Cleaver

Release v2.4

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

Apr 22, 2022

CONTENTS:

Cleaver is a free 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

INSTALLATION

Check the *Platform Specifications* for system requirements.

Installers are provided for Windows and Mac OS X. Linux users need to build Seg3D from *source*

CHAPTER

TWO

USER DOCUMENTATION

For information on how to use cleaver, checkout the Cleaver *Manual*.

2.1 Installing Cleaver from source

2.1.1 Dependencies

Qt

Qt binaries and packages are available on the Qt website or can be built from source code. Gcc/Clang/MSVC with C++11 support is required.

CMake

CMake versions 2.8 - 3.4 are supported.

ITK

ITK Insight Toolkit (ITK 4.7+ recommended)

2.1.2 Compiling From Source

Once you have obtained a compatible compiler and installed Qt on your system, you need to download and install CMake (http://www.cmake.org) to actually build the software. CMake is a platform independent configuring system that is used for generating Makefiles,Visual Studio project files, or Xcode project files.

Compiling ITK

Configure with:

```
CMAKE_CXX_FLAGS+="-std=c++11"
BUILD_SHARED_LIBS=FALSE
BUILD_EXAMPLES=FALSE
BUILD_TESTING=FALSE
ITKV3_COMPATIBILTY=TRUE
```

Then build ITK.

```
make -j12 all
```

You may need to use the CMake GUI in Windows. It is best to configure with `NMake Makefiles`. Once you have configured and generated, you can build in a command prompt.

```
cd C:\ITK_DIR
mkdir build
cd build
nmake all
```

Compiling Cleaver

Once CMake, Qt, ITK have been installed and/or built, run CMake from your build directory and give a path to the ShapeworksStudio directory containing the master CMakeLists.txt file.

Unix and OSX

```
mkdir Cleaver2/build
cd Cleaver2/build
cmake -D ITK_DIR=Path/To/Your/ITK/build -D QT_DIR=Path/To/Your/Qt5/build -D CMAKE_BUILD_
↪TYPE=Release ../src
make
```

Depending on how you obtained Qt, you may need to specify other Qt directories:

```
-D Qt5Widgets_DIR="Path/To/Qt/5.6/gcc/lib/cmake/Qt5Widgets"
-D Qt5OpenGL_DIR="Path/To/Qt/5.6/gcc/lib/cmake/Qt5OpenGL"
```

Windows

Open a Visual Studio 64 bit Native Tools Command Prompt. Follow these commands:

```
mkdir C:\Path\To\Cleaver2\build
cd C:\Path\To\Cleaver2\build
cmake -G "NMake Makefiles" -DITK_DIR="C:/Path/To/Your/ITK/build" -DQT_DIR="C:/Path/To/
↪Your/Qt5/build" -DCMAKE_BUILD_TYPE=Release ../src
nmake
```

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

```
C:\Qt5_DIR\msvc2015\5.6\bin\Qt5Widgets.dll
C:\Qt5_DIR\msvc2015\5.6\bin\Qt5Core.dll
C:\Qt5_DIR\msvc2015\5.6\bin\Qt5OpenGL.dll
C:\Qt5_DIR\msvc2015\5.6\bin\Qt5Gui.dll
```

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 application will be built in build/bin.

2.1.3 Testing

The repo comes with a set of regression tests to see if recent changes break expected results. To build the tests, you will need to set `BUILD_TESTING` to `ON` in either `ccmake` or when calling CMake:

```
cmake -DBUILD_TESTING=ON ../src
```

2.1.4 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 ../src
```

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

2.2 Cleaver Support

For questions and issues regarding building the software from source, please email our support list cleaver@sci.utah.edu

2.3 Using Cleaver

2.3.1 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
```

(continues on next page)

(continued from previous page)

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

2.3.2 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: ../_static/images/application.png 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.

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.

![Cleaver mesh](_static/images/mesh.png “Cleaver Mesh”]

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.

Data Manager

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

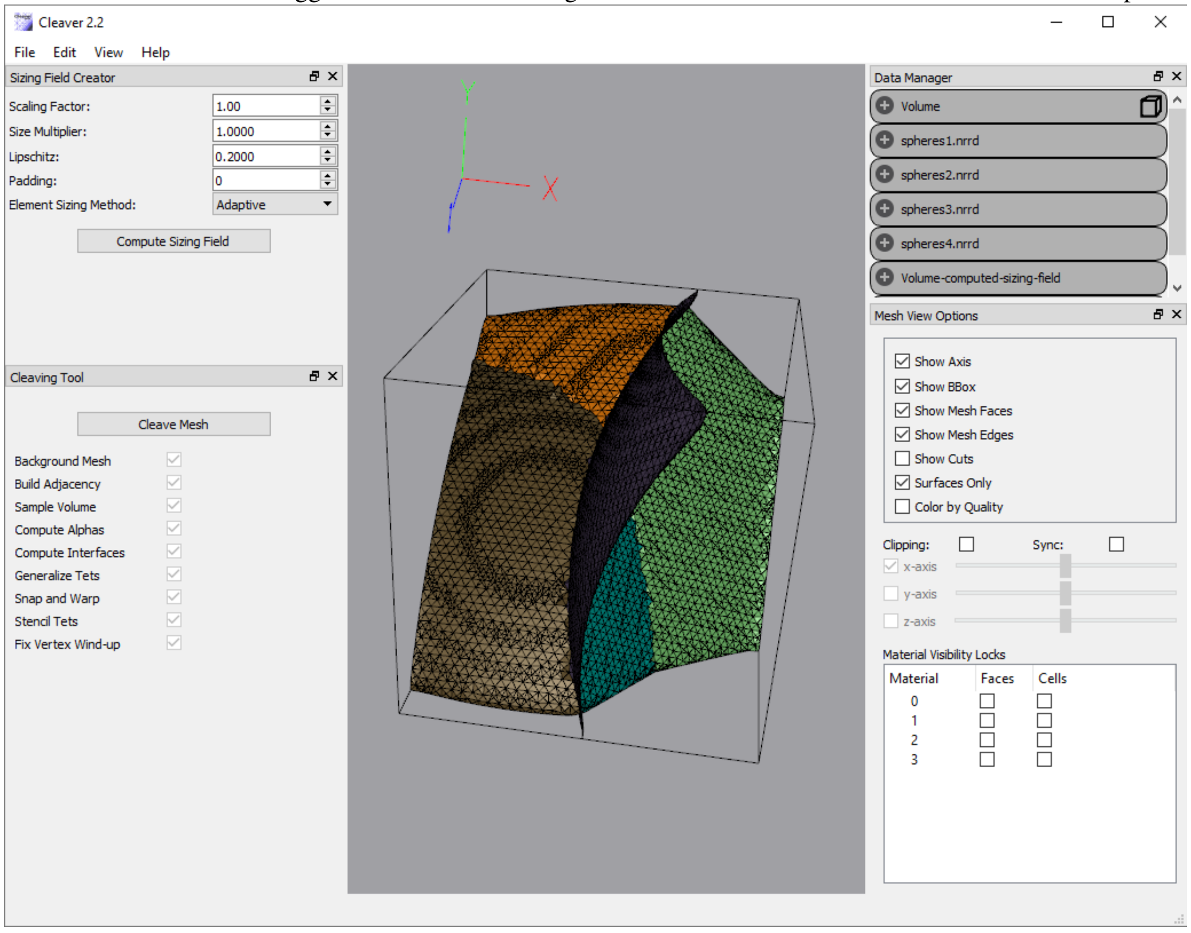
- *Mesh:* a mesh will have number of vertices, number of tetrahedra, and the min/max mesh boundaries.
- *Volume:* a volume will display the dimensions, origin, number of materials, the file names, and the associated sizing field (if any other than the default).

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.



Scaling Factor: 1.00
Size Multiplier: 1.0000
Lipschitz: 0.2000
Padding: 0
Element Sizing Method: Adaptive

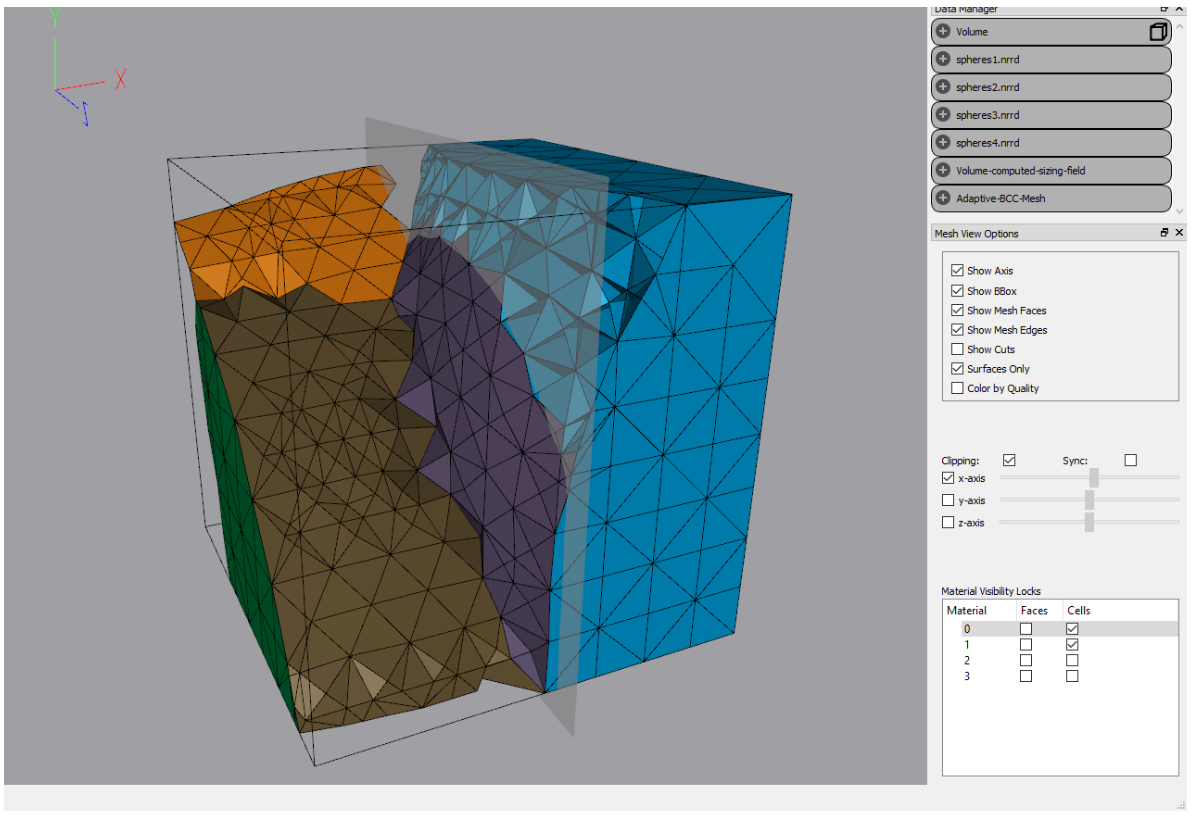
Background Mesh ✓
Build Adjacency ✓
Sample Volume ✓
Compute Alphas ✓
Compute Interfaces ✓
Generalize Tets ✓
Snap and Warp ✓
Stencil Tets ✓
Fix Vertex Wind-up ✓

☑ Show Axis
☑ Show BBox
☑ Show Mesh Faces
☑ Show Mesh Edges
☐ Show Cuts
☑ Surfaces Only
☐ Color by Quality

Clipping: ☐ Sync: ☐
☑ x-axis
☐ y-axis
☐ z-axis

Material	Faces	Cells
0	☐	☐
1	☐	☐
2	☐	☐
3	☐	☐

- *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.



- [✓] Show Axis
- [✓] Show BBox
- [✓] Show Mesh Faces
- [✓] Show Mesh Edges
- [ ] Show Cuts
- [✓] Surfaces Only
- [ ] Color by Quality

Clipping: [✓] Sync: [ ]

- [✓] x-axis
- [ ] y-axis
- [ ] z-axis

Material Visibility Locks

Material	Faces	Cells
0	☐	✓
1	☐	✓
2	☐	☐
3	☐	☐

- *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.

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.

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.

View

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

Help

Show information and documentation about Cleaver2, as well as issue reporting.

2.3.3 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(Cleaver2/src/lib/cleaver)
target_link_libraries(YOUR_TARGET ${your_libs} Cleaver2/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);
  mesher.stencilTets(verbose);
```

(continues on next page)

(continued from previous page)

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

Look at the `Cleaver2/src/cli/mesher/main.cpp` file for more details on how to apply and use the different options of the cleaver library.

2.3.4 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 following graphics cards are known to not support Cleaver:
  - AMD Radeon HD 6310 (Integrated Card)
  - AMD Radeon 7400 M
  - INTEL HD 3000 (Integrated Card)

2.4 Specifications for Cleaver

2.4.1 Minimum recommended system configuration:

- Windows 10+, OSX 10.12+, and OpenSuse 15.1+ Recommended. Other platforms may work, but are not officially supported.
- 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

2.4.2 Windows

The current source code must be compiled with the 64-bit version of Visual Studio 2015.

2.4.3 Mac OS X

The source code base was built with Xcode as well as GNU Make and works for both environments on OS X 10.12+.

2.4.4 Linux specifications

The code base has been tested for use with GCC, and this is the recommended compiler for linux. Compiler must support C++11.

Build from source

Cleaver can be (compiled) from source on Linux platforms (OpenSuSE, Ubuntu etc.), OSX, and Windows. It requires at least the following:

- C++11 64-bit compatible compiler
- Git 1.8 or higher (https://git-scm.com/)
- CMake 3.10.2 + (http://www.cmake.org/)
- Insight Toolkit (ITK 4.7+ recommended) (http://www.itk.org/)
- Qt 5.* (http://www.qt.io/developers/)
- NVIDIA card and drivers for Linux
- Graphics cards must support OpenGL 2.0 or greater (not available on older Intel embedded graphics cards).

Consult the distribution-specific section for additional package information and the developer documentation for build instructions.

OpenSuSE

We are currently testing on 64-bit Leap 42.1 (OpenSuSE package repository information).

OpenSuSE RPMs:

- gcc
- gcc-c++
- Make
- CMake
- git
- glu-devel
- libXmu-devel
- CMake-gui
- CMake

CHAPTER

THREE

CODE DOCUMENTATIONS

Doxygen Code Reference

CHAPTER FOUR

CONTRIBUTORS

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

CHAPTER

FIVE

ACKNOWLEDGEMENTS

Cleaver was supported by grants from the National Institute of General Medicine 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.

CHAPTER

SIX

BIBLIOGRAPHY

CHAPTER SEVEN

INDICES AND TABLES

- genindex
- modindex
- search