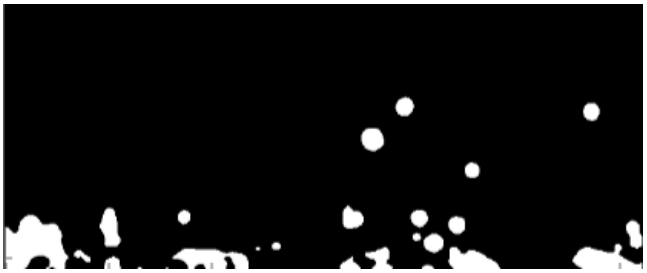
High Speed Image Analysis Documentation

Release 0.0.1

My Affiliation

Jul 17, 2017

Contents



This GitHub repository provides a template to add sphinx / Read The Docs documentation to any python project to generate this Module Docs.

These pages are written using reStructuredText that allows *emphasis*, **strong**, `literal` and many more styles.

You can add a reference *[A1]*, include equations like:

$$V(x) = \left(\frac{1-\eta}{\sigma\sqrt{2\pi}}\right) \cdot exp\left(\frac{x^2}{2\sigma^2}\right) + \eta \cdot \frac{\sigma}{2\pi} \cdot \frac{1}{x^2 + \left(\frac{\sigma}{2}\right)^2}$$

or

$$I_{white} = \int_{E_1}^{E_2} I(\theta, E) \cdot F(E)\, dE.$$

and tables:

Member	Type	Example
first	ordinal	1st
second	ordinal	2nd
third	ordinal	3rd

CHAPTER 1

Features

- List here
- the module features

CHAPTER 2

Contribute

- Documentation: https://github.com/decarlof/hspeed/tree/master/doc
- Issue Tracker: https://github.com/decarlof/hspeed/docs/issues
- Source Code: https://github.com/decarlof/hspeed/hspeed

CHAPTER 3

Content

About

This section describes what the HSpeed project is about.

Install

This section covers the basics of how to download and install Hspeed. We recommend you to install the Anaconda Python distribution.

Contents:

Installing from source

Clone Hispeed from GitHub repository:

```
git clone https://github.com/decarlof/hspeed.git hspeed
```

then:

```
cd hspeed
python setup.py install
```

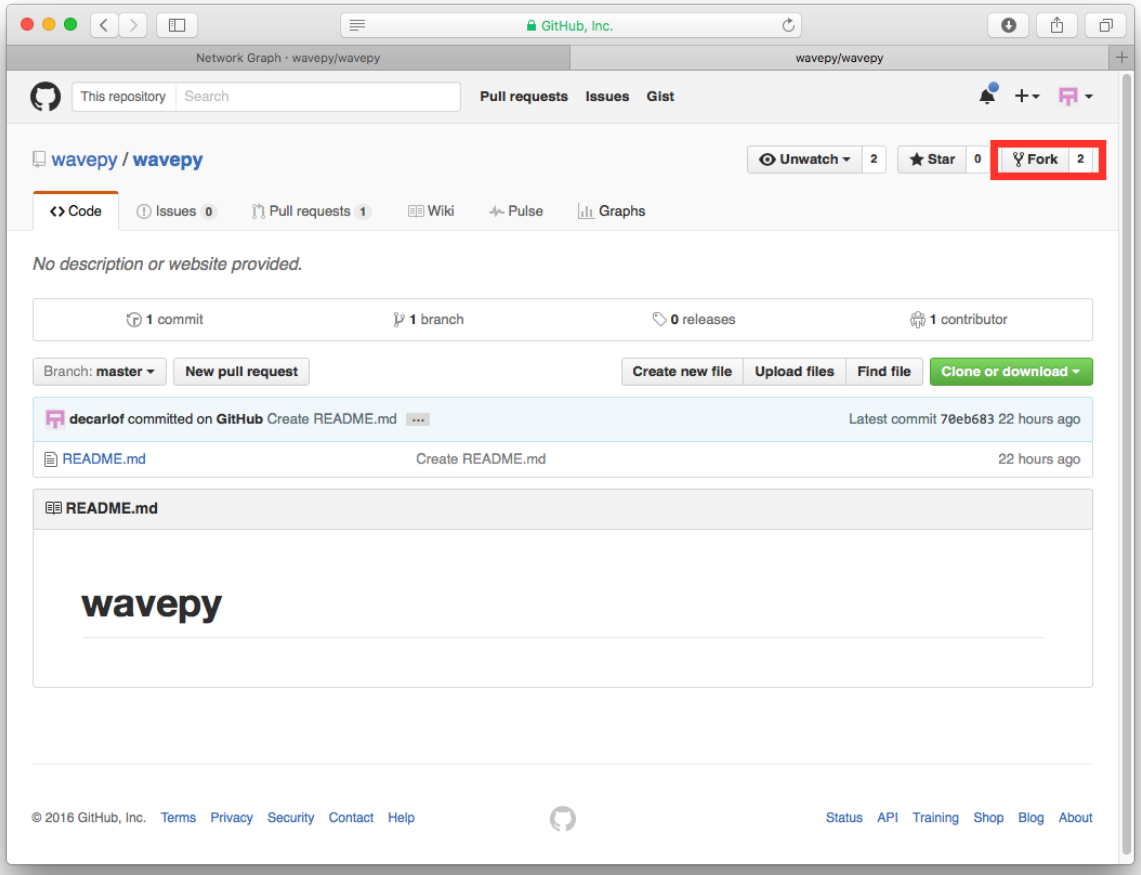
Development

This section explains the basics for developers who wish to contribute to a project mantained on GitHub and using the fork / pull request mechanism for accepting developer contributions.

Contents:

Fork the repository

The project is maintained on GitHub, which is a version control and a collaboration platform for software developers. To start first register on GitHub and fork the Project repository by clicking the **Fork** button in the header of the Project repository:

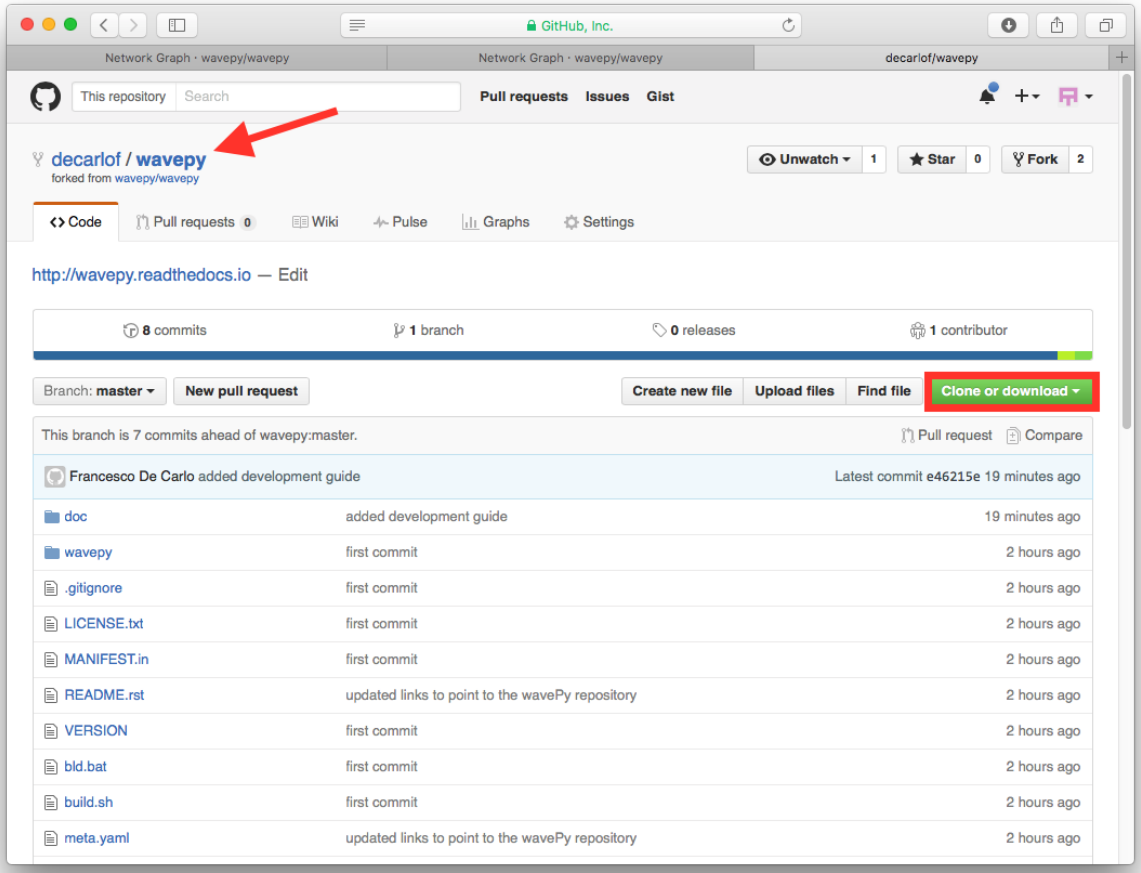


This successfully creates a copy of hspeed in your personal GitHub space.

Clone the repository

The next thing you want to do is to clone the repository you just created in your personal GitHub space to your local machine.

You can do this by clicking the **Clone in Desktop** button in the bottom of the right hand side bar:



This will launch the GitHub desktop application (available for both Mac and Win) and ask you where you want to save it. Select a location in your computer and feel comfortable with making modifications in the code.

Coding conventions

We try to keep a consistent and readable code. So, please keep in mind the following style and syntax guidance before you start coding.

First of all the code should be well documented, easy to understand, and integrate well into the rest of hspeed. For example, when you are writing a new function always describe the purpose and the parameters:

```
def my_awesome_func(a, b):
    """
    Adds two numbers.

    Parameters
    ----------
    a : scalar (float)
        First number to add

    b : scalar (float)
        Second number to add
```

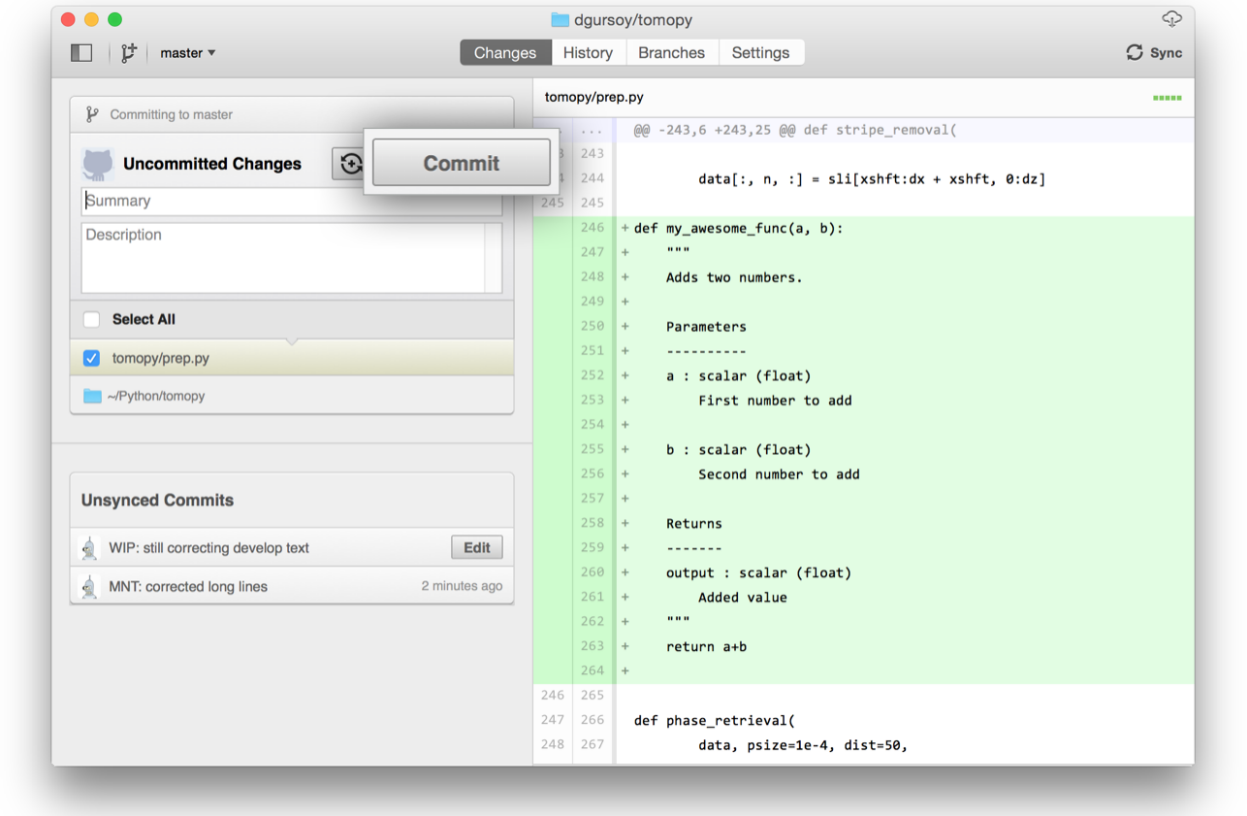
```
    Returns
    -------
    output : scalar (float)
        Added value
    """
    return a+b
```

Package versioning

We follow the X.Y.Z (Major.Minor.Patch) semantic for package versioning. The version should be updated before each pull request accordingly. The patch number is incremented for minor changes and bug fixes which do not change the software's API. The minor version is incremented for releases which add new, but backward-compatible, API features, and the major version is incremented for API changes which are not backward-compatible. For example, software which relies on version 2.1.5 of an API is compatible with version 2.2.3, but not necessarily with 3.2.4.

Commiting changes

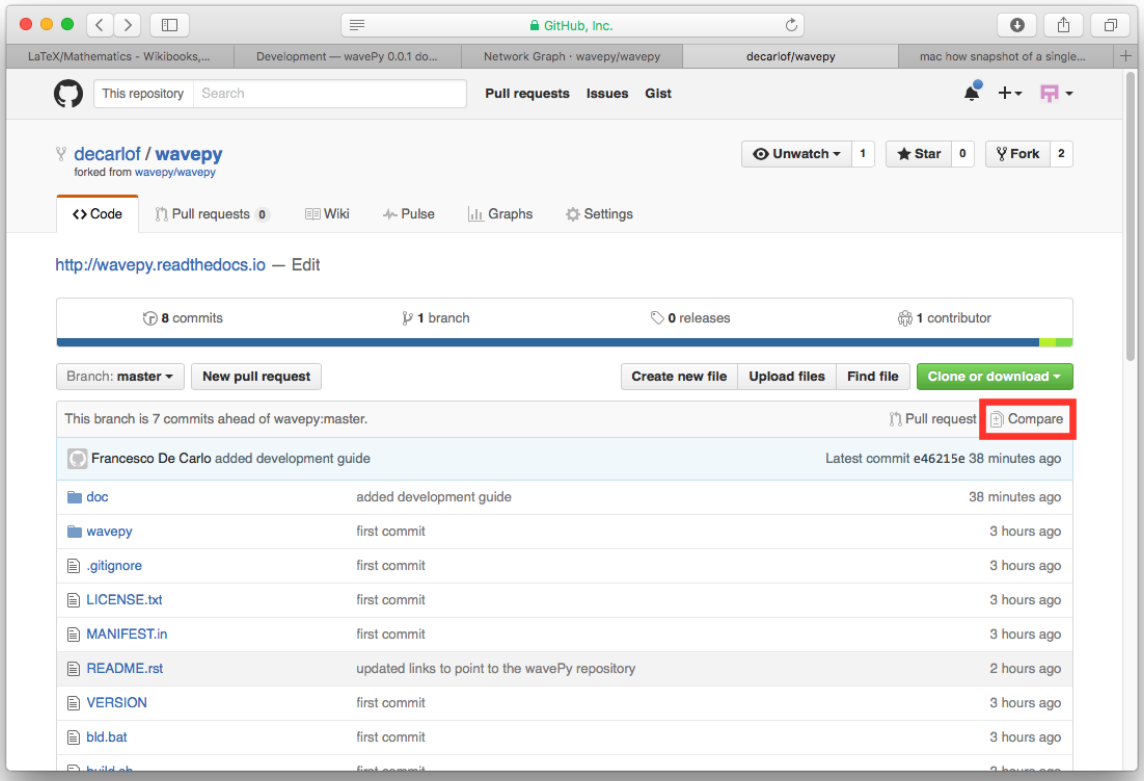
After making some changes in the code, you may want to take a *snapshot* of the edits you made. That's when you make a *commit*. To do this, launch the GitHub desktop application and it should provide you all the changes in your code since your last commit. Write a brief *Summary* and *Description* about the changes you made and click the **Commit** button:



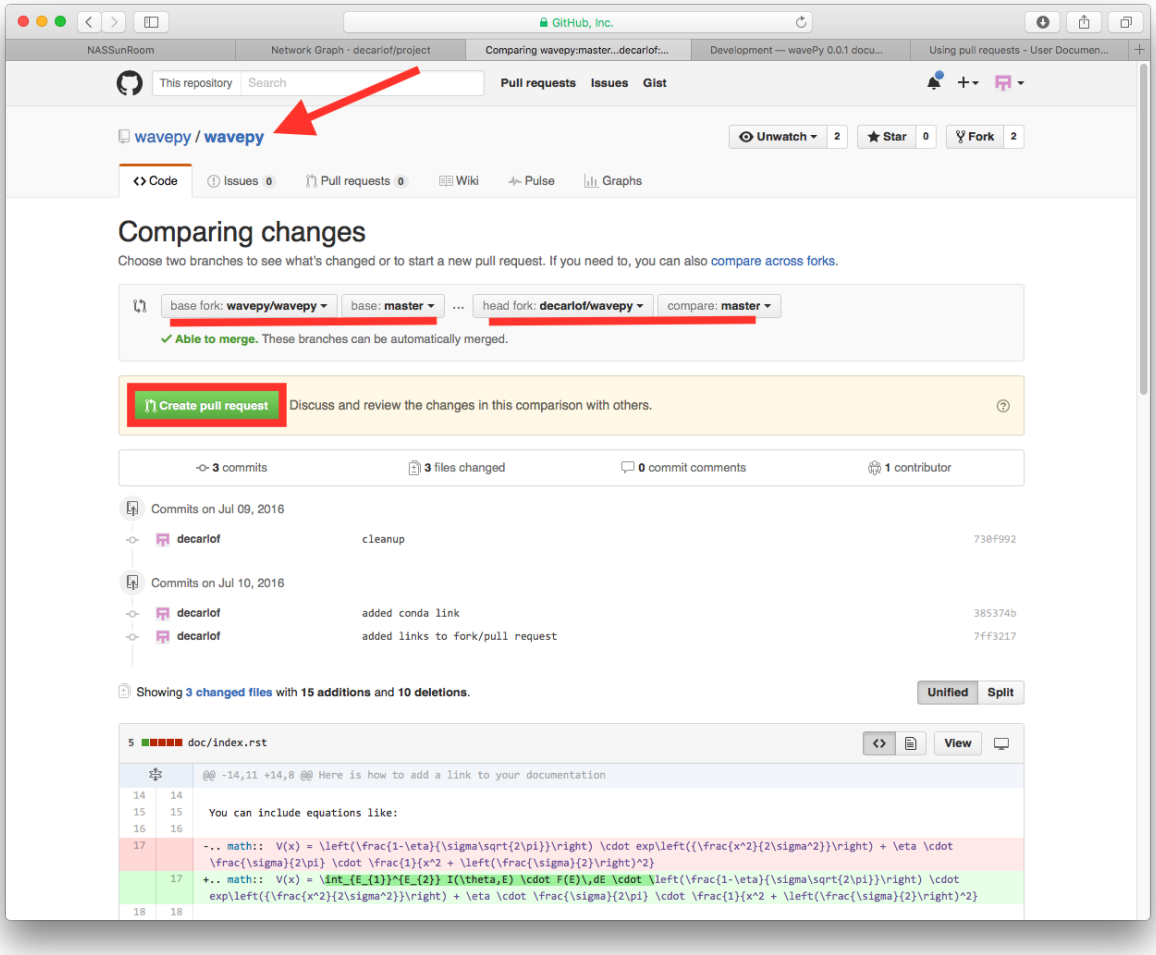
You can continue to make changes, add modules, write your own functions, and take more *Commit snapshots* of your code writing process.

Contributing back

Once you feel that the functionality you added would benefit the community, then you should consider contributing back to hspeed. For this, go to your online GitHub repository of hspeed and click on the *compare* button to compare, review and create a pull request.



After clicking on this button, you are presented with a review page where you can get a high-level overview of what exactly has changed between your forked branch and the original project repository. When you're ready to submit your pull request, click **Create pull request**:



Clicking on **Create pull request** sends you to a discussion page, where you can enter a title and optional description. It's important to provide as much useful information and a rationale for why you're making this Pull Request in the first place.

When you're ready typing out your heartfelt argument, click on **Send pull request**.

You're done!

API reference

hspeed Modules:

`hspeed.util`

Module for describing .....

Functions:

load_raw(top, index_start)	Function description.
shutter_off(image[, alpha, plot])	Function description.
particle_bed_location(image[, plot])	Function description.
laser_on(rdata, particle_bed_ref[, alpha])	Function description.
scale_to_one(ndata)	Function description.
sobel_stack(ndata)	Function description.
label(ndata[, blur_radius, threshold])	Function description.

hspeed.util.**load_raw**(*top*, *index_start*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.util.**shutter_off**(*image*, *alpha=0.7*, *plot=False*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.util.**particle_bed_location**(*image*, *plot=False*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.util.**laser_on**(*rdata*, *particle_bed_ref*, *alpha=1.0*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.util.**scale_to_one**(*ndata*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.util.**sobel_stack**(*ndata*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.util.**label**(*ndata*, *blur_radius=1.0*, *threshold=1*)
Function description.

Parameters

- **parameter_01** (*type*) – Description.
- **parameter_02** (*type*) – Description.
- **parameter_03** (*type*) – Description.

Returns *return_01* – Description.

hspeed.widget

Module for describing .....

Functions:

slider(data)	This is an example of a module level function.

class hspeed.widget.**slider**(*data*)
This is an example of a module level function.

Ref.: http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_numpy.html#example-numpy

Function parameters should be documented in the `Parameters` section. The name of each parameter is required. The type and description of each parameter is optional, but should be included if not obvious.

If *args or **kwargs are accepted, they should be listed as `*args` and `**kwargs`.

The format for a parameter is:

```
name : type
    description

    The description may span multiple lines. Following lines
    should be indented to match the first line of the description.
    The ": type" is optional.
```

```
    Multiple paragraphs are supported in parameter
    descriptions.
```

Parameters

- **parameter_01** (*int*) – The first parameter.
- **parameter_02** (`str`, optional) – The second parameter.
- **parameter_03** (`str`, optional) – The second parameter.
- ***args** – Variable length argument list.
- ****kwargs** – Arbitrary keyword arguments.

Returns

bool – True if successful, False otherwise.

The return type is not optional. The `Returns` section may span multiple lines and paragraphs. Following lines should be indented to match the first line of the description.

The `Returns` section supports any reStructuredText formatting, including literal blocks:

```
{
    'param1': param1,
    'param2': param2
}
```

Raises

- `AttributeError` – The `Raises` section is a list of all exceptions that are relevant to the interface.
- `ValueError` – If *param2* is equal to *param1*.

Examples

Examples should be written in doctest format, and should illustrate how to use the function.

```
>>> print([i for i in example_generator(4)])
[0, 1, 2, 3]
```

More Reference:

http://www.sphinx-doc.org/en/stable/domains.html#python-roles

More examples:

$$X(e^{j\omega}) = x(n)e^{-j\omega n}$$

Warning: Warning text.

Note: Note text.

update(*val*)

Examples

Here we describe what the examples are doing. You can cite with *[B1]*.

Example 01

This section contains the example_01 script.

Download file: `example_01.py`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
Example script
"""

from __future__ import print_function
import hspeed
import os
import sys
import argparse
import fnmatch

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("top", help="top directory where the tiff images are located: /data/")
    parser.add_argument("start", nargs='?', const=1, type=int, default=1, help="index of the first image: 10001 (default 1)")

    args = parser.parse_args()

    top = args.top
    index_start = int(args.start)

    # Read the raw data
    rdata = hspeed.load_raw(top, index_start)

    particle_bed_reference = hspeed.particle_bed_location(rdata[0], plot=False)
    print("Particle bed location: ", particle_bed_reference)

    # Cut the images to remove the particle bed
    cdata = rdata[:, 0:particle_bed_reference, :]

    # Find the image when the shutter starts to close
    dark_index = hspeed.shutter_off(rdata)
    print("Shutter CLOSED on image: ", dark_index)

    # Find the images when the laser is on
    laser_on_index = hspeed.laser_on(rdata, particle_bed_reference, alpha=0.8)
    print("Laser ON on image: ", laser_on_index)

```

```
if __name__ == "__main__":
    main(sys.argv[1:])
```

Example 02

This section contains the example_02 script.

Download file: `example_02.py`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

"""
Example script
"""

from __future__ import print_function

import os
import sys
import argparse
import fnmatch
import numpy as np
import tomopy

import hspeed

def main(arg):

    parser = argparse.ArgumentParser()
    parser.add_argument("top", help="top directory where the tiff images are located: /data/")
    parser.add_argument("start", nargs='?', const=1, type=int, default=1, help="index of the first image: 10001 (default 1)")

    args = parser.parse_args()

    top = args.top
    index_start = int(args.start)

    # Total number of images to read
    nfile = len(fnmatch.filter(os.listdir(top), '*.tif'))

    # Read the raw data
    rdata = hspeed.load_raw(top, index_start)

    particle_bed_reference = hspeed.particle_bed_location(rdata[0], plot=False)
    print("Particle bed location: ", particle_bed_reference)

    # Cut the images to remove the particle bed
    cdata = rdata[:, 0:particle_bed_reference, :]

    # Find the image when the shutter starts to close
    dark_index = hspeed.shutter_off(rdata)
    print("Shutter CLOSED on image: ", dark_index)

    # Find the images when the laser is on
```

```
    laser_on_index = hspeed.laser_on(rdata, particle_bed_reference, alpha=1.00)
    print("Laser ON on image: ", laser_on_index)

    # Set the [start, end] index of the blocked images, flat and dark.
    flat_range = [0, 1]
    data_range = [laser_on_index, dark_index]
    dark_range = [dark_index, nfile]

    flat = cdata[flat_range[0]:flat_range[1], :, :]
    proj = cdata[data_range[0]:data_range[1], :, :]
    dark = np.zeros((dark_range[1]-dark_range[0], proj.shape[1], proj.shape[2]))

    # if you want to use the shutter closed images as dark uncomment this:
    #dark = cdata[dark_range[0]:dark_range[1], :, :]

    ndata = tomopy.normalize(proj, flat, dark)
    ndata = tomopy.normalize_bg(ndata, air=ndata.shape[2]/2.5)
    ndata = tomopy.minus_log(ndata)
    hspeed.slider(ndata)

    ndata = hspeed.scale_to_one(ndata)
    ndata = hspeed.sobel_stack(ndata)
    hspeed.slider(ndata)

    ndata = tomopy.normalize(proj, flat, dark)
    ndata = tomopy.normalize_bg(ndata, air=ndata.shape[2]/2.5)
    ndata = tomopy.minus_log(ndata)

    blur_radius = 3.0
    threshold = .04
    nddata = hspeed.label(ndata, blur_radius, threshold)
    hspeed.slider(ndata)


if __name__ == "__main__":
    main(sys.argv[1:])
```

Credits

Citations

We kindly request that you cite the following article *[A1]* if you use project.

References

Bibliography

[A1] Cang Zhao, Kamel Fezzaa, Ross W. Cunningham, Haidan Wen, Francesco De Carlo, Lianyi Chen, Anthony D. Rollett, and Tao Sun. Real-time monitoring of laser powder bed fusion process using high-speed x-ray imaging and diffraction. *Scientific Reports*, 7(1):3602, 2017. URL: http://dx.doi.org/10.1038/s41598-017-03761-2, doi:10.1038/s41598-017-03761-2.

[B1] Cang Zhao, Kamel Fezzaa, Ross W. Cunningham, Haidan Wen, Francesco De Carlo, Lianyi Chen, Anthony D. Rollett, and Tao Sun. Real-time monitoring of laser powder bed fusion process using high-speed x-ray imaging and diffraction. *Scientific Reports*, 7(1):3602, 2017. URL: http://dx.doi.org/10.1038/s41598-017-03761-2, doi:10.1038/s41598-017-03761-2.

Python Module Index

h

Index