 doi:10.20944/preprints202005.0351.v1

EIT-MESHER – Segmented FEM Mesh generation and refinement

Paper Authors
1. Dowrick, Thomas[1];
2. Avery, James[2];
3. Faulkner, Mayo[3];
4. Holder, David[3];
5. Aristovich, Kirill[3]

Paper Author Roles and Affiliations
1. Wellcome/EPSRC Centre for Surgical and Interventional Sciences, University College London, UK
2. Department of Surgery and Cancer, Imperial College London, London, UK
3. Medical Physics and Biomedical Engineering, University College London, UK

Abstract
EIT-MESHER (https://github.com/EIT-team/Mesher) is C++ software, based on the CGAL library, which generates high quality Finite Element Model tetrahedral meshes from binary masks of 3D volume segmentations. Originally developed for biomedical applications in Electrical Impedance Tomography (EIT) to address the need for custom, non-linear refinement in certain areas (e.g. around electrodes), EIT-MESHER can also be used in other fields where custom FEM refinement is required, such as Diffuse Optical Tomography (DOT).

Keywords
Finite Element Models, C++, Electrical Impedance Tomography, Diffuse Optical Tomography, Mesh Generation

Introduction

The EIT-MESHER (https://github.com/EIT-team/Mesher) is a C++ based open source software which generates stable, good quality meshes (Figure 1) for solving the EIT forward solution. The software is based on the CGAL geometry processing kernel (https://www.cgal.org/) and utilises the extended Delaunay triangulation over binary domains in combination with a number of flexible post-triangulation optimisers [1]. In addition, the software performs user-defined mesh refinement, specific to the requirements of Electrical Impedance Tomography. The input to the software is a binary 3D mask (or segmentation) of the desired object, and the output is the fully optimised mesh ready for calculation of the forward problem. The mesh refinement process includes mesh optimisation near the electrodes, planar, gradient, cuboid or spherical sizing field refinement as well as ensuring the mesh quality [2] to be greater

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 May 2020 doi:10.20944/preprints202005.0351.v1

than 0.5 for every element, with an average 0.95 as standard. The combination of the above techniques makes EIT-MESHER it the best available tool for EIT-specific forward problems [3].

Comparable software for FEM generation include the MATLAB package - *iso2mesh* [4], which also includes a wrapper for a CGAL executable. However, at present this does not allow for the custom mesh refinement needed for EIT/DOT applications. Software suites EIDORS [5] for EIT and TOAST++ [6] for DOT both implement basic meshing capabilities, either through the *gmesh* or *tetgen* libraries, however it is not possible to mesh from binary volume data directly.

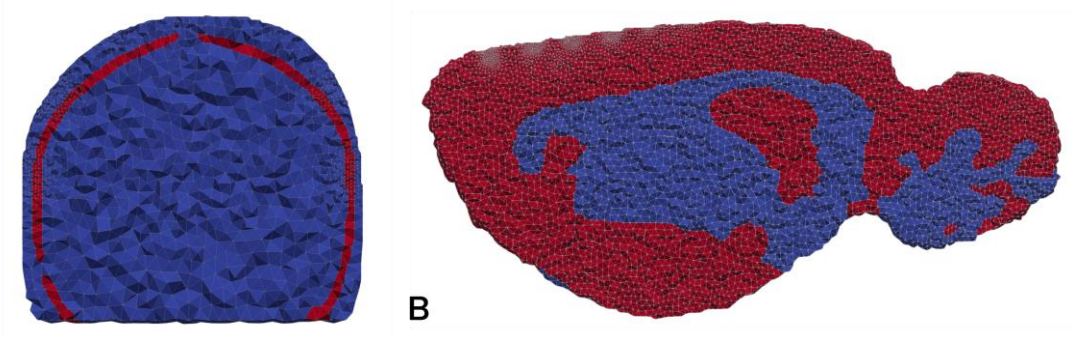

Figure 1 - Example generated meshes for A) neonate skull and B) rat brain with electrode refinement

Implementation and architecture

The main EIT-MESHER source code is organised into several source files in the *src/* directory:

Mesher_plus.cpp - Top level code

input_paremeters.cpp/h – Load input parameters from configuration file.

mesh_operations.cpp/h – Common mesh operations used by other files.

save_dgf.cpp/h – Output mesh in Dune Grid Format (used for EIT forward solver).

Sizing_fields.cpp/h – implements custom mesh refinement (spherical, elliptic, linear, cube)

mesh_stretcher.cpp/h - Implements mesh stretching.

deform_volume.cpp/h – Implements mesh deformation.

The EIT-MESHER executable takes three input files, a 3D binary volume in *.inr* format, a list of x,y,z electrode locations in CSV format, and a parameter file. A basic functional test is given in the *input/* directory, with realistic use cases given in *examples/*, which demonstrate the process of editing the parameter file, and the effects on the subsequent output mesh. Upon calling EIT-MESHER, the input volume is loaded using CGAL Image IO, and an element size value assigned to every point in the volume (the sizing field), as specified

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 May 2020 doi:10.20944/preprints202005.0351.v1

by the parameter file. Delaunay Triangulation, using the *CGAL C3T3* class modified to return additional point data, is then performed to create the initial tetrahedral mesh. Optimisations of the triangulation (*ODT/Llyod/Perturb/Exude*) are then performed to improve mesh quality, particularly in regions with large sizing field gradients. Finally, the output tetrahedral mesh and saved in formats required for EIT or DOT solvers.

To integrate with existing EIT workflows, code for interfacing with EIT-MESHER from MATLAB is contained in the *MATLAB/* folder. This folder also contains functions to convert CAD files into the required *.inr* format, to enable meshing of 3D printable phantoms [7].

Features

Variable mesh refinement

CGAL *Sizing Fields* are used to specify the size of elements within the FEM in different areas. Typically, this is used to specify a smaller element size around the surface electrodes in EIT or the sources and detectors in DOT applications, where the forward solutions are most sensitive. Element size within Regions of Interest (ROIs) internal to the volume can also be specified. These are commonly used either to create high density regions surrounding depth electrodes [8] or low density where the sensitivity is known to be negligible *a priori* [3] . Analytical functions for defining sizing fields are defined in *sizing_fields.cpp*, and the user can select which sizing field(s) to use in the parameter file.

Electrode refinement – Set a separate (usually smaller) element size around the electrodes.

Sphere refinement – a sphere, with specified coordinates, radius and cell size is placed within the mesh.

Cuboid refinement – a cuboid, with specified coordinates, dimensions and cell size is placed within the mesh.

Planar refinement – Create a linear gradient of cell size along a given dimension.

Depth refinement - The mesh size is based on the cartesian distance from the centre of the mesh. Elements at the centre have largest cell size, elements at

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 May 2020 doi:10.20944/preprints202005.0351.v1

the boundary have the smallest. User specifies a linear gradient between the two sizes from the centre.

Mesh deformation/stretching

EIT-MESHER can randomly deform an input volume to generate unique output meshes. This allows for multiple output models to be generated from a single input segmentation. The motivation behind this approach was to provide sufficient variety in input meshes to generate a dataset suitable for deep learning applications of EIT. Due to the mathematical complexity of the EIT problem, 1000s of segmented CT/MRI scans would be required to provide enough training data, which is impractical to collect. By deforming existing meshes, the required number can be obtained from a much smaller input set. The use of HPC resources to rapidly generate multiple meshes has been investigated, with EIT-MESHER successfully deployed on UCL's Myriad HPC service.

Two types of deformation are implemented:
Layer dilation – Where a segmented input has been provided, EIT-MESHER can dilate layer(s) within the mesh.
Mesh stretching – EIT-MESHER can apply one (or more) linear stretches, either along a single axis or some combination of x/y/z directions.

Output data

The output mesh contains the coordinates of each node and each tetrahedral element is assigned both the constituent nodes and a *material index* based on the value in the input volume domain. This index is used to assign layers of the mesh *e.g. scalp, skull, white and grey matter* different conductivity values during subsequent calculations. The primary output format is the *Dune Grid Format (DGF)* which is required for the EIT solver PEITS [9] along with accompanying parameter files. The mesh can also be written to *.vtu* for quick visualisation in *paraview (Kitware Inc. USA)* and to *.csv* files to read into MATLAB for use with EIDORS or TOAST++.

Quality control

Unit tests are included in the *tests/* folder and are built by default when compiling the code. The full range of tests is run on each update to the GitHub repository via a Travis CI script. The *travis.yml* script also runs a series of examples, using data in *examples/*, which are fully documented in the *examples/* folder with sample input and output data.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 22 May 2020 doi:10.20944/preprints202005.0351.v1

This software has been tested in Linux, both natively and under Windows Subsystem for Linux. It has also been deployed on the UCL Myriad HPC service. By running the tests and examples, as detailed in *travis.yml* and in the *examples/* folder, a user will be able to confirm that their build is functioning correctly and understand how to adapt the parameter files to their own application.

Each of the following examples demonstrates how to call EIT-MESHER through the Linux command line and through the relevant MATLAB functions. Examples on how to use the outputs with common forward solvers are also given where relevant.

Unit Cube – A simple demonstration with a known target volume to demonstrate basic functionality

Brain – Rat brain from MRI segmentation with white and grey matter layers created using Seg3D (seg3d.org). This is a typical use case in brain applications, demonstrating *electrode* and *default* refinement [8], [10].

Neonate scalp – A CAD model containing two layers – scalp and skull – each requiring conversion to *.inr* format before use. This is a typical use case to create meshes for 3D printable tanks or phantoms [7].

PEITS – Workflow example showing use of EIT-MESHER *.dgf* output in PEITS EIT forward solver [9].

EIDORS – Common MATLAB EIT solver, requires loading of *.csv* files and storing in specific structure.

TOAST++ – DOT software suite requires mesh in specific format and creation of grid basis function from input *.inr* file

(2) Availability

Operating system
Linux – tested on Ubuntu 14, 16, 18; Red Hat Enterprise Server 7.4 (UCL Myriad HPC); Windows Subsystem for Linux 1 & 2

Programming language
C++ 11
MATLAB Runtime v8.3 or later

 doi:10.20944/preprints202005.0351.v1

Additional system requirements
None

Dependencies
C++: CGAL
MATLAB: iso2mesh and MATLAB Image Processing Toolbox (For *.stl* to *.inr* conversion only)

List of contributors

Software location:
Name: EIT-MESHER
Persistent identifier: *https://github.com/EIT-team/Mesher*
Licence: *BSD 3*
Date published: *31/01/2020*
Code repository: GitHub

Language
C++, MATLAB

(3) Reuse potential
The EIT-MESHER software can be used to create high quality tetrahedral meshes for solving EIT and DOT forward problems. Primarily targeting biomedical applications, the software requires inputs of the form of 3D volume data such as those obtained from CT or MRI segmentations. However, conversion software from CAD format is included, thus enabling a much broader range of applications such as those from the robotics or tactile sensing fields [11]–[13]. The wrapper functions detailed in the provided examples demonstrate how the software can integrate into common EIT (PEITS, EIDORS) and DOT (TOAST++) workflows.

Potential extensions to this software are to defining specific element sizes per input layer, which would enable arbitrary sizing fields to be defined. This would be beneficial in applications with either a combination of depth and surface electrodes, or with an internal volume with much higher conductivity such as bone. Finally, CGAL also permits meshing from polyhedral surfaces, so a logical extension would be to mesh the CAD files directly instead of converting to volume data.

 doi:10.20944/preprints202005.0351.v1

Support for using this software is obtained by raising an issue through GitHub or by contacting the authors via email.

Acknowledgements

Funding statement
This work was supported by an ARCHER eCSE grant (eCSE13-11). James Avery was supported by the NIHR Imperial BRC.

Competing interests
The authors declare that they have no competing interests.

References

[1] B. Gärtner, R. Veltkamp, L. Rineau, and M. Yvinec, “A generic software design for Delaunay refinement meshing,” *Comput. Geom.*, vol. 38, no. 1, pp. 100–110, 2007.

[2] A. Liu and B. Joe, “Relationship between tetrahedron shape measures,” *BIT*, vol. 34, no. 2, pp. 268–287, Jun. 1994.

[3] K. Y. Aristovich, G. S. dos Santos, B. C. Packham, and D. S. Holder, “A method for reconstructing tomographic images of evoked neural activity with electrical impedance tomography using intracranial planar arrays.,” *Physiol. Meas.*, vol. 35, no. 6, pp. 1095–109, Jun. 2014.

[4] Q. Fang and D. A. Boas, “Tetrahedral mesh generation from volumetric binary and grayscale images,” in *Proceedings - 2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro, ISBI 2009*, 2009, pp. 1142–1145.

[5] A. Adler and W. R. B. Lionheart, “Uses and abuses of EIDORS: an extensible software base for EIT.,” *Physiol. Meas.*, vol. 27, no. 5, pp. S25-42, May 2006.

[6] M. Schweiger and S. Arridge, “The Toast++ software suite for forward and inverse modeling in optical tomography,” *J. Biomed. Opt.*, vol. 19, no. 4, p. 040801, 2014.

[7] J. Avery, K. Aristovich, B. Low, and D. Holder, “Reproducible 3D printed head tanks for electrical impedance tomography with realistic shape and conductivity distribution,” *Physiol. Meas.*, vol. 38, no. 6, pp. 1116–1131, Jun. 2017.

[8] A. Witkowska-Wrobel, K. Aristovich, M. Faulkner, J. Avery, and D. Holder, “Feasibility of imaging epileptic seizure onset with EIT and depth electrodes,” *Neuroimage*, vol. 173, no. February, pp. 311–321, Jun. 2018.

 doi:10.20944/preprints202005.0351.v1

[9] M. Jehl, A. Dedner, T. Betcke, K. Aristovich, R. Klofkorn, and D. Holder, "A Fast Parallel Solver for the Forward Problem in Electrical Impedance Tomography," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 1, pp. 126–137, Jan. 2015.

[10] M. Faulkner, S. Hannan, K. Aristovich, J. Avery, and D. Holder, "Feasibility of imaging evoked activity throughout the rat brain using electrical impedance tomography," *Neuroimage*, vol. 178, no. February, pp. 1–10, Sep. 2018.

[11] J. Avery, M. Runciman, A. Darzi, and G. P. Mylonas, "Shape Sensing of Variable Stiffness Soft Robots using Electrical Impedance Tomography," in *IEEE International Conference on Robotics and Automation*, 2019.

[12] D. Silvera-Tawil, D. Rye, M. Soleimani, and M. Velonaki, "Electrical impedance tomography for artificial sensitive robotic skin: A review," *IEEE Sens. J.*, vol. 15, no. 4, pp. 2001–2016, 2015.

[13] E. Coevoet, A. Escande, and C. Duriez, "Optimization-Based Inverse Model of Soft Robots With Contact Handling," *IEEE Robot. Autom. Lett.*, vol. 2, no. 3, pp. 1413–1419, Jul. 2017.