Background Cleaning Code – A Supplement.

Sections

1 – Setting up the code

The main code, 'background_cleaning.py' can sit in any directory, but must be surrounded by a particular directory structure in order to work properly. It must have other programs in the same directory, as well as a couple of folders containing files. These are all listed in more detail below.

Throughout this document, whenever I mention 'the code' inexplicably, I mean the main code that does the bulk of the work, 'background_cleaning.py'.

1.1 - Modules

The following modules must be installed in your version of Python;
-matplotlib
-math
-pylab
-numpy
-astropy
-collections
-urllib, urlparse
-aplpy
-PIL
-emcee
-scipy
-decimal
-pyfits

1.2 – Files needed around code

In the same directory as background_cleaning.py there must be the following files:
- binning.py
- triangle.py
- ccd4mask2.fit
- iphas-images.fits

- iphas-qc.fits

1.3 – Bright star table
This same directory must also contain the table that lists the bright stars that fall on or near images in the search area. The name of this file must be put into background_cleaning.py before executing it. The bright star table is made using the code 'stars2.py' which requires as inputs the minimum and maximum magnitudes you want to mask, as well as the name of the table of Tycho stars masked to the region of sky of interest. More information on running stars2.py is found within the code.

The file 'IPHASStars_0-16.fit' contains all cases where bright stars fall on or near any of the 'best' images as of DS2.

1.4 – Directories needed around the code
There must also be a directory called 'airglow_frames', containing the 12 dark-time frames and a second directory called 'confmaps', where the confidence maps will be stored.

1.5 – Editables inside the code
Before running the program, inside the code itself there are a few things to specify. These are marked inside but are;
-the name you're giving the targeted area.
-the confidence threshold, below which pixels are masked in the confidence cleaning step.
-the size of the bins in pixel to split the raw images into to find the background counts (usually 100).
-the sigma level at which to clip the binned images.
-the centre of the desired region in galactic coordinates.
-the size of the target area in degrees.
-the directory in which to build the directory structure for the code to run and store all raw, working and cleaned images. Must have lots of free space!!
-the location of the iphas-images fits table. Should just be included next to the main code.
-the location and name of the table containing the bright star information for the targeted region. The name depends on what you call it at the end of stars2.py.

2 – After running the code
2.1 – Should the code crash
When the code is executed, by running 'python background_cleaning.py' in the terminal, it will ask which number image to start at. This is almost always '0', unless there has been a bug causing the code to crash part way through the images, in which case it can be resumed at the point it crashed.

This is something I implemented to get around the 'Segmentation Fault' I kept getting in the terminal. It shouldn't be a problem, it seemed to be a fairly random problem that only my desktop suffered and John nor Leigh could get to the bottom of.

2.2 – Required network connection
Presuming the code is starting from the first image in the target area, and isn't starting part-way through due to a crash, it will download all the required raw images first. Once this part is complete, for the rest (and bulk) of it's duration it will not need a network connection.

2.3 – Montage

The directory structure at the end of the code is set up to allow an easy transition from this code that cleans the images, and Montage (http://montage.ipac.caltech.edu/) that mosaics them together. The only intermediate step is to bin the cleaned images and move them from the output directory of 'background_cleaning.py' which is called 'Ha_cleandir/' to the starting directory for Montage, 'rawdir/'. This is easily achieved by running the included program 'bin.py'.

2.3.1 – Binning

When running 'bin.py', the program will ask for 3 things:
-the root directory of the cleaned images, e.g: '~/test/'
-the folder containing the cleaned, unbinned images, e.g. 'Ha_cleandir/'
-the destination folder for the binned images, e.g. 'rawdir/'

After a little while of running, the binned images will be found in the output folder, while the unbinned, images remain as and where they were.