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.

© 2018 Published by Elsevier B.V.

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
2352-7110/© 2018 Published by Elsevier B.V.

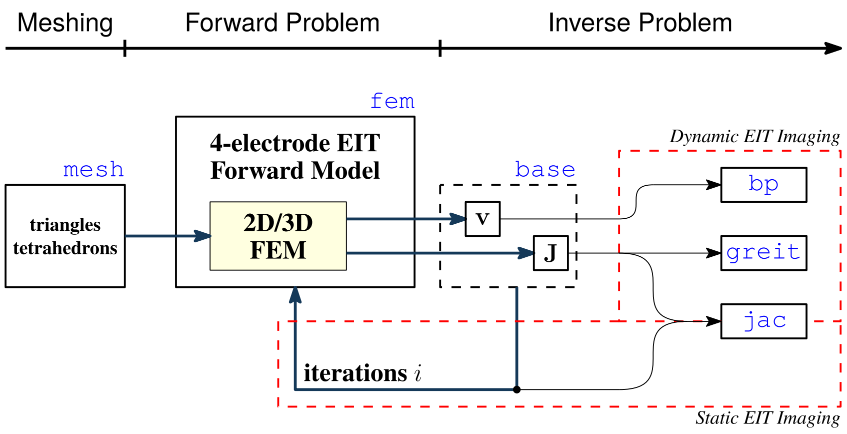

Fig. 1. The software architecture of pyEIT. pyEIT consists of 3 parts: meshing, solving forward and inverse problem. Blue texts denote corresponding Python modules.

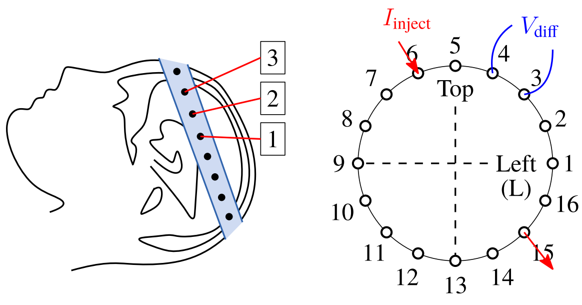

Fig. 2. A 16-electrode configuration EIT system for cerebral imaging.

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