Clarkson, and Jacob Fish. Computational approaches for composite materials. In *Composite Materials Qualification*. Begell House, 2021.
2. Sahar Bakhshian, Zhuofan Shi, Muhammad Sahimi, Theodore T Tsotsis, and Kristian Jessen. Image-based modeling of gas adsorption and deformation in porous media. *Scientific reports*, 8(1):1–12, 2018.
3. Julia Boonzaier, Petar I Petrov, Willem M Otte, Nickolay Smirnov, Sebastiaan FW Neggers, and Rick M Dijkhuizen. Design and evaluation of a rodent-specific transcranial magnetic stimulation coil: an in silico and in vivo validation study. *Neuromodulation: Technology at the Neural Interface*, 23(3):324–334, 2020.
4. Mar Cortes, Laura Dubreuil Vall, Giulio Ruffini, Douglas Labar, and Dylan Edwards. Transcranial direct current stimulation in chronic spinal cord injury: quantitative eeg study. *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation*, 10(1):e13, 2017.

5. Fotios Drakopoulos. *Finite element modeling driven by health care and aerospace applications*. PhD thesis, Old Dominion University, 2017.

6. Jacob Fish and Nan Hu. Multiscale modeling of femur fracture. *International Journal for Numerical Methods in Engineering*, 111(1):3–25, 2017.

7. Kathleen M Friel, Peter Lee, Disha Gupta, Hsing-Ching Kuo, Ana RP Smorenburg, and Dylan J Edwards. Combined transcranial direct current stimulation and upper extremity robotic therapy improves upper extremity function in an adult with cerebral palsy: a pilot study. *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation*, 10(1):e13, 2017.

8. Eloy García, Yago Diez, Oliver Diaz, Xavier Lladó, Robert Martí, Joan Martí, and Arnau Oliver. A step-by-step review on patient-specific biomechanical finite element models for breast mri to x-ray mammography registration. *Medical physics*, 45(1):e6–e31, 2018.

9. Ho Quang Nguyen, Tien Tuan Dao, Alain Rassineux, and Marie Christine Ho Ba Tho. Material-driven mesh of the lumbar spine derived from ct data. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 6(2):128–136, 2018.

10. Anh Phong Tran and Qianqian Fang. Fast and high-quality tetrahedral mesh generation from neuroanatomical scans. *arXiv preprint arXiv:1708.08954*, 2017.

11. Anh Phong Tran, Shijie Yan, and Qianqian Fang. Improving model-based functional near-infrared spectroscopy analysis using mesh-based anatomical and light-transport models. *Neurophotonics*, 7(1):015008, 2020.

12. Laura Dubreuil Vall, Mar Cortes, Dylan Edwards, Giulio Ruffini, and David Putrino. Eeg recordings during sham control transcranial direct current stimulation protocol. *Brain Stimulation: Basic, Translational, and Clinical Research in Neuromodulation*, 10(1):e13, 2017.

13. Jared Vicory, Ethan Murphy, and Ryan J Halter. Creation and visualization of high-quality tetrahedral meshes from segmentations using 3d slicer. *ELECTRICAL IMPEDANCE TOMOGRAPHY*, pages 55, 2018.

14. Charlotte E Vorwald, Shreeya Joshee, and J Kent Leach. Spatial localization of endothelial cells in heterotypic spheroids influences notch signaling. *Journal of Molecular Medicine*, 98(3):425–435, 2020.

15. Chuan Wang, Jie Zhu, Yanwen Guo, and Wenping Wang. Video vectorization via tetrahedral remeshing. *IEEE Transactions on Image Processing*, 26(4):1833–1844, 2017.

16. Karissa M Wang, Amanda J Rickards, Trevor Bingham, Jonathan D Tward, and Ryan G Price. Evaluation of a silicone-based custom bolus for radiation therapy of a superficial pelvic tumor. *Journal of Applied Clinical Medical Physics*, pages e13538, 2022.

17. Jing Xu. *Automatic Linear and Curvilinear Mesh Generation Driven by Validity Fidelity and Topological Guarantees*. PhD thesis, Old Dominion University, 2020.

18. Jing Xu and Andrey N Chernikov. Homeomorphic tetrahedralization of multi-material images with quality and fidelity guarantees. *Procedia engineering*, 203:40–52, 2017.

19. Jiayi Yao, Xiuju Wu, Daoqin Zhang, Lumin Wang, Li Zhang, Eric X Reynolds, Carlos Hernandez, Kristina I Boström, Yucheng Yao, and others. Elevated endothelial sox2 causes lumen disruption and cerebral arteriovenous malformations. *The Journal of clinical investigation*, 129(8):3121–3133, 2019.

CHAPTER

TWO

GETTING STARTED

2.1 Introduction

Consider reading the Cleaver *Manual*.

2.2 System requirements

- Windows 10+, macOS 10.12+, and Ubuntu 20.04 or OpenSuse 15.1+ Recommended.
- CPU: Core Duo or higher, recommended i5 or i7
- Memory: 4Gb, recommended 8Gb or more
- Dedicated Graphics Card (OpenGL 4.1+, Dedicated Shared Memory, no integrated graphics cards)
- Graphics Memory: minimum 128MB, recommended 256MB or more

Caution: The following graphics cards are known to not support Cleaver:

- AMD Radeon HD 6310 (Integrated Card)
- AMD Radeon 7400 M
- INTEL HD 3000 (Integrated Card)

2.3 Using Installer

1. Download the latest installers.
2. Learn about the Cleaver *Command Line Tool* and *Graphical Interface*.

Tip: If there is no installer available for your platform, you may consider *using python*, installing the *SlicerSegmentMesher* extension or building Cleaver from *source*.

2.4 Using Python

Cleaver is available as Python wheels distributed on PyPI for Windows, macOS and Linux for integrating in either an ITK or VTK filtering pipeline.

2.4.1 ITK

1. Create a virtual environment then install the `itk-cleaver` package:

```
pip install itk-cleaver
```

2. Generate multi-material tetrahedral mesh from a label image

```
import itk

image = itk.imread('./mickey.nrrd')

tet_mesh, triangle_mesh = itk.cleaver_image_to_mesh_filter(image)

itk.meshwrite(triangle_mesh, './triangle_mesh.vtk')
```

2.4.2 VTK

2.5 Using 3D Slicer

Download 3D Slicer and install the SlicerSegmentMesher extension that enables creating volumetric meshes from segmentation using Cleaver.

2.6 Using C++ and CMake

See *Building Cleaver* as well as the *Cleaver Library* manual.

CHAPTER

THREE

MANUAL

3.1 Command Line Tool

Using the sphere indicator functions in `src/test/test_data/input/`, you can generate a simple tet mesh using the following command:

```
bin/cleaver-cli --output_name spheres -i ../src/test/test_data/input/spheres*.nrrd
```

Type:

```
bin/cleaver-cli --help
```

For a list of command line tool options.

```
Command line flags:
-a [ --alpha ] arg                  initial alpha value
-s [ --alpha_short ] arg            alpha short value for constant element sizing method
-l [ --alpha_long ] arg             alpha long value for constant element sizing method
-b [ --background_mesh ] arg        input background mesh
-B [ --blend_sigma ] arg            blending sigma for input(s) to remove alias artifacts
-m [ --element_sizing_method ] arg  background element sizing method (adaptive [default],
↪ constant)
-F [ --feature_scaling ] arg        feature size scaling (higher values make a coaser␣
↪mesh)
-j [ --fix_tet_windup ]             ensure positive Jacobians with proper vertex wind-up
-h [ --help ]                       display help message
-i [ --input_files ] arg            material field paths or segmentation path
-L [ --lipschitz ] arg              maximum rate of change of element size (1 is uniform)
-f [ --output_format ] arg      output mesh format (tetgen [default], scirun,
    matlab, vtk, ply [surface mesh only])
-n [ --output_name ] arg            output mesh name (default 'output')
-o [ --output_path ] arg            output path prefix
-p [ --padding ] arg                volume padding
-r [ --record ] arg                 record operations on tets from input file
-R [ --sampling_rate ] arg          volume sampling rate (lower values make a coarser␣
↪mesh)
-S [ --segmentation ]               the input file is a segmentation file
   [--simple]                       use simple interface approximation
-z [ --sizing_field ] arg           sizing field path
-t [ --strict ]                     warnings become errors
-e [ --strip_exterior ]             strip exterior tetrahedra
```

(continues on next page)

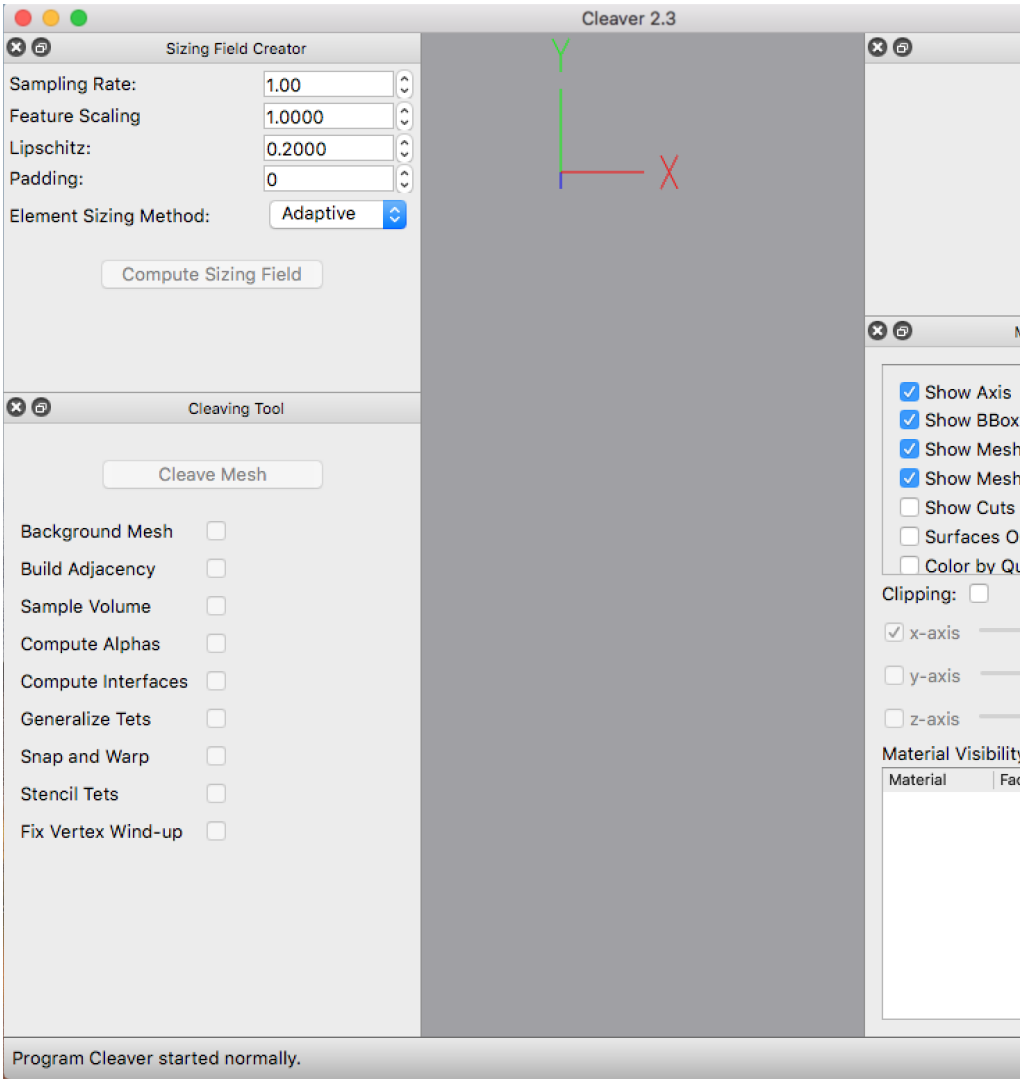
(continued from previous page)

```
-w [ --write_background_mesh ]      write background mesh
-v [ --verbose ]                    enable verbose output
-V [ --version ]                    display version information
```

3.2 Graphical Interface

You can run the GUI from the command line, or by double-clicking it in a folder.

```
gui/cleaver-gui.app
```



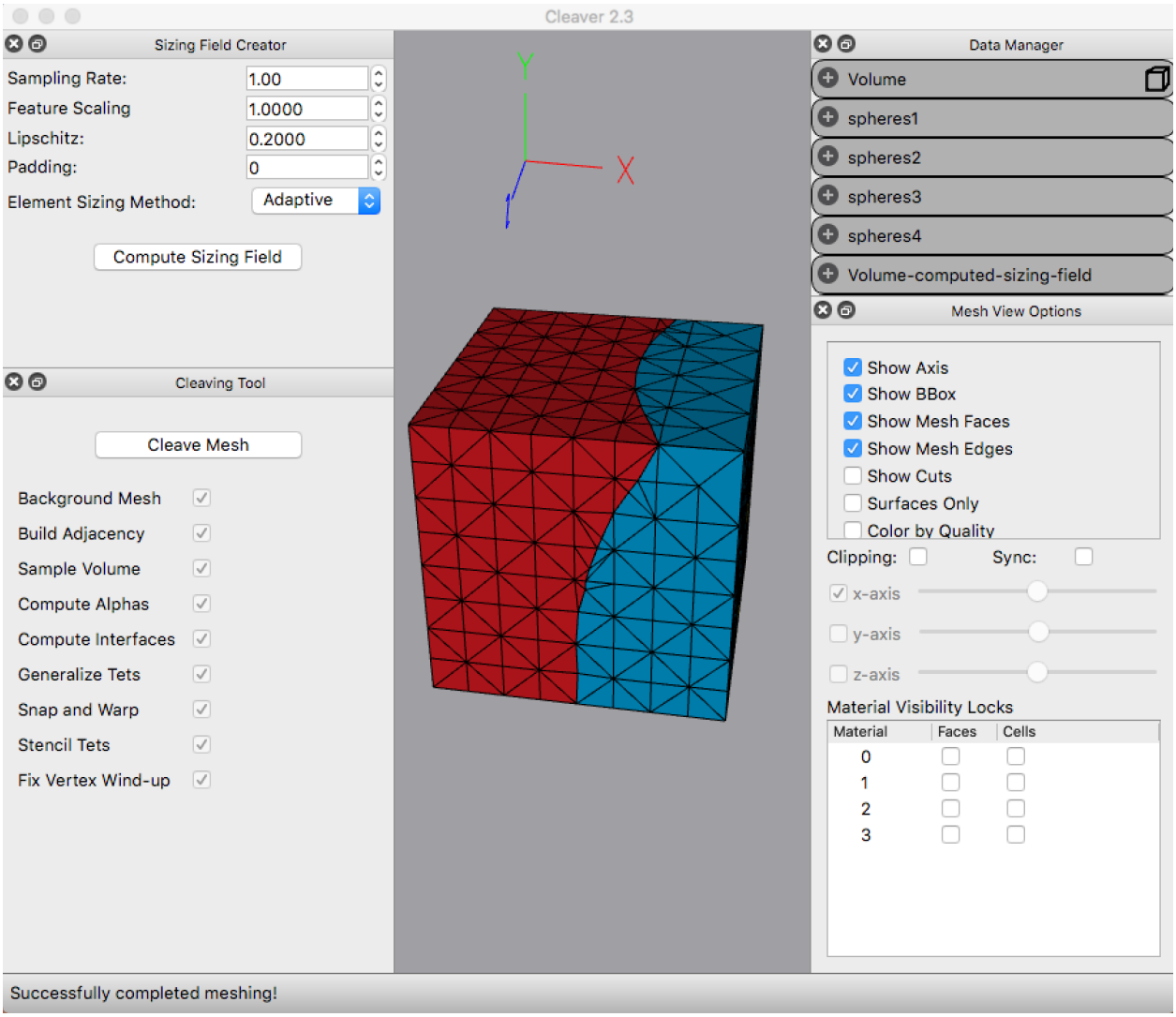
You should see a window similar to this: Load the spheres in `src/test/test_data/input` either with `ctrl+v` or `File -> Import Volume`, or load your own indicator functions or segmentation file.

Dialog Indicator Function Check: Click the check in the dialog if you are importing individual indicator functions. *Blending Function Sigma:* Choose a sigma for pre-process smoothing either your segmentation labels or indicator functions to avoid stair-step aliasing.

3.2.1 Sizing Field Creator

This tool allows a user to set parameters for the cleaving sizing field.

- *Sampling Rate:* the sampling rate of the input indicator functions or calculated indicator functions from segmentation files. The default sample rate will be the dimensions of the volume. Smaller sampling creates coarser meshes. Adjusting this parameter will also affect Cleaver's runtime, with smaller values running faster.
- *Feature Scaling:* scales features of the mesh effecting element size. Higher feature scaling creates coaser meshes.
- *Lipschitz:* the maximum rate of change of element size throughout a mesh. Helpful for meshes with high and low curvature. Will have no effect on meshes with constant element sizing methods.
- *Padding:* adds a volume buffer around the data. Useful when volumes intersect near the boundary.
- *Element Sizing Method:* select whether to adaptively/nonuniformly resize tetrahedra for more detail at volume interactions, or to keep tetrahedra sizes constant/uniform based on the sample scale.
- *Compute Sizing Field:* once you have your desired parameters, click this to create the sizing field. This is assuming a volume has been loaded (`ctrl+v or File->Import Volume`). New information will be added to the Data Manager at each step. If a sizing field is not created here, a default one will be created for you automatically before cleaving.



3.2.2 Cleaving Tool

This tab runs the cleaving algorithm and displays steps that have completed.

- *Cleave Mesh:* Run the cleaving algorithm. The steps are shown as complete with the check below. The rendering window will also update with each applicable step.

3.2.3 Data Manager

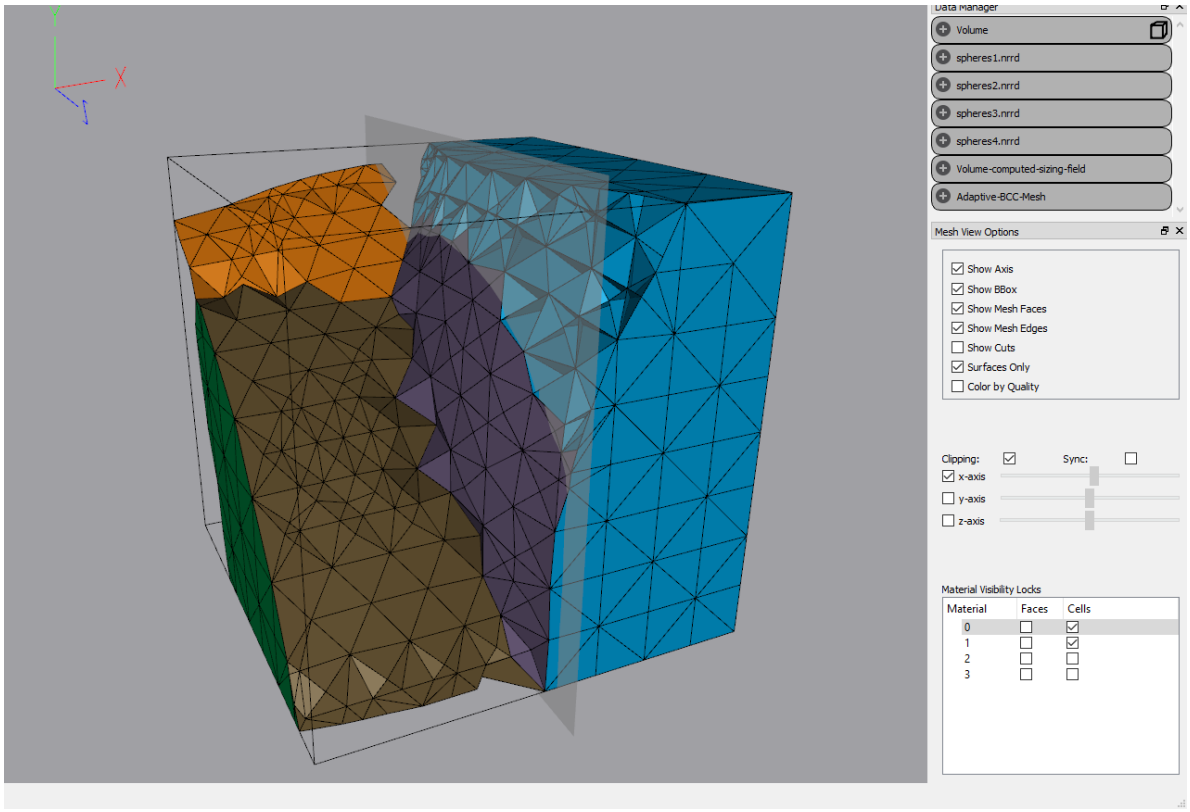
This tool displays information about meshes, volumes, and sizing fields loaded and created.

- *Mesh:* a mesh will have number of vertices, number of tetrahedra, and the min/max mesh boundaries.
- *Volume:* a volume will display the dimensions, origin, number of materials, the file names, and the associated sizing field (if any other than the default).

3.2.4 Mesh View Options

Here are are a number of options for visualizing the generated mesh.

- *Show Axis:* Toggle the rendering of the coordinate axis (x-y-z).
- *Show BBox:* Toggle the rendering of the mesh/volume bounding box.
- *Show Mesh Faces:* Toggle the rendering of the mesh's faces.
- *Show Edges:* Toggle the rendering of the mesh's edges.
- *Show Cuts:* Toggle the rendering of nodes where cuts took place.
- *Show Surfaces Only:* Toggle the rendering of the tets (volume) vs. the surface representing the interface between volumes.
- *Color by Quality:* Toggle the coloring of faces based on the quality of the tet vs. the material itself.



- *Clipping:* Toggle the clipping of tets based on the below sliders.
- *Sync:* When checked, faces will update during slider movement (slower). Otherwise, faces will update once the clipping plane has stopped moving (mouse is released).
- *X-Y-Z Axes:* Select which axis to clip the volume. The associated slider will permit clipping from one end of the bounding box to the other.
- *Material Visibility Locks:* A list of the materials is here. When the faces of a material is locked, clipping is ignored for that material and it is always visible. Locked cells refers to tets/volumes that will remain visible despite the clip.

3.2.5 File Menu

- *Import Volume:* Select 1-10 indicator function NRRDs, or 1 segmentation NRRD (if built in) to load in.
- *Import Sizing Field:* Load a sizing field NRRD to use for a Volume.
- *Import Mesh:* Import a tetgen mesh (*.node/*.ele pair) to visualize.
- *Export Mesh:* Write the current mesh to file in either node/ele (tetgen) format, or VTK format.

3.2.6 Edit Menu

- *Remove External Tets:* Removes tets that were created as padding around the volume.
- *Remove Locked Tets:* Removes tets that were not warped during cleaving.
- *Dihedral Angles:* Computes the min/max Dihedral angles. And displays them in the status bar.

3.2.7 View

Toggle view of the Sizing Field, Cleaving, Data, and Mesh View tools.

3.2.8 Help

Show information and documentation about Cleaver, as well as issue reporting.

3.3 Cleaver Library

To include the cleaver library, you should link to the library build, `libcleaver.a` or `cleaver.lib` and include the following headers in your project:

```
##CMake calls
include_directories(Cleaver/src/lib/cleaver)
target_link_libraries(YOUR_TARGET ${your_libs} Cleaver/build/lib/libcleaver.a)
```

There are other headers for different options, such as converting NRRD files to cleaver indicator functions. You may wish to write your own indicator function creation methods. The basic set of calls are in the following code snippet:

```
#include <cleaver/Cleaver.h>
#include <cleaver/CleaverMesher.h>
...
  //obtain your image fields before this line
  cleaver::Volume *volume = new cleaver::Volume(fields);
  cleaver::CleaverMesher mesher(volume);
  cleaver::AbstractScalarField *sizingField =
    cleaver::SizingFieldCreator::createSizingFieldFromVolume(
         volume,
         (float)(1.0/lipschitz), //defined previously
         (float)sampling_rate,   //defined previously
         (float)feature_scaling, //defined previously
         (int)padding,           //defined previously
         verbose);               //defined previously
  volume->setSizingField(sizingField);
  mesher.setRegular(false);
  bgMesh = mesher.createBackgroundMesh(verbose);
  mesher.buildAdjacency(verbose);
  mesher.sampleVolume(verbose);
  mesher.computeAlphas(verbose);
  mesher.computeInterfaces(verbose);
  mesher.generalizeTets(verbose);
  mesher.snapsAndWarp(verbose);
```

(continues on next page)

(continued from previous page)

```
  mesher.stencilTets(verbose);
  cleaver::TetMesh *mesh = mesher.getTetMesh();
  mesh->writeMesh(output_path + output_name,
    output_format, verbose);
...
```

Look at the `Cleaver/src/cli/mesher/main.cpp` file for more details on how to apply and use the different options of the cleaver library.

3.4 Known Issues

- On larger data sets with a potentially high number of quadruple points (> 3 material fields), some functions are failing to ensure valid tets and meshes, causing bad tets in the final output. This code is being debugged now for a future release.
- The graphics cards documented in *System requirements* are known to not support Cleaver.

CHAPTER

FOUR

BUILDING

4.1 Building ITK

Tip: Consider building ITK only if not already available through the system package manager.

4.1.1 Linux and macOS

Download ITK sources:

```
git clone -b v5.2.0 https://github.com/InsightSoftwareConsortium/ITK $HOME/ITK
```

Configure with:

```
cmake \
  -DBUILD_SHARED_LIBS=FALSE \
  -DBUILD_EXAMPLES=FALSE \
  -DBUILD_TESTING=FALSE \
  -S $HOME/ITK \
  -B $HOME/ITK-build
```

Then build ITK.

```
cmake --build $HOME/ITK-build --config Release --parallel 8
```

4.1.2 Windows

Download ITK sources:

```
git clone -b v5.2.0 https://github.com/InsightSoftwareConsortium/ITK %HOMEPATH%/ITK
```

Open a Visual Studio 64 bit Native Tools Command Prompt.

Configure with:

```
cmake -G "NMake Makefiles" ^
  -DBUILD_SHARED_LIBS=FALSE ^
  -DBUILD_EXAMPLES=FALSE ^
  -DBUILD_TESTING=FALSE ^
```

(continues on next page)

(continued from previous page)

```
  -S %HOMEPATH%/ITK ^
  -B %HOMEPATH%/ITK-build
```

Then build ITK.

```
cmake --build %HOMEPATH%/ITK-build --config Release --parallel 8
```

4.1.3 Getting Help

See https://itk.org/ITKSoftwareGuide/html/

4.2 Overview

Building Cleaver is the process of obtaining a copy of the source code of the project and use tools, such as compilers, project generators and build systems, to create binary libraries and executables.

Cleaver can be compiled from source on Linux platforms (OpenSuSE, Ubuntu etc.), macOS, and Windows.

CMake is a cross-platform build system generator that is used for generating Makefiles, Ninja, Visual Studio or Xcode project files.

The build-system generator provides options to selectively build Cleaver *Command Line Tool* and *Graphical Interface*.

Tip: Users of the Cleaver command line tool or graphical interface may not need to build the project as they can instead download and install pre-built packages as described in the *Getting Started* section.

4.3 Build Environment

Windows: 64-bit version of Visual Studio 2015 or newer.

macOS: macOS 10.12+ using either `Ninja`, `Xcode` or `Unix Makefiles` CMake generator.

Linux: GCC compiler supporting C++11 using either `Ninja` or `Unix Makefiles` CMake generator.

4.4 Build Options

The table below describes some of build options available when configuring Cleaver using CMake.

Option	Description	Default
`BUILD_CLI`	Build Cleaver Command Line Tool (CLI) application	`OFF`
`BUILD_GUI`	Build Cleaver Graphical Interface (GUI) application	`OFF`

Tip: By default, when both `BUILD_CLI` and `BUILD_GUI` are `OFF`, only the *Cleaver Library* is built.

4.5 Dependencies

4.5.1 Tools

- C++11 64-bit compatible compiler
- Git 1.8 or higher
- CMake 3.10.2+

4.5.2 Libraries

	Command Line Tool	Graphical Interface
ITK		
Qt		

Qt:

Qt 5 libraries are required to build the Cleaver Graphical Interface. The libraries may be installed by downloading the Qt universal installer and selecting the Qt 5.15.2 components.

Alternatively, the Qt libraries may be installed through the system package manager or as a least resort by building Qt from source.

ITK:

ITK Insight Toolkit 5.0+ is required. See *Building ITK*.

4.6 Building

Once CMake, Qt, ITK have been installed and/or built:

1. Download the Cleaver sources.
2. Run CMake to configure the project by specifying a build directory, the path to the Cleaver directory containing the `src/CMakeLists.txt` file.
3. Build the project

4.6.1 Linux and macOS

```
git clone https://github.com/SCIInstitute/Cleaver.git $HOME/Cleaver

cmake \
  -DITK_DIR:PATH=$HOME/ITK-build \
  -DQt5_DIR:PATH=/Path/To/Qt5/lib/cmake/Qt5 \
  -DCMAKE_BUILD_TYPE:STRING=Release \
  -DBUILD_CLI:BOOL=ON \
```

(continues on next page)

(continued from previous page)

```
  -DBUILD_GUI:BOOL=ON \
  -S $HOME/Cleaver/src \
  -B $HOME/Cleaver-build

cmake --build $HOME/Cleaver-build --config Release --parallel 8
```

4.6.2 Windows

Open a Visual Studio 64 bit Native Tools Command Prompt.

Follow these commands:

```
git clone https://github.com/SCIInstitute/Cleaver.git %HOMEPATH%/Cleaver

set Qt5_DIR=C:/Path/To/Qt/5.15.2/msvc2019_64/lib/cmake/Qt5

cmake -G "NMake Makefiles" ^
  -DQt5_DIR:PATH="%Qt5_DIR%" ^
  -DITK_DIR:PATH="%HOMEPATH%/ITK-build" ^
  -DCMAKE_BUILD_TYPE:STRING=Release ^
  -DBUILD_CLI:BOOL=ON ^
  -DBUILD_GUI:BOOL=ON ^
  -S %HOMEPATH%/Cleaver/src ^
  -B %HOMEPATH%/Cleaver-build

cmake --build %HOMEPATH%/Cleaver-build --config Release --parallel 8
```

Warning: Be sure to copy the Qt5 DLL files to the Executable directory for the program to run.

```
copy %Qt5_DIR%\..\..\..\bin\Qt5Core.dll %HOMEPATH%\Cleaver-build\bin\
copy %Qt5_DIR%\..\..\..\bin\Qt5Gui.dll %HOMEPATH%\Cleaver-build\bin\
copy %Qt5_DIR%\..\..\..\bin\Qt5OpenGL.dll %HOMEPATH%\Cleaver-build\bin\
copy %Qt5_DIR%\..\..\..\bin\Qt5Widgets.dll %HOMEPATH%\Cleaver-build\bin\
```

4.6.3 All Platforms

Your paths may differ slightly based on your Qt5 and ITK versions and where they are installed/built.

The console version `ccmake`, or GUI version can also be used. You may be prompted to specify your location of the Qt installation. If you installed Qt in the default location, it should find Qt automatically. After configuration is done, generate the make files or project files for your favorite development environment and build.

The Cleaver applications will be built in `Cleaver-build/bin`.

4.7 Testing

The repo comes with a set of regression tests to see if recent changes break expected results.

4.7.1 Linux and macOS

To build the tests, you may set `BUILD_TESTING` to `ON` in using either `ccmake` or when calling CMake:

```
cmake -DBUILD_TESTING=ON $HOME/Cleaver/src
```

4.8 Windows

The gtest library included in the repo needs to be built with forced shared libraries on Windows, so use the following:

```
cmake -DBUILD_TESTING=ON -Dgtest_forced_shared_crt=ON %HOMEPATH%/Cleaver/src
```

Be sure to include all other necessary CMake definitions as annotated above.

CHAPTER

FIVE

API

5.1 C++

Reference documentation for Cleaver C++ classes can be found in the Doxygen Code Reference.