galmask

Release 0.1.0

Yash Gondhalekar, Rafael S. de Souza

Mar 08, 2023

CONTENTS:

galmask is an open-source package written in Python that provides a simple way to remove unwanted background source detections from galaxy images. It builds on top of *astropy* and *photutils* astronomical Python libraries and the *opencv* and *skimage* image processing libraries.

galmask works as follows: Given a galaxy image, and optionally an estimate of the segmentation map, it gets rid of background detections around the galaxy which typically affect downstream analysis concerned with the morphology of galaxies. To achieve this, it efficiently uses the concepts of deblending and optionally, connected-component analysis.

CHAPTER

ONE

INSTALLATION

1.1 Requirements

- *astropy* for handling FITS I/O and general-purpose astronomical routines.
- *photutils* for photometry purposes and deblending detected sources.
- *opencv-python* for connected-component analysis.
- *skimage* for general image processing functionalities.

1.2 Via *pip*

galmask can be installed from PyPI via *pip* by running:

```
pip install galmask
```

1.3 Alternative method

galmask can also be installed by cloning the repository and doing a pip install in the project directory:

```
git clone https://github.com/Yash-10/galmask
cd galmask
pip install .
```

CHAPTER

TWO

EXAMPLE USAGE

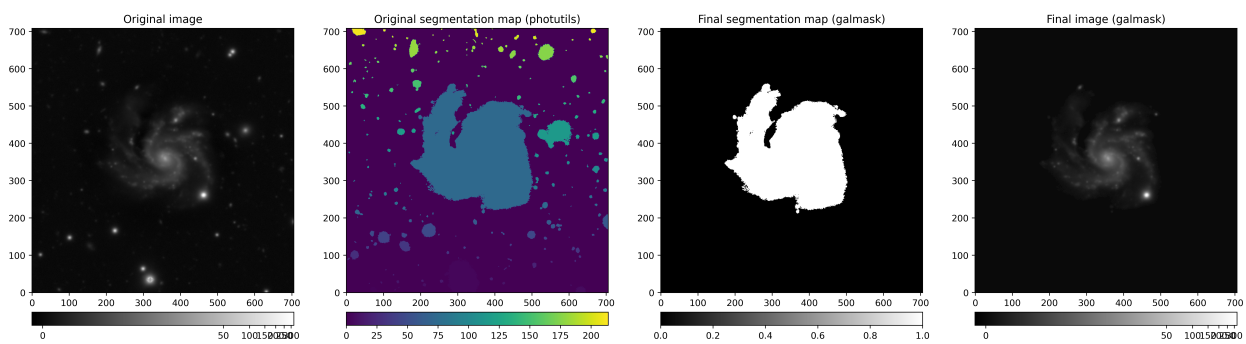
```
from astropy.io import fits
from astropy.visualization import AsinhStretch, ImageNormalize, ZScaleInterval,␣
↪LogStretch

import matplotlib.pyplot as plt
from mpl_toolkits.axes_grid1 import make_axes_locatable

# Import galmask
from galmask.galmask import galmask

def axes_colorbar(ax):
   divider = make_axes_locatable(ax)
   cax = divider.append_axes('bottom', size='5%', pad=0.3)
   return cax

filepath = 'example/gal2_R.fits'
image = fits.getdata(filepath)
npixels, nlevels, nsigma, contrast, min_distance, num_peaks, num_peaks_per_label,␣
↪connectivity, remove_local_max = 5, 32, 2., 0.15, 1, 10, 3, 4, True  # Parameters for␣
↪galmask
seg_image = None  # No segmentation map example
orig_segmap = fits.getdata('example/gal2_orig_segmap_R.fits')

galmasked, galsegmap = galmask(
   image, npixels, nlevels, nsigma, contrast, min_distance, num_peaks, num_peaks_per_
↪label,
   connectivity=4, kernel=fits.getdata('kernel.fits'), seg_image=seg_image, mode="1",
   remove_local_max=True, deblend=True
)

# Plotting result.
fig, ax = plt.subplots(1, 4, figsize=(24, 6))

# For keeping original and final images on same scale.
vmin = min(image.min(), galmasked.min())
vmax = max(image.max(), galmasked.max())

# fig.suptitle(filepath)
norm1 = ImageNormalize(image, vmin=vmin, vmax=vmax, interval=ZScaleInterval(),␣
↪stretch=LogStretch())
```

(continues on next page)

(continued from previous page)

```
im0 = ax[0].imshow(image, norm=norm1, origin='lower', cmap='gray')
ax[0].set_title("Original image")
cax0 = axes_colorbar(ax[0])
fig.colorbar(im0, cax=cax0, orientation='horizontal')

im1 = ax[1].imshow(orig_segmap, origin='lower')
ax[1].set_title("Original segmentation map (photutils)")
cax1 = axes_colorbar(ax[1])
fig.colorbar(im1, cax=cax1, orientation='horizontal')

im2 = ax[2].imshow(galsegmap, origin='lower', cmap='gray')
ax[2].set_title("Final segmentation map (galmask)")
cax2 = axes_colorbar(ax[2])
fig.colorbar(im2, cax=cax2, orientation='horizontal')

norm2 = ImageNormalize(galmasked, vmin=vmin, vmax=vmax, interval=ZScaleInterval(),␣
↪stretch=LogStretch())
im3 = ax[3].imshow(galmasked, norm=norm2, origin='lower', cmap='gray')
ax[3].set_title("Final image (galmask)")
cax3 = axes_colorbar(ax[3])
fig.colorbar(im3, cax=cax3, orientation='horizontal')

plt.show()
```

Output:



It returns the final segmentation map (column 3) along with the final galaxy image (last column) which can now be used in downstream analyses.

Note: It is important to note that `galmask` returns the final image in which the background pixels are set to zero. However, you could replace all such pixels with a background estimated from the original input image.

CHAPTER

THREE

GENERAL TIPS

Note that most of the parameters like `npixels`, `nlevels`, etc. are passed directly to downstream methods:

- `npixels` is passed to the `detect_sources` method from `photutils`.
- `npixels`, `nlevels`, and `contrast` are passed to the `deblend_sources` method from `photutils`.
- `min_distance`, `num_peaks`, and `num_peaks_per_label` are passed to the `peak_local_max` function from `skimage`.
- `connectivity` is used for the connected-component analysis from `opencv-python`.

Hence, a better understanding and usage of these parameters can be seen from their respective documentation.

Here are some empirical notes and tips that could be of interest:

1. You might want to set `deblend = True` if there are nearby sources in your image.
2. Using 8-connectivity tends to maximize connection of objects together. So use 4-connectivity if you do not want to maximize the connection.
3. For better performance, it might be helpful to input a custom `kernel` and `seg_image` since it alleviates some internal calculations.
4. If unsure, set `remove_local_max = True`.
5. The `mode` argument is essential since the results significantly depend on this value. `mode = 1` could work well for one image but not for the other. **Although mode = 1 is the default, you might want to experiment with the other options to choose the best option for your image. So, you might need to try all the three possible modes**. We empirically find at least one mode out of the possible three modes to work for any given image.
6. The results depend on the segmentation map, if any, input to galmask since it used as a basis for further cleaning the map. So please ensure that your segmentation map is plausible.
7. If you want to input a custom segmentation map, we would recommend using the NoiseChisel program, which does a great job detecting nebulous objects like irregular galaxies and helps particularly in detecting fainter outskirts of a galaxy.

CHAPTER

FOUR

RUNNING TESTS AND BUILDING THE DOCUMENTATION

To run tests locally, you would need to install pytest. Once done, you can navigate to the *tests/* directory and run, for example:

```
pytest test_galmask.py
```

and it should run without any failures!

If you would like to build the documentation locally, you can do:

```
cd docs/
make html
python -m http.server
```

You can then open the url http://0.0.0.0:8000/ in your browser.

CHAPTER

FIVE

REFERENCES AND ACKNOWLEDGMENTS

1. Astropy Collaboration, Robitaille, T. P., Tollerud, E. J., et al. 2013, A&A, 558, A33, doi: 10.1051/0004-6361/201322068
2. Astropy Collaboration, Price-Whelan, A. M., Sip˝ocz, B. M., et al. 2018, AJ, 156, 123, doi: 10.3847/1538-3881/aabc4f
3. Bradley, L., Sipocz, B., Robitaille, T., et al. 2016, Photutils: Photometry tools, Astrophysics Source Code Library, record ascl:1609.011. http://ascl.net/1609.011
4. Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. scikit-image: Image processing in Python. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453
5. Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.

- This research made use of Astropy, a community-developed core Python package for Astronomy (Robitaille et al., 2013, Price-Whelan et al., 2018)
- This research made use of Photutils, an Astropy package for detection and photometry of astronomical sources (Bradley et al. 2022).

CHAPTER

SIX

NOTE ON REPLACING BACKGROUND PIXEL VALUES

Instead of keeping the background pixels set to zero, if you would like to replace it with a background estimate post using galmask, you can do so using the below code template (it is inspired by GalClean's source code):

```
import numpy as np
import matplotlib.pyplot as plt

from photutils.segmentation import deblend_sources, detect_sources, detect_threshold
from astropy.convolution import convolve, Gaussian2DKernel
from astropy.stats import gaussian_fwhm_to_sigma

def get_segmap(image):
    """Creates a segmentation map of the original image, `image`."""
    npixels = 5
    nsigma = 3.
    sigma = 3.0 * gaussian_fwhm_to_sigma
    kernel = Gaussian2DKernel(sigma, x_size=3, y_size=3)
    kernel.normalize()
    kernel = kernel.array
    bkg_level = MedianBackground().calc_background(image)
    image_bkg_subtracted = image - bkg_level
    convolved_data = convolve(image_bkg_subtracted, kernel, normalize_kernel=True)
    threshold = detect_threshold(image_bkg_subtracted, nsigma=nsigma, background=0.0)
    objects = detect_sources(convolved_data, threshold, npixels=npixels).data
    return objects

objects = get_segmap(image)  # image is the original galaxy image input to galmask.
bkgmap = np.ma.masked_equal(objects, 0).mask  # Assume background pixels are set to zero.
bkgmap = bkgmap.astype(int)
to_replace = np.random.choice((bkgmap * image).ravel(), bkgmap.shape).ravel()  # Select␣
↪random elements from the background pixel map defined above.
if to_replace.shape[0] <= galmask_image[galmask_image == 0].shape[0]:  # If available␣
↪background pixels to sample is less than the required.
    extra = np.random.choice((bkgmap * image).ravel(), galmask_image[galmask_image == 0].
↪shape[0] - to_replace.shape[0])
    galmask_image[galmask_image == 0] = (np.concatenate([to_replace, extra]) / 100)
else:
    _shape = galmask_image[galmask_image == 0].shape
    galmask_image[galmask_image == 0] = (to_replace[:_shape[0]] / 100)  # Divide by some␣
↪large enough number (here, 100) to prevent background to be too high.
```

(continues on next page)

(continued from previous page)

```
plt.imshow(galmask_image)
plt.show()
```

Note: This is only a template and is **not** optimized or tested extensively. However, it can still be used (without guarantee) if you do not want all background pixels set to zero.

6.1 galmask

6.1.1 galmask package

Submodules

galmask.gal_mask module

galmask.utils module

Module contents

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

g

INDEX

G

M