amsaf Documentation

Release 0.1.0

Laura Hallock

Aug 09, 2018

Contents

The HART Lab's tools for registration-based segmentation

Contents:

CHAPTER 1

amsaf

The HART Lab's tools for registration-based segmentation

- Free software: MIT license
- Documentation: https://amsaf.readthedocs.io.

1.1 Features

- Easy interface for segmentation, registration, and parameter map tuning
- A passionate team of university researchers <3

1.2 TODO

- Good tests with Travis CI integration
- Web frontend for common jobs?

CHAPTER 2

Installation

2.1 Stable release

To install amsaf, run this command in your terminal:

```
$ pip install amsaf
```

This is the preferred method to install amsaf, 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 amsaf can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/hart-seg-reg/amsaf
```

Or download the tarball:

```
$ curl  -OL https://github.com/hart-seg-reg/amsaf/tarball/master
```

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

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

CHAPTER 3

Usage

To use amsaf in a project:

```
import amsaf

# image you want to segment
unsegmented_image = ...

# small segmentation slice from unsegmented image which we need in order to
# score each amsaf result
ground_truth = ...

# image which we want to map a segmentation from
segmented_image = ...

# segmentation corresponding to segmented_image
segmentation = ...

# create a generator for amsaf result computations
amsaf_results = amsaf.amsaf_eval(unsegmentd_image, ground_truth, segmented_image,␣
↪segmented)

# evaluate lazy computations, score them, and write them
amsaf.write_top_k(10, amsaf_results, '~/amsaf_results')
```

CHAPTER 4

amsaf

4.1 amsaf package

4.1.1 Subpackages

amsaf.parameter_maps package

Submodules

amsaf.parameter_maps.default module

Module contents

4.1.2 Submodules

4.1.3 amsaf.amsaf module

AMSAF is comprised of several utility functions which wrap SimpleITK and Elastix to facilitate easy registration, transformation, and segmentation of .nii images. It's core functionality, amsaf_eval, allows for quicker development of Elastix parameter maps by generating and ranking the results of parameter map instances in a caller-defined search space.

amsaf.amsaf.**amsaf_eval**(*unsegmented_image*, *ground_truth*, *segmented_image*, *segmentation*, *parameter_priors=None*, *verbose=False*, *memoize=False*)

Main AMSAF functionality

Generate and score new segmentations and corresponding Elastix parameter maps.

Parameters

- **unsegmented_image** (*SimpleITK.Image*) – The target for segmentation and scoring.

- **ground_truth** (*SimpleITK.Image*) – The segmentation slice of unsegmented_image used as a ground truth to score images generated by AMSAF.
- **segmented_image** (*SimpleITK.Image*) – The image we want to map a segmentation from.
- **segmentation** (*SimpleITK.Image*) – The segmentation corresponding to segmented_image.
- **parameter_priors** (*dict*) – An optional vector of 3 ParameterGrid-style dicts mapping Elastix parameter map keys to lists of values. Each value list will be substituted in for the corresponding key in a default dict so that the caller can specify specific combinations of values for some keys, usually to constrain the search space for testing or time consideration.
- **verbose** (*bool*) – Optional boolean flag to toggle verbose stdout printing from Elastix.
- **memoize** (*bool*) – Optional boolean flag to toggle memoized optimization. Warning: experimental

Returns A lazy stream of result (parameter map vector, result segmentation, segmentation score) lists.

Return type generator

amsaf.amsaf.**crop**(*img*, *start*, *end*, *padding=False*)

Crops image along a bounding box specified by start and end

Parameters

- **img** (*SimpleITK.Image*) – Image to be cropped
- **start** (*(int, int, int)*) – Tuple consisting of lower valued coordinates to define bounding box
- **end** (*(int, int, int)*) – Tuple consisting of higher valued coordinates to define bounding box
- **padding** (*bool*) – Optional boolean to specify zero padding

Return type SimpleITK.Image

amsaf.amsaf.**init_affine_transform**(*img*, *transform*, *center=None*)

Initializes an affine transform parameter map for a given image.

The transform fits the following format: T(x) = A(x-c) + c + t

Parameters

- **img** (*SimpleITK.Image*) – Image to be transformed
- **transform** (*numpy.ndarray*) – 4x3 numpy array consisting of affine matrix and a translational vector
- **center** (*(int, int, int)*) – Center of rotation. If none given, geometric center is used

Return type dict

amsaf.amsaf.**read_image**(*path*, *ultrasound_slice=False*)

Load image from filepath as SimpleITK.Image

Parameters

- **path** (*str*) – Path to .nii file containing image.

- **ultrasound_slice** – Optional. If True, image will be cast as sitkUInt16 for ultrasound images.

Returns Image object from path

Return type SimpleITK.Image

amsaf.amsaf.**register**(*fixed_image*, *moving_image*, *parameter_maps=None*, *auto_init=True*, *verbose=False*)

Register images using Elastix.

Parameters

- **parameter_maps** (*[SimpleITK.ParameterMap]*) – Optional vector of 3 parameter maps to be used for registration. If none are provided, a default vector of [rigid, affine, bspline] parameter maps is used.
- **auto_init** (*bool*) – Auto-initialize images. This helps with flexibility when using images with little overlap.
- **verbose** (*bool*) – Flag to toggle stdout printing from Elastix

Returns Tuple of (result_image, transform_parameter_maps)

Return type (SimpleITK.Image, [SimpleITK.ParameterMap])

amsaf.amsaf.**register_indv**(*fixed_image*, *moving_image*, *transform_type*, *parameter_map=None*, *auto_init=True*, *verbose=False*)

Register images using Elastix. Used to perform transforms individually Namely used for memoization to avoid redundant computation

Parameters

- **transform_type** (*String*) – Type of tranform to be performed
- **parameter_map** (*SimpleITK.ParameterMap*) – Optional parameter map to be used for registration. If none is provided, a default map based on transform type is used.
- **auto_init** (*bool*) – Auto-initialize images. This helps with flexibility when using images with little overlap.
- **verbose** (*bool*) – Flag to toggle stdout printing from Elastix

Returns Tuple of (result_image, transform_parameter_maps)

Return type (SimpleITK.Image, [SimpleITK.ParameterMap])

amsaf.amsaf.**seg_map**(*segmented_subject_dir*, *unsegmented_subject_dir*, *segmentation_dir*, *filenames*, *parameter_maps=None*, *strict=False*)

Intra-subject segmentation mappings from supplied filenames

Parameters

- **segmented_subject_dir** – Directory with data of segmented image
- **unsegmented_subject_dir** – Directory with data of unsegmented_image
- **segmentation_dir** – Directory with data of segmented image segmentation
- **filenames** – Iterable of filenames to map
- **parameter_maps** – Optional vector of 3 parameter maps to be used for registration. If none are provided, a default vector of [rigid, affine, bspline] parameter maps is used.
- **strict** – Default False. If True, a ValueError will be raised when some filename is not present in every supplied directory.

Return type [SimpleITK.Image]

```
>>> us_data = os.path.join(os.path.sep, 'srv', 'ultrasound_data')
>>> sub1 = os.path.join(us_data, 'sub1')
>>> sub2 = os.path.join(us_data, 'sub2')
>>> sub1_trials = os.path.join(sub1, 'trials')
>>> sub2_trials = os.path.join(sub2, 'trials')
>>> sub1_seg = os.path.join(sub1, seg)
>>> sub2_hand_shoulder_seg = seg_map(sub1_trials, sub2_trials, sub1_seg, [
↪'trial18_90_fs_volume.mha'])
```

amsaf.amsaf.**seg_map_all**(*segmented_subject_dir*, *unsegmented_subject_dir*, *segmentation_dir*, *parameter_maps=None*, *image_type='volume'*, *strict=False*)

Intra-subject segmentation mappings

Like seg_map, but selects all files of image_type in supplied directories as filename selection.

Parameters

- **segmented_subject_dir** – Directory with data of segmented image
- **unsegmented_subject_dir** – Directory with data of unsegmented_image
- **segmentation_dir** – Directory with data of segmented image segmentation
- **parameter_maps** – Optional vector of 3 parameter maps to be used for registration. If none are provided, a default vector of [rigid, affine, bspline] parameter maps is used.
- **image_type** – Either 'volume' or 'slice' corresponding to extensions '.mha' or '.nii', respectively
- **strict** – Default False. If True, a ValueError will be raised when some filename is not present in every supplied directory.

Return type [SimpleITK.Image]

```
>>> us_data = os.path.join(os.path.sep, 'srv', 'ultrasound_data')
>>> sub1 = os.path.join(us_data, 'sub1')
>>> sub2 = os.path.join(us_data, 'sub2')
>>> sub1_trials = os.path.join(sub1, 'trials')
>>> sub2_trials = os.path.join(sub2, 'trials')
>>> sub1_seg = os.path.join(sub1, seg)
>>> sub2_segs = seg_map_all(sub1_trials, sub2_trials, sub1_seg)
```

amsaf.amsaf.**segment**(*unsegmented_image*, *segmented_image*, *segmentation*, *parameter_maps=None*, *verbose=False*)

Segment image using Elastix

Parameters

- **segmented_image** (*SimpleITK.Image*) – Image with corresponding segmentation passed as the next argument
- **segmentation** (*SimpleITK.Image*) – Segmentation to be mapped from segmented_image to unsegmented_image
- **parameter_maps** (*[SimpleITK.ParameterMap]*) – Optional vector of 3 parameter maps to be used for registration. If none are provided, a default vector of [rigid, affine, bspline] parameter maps is used.
- **verbose** (*bool*) – Flag to toggle stdout printing from Elastix

Returns Segmentation mapped from segmented_image to unsegmented_image

Return type SimpleITK.Image

amsaf.amsaf.**split_x**(*img*, *midpoint_x*, *padding=False*)
Splits image into two separate images along an x-plane Returns both halves of the image, returning the image with lower x values first

Parameters

- **img** (*SimpleITK.Image*) – Image to be split
- **midpoint_x** (*int*) – x value specifying plane to split image along
- **padding** (*bool*) – Optional boolean to specify zero padding

Return type (SimpleITK.Image, SimpleITK.Image)

amsaf.amsaf.**split_y**(*img*, *midpoint_y*, *padding=False*)
Splits image into two separate images along an y-plane Returns both halves of the image, returning the image with lower y values first

Parameters

- **img** (*SimpleITK.Image*) – Image to be split
- **midpoint_y** (*int*) – y value specifying plane to split image along
- **padding** (*bool*) – Optional boolean to specify zero padding

Return type (SimpleITK.Image, SimpleITK.Image)

amsaf.amsaf.**split_z**(*img*, *midpoint_z*, *padding=False*)
Splits image into two separate images along an z-plane Returns both halves of the image, returning the image with lower z values first

Parameters

- **img** (*SimpleITK.Image*) – Image to be split
- **midpoint_z** (*int*) – z value specifying plane to split image along
- **padding** (*bool*) – Optional boolean to specify zero padding

Return type (SimpleITK.Image, SimpleITK.Image)

amsaf.amsaf.**top_k**(*k*, *amsaf_results*)
Get top k results of amsaf_eval

Parameters

- **k** (*int*) – Number of results to return. If k == 0, returns all results
- **amsaf_results** – Results in the format of amsaf_eval return value

Returns Top k result groups ordered by score

Return type [[SimpleITK.ParameterMap, SimpleITK.Image, float]]

amsaf.amsaf.**transform**(*image*, *parameter_maps*, *verbose=False*)
Transform an image according to some vector of parameter maps

Parameters

- **image** (*SimpleITK.Image*) – Image to be transformed
- **parameter_maps** (*[SimpleITK.ParameterMap]*) – Vector of 3 parameter maps used to dictate the image transformation

Returns Transformed image

Return type SimpleITK.Image

amsaf.amsaf.**write_image**(*image*, *path*)
Write an image to file

Parameters

- **image** (*SimpleITK.Image*) – Image to be written
- **path** (*str*) – Destination where image will be written to

Return type None

amsaf.amsaf.**write_result**(*amsaf_result*, *path*)
Write single amsaf_eval result to path

Writes parameter maps, segmentation, and score of AMSAF result as individual files at path.

Parameters

- **amsaf_results** – Results in the format of amsaf_eval return value
- **path** (*str*) – Filepath to write results at

Return type None

amsaf.amsaf.**write_top_k**(*k*, *amsaf_results*, *path*)
Write top k results to filepath

Results are written as subdirectories "result-i" for 0 < i <= k. Each subdirectory contains the result's corresponding parameter maps, segmentation, and score.

Parameters

- **k** (*int*) – Number of results to write. If k == 0, returns all results
- **amsaf_results** – Results in the format of amsaf_eval return value
- **path** (*str*) – Filepath to write results at

Return type None

4.1.4 amsaf.cli module

Console script for amsaf.

4.1.5 Module contents

Top-level package for amsaf.

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/hart-seg-reg/amsaf/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

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

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/hart-seg-reg/amsaf/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.

5.2 Get Started!

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

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

```
$ git clone git@github.com:your_name_here/amsaf.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 amsaf
$ cd amsaf/
$ 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 amsaf 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.

5.3 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. Check https://travis-ci.org/hart-seg-reg/amsaf/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ py.test tests.test_amsaf
```

CHAPTER 6

Credits

6.1 Development Lead

- Laura Hallock <hartsegproject@gmail.com>

6.2 Contributors

- Laura Hallock
- Daniel Ho
- Ian McDonald
- Evan Shu
- Thomas Li
- Neal Sanghvi

CHAPTER 7

History

7.1 0.1.0 (2018-02-03)

- First release on PyPI.

7.2 0.1.1 (2018-02-07)

- Updated with documentation and package structure updates

CHAPTER 8

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index