SoftwareX 7 (2018) 304–308

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

pyEIT: A python based framework for Electrical Impedance Tomography

Benyuan Liu, Bin Yang, Canhua Xu, Junying Xia, Meng Dai, Zhenyu Ji, Fusheng You, Xiuzhen Dong, Xuetao Shi, Feng Fu *

Department of Biomedical Engineering, Fourth Military Medical University, Xi'an, 710032, PR China

ARTICLE INFO

Article history:
Received 25 July 2018
Received in revised form 19 September 2018
Accepted 19 September 2018

Keywords:
Electrical Impedance Tomography
Inverse problems
Finite element method
Unstructrual mesh

ABSTRACT

We present a Python-based, open source Electrical Impedance Tomography (EIT) library called pyEIT. It is a multiplatform software released under the Apache License v2.0. pyEIT has a clean architecture and is well documented. It implements state-of-the-art EIT imaging algorithms and is also capable of simple 2D/3D meshing. pyEIT is written in Python. It accelerates the analysis of offline EIT data and can be incorporated into clinical EIT applications. In this paper, we focus on illustrating the fundamental design principles of pyEIT by using some intuitive examples about EIT forward computing and inverse solving.

Code metadata

Current code version	1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX_2018_114
Legal Code License	https://github.com/liubenyuan/pyEIT/blob/master/LICENSE.txt
Code versioning system used	git
Software code languages, tools, and services used	Python and the following Python packages: numpy, scipy, matplotlib, pandas
Compilation requirements, operating environments & dependencies	Linux, Windows, Mac OS
If available Link to developer documentation/manual	https://github.com/liubenyuan/pyEIT/tree/master/examples
Support email for questions	byliu@fmmu.edu.cn

1. Motivation and significance

Electrical impedance tomography (EIT) is a low-cost, non-invasive, radiation-free imaging method [1,2]. It is sensitive to the changes of internal electrical properties, which has potential in bedside monitoring during hospital care. Nowadays, lung EIT [3] and brain EIT [4] are two major clinical research directions. In lung EIT, the ventilation and perfusion distribution in the thorax are imaged and evaluated in real-time [3]. In brain EIT, the pathological intracranial changes, such as haemorrhage [5], ischemia [6] or infarction [7], can be continuously monitored and imaged using EIT. Most of the latest brain EIT researches are limited to phantom models or animal studies. *in-vivo* brain EIT is hard. The size of the skull is large, and the internal structure is complex. Moreover, the high resistivity of the skull [8] and the high conductivity of cerebrospinal fluid [9] create a shielding effect where only a small amount of current applies on the cerebral. But brain EIT is also life-saving. For example, the early identification of cerebral injuries [10] is of great value to clinical surgeons. To advance the development of brain EIT, we need to conduct large-scale 3D finite element (FE) simulations, implement various sophisticated EIT imaging algorithms and process a large amount of *in-vivo* data in a closed loop.

In this paper, we propose a Python-based EIT simulation and imaging framework called `pyEIT`. `pyEIT` ties the backend such as Finite Element Method (FEM) simulation, EIT inverse solving and imaging to the frontend applications. It may accelerate the evolution of *in-vivo* EIT studies.

Recently, we have used EIT *in-vivo* in cerebral imaging and monitoring during total aortic arch replacement [10]. The imaging speed of EIT is one frame per second. The data in [10] contain 42 subjects where the overall length is approximately 160 h. We constructed a pipeline processing where data filtering, meshing,

* Corresponding author.
E-mail address: fengfu@fmmu.edu.cn (F. Fu).

https://doi.org/10.1016/j.softx.2018.09.005

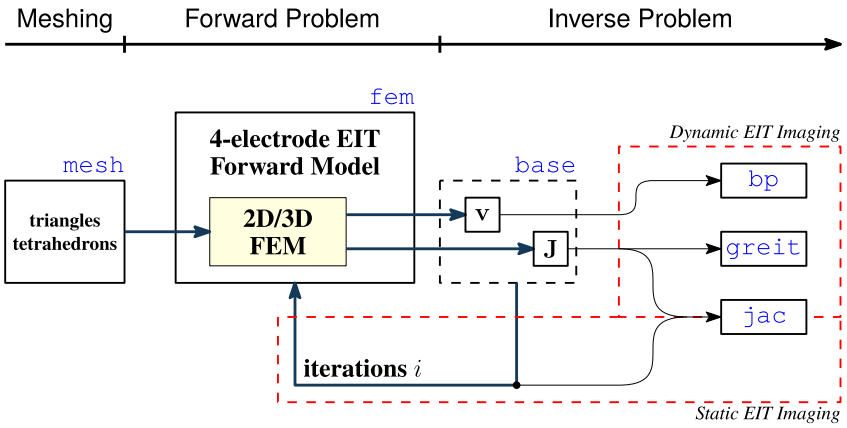

Fig. 1. The software architecture of pyEIT. pyEIT consists of 3 parts: meshing, solving forward and inverse problem. Blue texts denote corresponding Python modules.

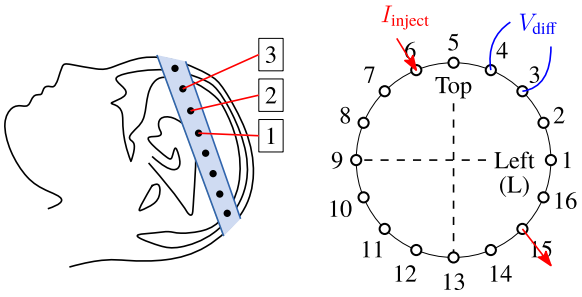

Fig. 2. A 16-electrode configuration EIT system for cerebral imaging.

EIT imaging, image postprocessing, feature extraction and classification are built upon `pyEIT` and other open source machine learning packages.

The `EIDORS` toolkit [11] has been proposed for nearly 20 years and it is widely used for developing and evaluating EIT algorithms. `pyEIT` is less developed compared to `EIDORS`. Some features such as Complete Electrode Model (CEM) and Total Variation (TV) regularization are missing in `pyEIT`. But, `pyEIT` is written in Python and extensible. These features can be added as a plugin module. Furthermore, `EIDORS` is based on MATLAB, which is essentially a functional programming language and has weak Object Oriented Programming (OOP) capability. In clinical EIT studies, most GUIs are written in C++ or Python. The algorithms developed in MATLAB need to be optimized and translated which consumes lots of work. `pyEIT` has clean IO and is suited for rapid prototyping EIT systems and benchmarking EIT reconstruction algorithms.

The architecture of `pyEIT` is introduced in Section 2. Illustrative examples are given in Section 3. The impact of `pyEIT` is highlighted in Section 4.

2. Software description

2.1. Software architecture

The architecture of `pyeit` is given in Fig. 1. The `mesh` module is capable of partitioning Ω into triangles (2D) or tetrahedrons (3D). `pyEIT` wraps around a linear `fem` module. `fem` solves the EIT forward problem using a 4-electrode model, and the intermediate variables such as boundary voltages $\mathbf{v}$ and the Jacobians $\mathbf{J}$ are recorded by the module `base`. `pyEIT` implements state-of-the-art EIT algorithms that support both dynamic EIT imaging (or time-difference imaging) and static EIT imaging.

2.2. Software functionalities

The `fem` module solves the forward problem of EIT. The mathematic model of EIT is formulated as a boundary value problem,

$$\nabla \cdot (\sigma \nabla u) = 0, \qquad \text{in } \Omega$$
$$\sigma \left.\frac{\partial u}{\partial n}\right|_{\partial\Omega} = g$$
$$\int_{\partial\Omega} u = 0$$

where Ω is the 2D or 3D domain to be imaged, and $\partial\Omega$ is the boundary. In EIT, we inject a safe current at a fixed frequency through a pair of electrodes attached to the boundary and measure the voltage differences on remaining electrode pairs. Fig. 2 shows a typical 16-electrode configuration. A frame of data, denoted by $\mathbf{v} \in \mathbb{R}^M$, is formed by rotating and repeating this process iteratively over all 16 electrodes.

EIT imaging is an inverse problem, which reconstructs the conductivities or the changes in conductivities inside the subject from boundary voltages,

$$\sigma = \min_{\sigma,\Omega} \|\mathbf{v} - f(\Omega, \sigma)\|_2^2 + \lambda \|\sigma - \sigma_0\|_2^2 \tag{1}$$

where σ_0 is the initial distribution of the conductivities, a forward operator f maps Ω and σ to boundary voltages $\mathbf{v}$. By assuming a perfect geometry (i.e., boundary shape and electrodes positions are known a priori), the jacobians of σ is computed as,

$$\mathbf{J} = \frac{\partial f(\sigma)}{\partial \sigma}$$

Gauss–Newton method is used to solve (1) iteratively,

$$\boldsymbol{\sigma}^{(k+1)} = \boldsymbol{\sigma}^{(k)} + (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T(\mathbf{v} - f(\boldsymbol{\sigma}^{(k)})) \tag{2}$$

Regularization terms can be incorporated into EIT easily by modifying the norms in (1) accordingly.

The `base` module records the boundary voltages and the Jacobians [12]. All the EIT imaging modules are built upon `base`. Static EIT imaging calculates (2) iteratively. A dynamic EIT imaging algorithm can image the changes of conductivities at two frames. In `pyEIT`, typical dynamic EIT imaging methods such as back projection (`bp`), GREIT [13] and NOSER [14] are implemented.

3. Illustrative examples

3.1. Creating triangle meshes on a unit circle

`pyEIT` reimplements `distmesh` using Python. It also provides a standard layered circle mesh. In the mesh module, `create` and

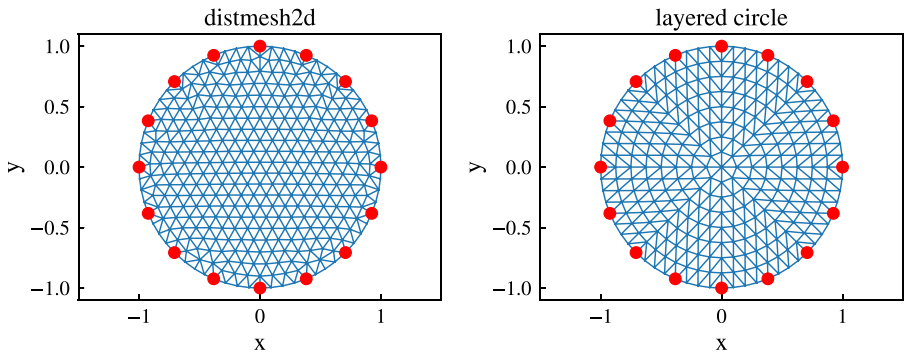

Fig. 3. Triangle meshes on a unit circle using the distmesh2d and a standard layered circle meshing method. The electrodes are highlighted using red nodes.

`layer_circle` are used. These functions return two objects, the first one is the mesh structure, which is a named tuple with `'node'`, `'element'` and `'perm'`. In 2D, `node` is a $N \times 2$ matrix specifies the xy coordinates of nodes, `element` is an $M \times 3$ matrix describes the connectivity structure of the mesh. The second one is `el_pos` which specifies the numbered locations of the electrodes (see Fig. 3).

```
mesh0, el_pos0 = create(n_el=16)
mesh1, el_pos1 = layer_circle(n_el=1, n_fan=8,
 n_layer=8)
xy = mesh0['node']  # Nx2 matrix
t = mesh0['element']  # Mx3 matrix
```

3.2. Solving EIT forward problems

We use the Finite Element Method (FEM) to solve EIT forward problems. In the forward class, we used a simple electrode model where the electrical current is assumed to flow into and out of the imaging area Ω through boundary nodes. Those special boundary nodes are called electrodes. It should be noted that this is a simplification and the complete electrode model (CEM) is preferred. We are planning to add CEM in a future version of `pyEIT`.

A single stimulation pattern is represented by a 2×1 row vector called $\mathbf{e}$. The source and the sink of the electrical current must be specified on two boundary electrodes. For example, $\mathbf{e} = [0, 7]$ means that the electrical current is injected into the 1st electrode and outflowed from the 8th electrode.

`pyEIT` supports multi-frequency EIT analysis where permittivities on Ω can be either real valued or complex valued. The permittivities are not mandatory. The forward solver class is initialized using the unstructured mesh object and the positions of electrodes. The `perms` on elements are initialized using the mesh object. The values of elements can be modified and we provided a method of the mesh class called `set_perm` (see Fig. 4).

```
# setup (AB) electrical current path
ex_line = [0, 7]  # Python index starts from 0
# calculate simulated data using FEM
fwd = Forward(mesh_obj, el_pos)
f, _ = fwd.solve(ex_line, perm=perm)
f = np.real(f)
```

The forward class has a method called `solve_eit`, it requires a parameter called `ex_mat` which thereafter we denoted it by $\mathbf{E}$. $\mathbf{E}$ is a $M \times 2$ matrix, where each row of it represents a stimulation pattern. For example, the ith row of $\mathbf{E}$ is $\mathbf{e}_i = [2, 5]$, which means that in our simple electrode model, the electrical current is injected through the electrode 3 and sank at the electrode 6. `solve_eit` iterates over $\mathbf{e}_i$, expanded it into boundary conditions and then solve the forward problem. The potentials at boundary electrodes are extracted. The voltage differences on two electrodes (specified by the parameter `step`) are computed and rearranged (specified by the parameter `parser`).

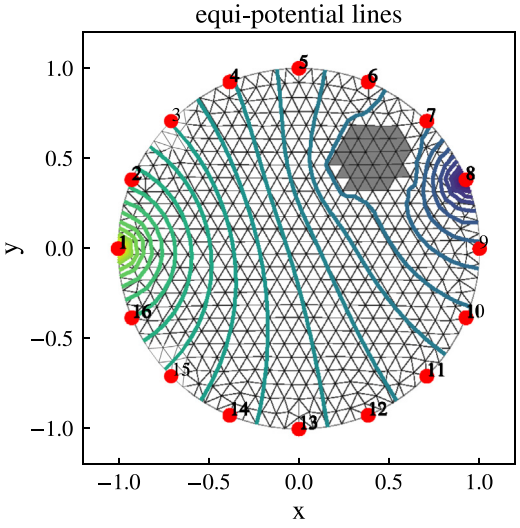

Fig. 4. Solving an EIT forward problem. The current is injected via electrode 1 (the indices of Python start from 0) and sank on electrode 8. Equi-potential lines are drawing in this figure for a better understanding of the distribution of electronic fields.

`pyEIT` allows users to construct any stimulation pattern. It has the flexibility of specifying any two electrodes as either a current source or sink in $\mathbf{e}_i$. The number of rows of $\mathbf{E}$ is not limited. We provide a simple wrapper called `eit_scan_lines` which produces a typical 16×2 stimulation patterns in EIT [2] where the distance from the electrical current source to the current sink is specified by a parameter called `dist`. $\mathbf{E}$ can be stacked, for example, $\mathbf{E} = [\mathbf{E}_1^T, \mathbf{E}_2^T]^T$. In brain EIT applications, we can use fused stimulation patterns. For example,

```
"""
stimulation pattern can be specified as a matrix
ex_mat = [[0, 1],
          [1, 2],
          [1, 7]],
or it can also be stacked using multiple ex_mat
"""
ex_mat1 = eit_scan_lines(16, dist=4)
ex_mat2 = eit_scan_lines(16, dist=7)
ex_mat = np.vstack([ex_mat1, ex_mat2])
p = fwd.solve_eit(ex_mat=ex_mat, parser='fmmu',
 step=1)
```

We reproduce the sensitivity analysis in [2]. In Fig. 5, multiple `'skip k'` stimulation patterns are simulated and the sensitivity of elements are visualized.

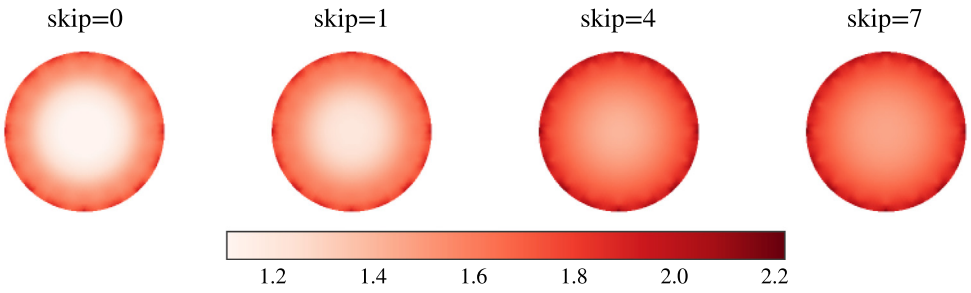

Fig. 5. Sensitivity of different skip-k patterns.

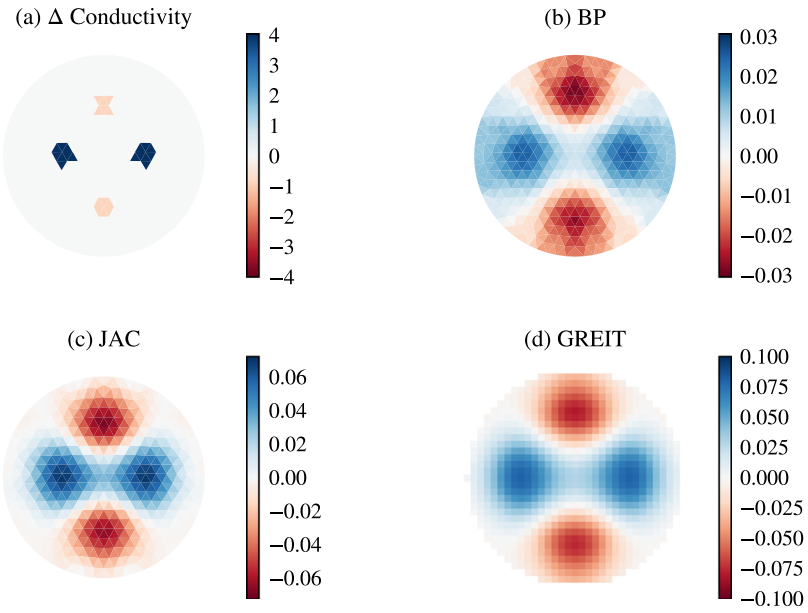

Fig. 6. Solving EIT inverse problems.

3.3. Solving EIT inverse problems

In the following, we present examples of dynamic EIT imaging. Dynamic EIT uses two frames and reconstructs the changes in the conductivities. It is also called time difference EIT imaging (see Fig. 6).

4. Impact

`pyEIT` is the first Python package in the fields of Electrical Impedance Tomography. It is OOP based and has clean architecture. The forward computing and inverse EIT imaging modules are stable. Advanced EIT imaging algorithms that use different regularization penalties can be easily incorporated.

The main impact of `pyEIT` is that it enables rapid prototyping of EIT systems and applications. The frontend GUI of EIT is usually developed using C++ or Python, the backends of these applications can use `pyEIT` directly for meshing, imaging or data pre-processing. Our group designed an EIT GUI using PyQT whose core functionalities are provided by `pyEIT`. Recently, an open source project called `OpenEIT`[1] uses `pyEIT` as its backend. `pyEIT` can also be used in Electrical Impedance Spectroscopy (EIS) and Electrical Capacitance Tomography (ECT) applications.

`pyEIT` accelerates offline EIT data analysis. A pipeline processing can be constructed using the state-of-the-art machine learning and image process Python packages. We have used brain EIT in monitoring the status of the brain non-invasively during cardiovascular surgeries [10]. The data consist approximately 160 hours recordings and are processed using `pyEIT`. We evaluated various image segmentation algorithms. The characteristics of EIT images are then analyzed to postoperative outcomes. A predictive model of early cerebral injuries is built. Then it is incorporated into the final *in-vivo* EIT application, which is beneficial in clinical brain EIT studies.

5. Conclusions

In this paper, we presented a Python-based platform called `pyEIT`. It is well documented and a complete set of examples regarding meshing, 2D/3D EIT forward simulation, dynamic or static EIT imaging are provided. It also has `IO` features that are capable of loading binary data or patient-specific meshes. The APIs of `pyEIT` are versatile and extensible. The members of the EIT community may benefit from `pyEIT` by sharing data and developed EIT algorithms. `pyEIT` is also the getting started point for academic or industry users who are unfamiliar with EIT.

Acknowledgments

This work was in part supported by Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2017JQ8008) and National Natural Science Foundation of China (Grant No. 61571445, 81570231, 81671846).

[1] https://github.com/OpenEIT.

References

[1] Holder DS. Electrical impedance tomography: methods, history and applications. CRC Press; 2004.

[2] Adler A, Boyle A. Electrical impedance tomography: Tissue properties to image measures. IEEE Trans Biomed Eng 2017;64(11):2494–504.

[3] Frerichs I, Amato MB, van Kaam AH, Tingay DG, Zhao Z, Grychtol B, et al. Chest electrical impedance tomography examination, data analysis, terminology, clinical use and recommendations: consensus statement of the translational eit development study group. Thorax 2016;1–11. http://dx.doi.org/10.1136/thoraxjnl-2016-208357.

[4] Fu F, Li B, Dai M, Hu SJ, Li X, Xu CH, et al. Use of electrical impedance tomography to monitor regional cerebral edema during clinical dehydration treatment. PLoS One 2014;9(12). e113202.

[5] Xu C, Wang L, Shi X, You F, Fu F, Liu R, et al. Real-time imaging and detection of intracranial haemorrhage by electrical impedance tomography in a piglet model. J Int Med Res 2010;38(5):1596–604.

[6] Shi X, You F, Fu F, Liu R, You Y, Dai M, et al. Preliminary research on monitoring of cerebral ischemia using electrical impedance tomography technique. In: 2008 30th annual international conference of the IEEE engineering in medicine and biology society; 2008. p. 1188–91 http://dx.doi.org/10.1109/IEMBS.2008.4649375.

[7] Yang B, Shi X, Dai M, Xu C, You F, Fu F, et al. Real-time imaging of cerebral infarction in rabbits using electrical impedance tomography. J Int Med Res 2014;42(1):173–83.

[8] Tang C, You F, Cheng G, Gao D, Fu F, Yang G, et al. Correlation between structure and resistivity variations of the live human skull. IEEE Trans Biomed Eng 2008;55(9):2286–92.

[9] Baumann SB, Wozny DR, Kelly SK, Meno FM. The electrical conductivity of human cerebrospinal fluid at body temperature. IEEE Trans Biomed Eng 1997;44(3):220–3.

[10] Li Y, Zhang D, Liu B, Jin Z, Duan W, Dong X, et al. Noninvasive cerebral imaging and monitoring using electrical impedance tomography during total aortic arch replacement. J Cardiothoracic Vascular Anesthesia 2018 [in press]. http://dx.doi.org/10.1053/j.jvca.2018.05.002.

[11] Adler A, Lionheart WR. Uses and abuses of EIDORS: an extensible software base for EIT. Physiological Measurement 2006;27(5):S25.

[12] Gómez-Laberge C, Adler A. Direct EIT Jacobian calculations for conductivity change and electrode movement. Physiological Measurement 2008;29(6):S89.

[13] Adler A, Arnold JH, Bayford R, Borsic A, Brown B, Dixon P, et al. GREIT: a unified approach to 2D linear EIT reconstruction of lung images. Physiol Measure 2009;30(6):S35.

[14] Cheney M, Isaacson D, Newell JC. Electrical impedance tomography. SIAM Rev 1999;41(1):85–101.