JointModels

June 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.

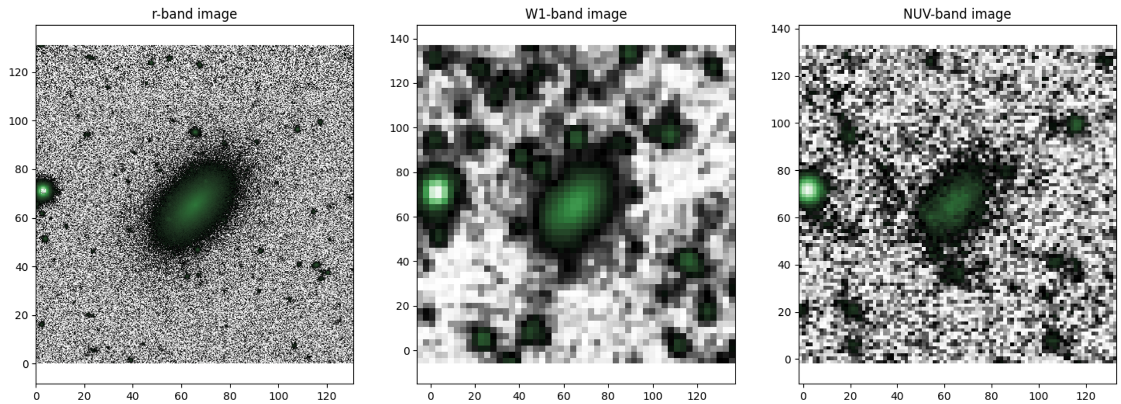
It is, of course, more work to set up a fit across multiple target images. However, the tradeoff can be well worth it. Perhaps there is space-based data with high resolution, but groundbased data has better S/N. Or perhaps each band individually does not have enough signal for a confident fit, but all 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",
    models = [model_r, model_W1, model_NUV],
    target = target_full,
)

model_full.initialize()
```

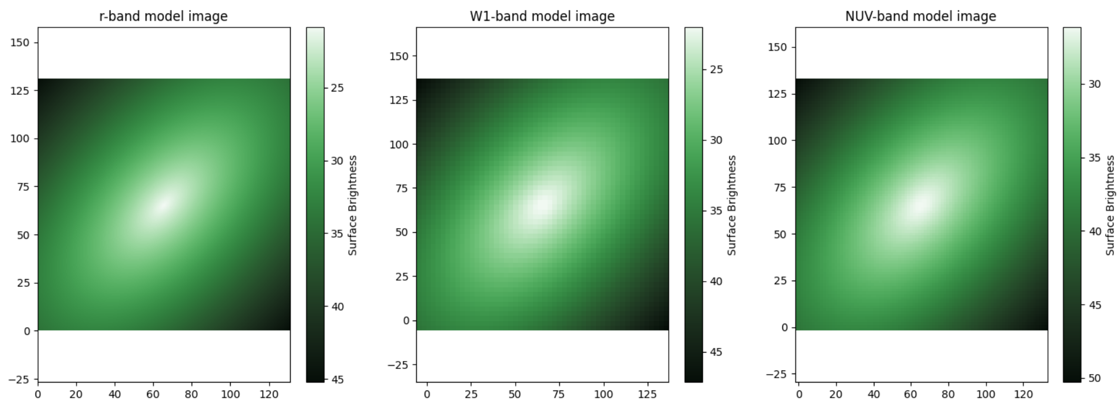
[6]:

```
result = ap.fit.LM(model_full, verbose = 1).fit()
print(result.message)
```

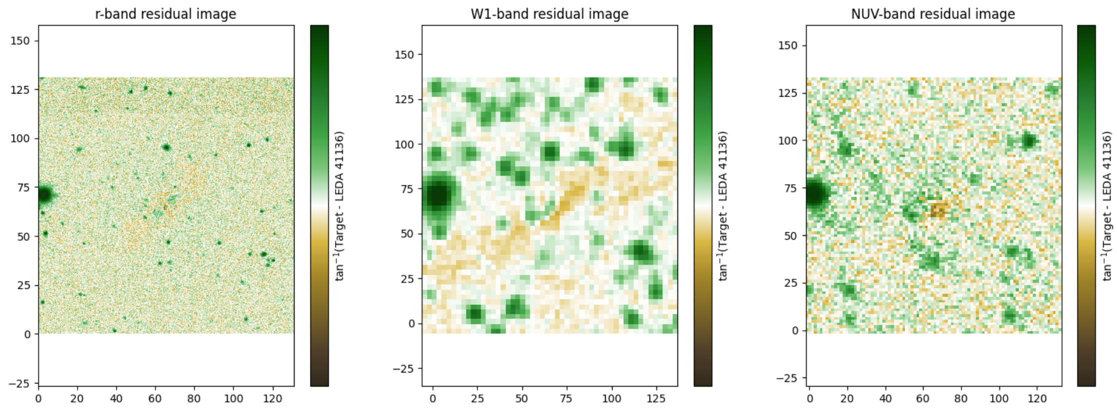
```
L: 1.0
---------init---------
LM loss: 93.27366451460108
max grad 29021.71841068653
L: 1.0
---------iter---------
LM loss: 93.26017887924537
accept
max grad 13344.687584180017
L: 0.1111111111111111
---------iter---------
LM loss: 93.24799279749257
accept
max grad 6119.109018353892
L: 0.012345679012345678
---------iter---------
LM loss: 93.24418499573945
accept
max grad 1244.8733828869752
L: 0.0013717421124828531
---------iter---------
LM loss: 93.24372669388836
```

```
accept
max grad 526.5826930598942
success
```

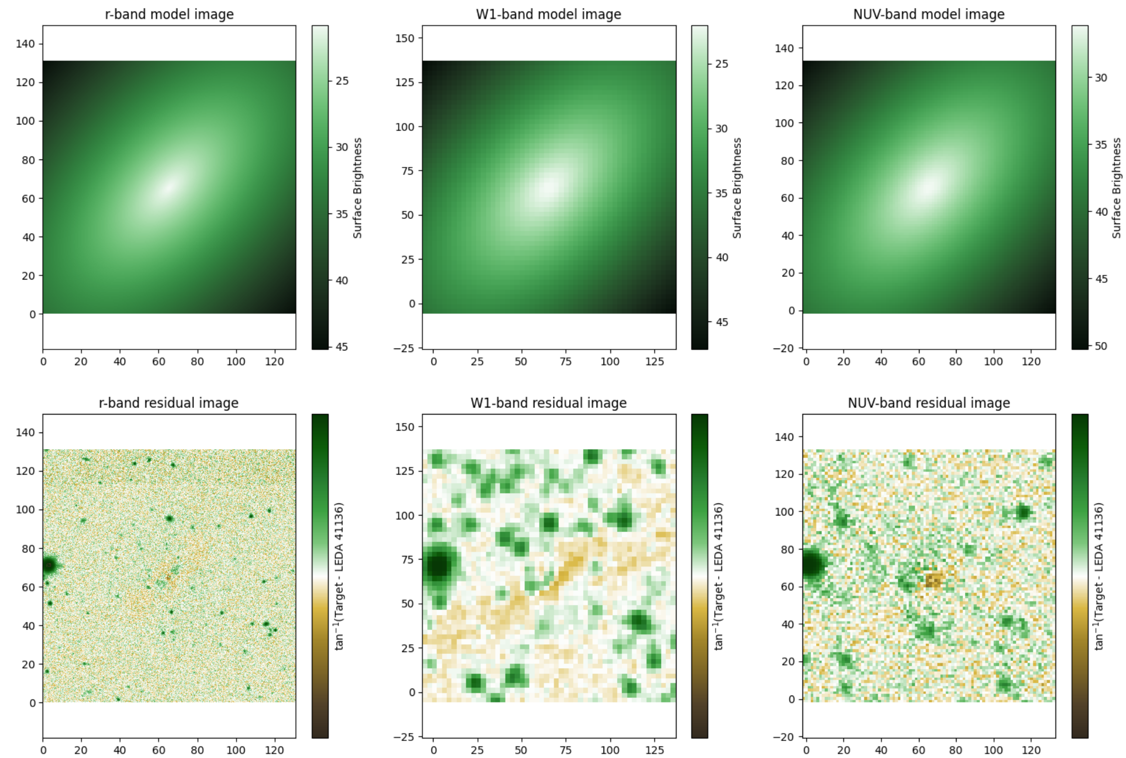
[7]:
```
# here we plot the results of the fitting, notice that each band has a␣
↪different PSF and pixelscale. Also, notice
# that the colour bars represent significantly different ranges since each␣
↪model was allowed to fit its own Ie.
# meanwhile the center, PA, q, and Re is the same for every model.
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()
```



[9]:
```
# Save a joint model just like any other model
model_full.save("jointsave.yaml")

# Load the joint model just like any other
model_reload = ap.models.AutoProf_Model(
    name = "reload LEDA 41136",
    filename = "jointsave.yaml",
)

# However, targets are not saved when saving a model, so those must be␣
↪re-assigned manually
# Assign the group target
model_reload.target = target_full
# Assign the sub-model targets
model_reload.models["rband model"].target = target_r
model_reload.models["W1band model"].target = target_W1
model_reload.models["NUVband model"].target = target_NUV

# You must also update the full model window before proceeding
model_reload.update_window()

# Plot everything again to check its working
fig1, ax1 = plt.subplots(2, 3, figsize = (18,12))
ap.plots.model_image(fig1, ax1[0], model_reload)
ax1[0][0].set_title("r-band model image")
ax1[0][1].set_title("W1-band model image")
ax1[0][2].set_title("NUV-band model image")
ap.plots.residual_image(fig1, ax1[1], model_reload)
ax1[1][0].set_title("r-band residual image")
ax1[1][1].set_title("W1-band residual image")
ax1[1][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.

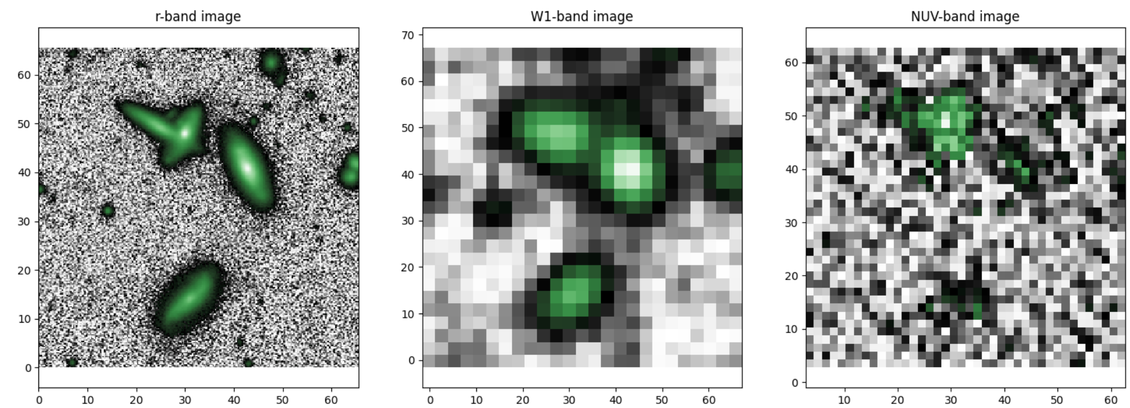
JointGroupModels

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

[10]:
```
# 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
```

```python
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))
```

```python
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()
```



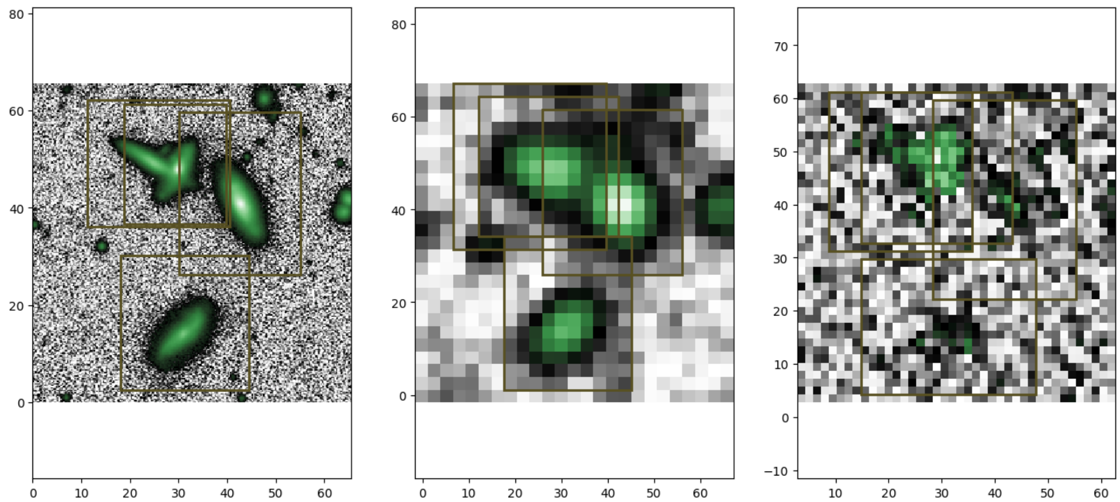
There is barely any signal in the GALEX data and it would be entirely impossible to analyze on its own. With simultaneous multiband fitting it is a breeze to get relatively robust results!

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.

[11]:
```python
# 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,
            models = 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,
    models = 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()
```



[12]:

```
MODEL.initialize()
result = ap.fit.LM(MODEL, verbose = 1, epsilon4 = 0.05).fit()
print(result.message)
```

```
L: 1.0
---------init---------
LM loss: 6.403416274358248
max grad 106920.54388187481
L: 1.0
---------iter---------
LM loss: 1.172537624069127e+28
reject
L: 11.0
---------iter---------
LM loss: 345029877.87082094
reject
L: 121.0
---------iter---------
LM loss: 6.287607175825135
accept
max grad 107019.09427263198
L: 13.444444444444445
```

```
---------iter---------
LM loss: 272946.59145345277
reject
L: 147.88888888888889
---------iter---------
LM loss: 6.197472479932939
accept
max grad 107160.3846512535
L: 16.432098765432098
---------iter---------
LM loss: 52359.97868749976
reject
L: 180.7530864197531
---------iter---------
LM loss: 6.126592147134424
accept
max grad 106911.76608244017
L: 20.083676268861453
---------iter---------
LM loss: 14609.19682039088
reject
L: 220.920438957476
---------iter---------
LM loss: 6.070282800551956
accept
max grad 106873.5935703212
L: 24.546715439719556
---------iter---------
LM loss: 1841.3824270887908
reject
L: 270.0138698369151
---------iter---------
LM loss: 6.025265454965052
accept
max grad 106827.68731297889
L: 30.001541092990564
---------iter---------
LM loss: 9.784013981101726
reject
L: 330.0169520228962
---------iter---------
LM loss: 5.989398170548845
accept
max grad 106838.4226739362
L: 36.66855022476624
---------iter---------
LM loss: 5.662486538856044
accept
```

```
max grad 106273.89368993299
L: 4.07428335830736
---------iter---------
LM loss: 237796.82576222107
reject
L: 44.81711694138096
---------iter---------
LM loss: 5.413879269444248
accept
max grad 105372.56939415312
L: 4.97967966015344
---------iter---------
LM loss: 3.9051814869820354
accept
max grad 83694.54805076298
L: 0.5532977400170489
---------iter---------
LM loss: inf
nan loss
L: 6.086275140187538
---------iter---------
LM loss: 3.3807306780070414
accept
max grad 72114.15166831565
L: 0.6762527933541709
---------iter---------
LM loss: 3.8245194603925905e+259
reject
L: 7.43878072689588
---------iter---------
LM loss: 3.0938888095051778
accept
max grad 63773.40711953317
L: 0.8265311918773199
---------iter---------
LM loss: 7.713655545647702e+22
reject
L: 9.091843110650519
---------iter---------
LM loss: 2.907340945893838
accept
max grad 57466.32338869121
L: 1.01020479007228
---------iter---------
LM loss: 2.157780590081726
accept
max grad 28238.819254240545
L: 0.11224497667469777
```

```
---------iter---------
LM loss: 6.712534062157312
reject
L: 1.2346947434216755
---------iter---------
LM loss: 1.8477469188759996
accept
max grad 57315.440931195575
L: 0.13718830482463062
---------iter---------
LM loss: 1.4426581183427085
accept
max grad 348164.40024065797
L: 0.015243144980514513
---------iter---------
LM loss: 1.090882163710096
accept
max grad 925189.5659103841
L: 0.0016936827756127237
---------iter---------
LM loss: 2.10611855990704
reject
L: 0.01863051053173996
---------iter---------
LM loss: 1.0343334434809464
accept
max grad 1904801.5916515943
L: 0.0020700567257488844
---------iter---------
LM loss: 2.80004493296601
reject
L: 0.022770623983237728
---------iter---------
LM loss: 1.0355335231058564
reject
L: 0.250476863815615
---------iter---------
LM loss: 1.0221876754269252
accept
max grad 1524272.5390304134
L: 0.027830762646179445
---------iter---------
LM loss: 1.0224589871354408
reject
L: 0.3061383891079739
---------iter---------
LM loss: 1.020491651193672
accept
```

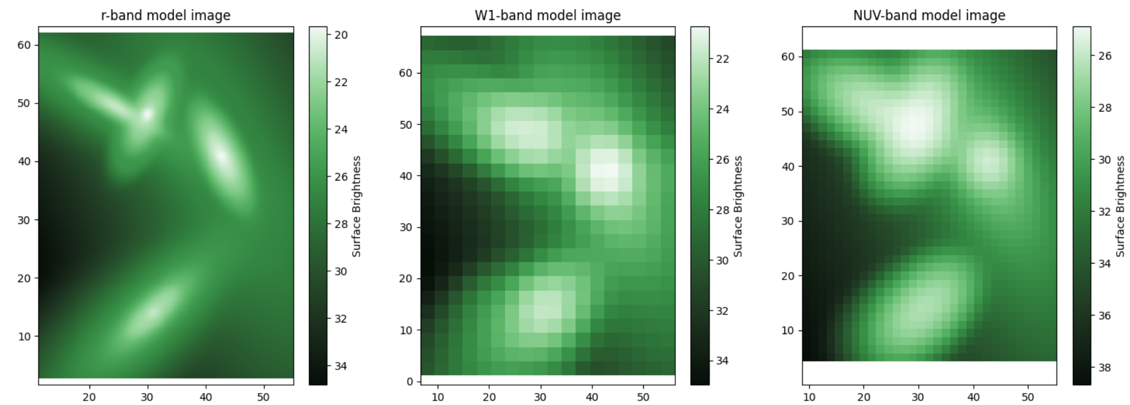
```
max grad 1136237.937198562
L: 0.03401537656755266
---------iter---------
LM loss: 1.0189221545329175
accept
max grad 1690464.4313854463
L: 0.0037794862852836286
---------iter---------
LM loss: 1.030379372630764
reject
L: 0.041574349138119915
---------iter---------
LM loss: 1.0173811375693644
accept
max grad 1336279.6041659098
L: 0.004619372126457768
---------iter---------
LM loss: 1.0209300449707326
reject
L: 0.050813093391035444
---------iter---------
LM loss: 1.0162375426824635
accept
max grad 911573.3450080042
L: 0.005645899265670605
---------iter---------
LM loss: 1.0161432018925132
reject
L: 0.06210489192237665
---------iter---------
LM loss: 1.0155300171195925
accept
max grad 583806.3584576884
L: 0.006900543546930739
---------iter---------
LM loss: 1.0150982281572118
accept
max grad 110452.84715493237
L: 0.0007667270607700821
---------iter---------
LM loss: 1.2754900629147259
reject
L: 0.008433997668470904
---------iter---------
LM loss: 1.0144045819964391
accept
max grad 52536.85566168313
L: 0.0009371108520523227
```

```
---------iter---------
LM loss: 1.1587581884629188
reject
L: 0.010308219372575549
---------iter---------
LM loss: 1.0135360893195922
accept
max grad 22178.424665240378
L: 0.00114535770806395
---------iter---------
LM loss: 1.0444972382060187
reject
L: 0.012598934788703449
---------iter---------
LM loss: 1.0135181934562798
reject
L: 0.13858828267573794
---------iter---------
LM loss: 1.0124505184696733
accept
max grad 10252.08423871252
L: 0.015398698075081993
---------iter---------
LM loss: 1.0125277488145015
reject
L: 0.16938567882590191
---------iter---------
LM loss: 1.0126620577399388
reject
L: 1.863242467084921
---------iter---------
LM loss: 1.012375430673546
accept
max grad 4870.110995600241
L: 0.20702694078721345
---------iter---------
LM loss: 1.0126695201546119
reject
L: 2.277296348659348
---------iter---------
LM loss: 1.0123628696480986
accept
max grad 3254.5587049863643
L: 0.2530329276288164
---------iter---------
LM loss: 1.012667996152371
reject
L: 2.7833622039169805
```

```
---------iter---------
LM loss: 1.012356057789811
accept
max grad 2722.2180276695226
L: 0.3092624671018867
---------iter---------
LM loss: 1.0126724642835296
reject
L: 3.401887138120754
---------iter---------
LM loss: 1.0126822799319963
reject
L: 37.420758519328295
---------iter---------
LM loss: 1.0123555622076896
accept
max grad 2683.217602918642
L: 4.157862057703144
---------iter---------
LM loss: 1.012681824178603
reject
L: 45.736482634734585
---------iter---------
LM loss: 1.0123551678736482
accept
max grad 2653.5274135744185
L: 5.081831403859399
---------iter---------
LM loss: 1.0126815473080384
reject
L: 55.900145442453386
---------iter---------
LM loss: 1.0123548511607543
accept
max grad 2630.6047180239893
L: 6.21112727138371
---------iter---------
LM loss: 1.0126813745860384
reject
L: 68.32239998522081
---------iter---------
LM loss: 1.0123545954265312
accept
max grad 2612.709767881607
L: 7.591377776135646
---------iter---------
LM loss: 1.012352500008568
accept
```

```
max grad 2488.156606771391
L: 0.8434864195706273
---------iter---------
LM loss: 1.0126683615359033
reject
L: 9.278350615276901
---------iter---------
LM loss: 1.012679629866087
reject
L: 102.06185676804591
---------iter---------
LM loss: 1.0126811112282348
reject
L: 1122.6804244485052
---------iter---------
LM loss: 1.0123524857772428
accept
max grad 2487.398321055794
success
```

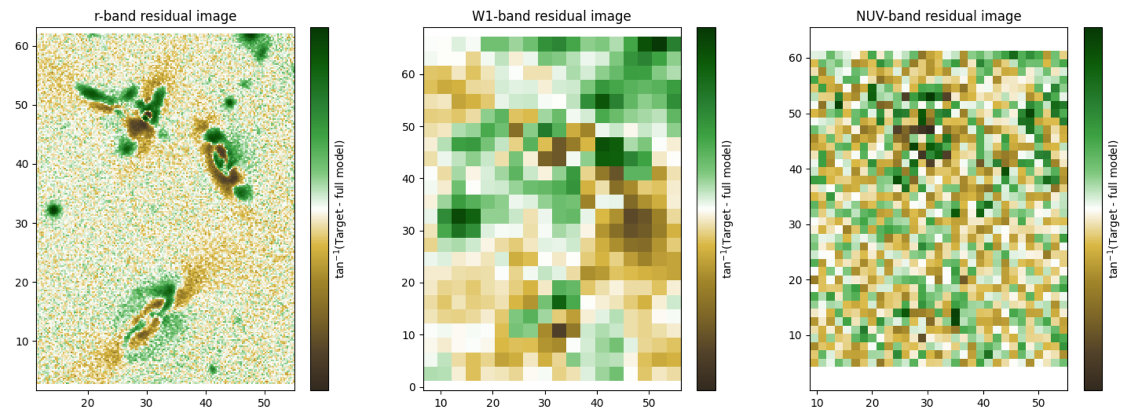
[13]:
```
fig1, ax1 = plt.subplots(1, 3, figsize = (18,6))
ap.plots.model_image(fig1, ax1, MODEL)
ax1[0].set_title("r-band model image")
ax1[1].set_title("W1-band model image")
ax1[2].set_title("NUV-band model image")
plt.show()
```



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!

[14]:
```
fig, ax = plt.subplots(1, 3, figsize = (18,6))
ap.plots.residual_image(fig, ax, MODEL)
ax[0].set_title("r-band residual image")
```

```python
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!

```
[ ]:
```