JointModels

May 10, 2023

1 Joint Modelling

In this tutorial you will learn how to set up a joint modelling fit which encoporates the data from multiple images. These use `Group_Model` objects just like in the `GroupModels.ipynb` tutorial, the main difference being how the `Target_Image` object is constructed and that more care must be taken when assigning targets to models.

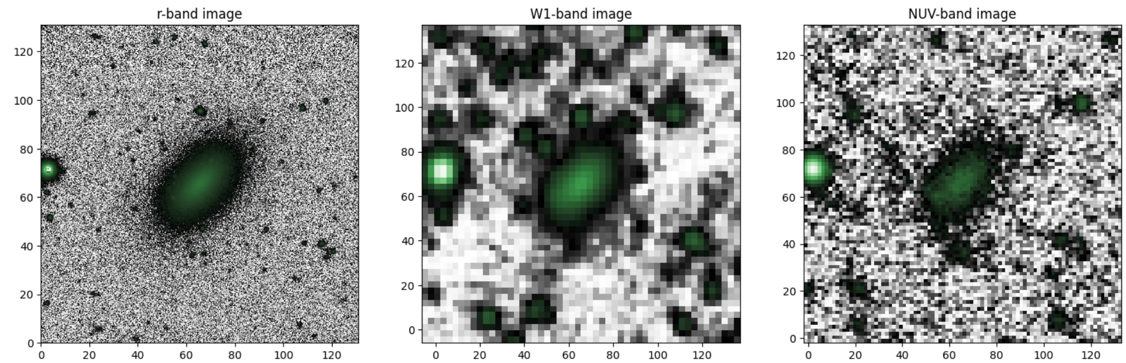
It is, of course, more work to set up a fit across multiple target images. However, the tradeoff can be well worth it. Perhaps there is space-based data with high resolution, but groundbased data has better S/N. Or perhaps each band individually does not have enough signal for a confident fit, but all three together just might. Perhaps colour information is of paramount importance for a science goal, one would hope that both bands could be treated on equal footing but in a consistent way when extracting profile information. There are a number of reasons why one might wish to try and fit a multi image picture of a galaxy simultaneously.

When fitting multiple bands one often resorts to forced photometry, somtimes also blurring each image to the same approximate PSF. With AutoProf this is entirely unecessary as one can fit each image in its native PSF simultaneously. The final fits are more meaningful and can encorporate all of the available structure information.

[1]:
```
import autoprof as ap
import numpy as np
import torch
from astropy.io import fits
import matplotlib.pyplot as plt
from scipy.stats import iqr
```

[2]:
```
# First we need some data to work with, let's use LEDA 41136 as our example galaxy

# Our first image is from the DESI Legacy-Survey r-band. This image has a pixelscale of 0.262 arcsec/pixel and is 500 pixels across
target_r = ap.image.Target_Image(
    data = np.array(fits.open("https://www.legacysurvey.org/viewer/fits-cutout?ra=187.3119&dec=12.9783&size=500&layer=ls-dr9&pixscale=0.262&bands=r")[0].data, dtype = np.float64),
    pixelscale = 0.262,
    zeropoint = 22.5,
```

```python
    variance = np.ones((500,500))*0.008**2, # note that the variance is important to ensure all images are compared with proper statistical weight. Here we just use the IQR^2 of the pixel values as the variance, for science data one would use a more accurate variance value
    psf = ap.utils.initialize.gaussian_psf(1.12/2.355, 51, 0.262) # we construct a basic gaussian psf for each image by giving the simga (arcsec), image width (pixels), and pixelscale (arcsec/pixel)
)

# The second image is a unWISE W1 band image. This image has a pixelscale of 2.75 arcsec/pixel and is 52 pixels across
target_W1 = ap.image.Target_Image(
    data = np.array(fits.open("https://www.legacysurvey.org/viewer/fits-cutout?ra=187.3119&dec=12.9783&size=52&layer=unwise-neo7&pixscale=2.75&bands=1")[0].data, dtype = np.float64),
    pixelscale = 2.75,
    zeropoint = 25.199,
    variance = np.ones((52,52))*4.9**2,
    psf = ap.utils.initialize.gaussian_psf(6.1/2.355, 21, 2.75),
    origin = (np.array([500,500]))*0.262/2 - (np.array([52,52]))*2.75/2, # here we ensure that the images line up by slightly adjusting the origin
)

# The third image is a GALEX NUV band image. This image has a pixelscale of 1.5 arcsec/pixel and is 90 pixels across
target_NUV = ap.image.Target_Image(
    data = np.array(fits.open("https://www.legacysurvey.org/viewer/fits-cutout?ra=187.3119&dec=12.9783&size=90&layer=galex&pixscale=1.5&bands=n")[0].data, dtype = np.float64),
    pixelscale = 1.5,
    zeropoint = 20.08,
    variance = np.ones((90,90))*0.0007**2,
    psf = ap.utils.initialize.gaussian_psf(5.4/2.355, 21, 1.5),
    origin = (np.array([500,500]))*0.262/2 - (np.array([90,90]))*1.5/2,
)

fig1, ax1 = plt.subplots(1, 3, figsize = (18,6))
ap.plots.target_image(fig1, ax1[0], target_r)
ax1[0].set_title("r-band image")
ap.plots.target_image(fig1, ax1[1], target_W1)
ax1[1].set_title("W1-band image")
ap.plots.target_image(fig1, ax1[2], target_NUV)
ax1[2].set_title("NUV-band image")
plt.show()
```



[3]:
```
# The joint model will need a target to try and fit, but now that we have␣
↪multiple images the "target" is
# a Target_Image_List object which points to all three.
target_full = ap.image.Target_Image_List((target_r, target_W1, target_NUV))
# It doesn't really need any other information since everything is already␣
↪available in the individual targets
```

[4]:
```
# To make things easy to start, lets just fit a sersic model to all three. In␣
↪principle one can use arbitrary
# group models designed for each band individually, but that would be␣
↪unecessarily complex for a tutorial

model_r = ap.models.AutoProf_Model(
    name = "rband model",
    model_type = "sersic galaxy model",
    target = target_r,
    psf_mode = "full",
)
model_W1 = ap.models.AutoProf_Model(
    name = "W1band model",
    model_type = "sersic galaxy model",
    target = target_W1,
    psf_mode = "full",
)
model_NUV = ap.models.AutoProf_Model(
    name = "NUVband model",
    model_type = "sersic galaxy model",
    target = target_NUV,
    psf_mode = "full",
)

# At this point we would just be fitting three separate models at the same␣
↪time, not very interesting. Next
```

```
# we add constraints so that some parameters are shared between all the models.␣
↪It makes sense to fix
# structure parameters while letting brightness parameters vary between bands␣
↪so that's what we do here.
model_W1.add_equality_constraint(model_r, ["center", "q", "PA", "n", "Re"])
model_NUV.add_equality_constraint(model_r, ["center", "q", "PA", "n", "Re"])
# Now every model will have a unique Ie, but every other parameter is shared␣
↪for all three
```

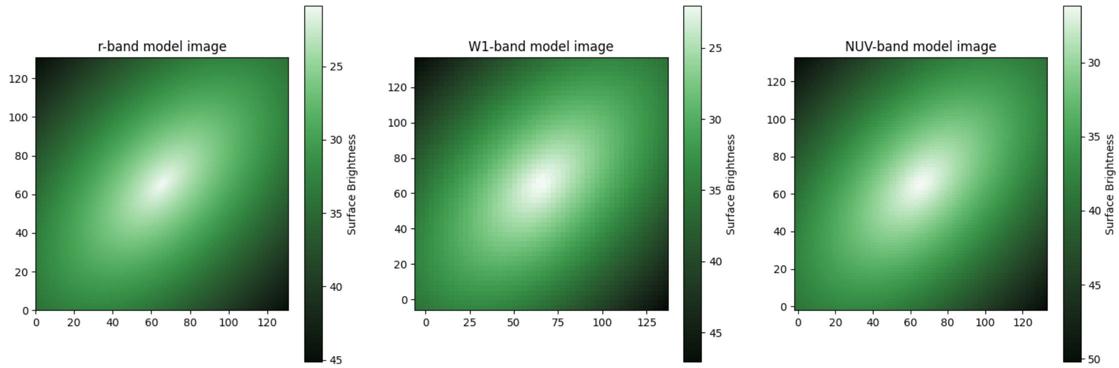
[5]:
```
# We can now make the joint model object

model_full = ap.models.AutoProf_Model(
    name = "LEDA 41136",
    model_type = "group model",
    model_list = [model_r, model_W1, model_NUV],
    target = target_full,
)

model_full.initialize()
```

[6]:
```
result = ap.fit.LM(model_full, verbose = 1).fit()
print(result.message)
```

```
L: 1.0
---------init---------
LM loss: 93.27189078643637
L: 1.0
---------iter---------
LM loss: 93.25876606427266
accept
L: 0.1111111111111111
---------iter---------
LM loss: 93.24679451536917
accept
L: 0.012345679012345678
---------iter---------
LM loss: 93.24392242400215
accept
L: 0.0013717421124828531
---------iter---------
LM loss: 93.2437177834453
accept
success
```

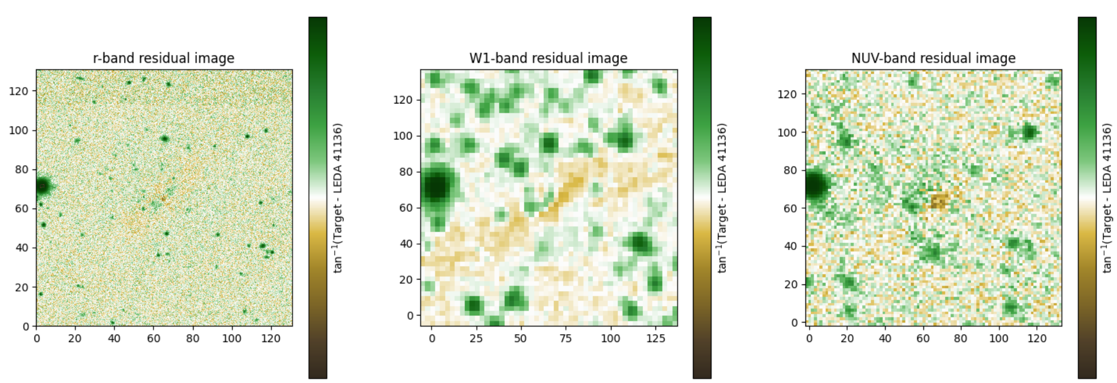
[7]:
```
# here we plot the results of the fitting, notice that each band has a␣
↪different PSF and pixelscale. Also, notice
```

```
# that the colour bars represent significantly different ranges since each model was allowed to fit its own Ie.
# meanwhile the center, PA, q, and Re is the same for every model.
fig1, ax1 = plt.subplots(1, 3, figsize = (18,6))
ap.plots.model_image(fig1, ax1, model_full)
ax1[0].set_title("r-band model image")
ax1[1].set_title("W1-band model image")
ax1[2].set_title("NUV-band model image")
plt.show()
```



[8]:

```
# We can also plot the residual images. As can be seen, the galaxy is fit in all three bands simultaneously
# with the majority of the light removed in all bands. A residual can be seen in the r band. This is likely
# due to there being more structure in the r-band than just a sersic. The W1 and NUV bands look excellent though
fig1, ax1 = plt.subplots(1, 3, figsize = (18,6))
ap.plots.residual_image(fig1, ax1, model_full)
ax1[0].set_title("r-band residual image")
ax1[1].set_title("W1-band residual image")
ax1[2].set_title("NUV-band residual image")
plt.show()
```



1.1 Joint models with multiple models

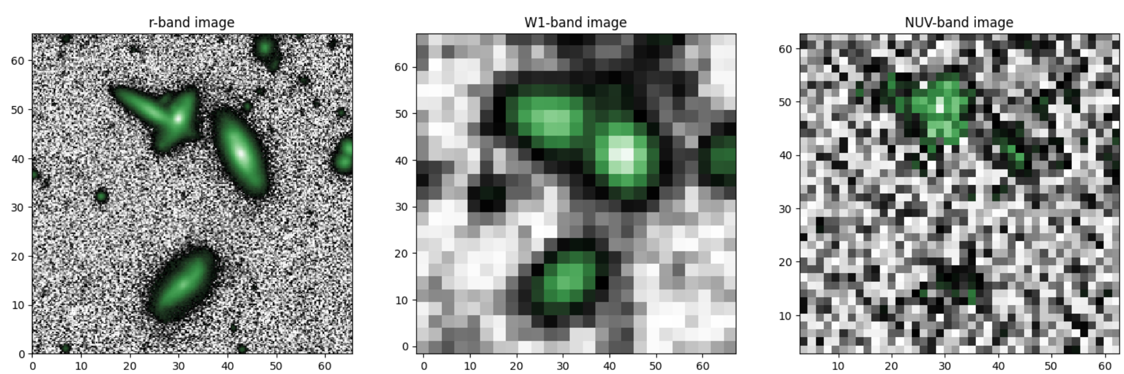
If you want to analyze more than a single astronomical object, you will need to combine many models for each image in a reasonable structure. There are a number of ways to do this that will work, though may not be as scalable. For small images, just about any arrangement is fine when using the LM optimizer. But as images and number of models scales very large, it may be neccessary to sub divide the problem to save memory. To do this you should arrange your models in a hierarchy so that AutoProf has some information about the structure of your problem. There are two ways to do this. First, you can create a group of models where each sub-model is a group which holds all the objects for one image. Second, you can create a group of models where each sub-model is a group which holds all the representations of a single astronomical object across each image. The second method is preferred. See the diagram below to help clarify what this means.

JointGroupModels

Here we will see an example of a multiband fit of an image which has multiple astronomical objects.

[9]:
```
# First we need some data to work with, let's use another LEDA object, this␣
↪time a group of galaxies: LEDA 389779, 389797, 389681

RA = 320.5003
DEC = -57.4585
# Our first image is from the DESI Legacy-Survey r-band. This image has a␣
↪pixelscale of 0.262 arcsec/pixel
rsize = 250
target_r = ap.image.Target_Image(
    data = np.array(fits.open(f"https://www.legacysurvey.org/viewer/fits-cutout?
↪ra={RA}&dec={DEC}&size={rsize}&layer=ls-dr9&pixscale=0.262&bands=r")[0].
↪data, dtype = np.float64),
    pixelscale = 0.262,
    zeropoint = 22.5,
    variance = np.ones((rsize,rsize))*0.008**2, # note that the variance is␣
↪important to ensure all images are compared with proper statistical weight.␣
↪Here we just use the IQR^2 of the pixel values as the variance, for science␣
↪data one would use a more accurate variance value
    psf = ap.utils.initialize.gaussian_psf(1.12/2.355, 51, 0.262) # we␣
↪construct a basic gaussian psf for each image by giving the simga (arcsec),␣
↪image width (pixels), and pixelscale (arcsec/pixel)
)

# The second image is a unWISE W1 band image. This image has a pixelscale of 2.
↪75 arcsec/pixel
wsize = 25
target_W1 = ap.image.Target_Image(
```

```
    data = np.array(fits.open(f"https://www.legacysurvey.org/viewer/fits-cutout?
↪ra={RA}&dec={DEC}&size={wsize}&layer=unwise-neo7&pixscale=2.75&bands=1")[0].
↪data, dtype = np.float64),
    pixelscale = 2.75,
    zeropoint = 25.199,
    variance = np.ones((wsize,wsize))*4.9**2,
    psf = ap.utils.initialize.gaussian_psf(6.1/2.355, 21, 2.75),
    origin = (np.array([rsize,rsize]))*0.262/2 - (np.array([wsize,wsize]))*2.75/
↪2, # here we ensure that the images line up by slightly adjusting the origin
)

# The third image is a GALEX NUV band image. This image has a pixelscale of 1.5␣
↪arcsec/pixel
gsize = 40
target_NUV = ap.image.Target_Image(
    data = np.array(fits.open(f"https://www.legacysurvey.org/viewer/fits-cutout?
↪ra={RA}&dec={DEC}&size={gsize}&layer=galex&pixscale=1.5&bands=n")[0].data,␣
↪dtype = np.float64),
    pixelscale = 1.5,
    zeropoint = 20.08,
    variance = np.ones((gsize,gsize))*0.0007**2,
    psf = ap.utils.initialize.gaussian_psf(5.4/2.355, 21, 1.5),
    origin = (np.array([rsize,rsize]))*0.262/2 - (np.array([gsize,gsize]))*1.5/
↪2,
)
target_full = ap.image.Target_Image_List((target_r, target_W1, target_NUV))

fig1, ax1 = plt.subplots(1, 3, figsize = (18,6))
ap.plots.target_image(fig1, ax1, target_full)
ax1[0].set_title("r-band image")
ax1[1].set_title("W1-band image")
ax1[2].set_title("NUV-band image")
plt.show()
```

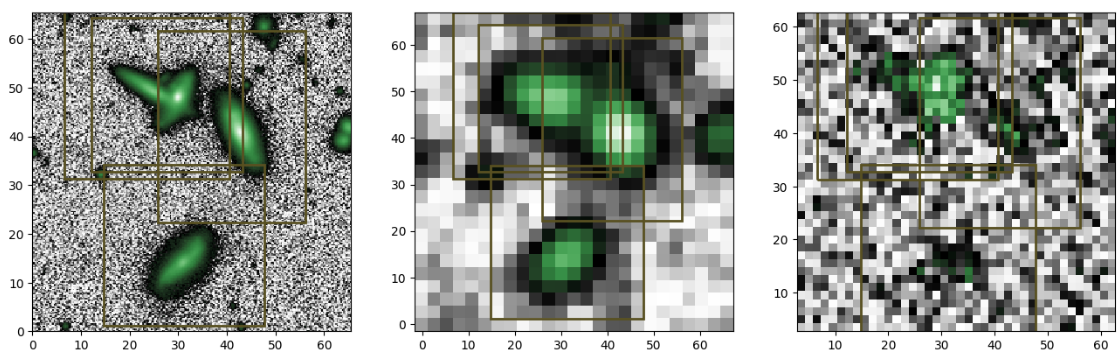


There is barely any signal in the GALEX data and it would be entirely impossible to analyze on its own. With simultaneous multiband fitting it is a breeze to get relatively robust results!

Next we need to construct models for each galaxy. This is understandably more complex than in the single band case, since now we have three times the amout of data to keep track of. Recall that we will create a number of joint models to represent each astronomical object, then put them all together in a larger group model.

[10]:
```
# Here we enter the window parameters by hand, in general one would use a␣
↪segmentation map or some other automated proceedure to pick out the area for␣
↪many objects
windows = [
    {"r":[[72,152],[140,234]], "W1": [[5,16],[13,24]], "NUV": [[8,27],[20,39]]},
    {"r":[[43,155],[138,237]], "W1": [[3,15],[12,25]], "NUV": [[4,22],[19,39]]},
    {"r":[[115,210],[100,228]], "W1": [[10,21],[10,23]], "NUV":␣
↪[[17,35],[13,38]]},
    {"r":[[69,170],[10,115]], "W1": [[7,17],[1,13]], "NUV": [[8,30],[1,18]]},
]

model_list = []

for i, window in enumerate(windows):
    # create the submodels for this object
    sub_list = []
    sub_list.append(
        ap.models.AutoProf_Model(
            name = f"rband model {i}",
            model_type = "spline galaxy model", # we use spline models for the␣
↪r-band since it is well resolved
            target = target_r,
            window = window["r"],
            psf_mode = "full",
        )
    )
    sub_list.append(
        ap.models.AutoProf_Model(
            name = f"W1band model {i}",
            model_type = "sersic galaxy model", # we use sersic models for W1␣
↪and NUV since there isn't much visible detail, a simple model is sufficient
            target = target_W1,
            window = window["W1"],
            psf_mode = "full",
        )
    )
    sub_list.append(
        ap.models.AutoProf_Model(
            name = f"NUVband model {i}",
            model_type = "sersic galaxy model",
```

```
            target = target_NUV,
            window = window["NUV"],
            psf_mode = "full",
        )
    )
    # ensure equality constraints
    sub_list[1].add_equality_constraint(sub_list[0], ["center", "q", "PA"])
    sub_list[2].add_equality_constraint(sub_list[0], ["center", "q", "PA"])
    # Make the multiband model for this object
    model_list.append(
        ap.models.AutoProf_Model(
            name = f"model {i}",
            model_type = "group model",
            target = target_full,
            model_list = sub_list,
        )
    )
# Make the full model for this system of objects
MODEL = ap.models.AutoProf_Model(
    name = f"full model",
    model_type = "group model",
    target = target_full,
    model_list = model_list,
)
fig, ax = plt.subplots(1,3, figsize = (16,7))
ap.plots.target_image(fig, ax, MODEL.target)
ap.plots.model_window(fig, ax, MODEL)
ax1[0].set_title("r-band image")
ax1[1].set_title("W1-band image")
ax1[2].set_title("NUV-band image")
plt.show()
```



[11]:
```
MODEL.initialize()
result = ap.fit.LM(MODEL, verbose = 1, epsilon4 = 0.05).fit()
```

```python
print(result.message)
```

```
L: 1.0
---------init---------
LM loss: 6.614283194453883
L: 1.0
---------iter---------
LM loss: 61974424.04790079
reject
L: 11.0
---------iter---------
LM loss: 5.343013355499507
accept
L: 1.2222222222222223
---------iter---------
LM loss: 2228268.716430656
reject
L: 13.444444444444446
---------iter---------
LM loss: 4.718589336258133
accept
L: 1.4938271604938274
---------iter---------
LM loss: 639570.12796616
reject
L: 16.4320987654321
---------iter---------
LM loss: 4.286010682555317
accept
L: 1.825788751714678
---------iter---------
LM loss: 2613.5315651224096
reject
L: 20.08367626886146
---------iter---------
LM loss: 3.940260728579258
accept
L: 2.231519585429051
---------iter---------
LM loss: 2.905211826311027
accept
L: 0.2479466206032279
---------iter---------
LM loss: 8197210.229638672
reject
L: 2.727412826635507
---------iter---------
```

```
LM loss: 2.6284840718047606
accept
L: 0.3030458696261674
---------iter---------
LM loss: 8755107.447491184
reject
L: 3.3335045658878415
---------iter---------
LM loss: 2.4620521148165246
accept
L: 0.3703893962097602
---------iter---------
LM loss: 8647366.3672287
reject
L: 4.074283358307362
---------iter---------
LM loss: 2.3716787735326537
accept
L: 0.4526981509230402
---------iter---------
LM loss: 8592091.538973933
reject
L: 4.979679660153442
---------iter---------
LM loss: 2.248851023502802
accept
L: 0.5532977400170491
---------iter---------
LM loss: 8511039.630616788
reject
L: 6.08627514018754
---------iter---------
LM loss: 2.2100794371807995
accept
L: 0.6762527933541711
---------iter---------
LM loss: 8274717.406400003
reject
L: 7.438780726895882
---------iter---------
LM loss: 2.1767836521967765
accept
L: 0.8265311918773202
---------iter---------
LM loss: 7845304.350631175
reject
L: 9.091843110650522
---------iter---------
```

```
LM loss: 2.15116256845713
accept
L: 1.0102047900722804
---------iter---------
LM loss: 6746642.9278644025
reject
L: 11.112252690795085
---------iter---------
LM loss: 2.108461869687026
accept
L: 1.2346947434216762
---------iter---------
LM loss: 1466235.4190541867
reject
L: 13.581642177638438
---------iter---------
LM loss: 2.0756463436869095
accept
L: 1.5090713530709374
---------iter---------
LM loss: 1.9739142702611336
accept
L: 0.16767459478565971
---------iter---------
LM loss: 1.4862508748507375
accept
L: 0.018630510531739967
---------iter---------
LM loss: 14.353349260196437
reject
L: 0.20493561584913964
---------iter---------
LM loss: 1.322716770871229
accept
L: 0.022770623983237738
---------iter---------
LM loss: 8.958585331893877
reject
L: 0.2504768638156151
---------iter---------
LM loss: 1.2496423993359536
accept
L: 0.02783076264617946
---------iter---------
LM loss: 31.244170862034323
reject
L: 0.30613838910797403
---------iter---------
```

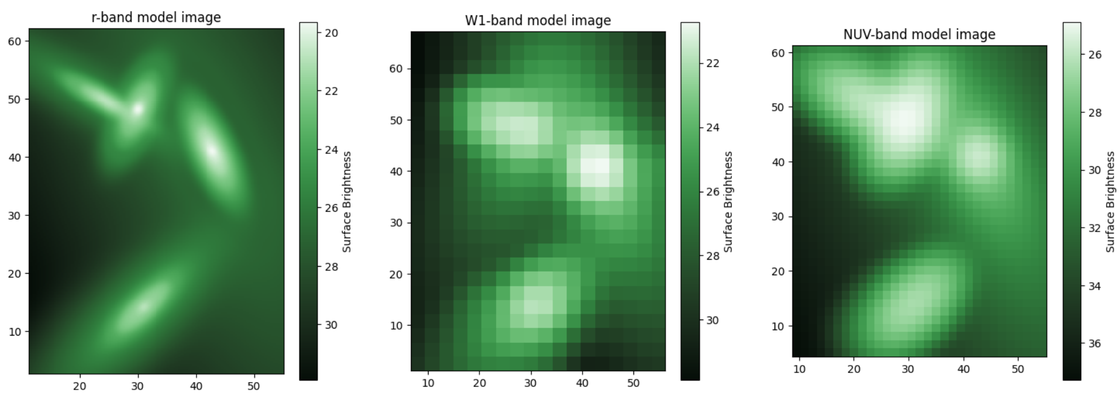
```
LM loss: 1.1908732161609792
accept
L: 0.03401537656755267
---------iter---------
LM loss: 7.293291139299581
reject
L: 0.3741691422430794
---------iter---------
LM loss: 1.1654830892643069
accept
L: 0.041574349138119936
---------iter---------
LM loss: 73201.84729726113
reject
L: 0.4573178405193193
---------iter---------
LM loss: 1.152340492262594
accept
L: 0.05081309339103548
---------iter---------
LM loss: 9424066835.010786
reject
L: 0.5589440273013903
---------iter---------
LM loss: 1.140970096481292
accept
L: 0.0621048919223767
---------iter---------
LM loss: 2.7236481207062293
reject
L: 0.6831538111461437
---------iter---------
LM loss: 1.0750957987511083
accept
L: 0.07590597901623819
---------iter---------
LM loss: 1.0693348271682779
accept
L: 0.00843399766847091
---------iter---------
LM loss: 1.0498354230638112
accept
L: 0.0009371108520523232
---------iter---------
LM loss: nan
nan loss
L: 0.010308219372575556
---------iter---------
```

```
LM loss: 1.0351355601783978
accept
L: 0.0011453577080639506
---------iter---------
LM loss: 1.3792005519189724e+239
reject
L: 0.012598934788703458
---------iter---------
LM loss: 1.6693248319936967e+184
reject
L: 0.13858828267573803
---------iter---------
LM loss: 9.028255882972026e+21
reject
L: 1.5244711094331183
---------iter---------
LM loss: 1.0030347882739288
accept
L: 0.16938567882590203
---------iter---------
LM loss: 1.0022821231726586
accept
L: 0.01882063098065578
---------iter---------
LM loss: 0.9972510206199382
accept
L: 0.0020911812200728646
---------iter---------
LM loss: 1.0082520500714431
reject
L: 0.02300299342080151
---------iter---------
LM loss: 0.9967343031858382
accept
L: 0.0025558881578668347
---------iter---------
LM loss: 1.0031823939532645
reject
L: 0.02811476973653518
---------iter---------
LM loss: 0.9964956343632326
accept
L: 0.0031238633040594644
---------iter---------
LM loss: 1.0011739467676926
reject
L: 0.03436249634465411
---------iter---------
```

```
LM loss: 0.9963870803664229
accept
L: 0.003818055149406012
---------iter---------
LM loss: 0.9992297297088406
reject
L: 0.041998606643466135
---------iter---------
LM loss: 0.996331512679291
accept
L: 0.004666511849274015
---------iter---------
LM loss: 0.9987247272722547
reject
L: 0.05133163034201416
---------iter---------
LM loss: 0.9963324083202644
reject
L: 0.5646479337621558
---------iter---------
LM loss: 0.9959114267793887
accept
L: 0.0627386593069062
---------iter---------
LM loss: 0.9959082656732073
reject
L: 0.6901252523759682
---------iter---------
LM loss: 0.9958570032630536
accept
L: 0.07668058359732981
---------iter---------
LM loss: 0.9958827688480423
reject
L: 0.8434864195706279
---------iter---------
LM loss: 0.9958472243887082
accept
L: 0.09372071328562531
---------iter---------
LM loss: 0.9958461941088542
reject
L: 1.0309278461418785
---------iter---------
LM loss: 0.9958409487588651
accept
L: 0.11454753846020872
---------iter---------
```

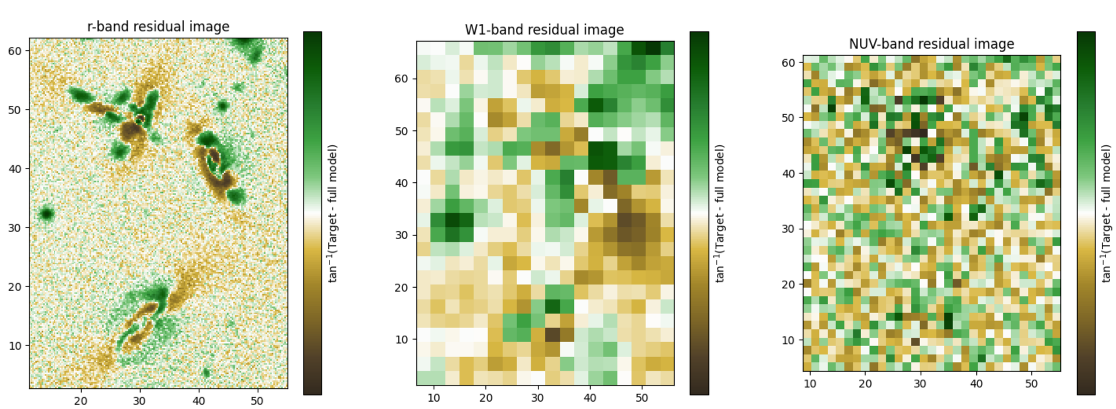
```
LM loss: 0.9958234265806855
accept
success
```

[12]:
```python
fig1, ax1 = plt.subplots(1, 3, figsize = (18,6))
ap.plots.model_image(fig1, ax1, MODEL)
ax1[0].set_title("r-band model image")
ax1[1].set_title("W1-band model image")
ax1[2].set_title("NUV-band model image")
plt.show()
```



The models look excellent! The power of multiband fitting lets us know that we have extracted all the available information here, no forced photometry required!

[13]:
```python
fig, ax = plt.subplots(1, 3, figsize = (18,6))
ap.plots.residual_image(fig, ax, MODEL)
ax[0].set_title("r-band residual image")
ax[1].set_title("W1-band residual image")
ax[2].set_title("NUV-band residual image")
plt.show()
```



The residuals look acceptable, but clearly there is more structure to be found in these galaxies, this is especially apparent in the r-band data. At least for the lower galaxy, we can see in the observed image that there are spiral arms, those can easily cause large scale residual patterns.

1.1.1 Dithered images

Note that it is not necessary to use images from different bands. Using dithered images one can effectively achieve higher resolution. It is possible to simultaneously fit dithered images with AutoProf instead of postprocessing the two images together. This will of course be slower, but may be worthwhile for cases where extra care is needed.

1.1.2 Stacked images

Like dithered images, one may wish to combine the statistical power of multiple images but for some reason it is not clear how to add them. In this case one can simply have AutoProf fit the images simultaneously. Again this is slower than if the image could be combined, but should extract all the statistical power from the data.

1.1.3 Time series

Some objects change over time. For example they may get brighter and dimmer, or may have a transient feature appear. However, the structure of an object may remain constant. An example of this is a supernova and its host galaxy. The host galaxy likely doesn't change across images, but the supernova does. It is possible to fit a time series dataset with a shared galaxy model across multiple images, and a shared position for the supernova, but a variable brightness for the supernova over each image.

It is possible to get quite creative with joint models as they allow one to fix selective features of a model over a wide range of data. If you have a situation which may benefit from joint modelling but are having a hard time determining how to format everything, please do contact us!

[ ]:
```

```