dask-ndfourier Documentation

Release 0.1.1+16.g66b3d00.dirty

John Kirkham

Jun 20, 2017

Contents

Contents:

CHAPTER 1

dask-ndfourier

A library for using N-D Fourier filter with Dask Arrays

- Free software: BSD 3-Clause
- Documentation: https://dask-ndfourier.readthedocs.io.

Features

- TODO

Credits

This package was created with Cookiecutter and the dask-image/dask-image-cookiecutter project template.

CHAPTER 2

Installation

Stable release

To install dask-ndfourier, run this command in your terminal:

```
$ pip install dask-ndfourier
```

This is the preferred method to install dask-ndfourier, 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.

From sources

The sources for dask-ndfourier can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/dask-image/dask-ndfourier
```

Or download the tarball:

```
$ curl  -OL https://github.com/dask-image/dask-ndfourier/tarball/master
```

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

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

CHAPTER 3

Usage

To use dask-ndfourier in a project:

```
import dask_ndfourier
```

CHAPTER 4

API

dask_ndfourier package

dask_ndfourier.**fourier_gaussian**(*input*, *sigma*, *n=-1*, *axis=-1*)

Multi-dimensional Gaussian fourier filter.

The array is multiplied with the fourier transform of a Gaussian kernel.

Parameters

- **input** (*array_like*) – The input array.
- **sigma** (*float or sequence*) – The sigma of the Gaussian kernel. If a float, *sigma* is the same for all axes. If a sequence, *sigma* has to contain one value for each axis.
- **n** (*int, optional*) – If *n* is negative (default), then the input is assumed to be the result of a complex fft. If *n* is larger than or equal to zero, the input is assumed to be the result of a real fft, and *n* gives the length of the array before transformation along the real transform direction.
- **axis** (*int, optional*) – The axis of the real transform.

Returns **fourier_gaussian**

Return type Dask Array

Examples

```
>>> from scipy import ndimage, misc
>>> import numpy.fft
>>> import matplotlib.pyplot as plt
>>> fig, (ax1, ax2) = plt.subplots(1, 2)
>>> plt.gray()  # show the filtered result in grayscale
>>> ascent = misc.ascent()
>>> input_ = numpy.fft.fft2(ascent)
```

```
>>> result = ndimage.fourier_gaussian(input_, sigma=4)
>>> result = numpy.fft.ifft2(result)
>>> ax1.imshow(ascent)
```

dask_ndfourier.**fourier_shift**(*input*, *shift*, *n=-1*, *axis=-1*)

Multi-dimensional fourier shift filter.

The array is multiplied with the fourier transform of a shift operation.

Parameters

- **input** (*array_like*) – The input array.
- **shift** (*float or sequence*) – The size of the box used for filtering. If a float, *shift* is the same for all axes. If a sequence, *shift* has to contain one value for each axis.
- **n** (*int, optional*) – If *n* is negative (default), then the input is assumed to be the result of a complex fft. If *n* is larger than or equal to zero, the input is assumed to be the result of a real fft, and *n* gives the length of the array before transformation along the real transform direction.
- **axis** (*int, optional*) – The axis of the real transform.

Returns **fourier_shift**

Return type Dask Array

Examples

```
>>> from scipy import ndimage, misc
>>> import matplotlib.pyplot as plt
>>> import numpy.fft
>>> fig, (ax1, ax2) = plt.subplots(1, 2)
>>> plt.gray()  # show the filtered result in grayscale
>>> ascent = misc.ascent()
>>> input_ = numpy.fft.fft2(ascent)
>>> result = ndimage.fourier_shift(input_, shift=200)
>>> result = numpy.fft.ifft2(result)
>>> ax1.imshow(ascent)
>>> ax2.imshow(result.real)  # the imaginary part is an artifact
>>> plt.show()
```

dask_ndfourier.**fourier_uniform**(*input*, *size*, *n=-1*, *axis=-1*)

Multi-dimensional uniform fourier filter.

The array is multiplied with the fourier transform of a box of given size.

Parameters

- **input** (*array_like*) – The input array.
- **size** (*float or sequence*) – The size of the box used for filtering. If a float, *size* is the same for all axes. If a sequence, *size* has to contain one value for each axis.
- **n** (*int, optional*) – If *n* is negative (default), then the input is assumed to be the result of a complex fft. If *n* is larger than or equal to zero, the input is assumed to be the result of a real fft, and *n* gives the length of the array before transformation along the real transform direction.
- **axis** (*int, optional*) – The axis of the real transform.

Returns **fourier_uniform** – The filtered input. If *output* is given as a parameter, None is returned.

Return type Dask Array

Examples

```
>>> from scipy import ndimage, misc
>>> import numpy.fft
>>> import matplotlib.pyplot as plt
>>> fig, (ax1, ax2) = plt.subplots(1, 2)
>>> plt.gray()  # show the filtered result in grayscale
>>> ascent = misc.ascent()
>>> input_ = numpy.fft.fft2(ascent)
>>> result = ndimage.fourier_uniform(input_, size=20)
>>> result = numpy.fft.ifft2(result)
>>> ax1.imshow(ascent)
>>> ax2.imshow(result.real)  # the imaginary part is an artifact
>>> plt.show()
```

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/dask-image/dask-ndfourier/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/dask-image/dask-ndfourier/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 :)

Get Started!

Ready to contribute? Here's how to set up *dask-ndfourier* for local development.

1. Fork the *dask-ndfourier* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/dask-ndfourier.git
```

3. Install your local copy into an environment. Assuming you have conda installed, this is how you set up your fork for local development (on Windows drop *source*). Replace *"<some version>"* with the Python version used for testing.:

```
$ conda create -n dask-ndfourierenv python="<some version>"
$ source activate dask-ndfourierenv
$ 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:

```
$ flake8 dask_ndfourier tests
$ python setup.py test or py.test
```

   To get flake8, just conda install it into your environment.

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.

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. Check https://travis-ci.org/dask-image/dask-ndfourier/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests/test_core.py
```

CHAPTER 6

Credits

Development Lead

- John Kirkham, Howard Hughes Medical Institute <kirkhamj@janelia.hhmi.org>

Contributors

None yet. Why not be the first?

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

Index

D

F