astrocut Documentation

Release 0.3

MAST Archive Developers

Jun 21, 2019

CONTENTS

Tools for making image cutouts from sets of TESS full frame images.

This package is under active development, and will ultimately grow to encompass a range of cutout activities relevant to images from many missions, however at this time it is focussed on the specific problem of creating Target Pixel File cutouts from sectors of TESS full frame images.

Astrocut lives on GitHub at: github.com/spacetelescope/astrocut.

Part I

Documentation

INSTALLATION

1.1 Requirements

Astrocut has the following requirements:

- [Astropy](#)
- [NumPy](#)

1.2 Installing astrocut

1.2.1 Using pip

The easiest way to install Astrocut is using pip:

```
pip install astrocut
```

1.2.2 From source

To install the bleading edge version from github without donloading, run the following command:

```
pip git+https://github.com/spacetelescope/astrocut.git
```

The latest development version of astrocut can be cloned from github using this command:

```
git clone https://github.com/spacetelescope/astrocut.git
```

To install astrocut (from the root of the source tree):

```
python setup.py install
```

ASTROCUT DOCUMENTATION

2.1 Introduction

Astrocut contains tools for creating image cutouts from sets images with shared footprints. This package is under active development, and will ultimately grow to encompass a range of cutout activities relevant to images from many missions, however at this time it is focused on the specific problem of creating image cutouts from sectors of TESS full frame images (FFIs).

There are two parts to this package, the `CubeFactory` class allows you to create a large image cube from a list of FFI files. This is what allows the cutout operation to be performed efficiently. The `CutoutFactory` class performs the actual cutout and builds a target pixel file (TPF) that is compatible with TESS pipeline TPFs.

2.2 Getting Started

The basic workflow is to first create an image cube from individual FFI files (this is one-time work), and then make individual cutout TPFs from this large cube file. If you are doing a small number of cutouts, it may make sense for you to use our tesscut web service: mast.stsci.edu/tesscut

2.2.1 Making image cubes

Making an image cube is a simple operation, but comes with a vert important limitation:

> **Warning: Memory Requirements**
>
> The entire cube file must be able to fit in your computer's memory!
>
> For a sector of TESS FFI images from a single camera/chip combination this is ~50 GB.

This operation can also take some time to run. For the 1348 FFI images of the TESS ete-6 simulated sector, it takes about 12 minutes to run on a computer with 65 GB of memory.

By default *make_cube* runs in verbose mode and prints out it's progress, however setting verbose to false will silence all output.

```
>>> from astrocut import CubeFactory
>>> from glob import glob
>>> from astropy.io import fits
>>>
>>> my_cuber = CubeFactory()
```

```
>>> input_files = glob("data/*ffic.fits")
>>>
>>> cube_file = my_cuber.make_cube(input_files)
Completed file 0
Completed file 1
Completed file 2
.
.
.
Completed file 142
Completed file 143
Total time elapsed: 46.42 sec
File write time: 8.82 sec

>>> print(cube_file)
img-cube.fits

>>> cube_hdu = fits.open(cube_file)
>>> cube_hdu.info()
Filename: img-cube.fits
No.    Name      Ver    Type      Cards   Dimensions   Format
0  PRIMARY       1 PrimaryHDU      28   ()
1                1 ImageHDU         9   (2, 144, 2136, 2078)   float32
2                1 BinTableHDU    302   144R x 147C   [24A, J, J, J, J, J, J, D, 24A, J, 24A, 24A, J, J,
→ D, 24A, 24A, 24A, J, D, 24A, D, D, D, D, 24A, 24A, D, D, D, D, D, 24A, D, D, D, D, J, D, D, D, D, D,
→D, D, D, D, D, D, D, J, J, D, J, J, J, J, J, J, J, J, J, J, D, J, J, J, J, J, J, D, J, J, J, J, J, J,
→D, J, J, J, J, J, J, D, J, J, J, J, J, J, J, J, 24A, D, J, 24A, 24A, D, D, D, D, D, D, D, D, J, J, D,
→D, D, D, D, D, J, J, D, D, D, D, D, D, D, D, D, D, 24A, J, 24A, 24A, J, J, D, 24A, 24A, J, J, D,
→ D, D, D, J, 24A, 24A, 24A]
```

2.2.2 Making cutout target pixel files

To make a cutout, you must already have an image cube to cut out from. Assuming that that step has been completed, you simply give the central coordinate and cutout size (in either pixels or angular `Quanitity`) to the *cube_cut* function.

You can either specify a target pixel file name, or it will be built as: "<cube_file_base>_<ra>_<dec>_<cutout_size>_astrocut.fits". You can optionally also specify a output path, the directory in which the target pixel file will be saved, if unspecified it defaults to the current directory.

```
>>> from astrocut import CutoutFactory
>>> from astropy.io import fits
>>>
>>> my_cutter = CutoutFactory()
>>> cube_file = "img-cube.fits"
>>>
>>> cutout_file = my_cutter.cube_cut(cube_file, "251.51 32.36", 5, verbose=True)
Cutout center coordinate: 251.51,32.36
xmin,xmax: [26 31]
ymin,ymax: [149 154]
Image cutout cube shape: (144, 5, 5)
Uncertainty cutout cube shape: (144, 5, 5)
Target pixel file: img_251.51_32.36_5x5_astrocut.fits
Write time: 0.016 sec
Total time: 0.18 sec
```

```
>>> cutout_hdu = fits.open(cutout_file)
>>> cutout_hdu.info()
Filename: img_251.51_32.36_5x5_astrocut.fits
No.    Name       Ver    Type      Cards   Dimensions   Format
0  PRIMARY        1 PrimaryHDU      42   ()
1  PIXELS         1 BinTableHDU    222   144R x 12C   [D, E, J, 25J, 25E, 25E, 25E, 25E, J, E, E, 38A]
2  APERTURE       1 ImageHDU        45   (5, 5)   float64
```

2.3 astrocut Package

2.3.1 Classes

CubeFactory	Class for creating image cubes.
CutoutFactory()	Class for creating image cutouts.

CubeFactory

class astrocut.**CubeFactory**

> Bases: object
>
> Class for creating image cubes.
>
> This class emcompasses all of the cube making functionality. In the current version this means creating image cubes fits files from TESS full frame image sets. Future versions will include more generalized cubing functionality.

Methods Summary

make_cube(self, file_list[, cube_file, . . .])	Turns a list of fits image files into one large data-cube.

Methods Documentation

make_cube(*self*, *file_list*, *cube_file='img-cube.fits'*, *sector=None*, *verbose=True*)

> Turns a list of fits image files into one large data-cube. Input images must all have the same footprint and resolution. The resulting datacube is transposed for quicker cutouts. This function can take some time to run and requires enough memory to hold the entire cube in memory. (For full TESS sectors this is about 40 GB)
>
> > **Parameters**
> >
> > > **file_list**
> > > [array] The list of fits image files to cube. Assumed to have the format of a TESS FFI: - A primary HDU consisting only of a primary header - An image HDU containing the image - A second image HDU containing the uncertainty image
> > >
> > > **cube_file**
> > > [string] Optional. The filename/path to save the output cube in.

sector
[int] Optional. TESS sector to add as header keyword (not present in FFI files).

verbose
[bool] Optional. If true intermediate information is printed.

Returns

response: string or None
If successful, returns the path to the cube fits file, if unsuccessful returns None.

CutoutFactory

class astrocut.**CutoutFactory**
Bases: `object`

Class for creating image cutouts.

This class emcompasses all of the cutout functionality. In the current version this means creating cutout target pixel files from TESS full frame image cubes. Future versions will include more generalized cutout functionality.

Initiazation function.

Methods Summary

cube_cut(self, cube_file, coordinates, . . .)	Takes a cube file (as created by CubeFactory), and makes a cutout target pixel file of the given size around the given coordinates.

Methods Documentation

cube_cut(*self*, *cube_file*, *coordinates*, *cutout_size*, *target_pixel_file=None*, *output_path='.'*, *verbose=False*)
Takes a cube file (as created by CubeFactory), and makes a cutout target pixel file of the given size around the given coordinates. The target pixel file is formatted like a TESS pipeline target pixel file.

Parameters

cube_file
[str] The cube file containing all the images to be cutout. Must be in the format returned by ~astrocut.make_cube.

coordinates
[str or `astropy.coordinates.SkyCoord` object] The position around which to cutout. It may be specified as a string ("ra dec" in degrees) or as the appropriate SkyCoord object.

cutout_size
[int, array-like, Quantity] The size of the cutout array. If cutout_size is a scalar number or a scalar Quantity, then a square cutout of cutout_size will be created. If cutout_size has two elements, they should be in (ny, nx) order. Scalar numbers in cutout_size are assumed to be in units of pixels. Quantity objects must be in pixel or angular units.

target_pixel_file
[str] Optional. The name for the output target pixel file. If no name is supplied, the file

will be named: `<cube_file_base>_<ra>_<dec>_<cutout_size>_astrocut.fits`

output_path
> [str] Optional. The path where the output file is saved. The current directory is default.

verbose
> [bool] Optional. If true intermediate information is printed.

Returns

> **response: string or None**
> > If successfull, returns the path to the target pixel file, if unsuccessful returns None.

LICENSE

3.1 AstroCut License

AstroCut is licensed under a 3-clause BSD style license:

Copyright (c) 2018, MAST Archive Developers All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

- Neither the name of the Astropy Team nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

a

A

C

M