staple Documentation

Release 0.3.2

Fernando Perez-Garcia

Nov 07, 2019

Contents:

CHAPTER 1

STAPLE

Python implementation of the Simultaneous Truth and Performance Level Estimation (STAPLE) algorithm for generating ground truth volumes from a set of binary segmentations.

The STAPLE algorithm is described in S. Warfield, K. Zou, W. Wells, Validation of image segmentation and expert quality with an expectation-maximization algorithm in MICCAI 2002: Fifth International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer-Verlag, Heidelberg, Germany, 2002, pp. 298-306.

1.1 Installation

```
$ pip install staple
```

1.2 Usage

```
$ staple seg_1.nii.gz seg_2.nii.gz seg_3.nii.gz result.nii.gz
```

1.3 Caveats

- The SimpleITK implementation is about 16 times faster for the test images (0.7 s vs 11.8 s). The implementation in this repository is mostly for educational purposes.

- Markov random field (MRF) postprocessing is not implemented (nor is it in the ITK version). If you need STAPLE with MRF, check out Jorge Cardoso's NiftySeg.

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

2.1 Stable release

To install staple, run this command in your terminal:

```
$ pip install staple
```

This is the preferred method to install staple, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for staple can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/fepegar/staple
```

Or download the tarball:

```
$ curl  -OL https://github.com/fepegar/staple/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use staple in a project:

```
import staple
```

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/fepegar/staple/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

4.1.4 Write Documentation

staple could always use more documentation, whether as part of the official staple docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/fepegar/staple/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

4.2 Get Started!

Ready to contribute? Here's how to set up *staple* for local development.

1. Fork the *staple* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/staple.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv staple
$ cd staple/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 staple tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

4.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.7, 3.4, 3.5 and 3.6, and for PyPy. Check https://travis-ci.org/fepegar/staple/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ py.test tests.test_staple
```

4.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 5

Credits

5.1 Development Lead

- Fernando Perez-Garcia <fernando.perezgarcia.17@ucl.ac.uk>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

Indices and tables

- genindex
- modindex
- search