



# The new **cleanest** code in Python

**N. Cardiel**

Collaborators: M. Chillarón, C. Cabello, S. Pascual, J. Gallego, M. Ceballos



Grant PID2022-138621NB-I00 funded by MCIN/AEI/10.13039/501100011033/FEDER,EU



# The old **cleanest** (in FORTRAN 77)

<https://cleanest.readthedocs.io/en/latest/index.html>

cleanest

latest

Search docs

License agreement

Installation

Using the program

The best way to build AI-powered apps

GenAI apps + MongoDB Atlas You don't need a separate database to start building GenAI-powered apps.

Ad by EthicalAds

Welcome to cleanest's documentation!

Edit on GitHub

## Welcome to cleanest's documentation!

cleanest is a Fortran 77 program specially devoted to the semiautomatic removal of cosmic rays in astronomical images.

This program was initially developed as part of the REDUCEME package. Due to historical reasons, REDUCEME images are not stored as FITS files. For that reason, and in order to facilitate the use of cleanest without the need to convert the images to/from REDUCEME format to FITS, cleanest has been detached from the REDUCEME package and converted in a stand-alone program.

This software was created by Nicolás Cardiel as part of his thesis work, developed under the supervision of Javier Gorgas, at the Departamento de Astrofísica de the Universidad Complutense de Madrid. If you use this program, please cite Cardiel (2020).

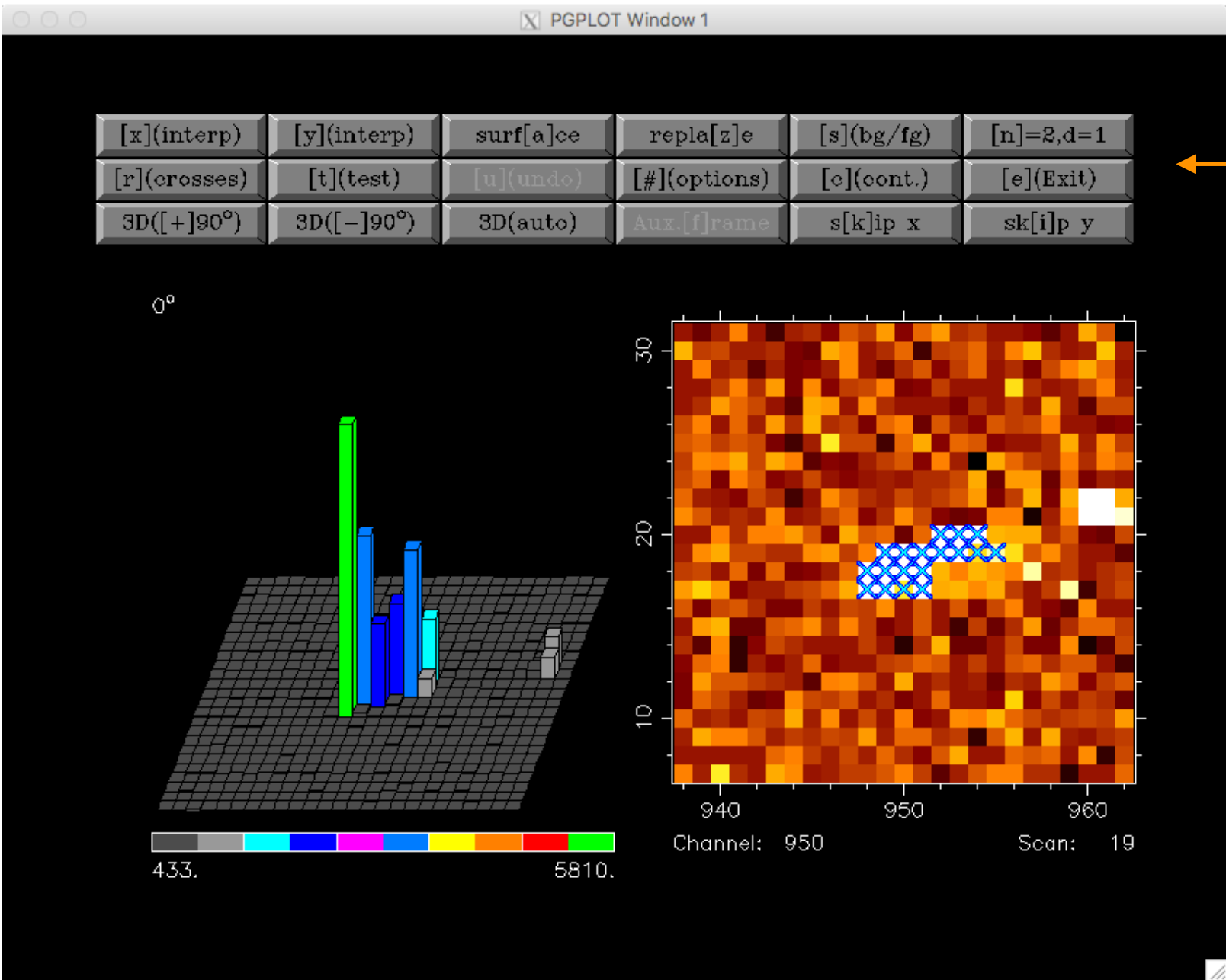
### Documentation outline

- License agreement
- Installation
  - Requirements
  - Installation of cleanest
- Using the program
  - General description
  - Execution example
    - Option `[l]ook` in main menu
    - Option `[s]tart` in main menu
    - Option `[r]egion` in main menu
    - Option `[w]indow` in main menu
    - Option `[a]utomatic` in main window
    - Option `[o]ptions` in main menu
    - Option `sa[v]e` in main menu
    - Option `histo-->[1]` in main menu
    - Option `histo-->[2]` in main menu
    - Option `[t]op1000` in main menu
    - Option `plotsp3[d]` in main menu
    - Option `[q]uit` in main menu

Next

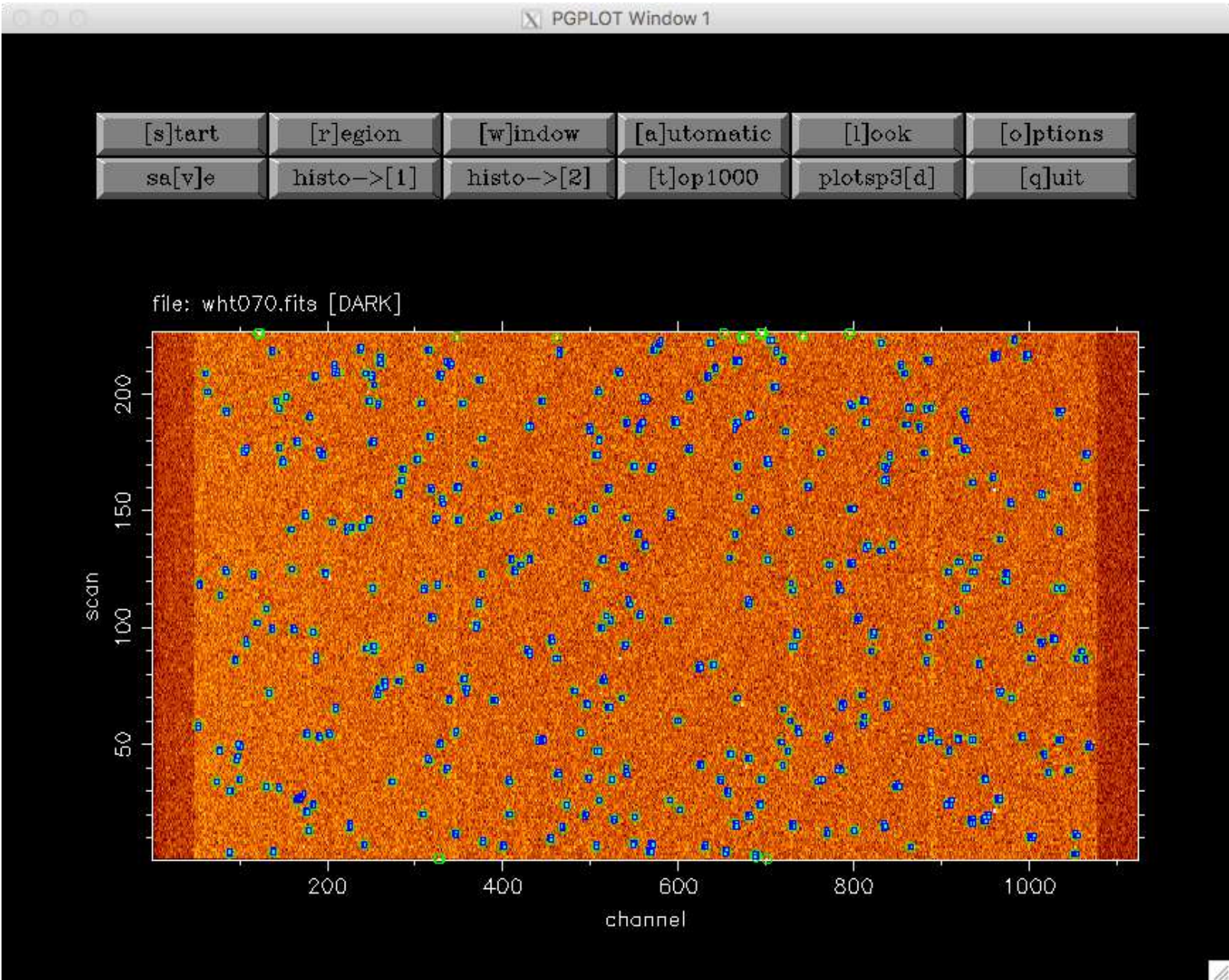
© Copyright 2017, Nicolás Cardiel. Revision d41172d2.

33 years old!




GUI with buttons!

Automatic detection of CR pixels  
+  
Interactive supervision



Automatic cleaning of images





Search

+ K

The package teareduce

Auxiliary functions and classes

Simulation of CCD images

Interactive Cosmic Ray Removal in Single or Double Exposures

Automatic Cosmic Ray Removal in Double Exposures

C distortion and wavelength calibration

Adaptive spline fit

Bibliography

Contents

Purpose

Installation

Table of contents

# The package teareduce

The Python package **teareduce** has been initially developed by Nicolás Cardiel to facilitate the reduction of astronomical images within the course *Experimental Techniques in Astrophysics*, which is part of the [Master's Degree in Astrophysics](#) at the [Universidad Complutense of Madrid \(UCM\)](#).

Different people at UCM have also contributed to the development and testing of this package: Sergio Pascual, María Chillarón, Cristina Cabello, Jesús Gallego and Jaime Zamorano. Acknowledgment is also given to Maite Ceballos (IFCA) for her help in setting up this documentation website.

Thanks are also extended to many students of the Master's in Astrophysics at UCM who, in recent years, have used this code in their practical work associated with the reduction of astronomical observations obtained with different instruments and telescopes.

## Purpose

This package is not intended to be a general-purpose image reduction code. It only includes specific operations required in certain steps of the traditional astronomical image reduction process that, at the time of its creation, were not available in more established packages such as [ccdproc](#). Students can examine the Python code and introduce modifications in order to gain a deeper understanding of the operations performed during the astronomical image reduction process. In addition, it also offers alternative ways to perform certain tasks that we have found to be more practical for use in Master's level classes.

## Installation

It is advisable to install this package in a Python environment. For example:

```
$ python3 -m venv venv_tea
$ . tea/bin/activate
(venv_tea) $ pip install teareduce
```

Warning

If you are planning to use **tea-cleanest**, you need to install this package with extra dependencies. In this case employ:

```
(venv_tea) $ pip install 'teareduce[cleanest]'
```

Astronomical Data Analysis Software & Systems ADASS 2025

## Teareduce: a Python package with utilities for teaching reduction techniques in Astronomy

Nicolás Cardiel<sup>1,2</sup>, Sergio Pascual<sup>1,2</sup>, María Chillarón-Víctor<sup>1,2</sup>, Cristina Cabello<sup>1,2</sup>, Jesús Gallego<sup>1,2</sup>, Jaime Zamorano<sup>1</sup>, M<sup>a</sup> Teresa Ceballos<sup>3</sup>

<sup>1</sup>Departamento de Física de la Tierra y Astrofísica, Facultad CC. Físicas Universidad Complutense de Madrid, Spain  
<sup>2</sup>Instituto de Física de Partículas y del Cosmos, IPARCOS, Facultad CC. Físicas, Universidad Complutense de Madrid, Spain  
<sup>3</sup>Instituto de Física de Cantabria, CSIC-Universidad de Cantabria, Spain

The Python package **teareduce** has been developed to support teaching activities related to the reduction of astronomical data. Specifically, it serves as **instructional material** for students participating in practical classes on the processing of astronomical images acquired with various instruments and telescopes. These classes are part of the course Experimental Techniques in Astrophysics, which belongs to the Master's Degree in Astrophysics at the Complutense University of Madrid. The code is publicly available on GitHub, accompanied by a documentation page that includes Jupyter notebooks demonstrating the use of its various classes and functions.

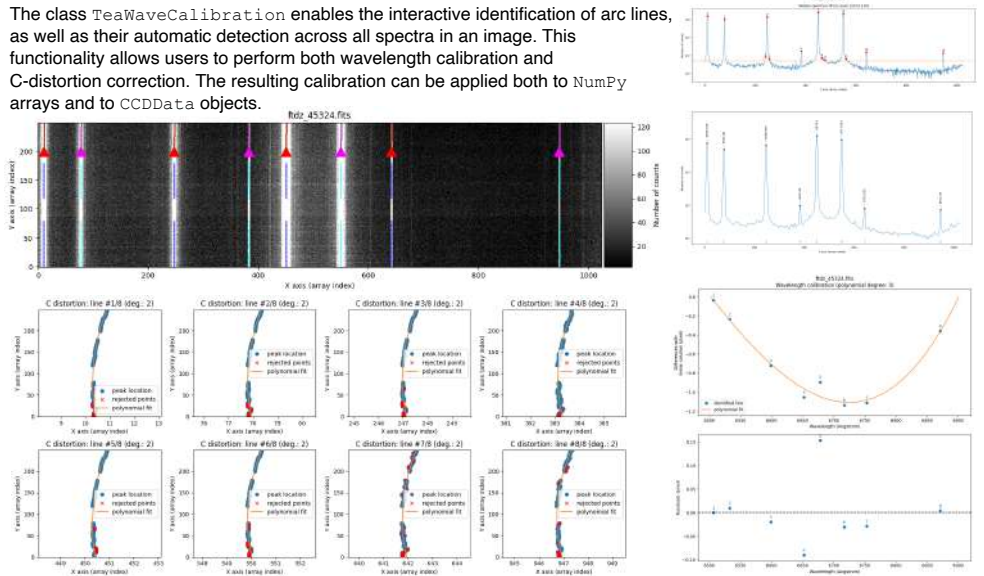
Python is widely used in the scientific community and offers a powerful ecosystem for astronomical data processing. **Astropy** and affiliated packages provide ready-to-use tools for tasks like handling FITS files, performing calibrations, and analyzing photometric data, which significantly simplifies the image reduction workflow. Additionally, Python integrates seamlessly with other scientific packages (e.g., **NumPy**, **SciPy**, **Matplotlib**) to enhance data manipulation and visualization. Its cross-platform compatibility with Windows, macOS, and Linux further ensures flexibility and accessibility for researchers working in different computing environments, making Python an ideal choice for astronomical image reduction.

As part of the course *Técnicas Experimentales en Astrofísica* (TEA; Experimental Techniques in Astrophysics), included in the Master's Degree in Astrophysics at the Universidad Complutense de Madrid, students prepare observing proposals, carry out astronomical observations using the CAFOS instrument installed on the 2.2 m telescope at the Calar Alto Observatory (Spain), reduce the acquired images, and analyse the results. To support the data reduction and analysis process, we have developed the **teareduce** package. Its main goal of is to serve as an educational tool.

**teareduce** is not intended as a general-purpose image reduction tool. Rather, it includes a set of specific operations needed for certain steps in the traditional astronomical image reduction workflow. At the time of its creation, these features were not available in more established packages such as **ccdproc**. Students can examine the Python code and introduce modifications in order to gain a deeper understanding of the operations performed during the astronomical image reduction process. The package also introduces alternative approaches to some tasks, which have proven to be more practical for use in Master's-level instruction. Some examples are provided below.

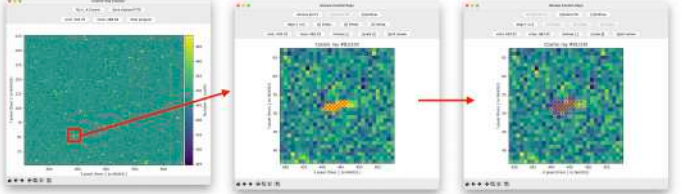
### Wavelength Calibration

The class `TeaWaveCalibration` enables the interactive identification of arc lines, as well as their automatic detection across all spectra in an image. This functionality allows users to perform both wavelength calibration and C-distortion correction. The resulting calibration can be applied both to NumPy arrays and to `CCDData` objects.

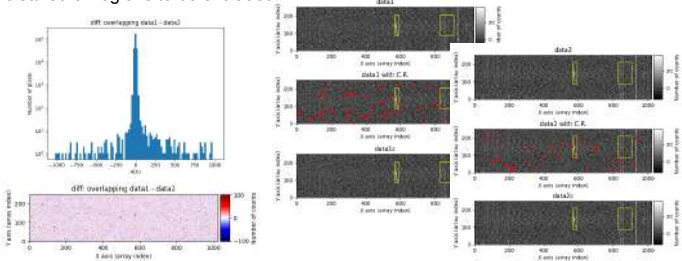



### Cosmic Ray removal

The installation of the **teareduce** package includes an auxiliary program called `tea-cleanest`, which enables the interactive cleaning of cosmic rays in individual exposures. This code is inspired by the `cleanest` code (Cardiel 2020; <https://cleanest.readthedocs.io/>) although the approach to detecting cosmic rays differs. In particular, this program uses the L.A. Cosmic algorithm (van Dokkum 2001) to identify pixels suspected of being affected by cosmic-ray hits. In the example shown in the figure below, we observe an image containing multiple cosmic-ray pixels (left panel), the identification of a specific cosmic ray (central panel); the affected pixels are pre-flagged and marked with red x's; the user can manually select or deselect suspicious pixels, and the interpolation of the affected pixels (right panel); the interpolated pixels are marked with x's, while the pixels used to compute the interpolation are indicated with magenta dots).



The `cc2images` function provides an alternative method for automatically removing cosmic rays in cases where two equivalent exposures are available. In such situations, the cosmic-ray pixels in each exposure can be replaced with the signal from the other exposure. This function also allows users to define regions to be cleaned or regions to be excluded.






Nicolás Cardiel

Madrid 17/12/2025

IV GUAIX Meeting

3





Search

% + K

The package teareduce

Auxiliary functions and classes

Simulation of CCD images

Interactive Cosmic Ray Removal in Single or Double Exposures

Automatic Cosmic Ray Removal in Double Exposures

C distortion and wavelength calibration

Adaptive spline fit

Bibliography

Contents

Purpose

Installation

Table of contents

# The package teareduce

The Python package **teareduce** has been initially developed by Nicolás Cardiel to facilitate the reduction of astronomical images within the course *Experimental Techniques in Astrophysics*, which is part of the [Master's Degree in Astrophysics](#) at the [Universidad Complutense of Madrid \(UCM\)](#).

Different people at UCM have also contributed to the development and testing of this package: Sergio Pascual, María Chillarón, Cristina Cabello, Jesús Gallego and Jaime Zamorano. Acknowledgment is also given to Maite Ceballos (IFCA) for her help in setting up this documentation website.

Thanks are also extended to many students of the Master's in Astrophysics at UCM who, in recent years, have used this code in their practical work associated with the reduction of astronomical observations obtained with different instruments and telescopes.

## Purpose

This package is not intended to be a general-purpose image reduction code. It only includes specific operations required in certain steps of the traditional astronomical image reduction process that, at the time of its creation, were not available in more established packages such as [ccdproc](#). Students can examine the Python code and introduce modifications in order to gain a deeper understanding of the operations performed during the astronomical image reduction process. In addition, it also offers alternative ways to perform certain tasks that we have found to be more practical for use in Master's level classes.

## Installation

It is advisable to install this package in a Python environment. For example:

```
$ python3 -m venv venv_tea
$ . tea/bin/activate
(venv_tea) $ pip install teareduce
```

Warning

If you are planning to use **tea-cleanest**, you need to install this package with extra dependencies. In this case employ:

```
(venv_tea) $ pip install 'teareduce[cleanest]'
```

Astronomical Data Analysis Software & Systems ADASS 2025

## Teareduce: a Python package with utilities for teaching reduction techniques in Astronomy

Nicolás Cardiel<sup>1,2</sup>, Sergio Pascual<sup>1,2</sup>, María Chillarón-Víctor<sup>1,2</sup>, Cristina Cabello<sup>1,2</sup>, Jesús Gallego<sup>1,2</sup>, Jaime Zamorano<sup>1</sup>, M<sup>a</sup> Teresa Ceballos<sup>3</sup>

<sup>1</sup>Departamento de Física de la Tierra y Astrofísica, Facultad CC. Físicas Universidad Complutense de Madrid, Spain  
<sup>2</sup>Instituto de Física de Partículas y del Cosmos, IPARCOS, Facultad CC. Físicas, Universidad Complutense de Madrid, Spain  
<sup>3</sup>Instituto de Física de Cantabria, CSIC-Universidad de Cantabria, Spain

The Python package **teareduce** has been developed to support teaching activities related to the reduction of astronomical data. Specifically, it serves as **instructional material** for students participating in practical classes on the processing of astronomical images acquired with various instruments and telescopes. These classes are part of the course Experimental Techniques in Astrophysics, which belongs to the Master's Degree in Astrophysics at the Complutense University of Madrid. The code is publicly available on GitHub, accompanied by a documentation page that includes Jupyter notebooks demonstrating the use of its various classes and functions.

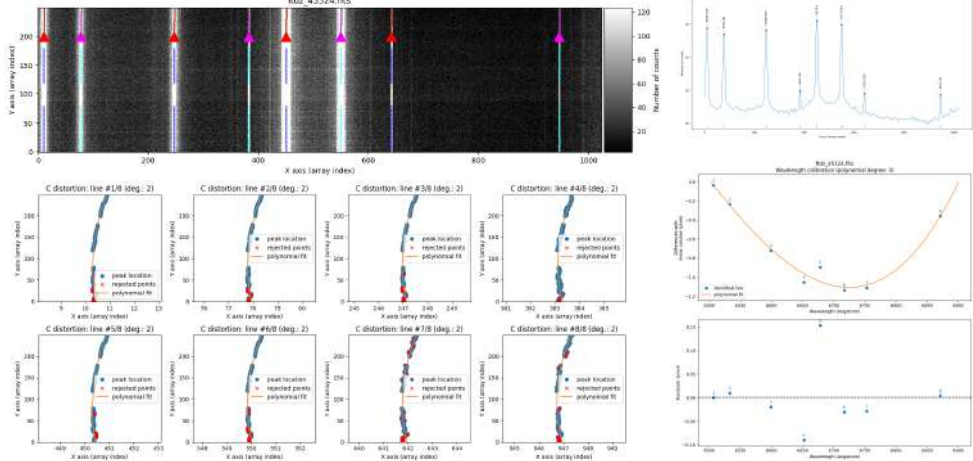
Python is widely used in the scientific community and offers a powerful ecosystem for astronomical data processing. **Astropy** and affiliated packages provide ready-to-use tools for tasks like handling FITS files, performing calibrations, and analyzing photometric data, which significantly simplifies the image reduction workflow. Additionally, Python integrates seamlessly with other scientific packages (e.g., **NumPy**, **SciPy**, **Matplotlib**) to enhance data manipulation and visualization. Its cross-platform compatibility with Windows, macOS, and Linux further ensures flexibility and accessibility for researchers working in different computing environments, making Python an ideal choice for astronomical image reduction.

As part of the course *Técnicas Experimentales en Astrofísica* (TEA; Experimental Techniques in Astrophysics), included in the Master's Degree in Astrophysics at the Universidad Complutense de Madrid, students prepare observing proposals, carry out astronomical observations using the CAFOS instrument installed on the 2.2 m telescope at the Calar Alto Observatory (Spain), reduce the acquired images, and analyse the results. To support the data reduction and analysis process, we have developed the **teareduce** package. Its main goal of is to serve as an educational tool.

**teareduce** is not intended as a general-purpose image reduction tool. Rather, it includes a set of specific operations needed for certain steps in the traditional astronomical image reduction workflow. At the time of its creation, these features were not available in more established packages such as **ccdproc**. Students can examine the Python code and introduce modifications in order to gain a deeper understanding of the operations performed during the astronomical image reduction process. The package also introduces alternative approaches to some tasks, which have proven to be more practical for use in Master's-level instruction. Some examples are provided below.

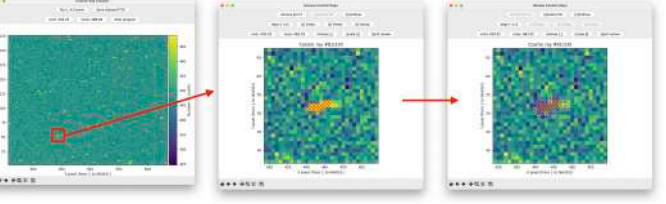
### Wavelength Calibration

The class `TeaWaveCalibration` enables the interactive identification of arc lines, as well as their automatic detection across all spectra in an image. This functionality allows users to perform both wavelength calibration and C-distortion correction. The resulting calibration can be applied both to NumPy arrays and to `CCDData` objects.



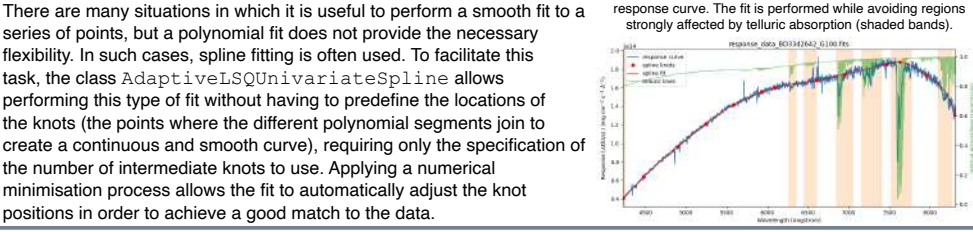
### Cosmic Ray removal

The installation of the **teareduce** package includes an auxiliary program called `tea-cleanest`, which enables the interactive cleaning of cosmic rays in individual exposures. This code is inspired by the `cleanest` code (Cardiel 2020; <https://cleanest.readthedocs.io/>) although the approach to detecting cosmic rays differs. In particular, this program uses the L.A. Cosmic algorithm (van Dokkum 2001) to identify pixels suspected of being affected by cosmic-ray hits. In the example shown in the figure below, we observe an image containing multiple cosmic-ray pixels (left panel), the identification of a specific cosmic ray (central panel); the affected pixels are pre-flagged and marked with red x's; the user can manually select or deselect suspicious pixels, and the interpolation of the affected pixels (right panel); the interpolated pixels are marked with x's, while the pixels used to compute the interpolation are indicated with magenta dots).



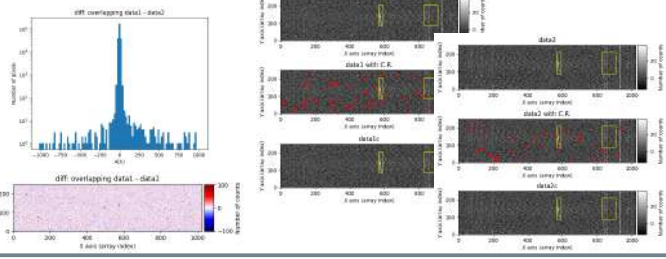
### Adaptive spline fit








There are many situations in which it is useful to perform a smooth fit to a series of points, but a polynomial fit does not provide the necessary flexibility. In such cases, spline fitting is often used. To facilitate this task, the class `AdaptiveLSQuivariateSpline` allows performing this type of fit without having to predefine the locations of the knots (the points where the different polynomial segments join to create a continuous and smooth curve), requiring only the specification of the number of intermediate knots to use. Applying a numerical minimisation process allows the fit to automatically adjust the knot positions in order to achieve a good match to the data.



### CC2images

The `cc2images` function provides an alternative method for automatically removing cosmic rays in cases where two equivalent exposures are available. In such situations, the cosmic-ray pixels in each exposure can be replaced with the signal from the other exposure. This function also allows users to define regions to be cleaned or regions to be excluded.





Project: P17/21 TAU-CM P17T-CM

Financiado por la Unión Europea NextGenerationEU

Proyecto PID2020-138631NB-I00 funded by:


Nicolás Cardiel

Madrid 17/12/2025

IV GUAIX Meeting

4





Search

\* + K

The package teareduce

Auxiliary functions and classes

Simulation of CCD images

Interactive Cosmic Ray Removal in Single or Double Exposures

Automatic Cosmic Ray Removal in Double Exposures

C distortion and wavelength calibration

Adaptive spline fit

Bibliography

# Interactive Cosmic Ray Removal in Single or Double Exposures

The installation of the **teareduce** package also includes an auxiliary program called **tea-cleanest**, which enables the interactive cleaning of cosmic rays. This code is inspired by the [cleanest](#) code [Cardiel20], although the approach to detecting cosmic rays differs. In particular, **tea-cleanest** uses the [L.A. Cosmic algorithm](#) [vanDokkum01] to identify pixels suspected of being affected by cosmic-ray hits.

The program also allows alternatively loading a mask that indicates the location of the cosmic-ray pixels. In this case, the mask will be read from an external FITS file containing an array of integers, where any pixel with a value other than 0 is considered a cosmic-ray pixel.

The identified cosmic rays can be interpolated either automatically or under user supervision. To replace the signal in the cosmic-ray pixels, information from the neighboring pixels is used, and any of the following methods may be applied:

- a constant average value: the mean or median
- one-dimensional interpolation: along the X-axis or the Y-axis
- 2D interpolation using a plane (degree 1)
- the values determined by L.A.Cosmic
- the values computed using the method described by [vanDokkumPasha24]
- the values present in an auxiliary image

Note

Although **tea-cleanest** was initially developed to clean individual exposures, it now also allows the use of a second (auxiliary) exposure, whose information can be used to replace pixels affected by cosmic rays in the first image.

In the case of equivalent double exposures, it is possible to remove the cosmic rays from each exposure employing the information in the other exposure as an auxiliary image. By swapping the roles of main and auxiliary image, we can remove the cosmic rays from the second exposure as well. This method should work properly as long as the number of cosmic rays is not so high that the same pixel is likely to be affected by a cosmic ray in both exposures.

When three or more equivalent exposures are available, **tea-cleanest** can be used by first cleaning a pair of those images and then using the result as the auxiliary image for cleaning the remaining exposures. In any case, in this scenario it may be preferable to use the median of the available exposures instead of **tea-cleanest**. It is also possible to use slightly more sophisticated algorithms, such as the one implemented in [numina-crmasks](#).

Contents

Simple execution (single image)

Automatic cleaning of the detected CRs

Manual cleaning of the detected CRs

Advanced execution (two images)

Description of the main window button actions

Using the cleanest functionality programmatically

(venv\_tea) \$ tea-cleanest --help

Usage: tea-cleanest [-h] [--extension EXTENSION] [--auxfile AUXFILE] [--extension\_auxfile EXTENSION\_AUXFILE] [--fontfamily FONTFAMILY] [--fontsize FONTSIZE] [--width WIDTH] [--height HEIGHT] [--version] [--verbose] [input\_fits]

Interactive cosmic ray cleaner for FITS images.

Positional Arguments:

input\_fits

Path to the FITS file to be cleaned.

Options:

-h, --help

show this help message and exit

--extension EXTENSION

FITS extension to use (default: 0).

--auxfile AUXFILE

Auxiliary FITS file

--extension\_auxfile EXTENSION\_AUXFILE

FITS extension for auxiliary file (default: 0).

--fontfamily FONTFAMILY

Font family for the GUI (default: Helvetica).

--fontsize FONTSIZE

Font size for the GUI (default: 14).

--width WIDTH

Width of the GUI window in pixels (default: 800).

--height HEIGHT

Height of the GUI window in pixels (default: 700).

--version

show program's version number and exit

--verbose, -v

Enable verbose output.

Cosmic Ray Cleaner (TEA version 0.5.9)

Run L.A.Cosmic

Load auxdata

Load CR mask

Replace detected CRs

Review detected CRs

[c]ursor: OFF

[t]oggle data

[a]spect: equal

Save cleaned FITS

Stop program

vmín: -16.33

vmáx: 32.00

minmax [,]

zscale [/]

CR overlay: ON

Help

data: examplecr1.fits

Y pixel (from 1 to NAXIS2)

1400

1200

1000

800

600

400

200

250

500

750

1000

1250

1500

1750

2000

X pixel (from 1 to NAXIS1)

Number of counts

30

20

10

0

-10

Navigation icons

Cosmic Ray Cleaner (TEA version 0.5.9)

Run L.A.Cosmic

Load auxdata

Load CR mask

Replace detected CRs

Review detected CRs

[c]ursor: ON

[t]oggle data

[a]spect: equal

Save cleaned FITS

Stop program

vmín: -16.33

vmáx: 32.00

minmax [,]

zscale [/]

CR overlay: ON

Help

data: examplecr1.fits

Y pixel (from 1 to NAXIS2)

1400

1200

1000

800

600

400

200

250

500

750

1000

1250

1500

1750

2000

X pixel (from 1 to NAXIS1)

Number of counts

30

20

10

0

-10

Navigation icons

(x, y) = (518, 1490) [-0.6]

Nicolás Cardiel

Madrid 17/12/2025

IV GUAIX Meeting

5



## Cosmic-Ray Rejection by Laplacian Edge Detection

PIETER G. VAN DOKKUM<sup>1</sup>

California Institute of Technology, MS 105-24, Pasadena, CA 91125; pgd@astro.caltech.edu

Received 2001 May 1; accepted 2001 July 31

**ABSTRACT.** Conventional algorithms for rejecting cosmic rays in single CCD exposures rely on the contrast between cosmic rays and their surroundings and may produce erroneous results if the point-spread function is smaller than the largest cosmic rays. This paper describes a robust algorithm for cosmic-ray rejection, based on a variation of Laplacian edge detection. The algorithm identifies cosmic rays of arbitrary shapes and sizes by the sharpness of their edges and reliably discriminates between poorly sampled point sources and cosmic rays. Examples of its performance are given for spectroscopic and imaging data, including *Hubble Space Telescope* Wide Field Planetary Camera 2 images.

### 1. INTRODUCTION

Various methods are in use for identifying and replacing cosmic-ray hits in CCD data. The most straightforward approach is to obtain multiple exposures of the same field (or multiple nondestructive readouts during a single exposure; e.g., Fixsen et al. 2000). In general, a given pixel will suffer from a cosmic-ray hit in only one or two of the exposures, and the remaining exposures can be used to obtain its replacement value (e.g., Zhang 1995). Methods for combining multiple exposures have reached a high degree of sophistication, particularly those developed for dithered *Hubble Space Telescope* (HST) data (e.g., Windhorst, Franklin, & Neuschaefer 1994; Freudling 1995; Fruchter & Hook 1997).

However, there are circumstances when cosmic-ray identification in single exposures is required or desirable. The object of interest may be varying or moving on short timescales, and in the case of long-slit spectra, the positions and intensities of sky lines and object spectra may change (e.g., Croke 1995). Furthermore, pixels can be hit by cosmic rays in more than one exposure, and some affected pixels may remain after combining individual images. Cosmic-ray removal in individual exposures may also be desirable if the images are shifted with respect to each other by a noninteger number of pixels or if the seeing varied significantly between the exposures (see Rhoads 2000). Finally, multiple exposures are simply not always available.

Methods for identifying cosmic rays in single images or spectra include median filtering (e.g., QZAP by M. Dickinson), filtering by adapted point-spread functions (PSFs) (e.g., Rhoads 2000), trained neural networks (Salzberg et al. 1995), and the interpolation of neighboring pixels (e.g., the COSMICRAYS

task in the IRAF package). All these methods effectively remove small cosmic rays from well-sampled data.

The most widely used methods are based on some form of median filtering and usually include adaptations to exclude stars and emission lines from the list of cosmic rays. However, problems arise when cosmic rays affect more than half the area of the filter or when the PSF is smaller than the filter. The size of the filter is therefore a trade-off between detecting large cosmic rays and limiting contamination by structure on the scale of the PSF.

In this paper, a new algorithm for rejecting cosmic rays in single exposures is described. It is based on Laplacian edge detection, which is a widely used method for highlighting boundaries in digital images (see, e.g., Gonzalez & Woods 1992). The strength of the method is that it relies on the sharpness of the *edges* of cosmic rays rather than the contrast between entire cosmic rays and their surroundings. Therefore, it is largely independent of the morphology of cosmic rays. This property is very useful and forms the basis for a robust discrimination between poorly sampled point sources and cosmic rays.

### 2. THE LAPLACIAN

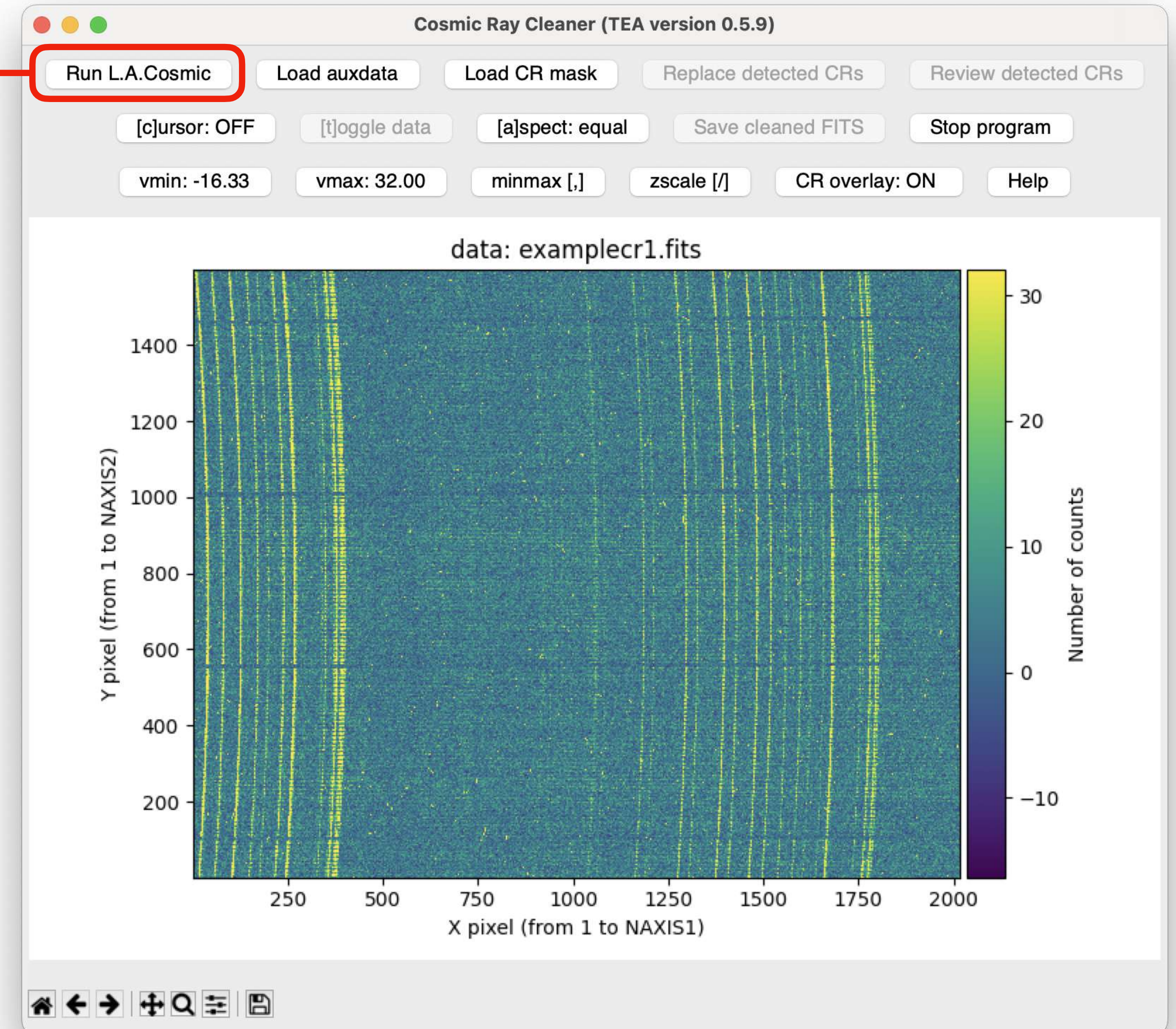
The Laplacian of a two-dimensional function is a second-order derivative defined as

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}. \quad (1)$$

The Laplacian is commonly used for edge detection in digital images. In this application, the image is convolved with the

<sup>1</sup> Hubble Fellow.

The detection of CR pixels is performed making use of the **Laplacian Edge Detection Algorithm** ([van Dokkum 2001](#)).





# The detection algorithm is executed twice to detect CR tails

Cosmic Ray Mask Generation Parameters

L.A.Cosmic Parameters

Parameter	Run 1	Run 2	Type
sigclip:	5.0	3.0	(float)
sigfrac:	0.3	0.3	(float)
objlim:	5.0	5.0	(float)
gain:	1.0	1.0	(float)
readnoise:	6.5	6.5	(float)
satlevel:	65535	65535	(float)
niter:	4	4	(int)
sepmed:	True	True	(bool)
cleantype:	meanmask	meanmask	(str)

Parameter	Run 1	Run 2	Type
fsmode:	median	median	(str)
psfmodel:	gaussxy	gaussxy	(str)
psffwhm_x:	2.5	2.5	(float)
psffwhm_y:	2.5	2.5	(float)
psfsize:	7	7	(int, odd)
psfbeta:	4.765	4.765	(float)
verbose:	False	False	(bool)
inbkg:			
invar:			

Additional Parameters

Dilation:	0	(int)
Border Padding:	10	(int)

Region to be Examined

xmin:	1	(int) --> [1, 2016]
xmax:	2016	(int) --> [1, 2016]
ymin:	1	(int) --> [1, 1596]
ymax:	1596	(int) --> [1, 1596]

OK

Cancel

Reset



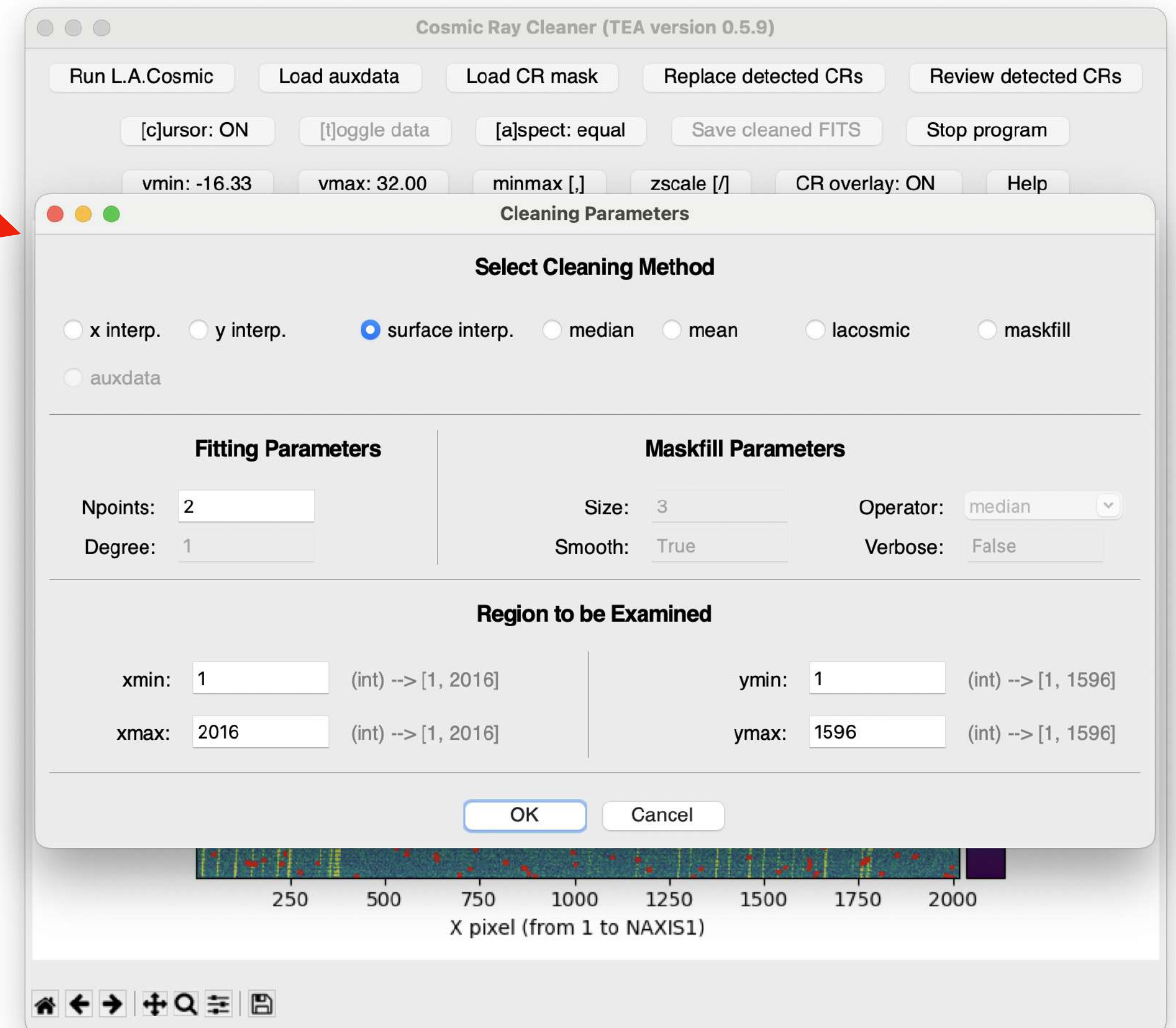
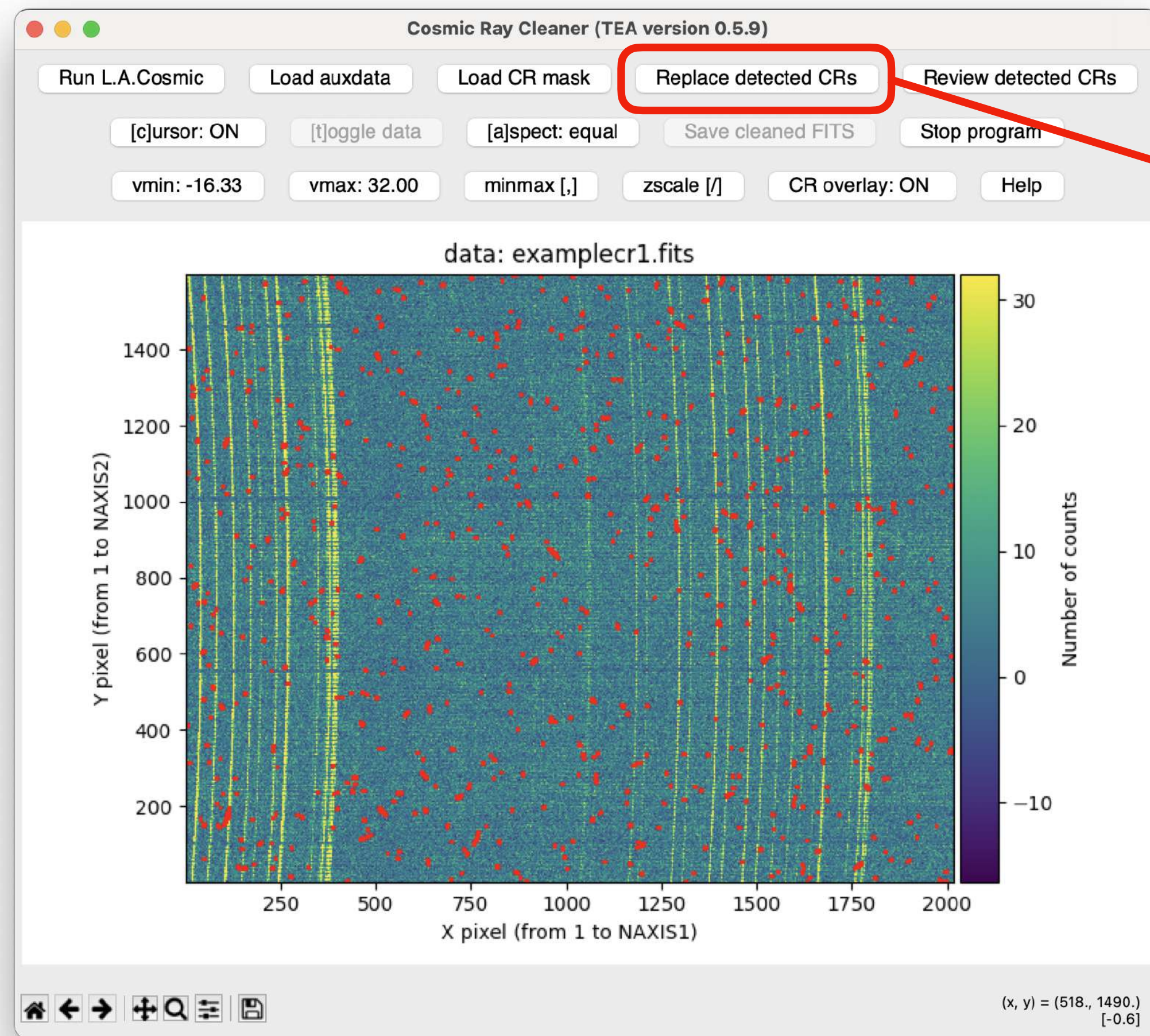
# After detecting the CR pixels

- Connected pixels are grouped to form CR features
- The user can interpolate the CR pixels using 

{



Automatic (unsupervised) interpolation

Interactive (supervised) cleaning





## A Robust and Simple Method for Filling in Masked Data in Astronomical Images

Pieter van Dokkum  and Imad Pasha   
Astronomy Department, Yale University, 219 Prospect St, New Haven, CT 06511, USA  
*Received 2023 December 4; accepted 2024 February 12; published 2024 March 13*

### Abstract

Astronomical images often have regions with missing or unwanted information, such as bad pixels, bad columns, cosmic rays, masked objects, or residuals from imperfect model subtractions. In certain situations it can be essential, or preferable, to fill in these regions. Most existing methods use low order interpolations for this task. In this paper a method is described that uses the full information that is contained in the pixels just outside masked regions. These edge pixels are extrapolated inwards, using iterative median filtering. This leads to a smoothly varying spatial resolution within the filled-in regions, and ensures seamless transitions between masked pixels and good pixels. Gaps in continuous, narrow features can be reconstructed with high fidelity, even if they are large. The method is implemented in `maskfill`, an open-source MIT licensed Python package (<https://github.com/dokkum/maskfill>). Its performance is illustrated with several examples, and compared to several alternative interpolation schemes.

*Unified Astronomy Thesaurus concepts:* [Direct imaging \(387\)](#); [Astronomical techniques \(1684\)](#); [Astronomy data reduction \(1861\)](#); [Astronomy data analysis \(1858\)](#)

### 1. Introduction

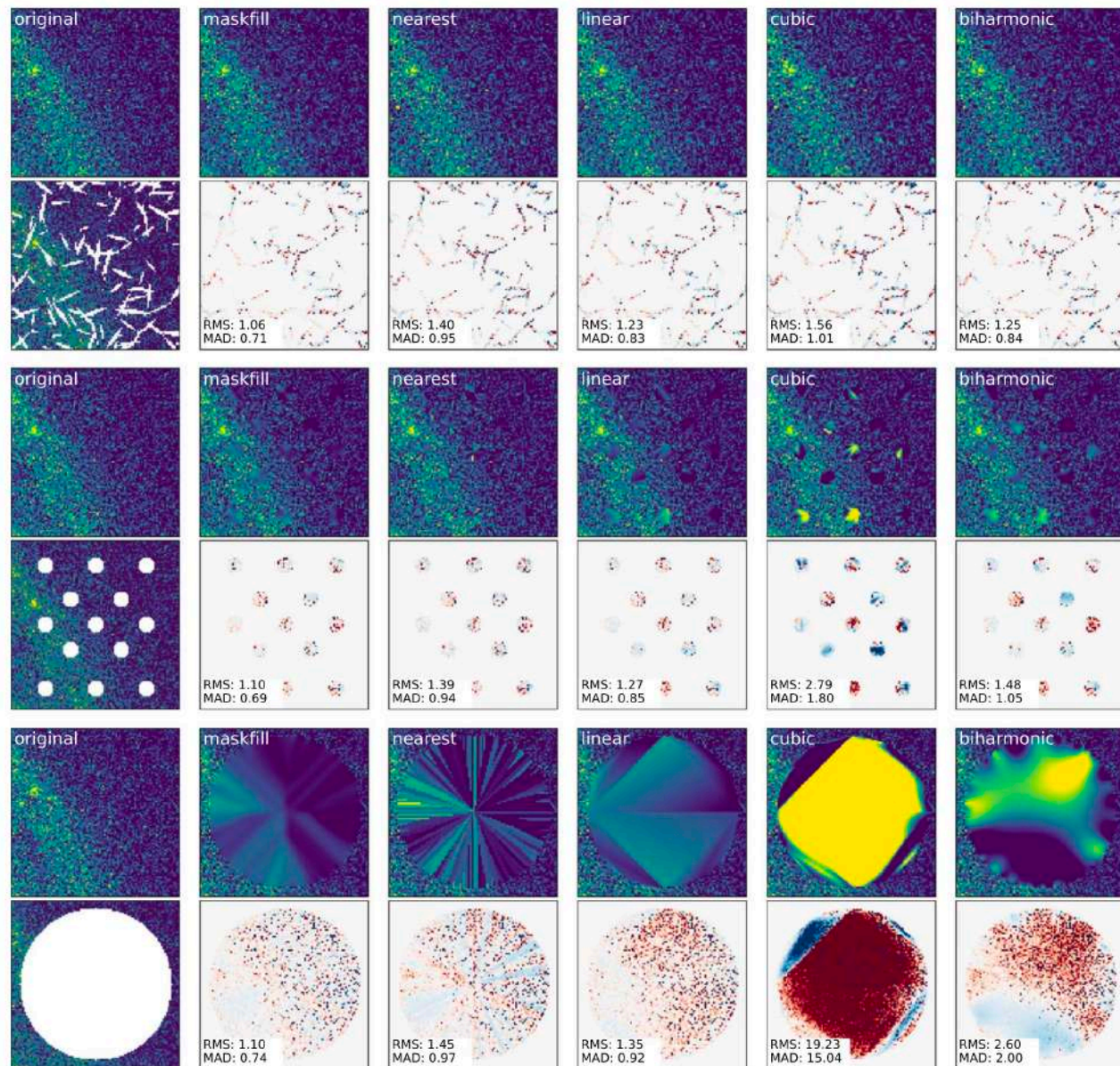
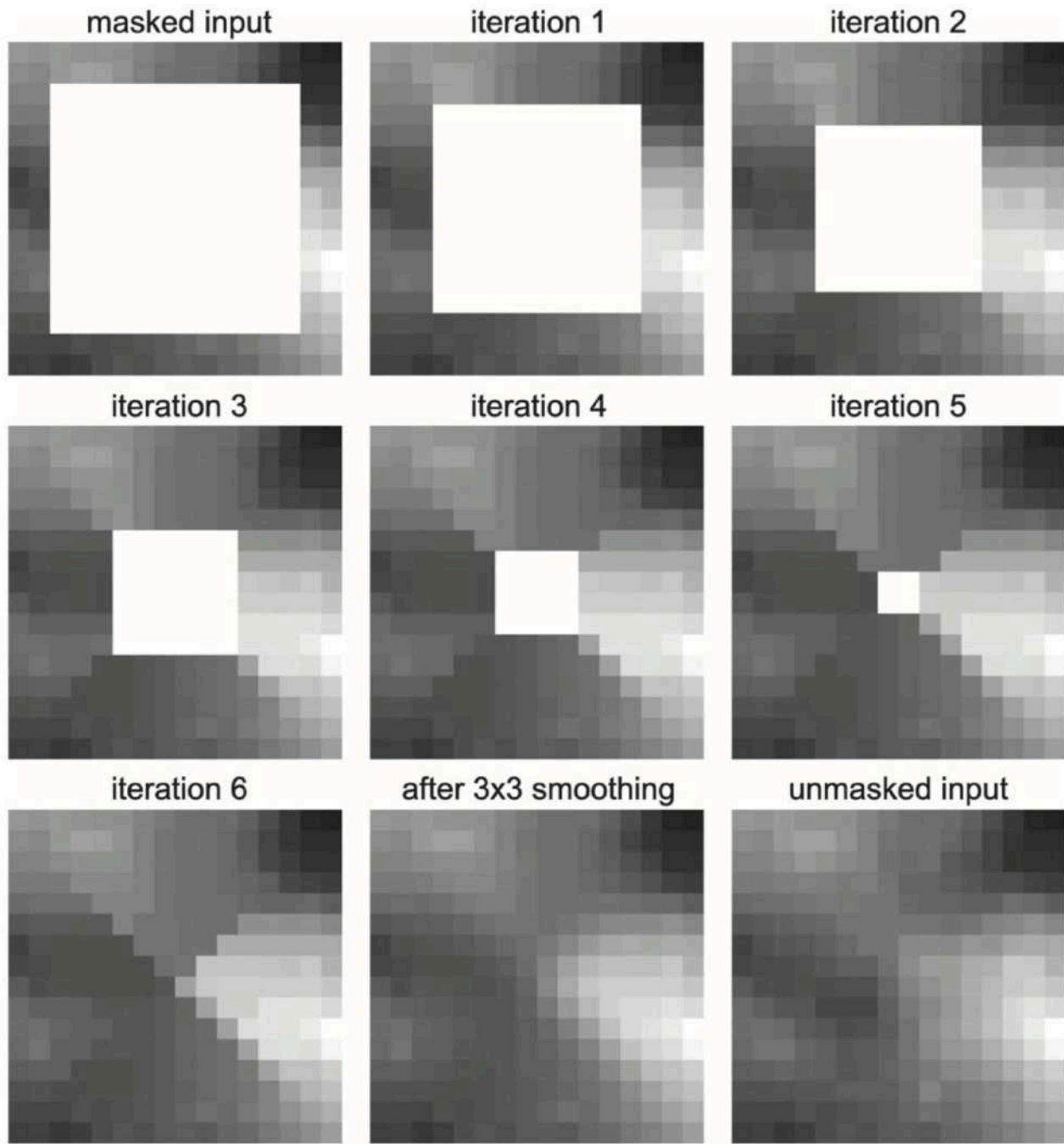
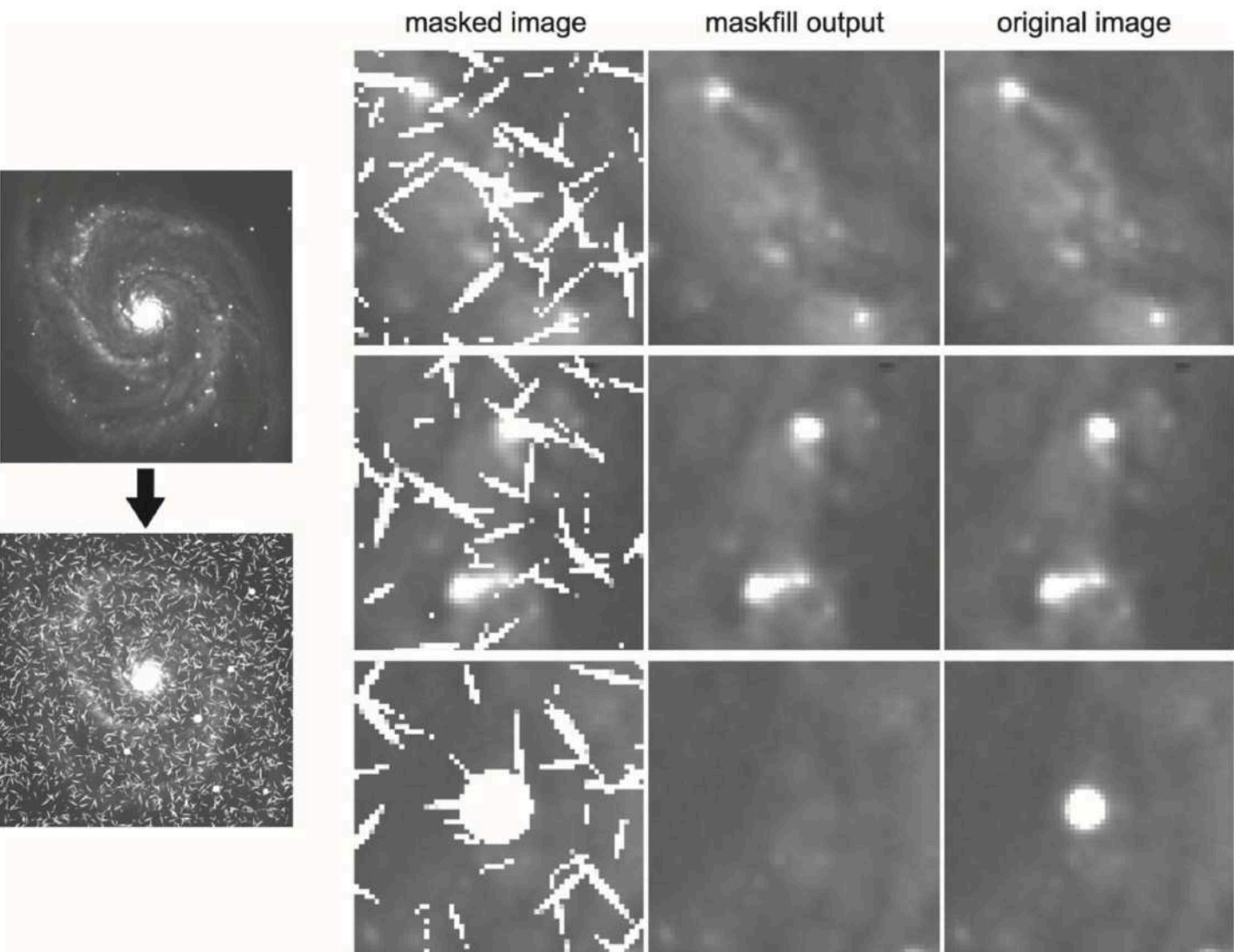
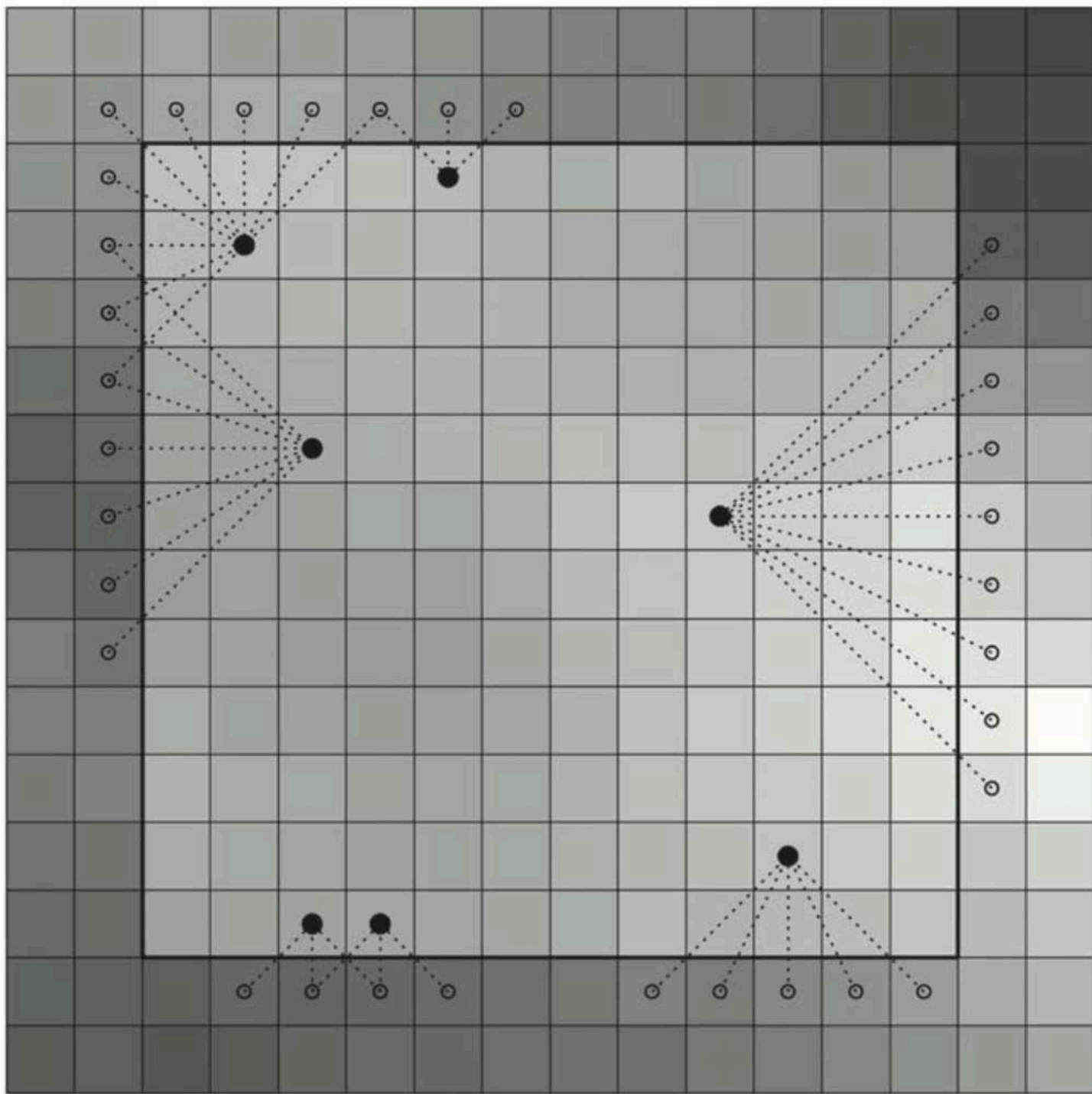
Image masking serves various purposes. Detector defects, such as hot pixels, bad pixels, or bad columns, result in predictable locations where data cannot be trusted or are missing altogether. Cosmic ray hits can occur anywhere on the detector, producing short trails of very bright pixels (Leach & Gursky 1979). Both detector defects and cosmic rays are routinely masked in the early stages of the data reduction process.

Another reason for masking is if certain objects are unwanted. An example is masking of bright stars and galaxies in data that are searched for low surface brightness emission (Greco et al. 2018; Montes & Trujillo 2018; Danieli & van Dokkum 2019). A variation on this theme is the masking of residuals after image subtraction. Image subtraction is routinely performed in transient photometry (Kessler et al. 2015), searches for faint or spatially extended objects near bright ones (Marois et al. 2006; van Dokkum et al. 2020), continuum correction of narrow band data (James et al. 2004; Garner et al. 2022; Lokhorst et al. 2022), and a wide range of other contexts. In all these applications, regions where the subtraction is not satisfactory (such as the centers of bright stars) are

typically masked, so they do not impact the subsequent analysis.

In many cases masked pixels do not need to be filled in, either because there are independent, redundant, observations of the same sky positions (e.g., Fruchter & Hook 2002), or because subsequent image analysis steps can explicitly handle masked data, e.g., profile fitting software such as `GALFIT` (Peng et al. 2002) and `pyrsenic` (Pasha & Miller 2023).

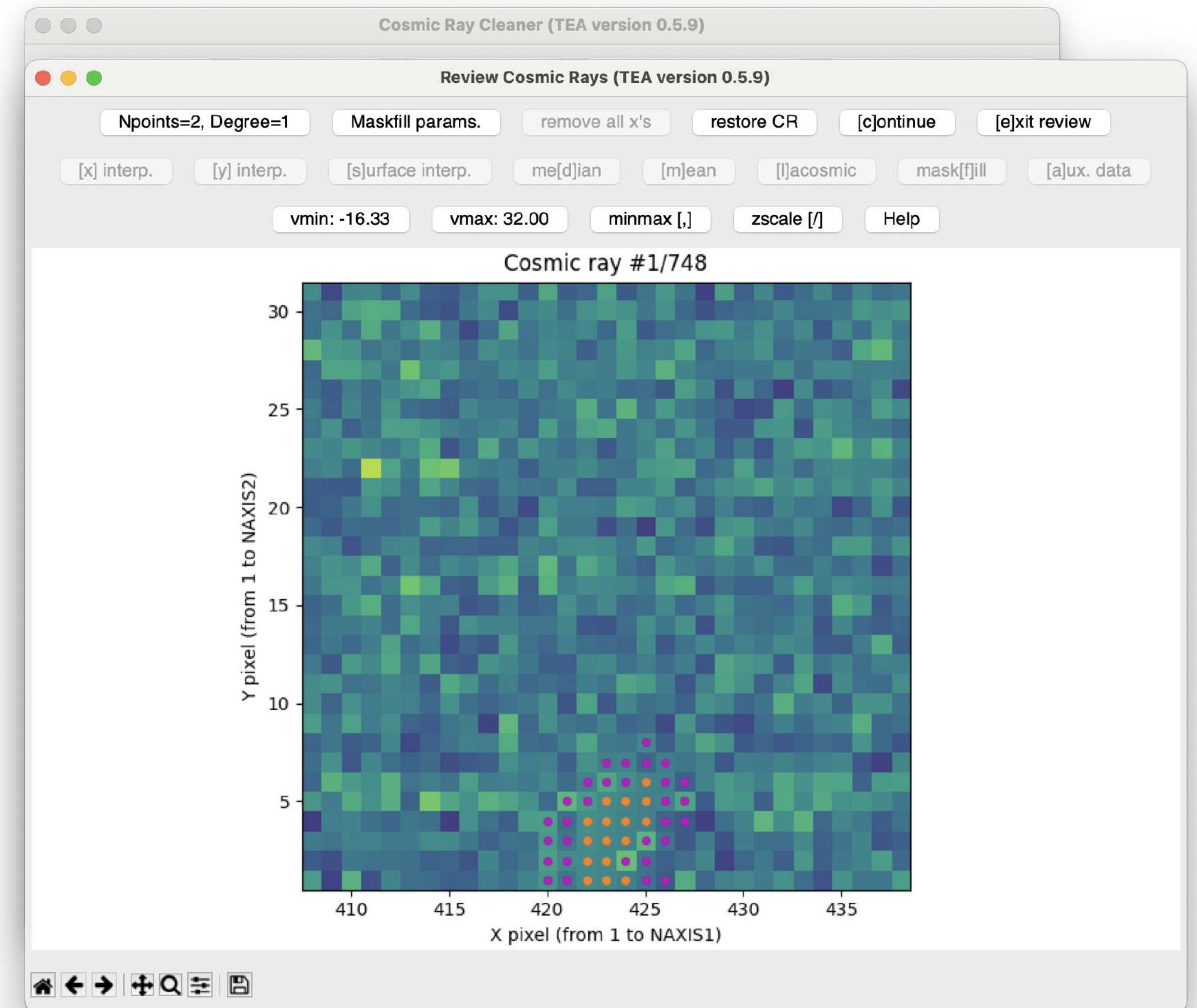
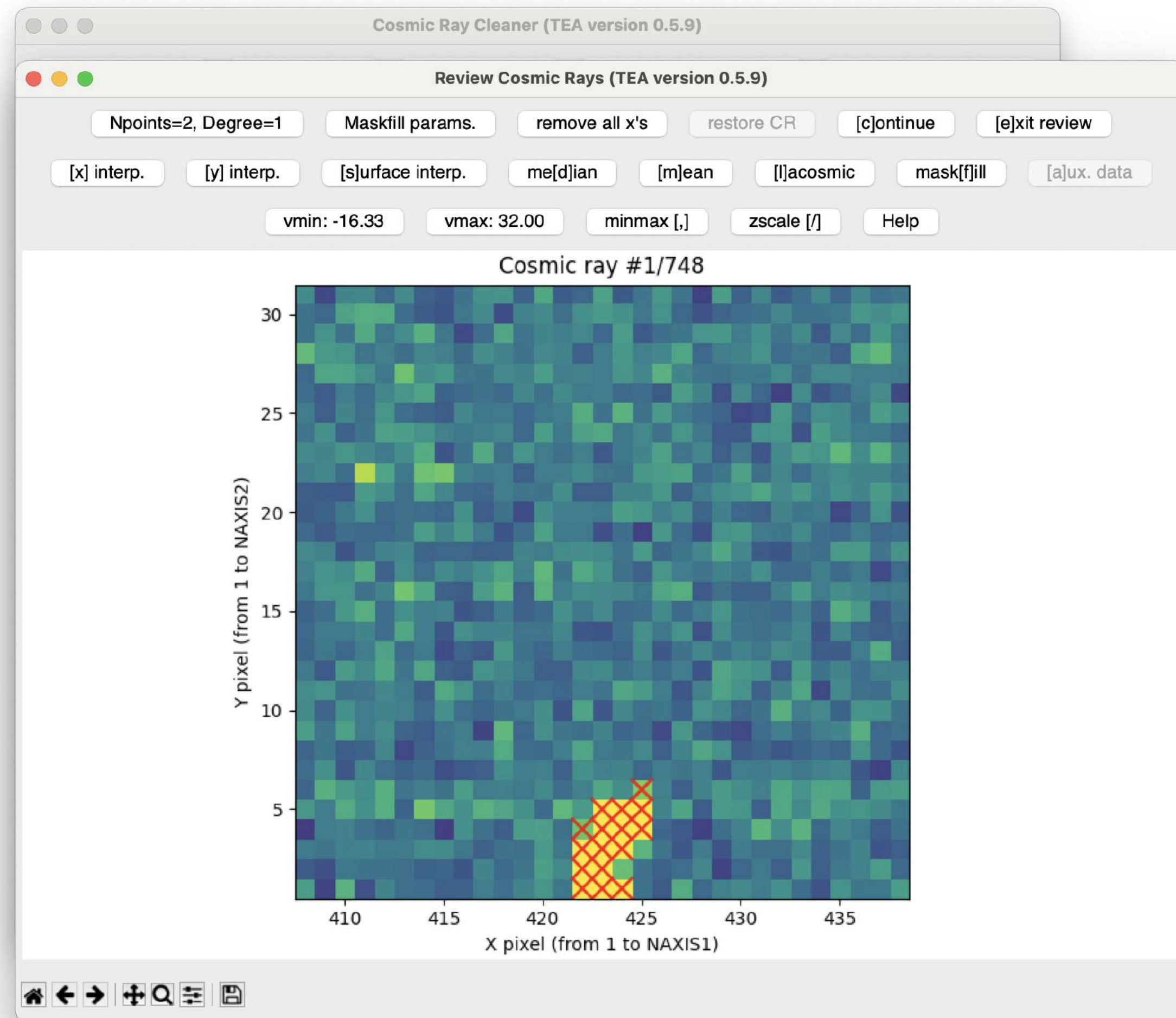
However, there are exceptions when mask-filling (also known as inpainting) is desirable or necessary. Multiple redundant exposures are not always available, for instance in time series data. Also, detector defects and cosmic rays are sharp features that are best identified before resampling the data; this is why reduction pipelines often remove these features from individual frames, even if multiple exposures are available (Kelson 2003; Neill et al. 2023). Turning to masked stars and galaxies, convolutions such as Gaussian smoothing produce artifacts on both sides of sharp boundaries, and these can be greatly reduced when the masks are (temporarily) filled in beforehand. Another reason to fill in masks is to properly account for missing flux when measuring intracluster light (Montes & Trujillo 2018) or Galactic cirrus emission (Liu et al. 2023). A recently developed application is to obtain the best-possible local background for photometry; Saydjari & Finkbeiner (2022) show that inpainting can lead to significant improvements in photometric accuracy when dealing with





After detecting the CR pixels

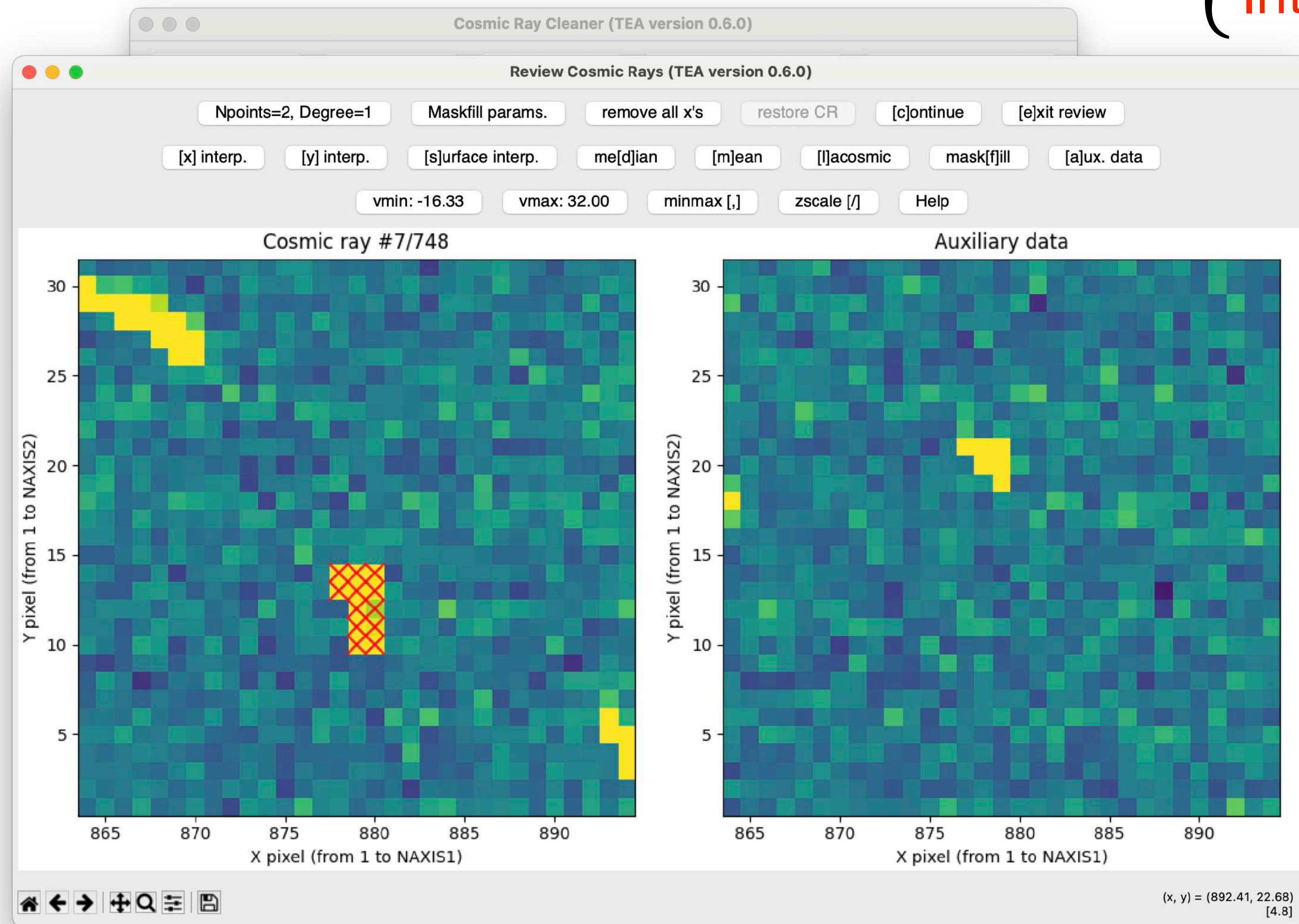
- Connected pixels are grouped to form CR features
- The user can interpolate the CR pixels using  $\left\{ \begin{array}{l} \text{Automatic (unsupervised) interpolation} \\ \text{Interactive (supervised) cleaning} \end{array} \right.$





## After detecting the CR pixels

- Connected pixels are grouped to form CR features
- The user can interpolate the CR pixels using  $\left\{ \begin{array}{l} \text{Automatic (unsupervised) interpolation} \\ \text{Interactive (supervised) cleaning} \end{array} \right.$

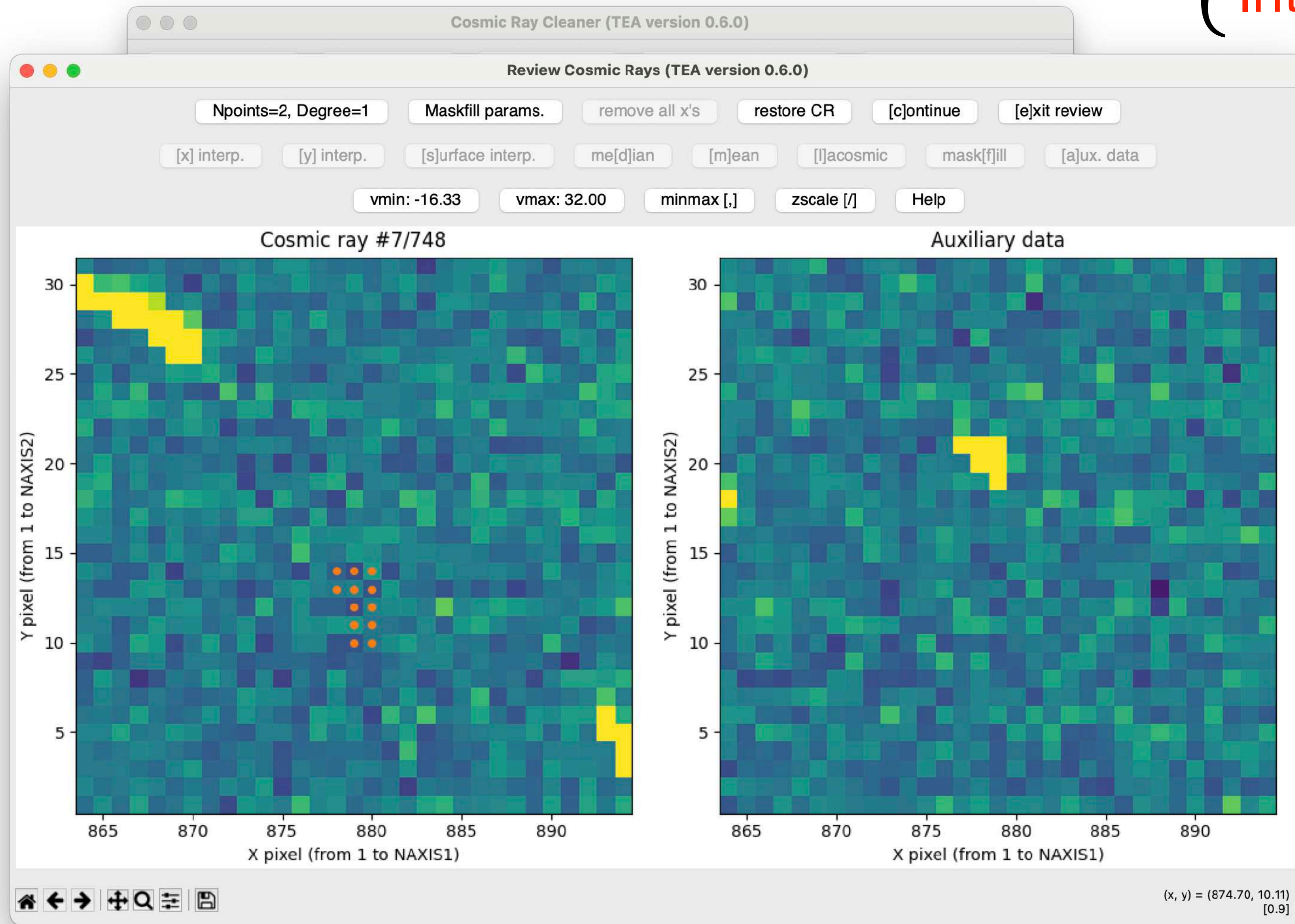


It is possible to use an **auxiliary image**, which allows cleaning pairs of equivalent exposures (assuming the CR affecting both exposures do not affect the same pixels).



## After detecting the CR pixels

- Connected pixels are grouped to form CR features
- The user can interpolate the CR pixels using  $\left\{ \begin{array}{l} \text{Automatic (unsupervised) interpolation} \\ \text{Interactive (supervised) cleaning} \end{array} \right.$



It is possible to use an **auxiliary image**, which allows cleaning pairs of equivalent exposures (assuming the CR affecting both exposures do not affect the same pixels).

