PyBDSM Cookbook

David Rafferty

February 21, 2011

1 Introduction

PyBDSM (**Py**thon **B**lob **D**etection and **S**ource **M**anagement) is a Python source-finding software package written by Niruj Mohan and Alexander Usov and currently maintained by David Rafferty. PyBDSM can process FITS and CASA images and can output source lists in BBS, FITS, ASCII and ds9 region formats. It can be used in the IPython environment or in Python scripts. Details of the algorithms used in PyBDSM can be found in Section 2.7 of the Anaamika manual located at `http://www.strw.leidenuniv.nl/~mohan/anaamika_manual.pdf` (the Anaamika package includes the Fortran version of PyBDSM, as well as other related software, and is no longer being maintained).

2 Setup

Currently, PyBDSM is installed on the offline cluster in `/home/rafferty/PyBDSM`.[1] To initialize your environment for PyBDSM, run:

```
> use LofIm
> source /home/rafferty/PyBDSM/init_pybdsm
```

PyBDSM should now be visible to Python and can be imported into Python scripts or IPython with the command `import bdsm`.

3 Usage

A standard analysis is done using the `bdsm.process_image()` task. This task reads in the input image, calculates a background rms image, finds islands of emission, fits Gaussians to the islands, and groups the Gaussians into sources. The general call is (with defaults shown):

```
img = bdsm.process_image(input_file, beam=None, freq=None,
      thresh_isl=3.0, thresh_pix=5.0, thresh_gaus=None,
```

[1] Eventually, PyBDSM will be part of the nightly build; at that point, it may be initialized with `use LUS`.

```
use_rms_map=None, rms_box=None, rms=None, extended=False,
gaussian_maxsize=10.0)
```

where `input_file` is the name of the image (in FITS or CASA format) and the other (optional) arguments are defined as follows:

`beam` The restoring beam, defined as (major axis FWHM [deg], minor axis FWHM [deg], angle east from north of major axis [deg]). For example, `beam = (0.002916, 0.002654, -173.36)`. If not given (i.e., `beam = None`, the default), the beam information is read from the image header.

`freq` The frequency in Hz of input image (use a list if input image is a cube: e.g., `[30e6, 35e6, 72e6]`). If not given (i.e., `freq = None`, the default), the frequency is read from the image header.

`thresh_isl` The threshold for island detection in number of times the background rms (default = 3.0).

`thresh_pix` At least one pixel in each island must exceed this threshold of number of times the background rms (default = 5.0).

`thresh_gaus` The peak threshold in sigma for adding more Gaussians to an island. If not given (the default), the threshold will be estimated automatically.

`use_rms_map` Use a 2-D map for the background rms or a constant value? `None/True/False` (default = `None` = determine automatically). If set to `None` or `True`, the rms map is calculated using a moving-window algorithm, where the mean and rms are calculated within a window of a size (`box_size` × `box_size`), and the window is stepped within the image by steps of size `box_step`. The `box_size` and `box_step` values can be calculated automatically or set with the `rms_box` option.

`rms_box` The box size and box step to use when constructing the rms map, defined as (`box_size` [pixels], `box_step` [pixels]). For example, `rms_box = (25, 10)` will use a 25-pixel box stepped across the image in 10-pixel steps. If set to `None` (the default), PyBDSM estimates the box automatically (in this case, see the log file for the box information).

`rms` The background rms value in Jy/beam to use if `use_rms_map = False`. If set to `None` (the default), the rms is estimated from the input image.

`gaussian_maxsize` Maximum allowable size of fitted Gaussians in number of beam areas (default = 10).

`extended` If significant extended emission is expected, try setting this option to `True` (default = `False`). This option simply sets `use_rms_map = False` (to use a constant rms rather than a 2-D map that may be biased by the extended emission) and `gaussian_maxsize = 100` (to allow extended Gaussians to be fit).

A few other, more specialized, options are available. Enter `bdsm.process_image?` or `help(bdsm.process_image)` in IPython for a description of all current options.

If the fit is successful, PyBDSM will return an "Image" object (in this example named "img") which contains the results of the fit (among many other things). A log file with additional information about the fit will be saved in the current directory. A number of tools are available for examining the fit and writing out the source list, residual image, etc. These tools are methods of the Image object returned by `bdsm.process_image()` and are described below:

`img.showfit()` This method shows a quick summary of the fit by plotting the input image with the islands and Gaussians found, along with the model and residual images.

`img.showrms()` Either prints the rms value used (if constant) or shows the 2-D rms map, along with the input image with the islands and Gaussians overlaid.

`img.write_model_img()` Write the model image to a FITS file.

`img.write_resid_img()` Write the residual image to a FITS file.

`img.write_rms_img()` Write the 2-D background rms image to a FITS file.

`img.write_ch0_img()` Write the `ch0` image (the image used for detection) to a FITS file.[2]

`img.write_gaul()` This method writes the Gaussian list to a file. Available options are:

`format = "ascii"/"bbs"/"ds9"/"fits"` Sets the output file format. Default = `"fits"`.

`patches = True/False` If set to `True` (and `format = "bbs"`), patches are used in the BBS sky model. Default = `False`.

`patch_type = "single"/"gaussian"/"source"` Determines whether a single patch is created for all Gaussians or a separate patch is created for each Gaussian or for each source. If patches are defined by source, a patch includes all the Gaussians that belong to a given source. Default = `"source"`.

The information included in the output varies depending on the format used: with the ASCII and FITS formats, all Gaussian parameters are included; with the BBS and ds9 formats, only a subset of parameters are included. Additionally, in the BBS and ds9 output files, sources with sizes smaller than the beam are denoted as point sources. For details of the entries in the Gaussian lists made with `img.write_gaul()`, see Section 5.

[2]Typically the `ch0` image is just the original input image. However, if the input image is a 3- or 4-D cube, the `ch0` image is the result of an average over the frequency axis of the cube.

4 Examples

4.1 Simple Example

A simple example of using PyBDSM on a LOFAR image (an HBA image of 3C61.1) is shown below.

```
> ipython
Python 2.5.2 (r252:60911, Jan 20 2010, 23:14:04)
Type "copyright", "credits" or "license" for more information.
IPython 0.8.1 -- An enhanced Interactive Python.
?       -> Introduction to IPython's features.
%magic  -> Information about IPython's 'magic' % functions.
help    -> Python's own help system.
object? -> Details about 'object'. ?object also works, ?? prints more.

In [1]: import bdsm

In [2]: img_3C61 = bdsm.process_image("sb48.fits")
Opened sb48.fits
Image size : (256, 256) pixels
Beam shape (major, minor, pos angle) : (0.002916, 0.002654, -173.36) degrees
Number of islands found : 4
Fitting islands with Gaussians : #0...#1...#2...#3...done.
Total number of Gaussians fit to image : 12
Number of sources formed by grouping Gaussians : 6

In [3]: img_3C61.showfit()
```

The figure made by `img_3C61.showfit()` is shown in Figure 1. In the plot window, one can zoom in, save the plot to a file, etc. The list of best-fit Gaussians found by PyBDSM may be written to a file for use in other programs, such as BBS, as follows:

```
In [4]: img_3C61.write_gaul(format="bbs")
--> Wrote BBS sky model sb48.pybdsm.gaul.sky
```

The output Gaussian lists contain source positions, fluxes, etc. (see Section 5 for details). Neighboring Gaussians are automatically grouped into sources that can be used as patches in BBS sky models. For example, the command:

```
In [4]: img_3C61.write_gaul(format="bbs", patches=True)
--> Wrote BBS sky model sb48.pybdsm.gaul.sky
```

will write out the same Gaussian list but with the 12 Gaussians grouped into six patches that correspond to the six sources determined by PyBDSM (setting `patch_type = "gaussian"` in the same call will force a separate patch for each Gaussian, and `patch_type = "single"` will put all Gaussians in a single patch).

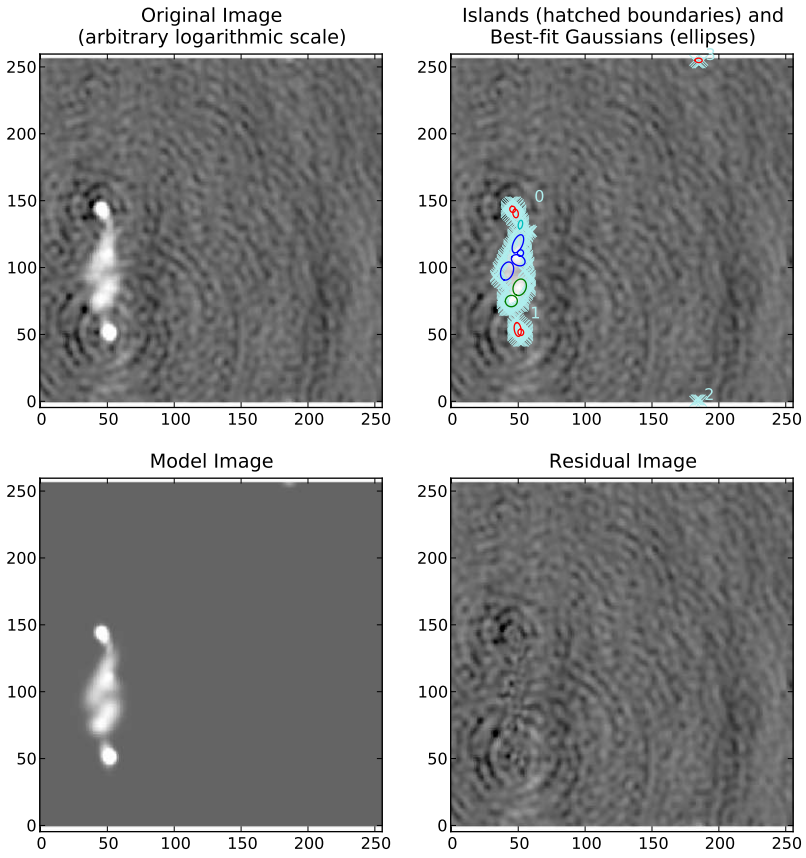

Figure 1: Output of `img_3C61.showfit()`, showing the original image with and without sources, the model image, and the residual (original minus model) image. Boundaries of the islands of emission found by PyBDSM are shown in light blue (with the corresponding island index, which starts counting from zero, also shown) The fitted Gaussians are shown for each island as ellipses (the sizes of which correspond to the FWHMs of the Gaussians). Gaussians that have been grouped together into a source are shown with the same color. For example, the two red Gaussians of island #1 have been grouped together into one source, and the nine Gaussians of island #0 have been grouped into 4 separate sources.

4.2 Image with Artifacts

Occasionally, a PyBDSM run with the default parameters does not produce good results. For example, if there are significant deconvolution artifacts in the image, the `thresh_isl`, `thresh_pix`, or `rms_box` parameters might need to be changed to prevent PyBDSM from fitting Gaussians to such artifacts. An example of running PyBDSM with the default parameters on such an image is shown in Figure 2. It is clear that a number of spurious sources are being detected. Simply raising the threshold for island detection (using the `thresh_isl` parameter) would remove these sources but would also remove many real but faint sources in regions of low rms. Instead, by setting the `rms_box` parameter to better match the typical scale over which the artifacts vary significantly, one obtains much better results. In this example, the scale of the regions affected by artifacts is approximately 20 pixels, whereas PyBDSM used a `rms_box` of 61 pixels when run with the default parameters (as reported in the log file), resulting in an rms map that is over-smoothed. Therefore, one could call `bdsm.process_image()` as follows:

```
img_3C196 = bdsm.process_image("3C196.image", rms_box=(20, 10))
```

Here the rms map is computed using a box of 20 pixels in size with a step size of 10 pixels (i.e., the box is moved across the image in 10-pixel steps). See Figure 3 for a summary of the results of this call.

4.3 Image with Extended Emission

If there is extended emission that fills a significant portion of the image, the background rms map will likely be biased high in regions where extended emission is present (this can be checked with the `img.showrms()` method). Setting `extended = True` in the `bdsm.process_image()` call will force PyBDSM to use a constant rms value across the whole image (i.e., `use_rms_map = False`) and will allow for large Gaussians (`gaussian_maxsize` is set to 100 times the beam area). Depending on the source structure, the `thresh_isl`, `thresh_pix`, and `thresh_gaus` parameters may also have to be adjusted as well to ensure that PyBDSM finds and fits islands of emission properly. An example analysis of an image with significant extended emission is shown in Figure 4.

5 Gaussian List Description

The ASCII and FITS Gaussian lists made with `img.write_gaul()` include the following entries for each Gaussian:

`Gaul_id` Index number of the Gaussian (starts from zero).

`Island_id` Index number of the island the Gaussian belongs to (starts from zero). Islands can be fit with multiple Gaussians.

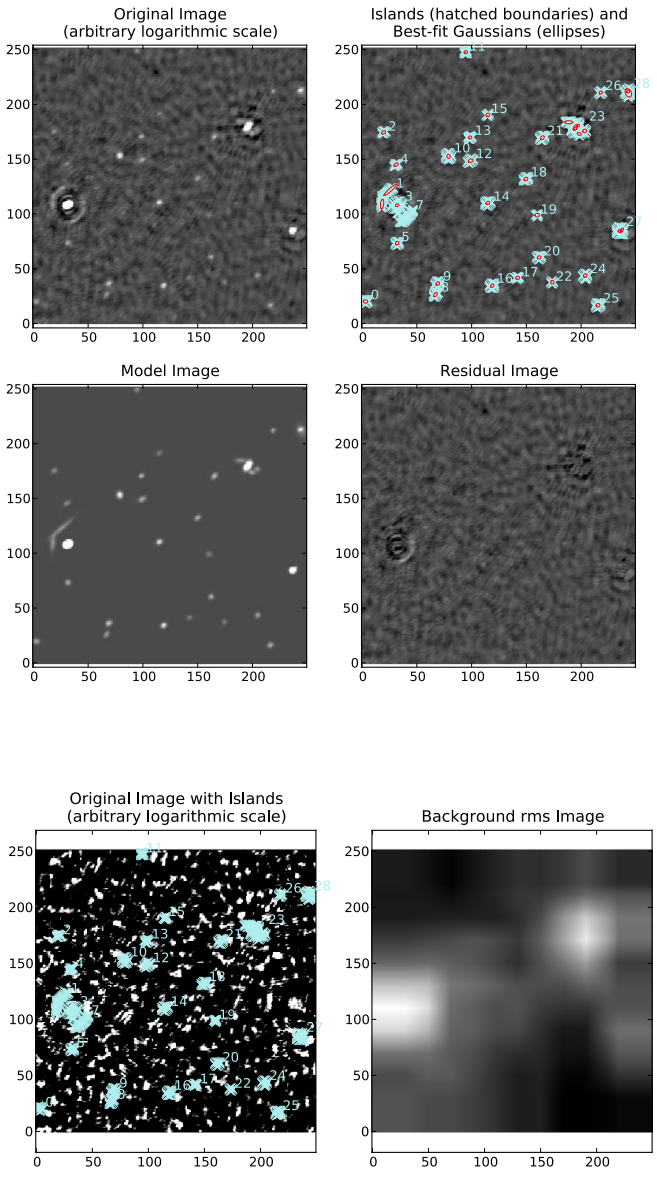

Figure 2: Example fit with default parameters of an image with strong artifacts around bright sources. A number of artifacts near the bright sources are picked up as sources. The background rms map for the same region (produced using `img.showrms()`) is shown in the lower panel: the rms varies fairly slowly across the image, whereas ideally it would increase strongly near the bright sources (reflecting the increased rms in those regions due to the artifacts).

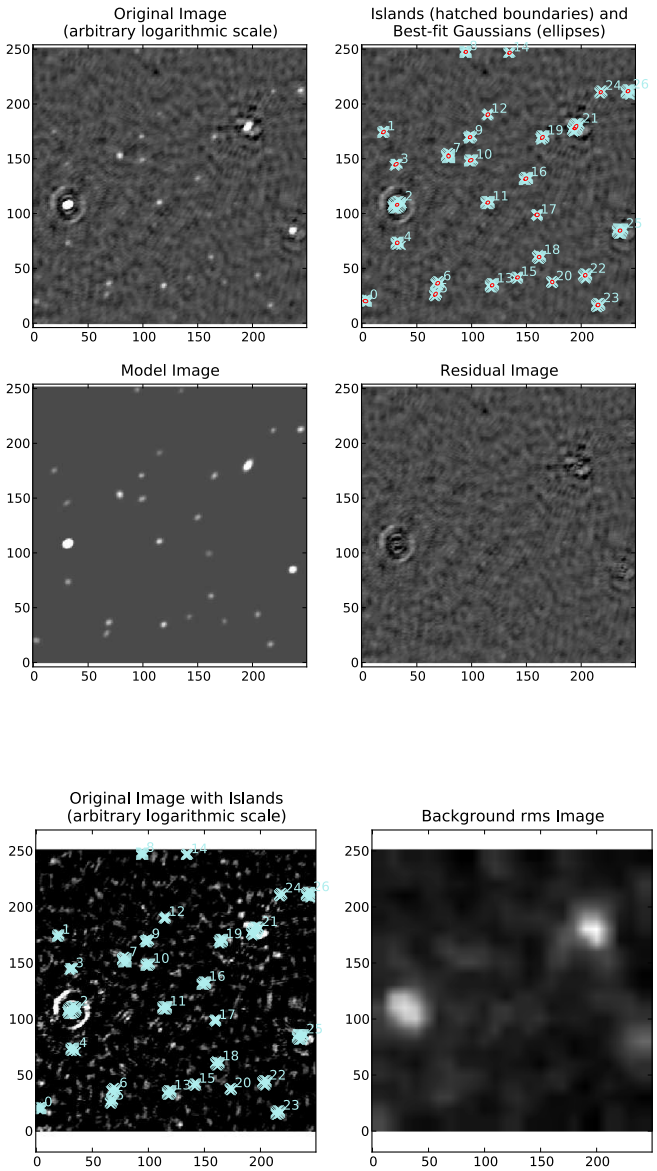

Figure 3: The same as Figure 2. but with `rms_box = (20, 10)`. The rms map now varies on scales similar to that of the regions affected by the artifacts, and both bright and faint sources are recovered properly.

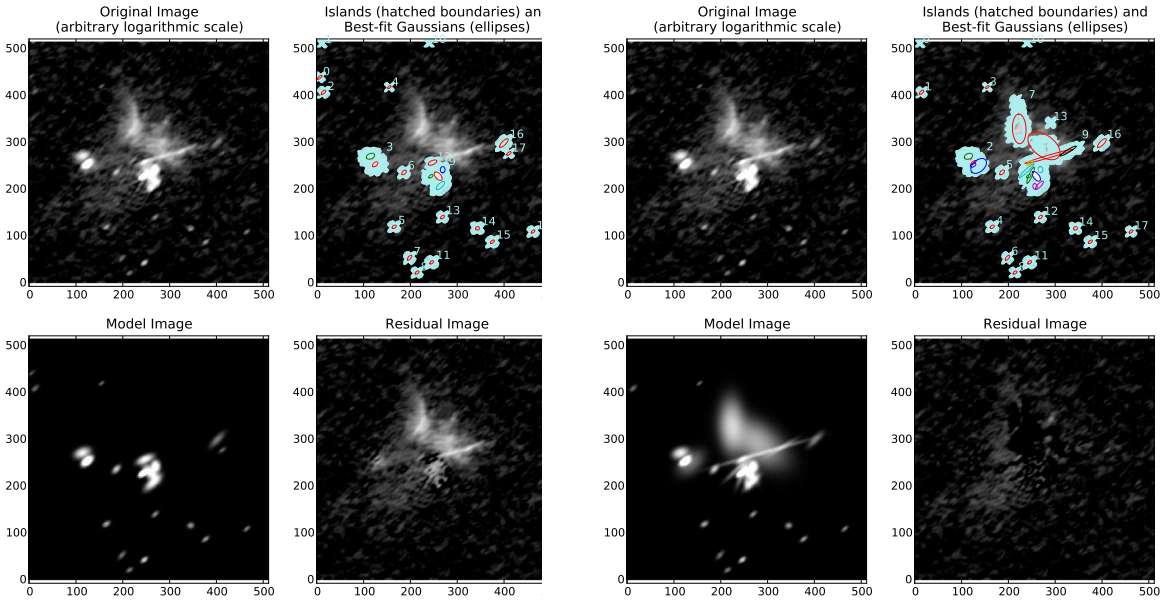

Figure 4: Example fit of an image of A2256 using the default settings of `bdsm.process_image()` (*left*) and using the `extended = True` option (*right*).

`Flag` If flag is zero, the Gaussian is valid. For the meaning of nonzero flags, see Section 2.5.2 of `http://www.strw.leidenuniv.nl/~mohan/anaamika_manual.pdf`.

`Total_flux` Total flux density, in Jy.

`Err_total_flux` Error in total flux density, in Jy.

`Peak_flux` Peak flux density per beam, in Jy/beam.

`Err_peak_flux` Error in peak flux density per beam, in Jy/beam.

`RA` R.A. of center of fitted Gaussian, in degrees.

`Err_RA` Error in R.A., in degrees.

`DEC` Dec of center of fitted Gaussian, in degrees.

`Err_DEC` Error in Dec, in degrees.

`Xpos` X-position of center of fitted Gaussian, in pixels.

`Err_xpos` Error in x-position, in pixels.

`Ypos` Y-position of center of fitted Gaussian, in pixels.

`Err_ypos` Error in y-position, in pixels.

`Bmaj_fw` FWHM of major axis of fitted Gaussian, in arcsec.

`Err_bmaj` Error in FWHM of major axis, in arcsec.

`Bmin_fw` FWHM of minor axis of fitted Gaussian, in arcsec.

`Err_bmin` Error in FWHM of minor axis, in arcsec.

`Bpa` Position angle of fitted Gaussian, east from north, in degrees.

`Err_bpa` Error in position angle, in degrees.