AdvancedPSFModels

June 17, 2023

1 Advanced PSF modeling

Ideally we always have plenty of well separated bright, but not oversaturated, stars to use to construct a PSF model. These models are incredibly important for certain science objectives that rely on precise shape measurements and not just total light measures. Here we demonstrate some of the special capabilities AutoPhot has to handle challenging scenarios where a good PSF model is needed but there are only very faint stars, poorly placed stars, or even no stars to work with!

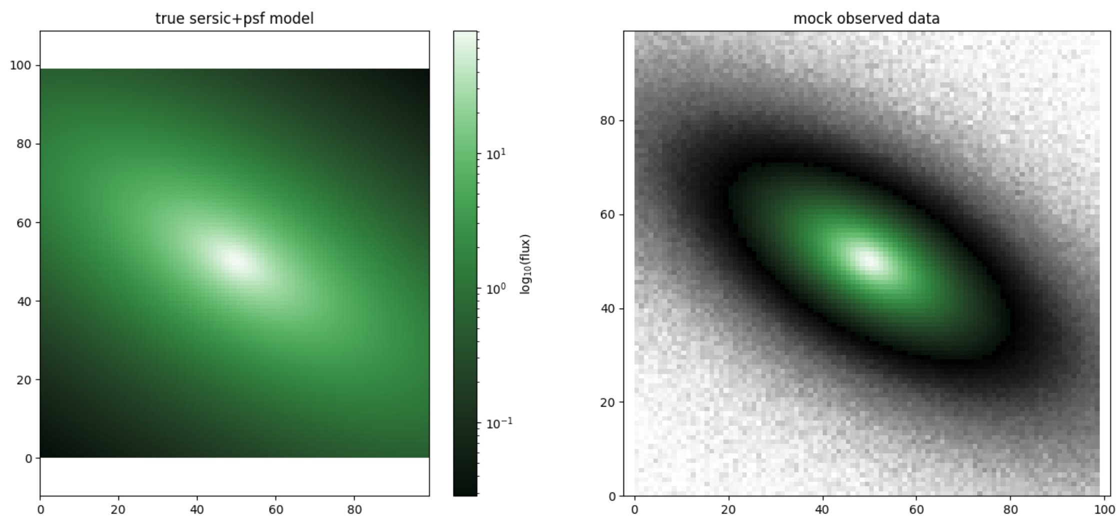
[1]:
```
import autophot as ap
import numpy as np
import torch
from astropy.io import fits
import matplotlib.pyplot as plt
from time import time
%matplotlib inline
```

2 PSF modeling without stars

Can it be done? Let's see!

[2]:
```
# Lets make some data that we need to fit

true_psf = ap.utils.initialize.moffat_psf(
    2., # n                                 !!!!! Take note, we want to get n =
 ↪2. !!!!!!
    3., # Rd                                !!!!! Take note, we want to get Rd =
 ↪3.!!!!!!
    51, # pixels
    1.  # pixelscale
)

target = ap.image.Target_Image(
    data = torch.zeros(99,99),
    pixelscale = 1.,
    psf = true_psf,
)
```

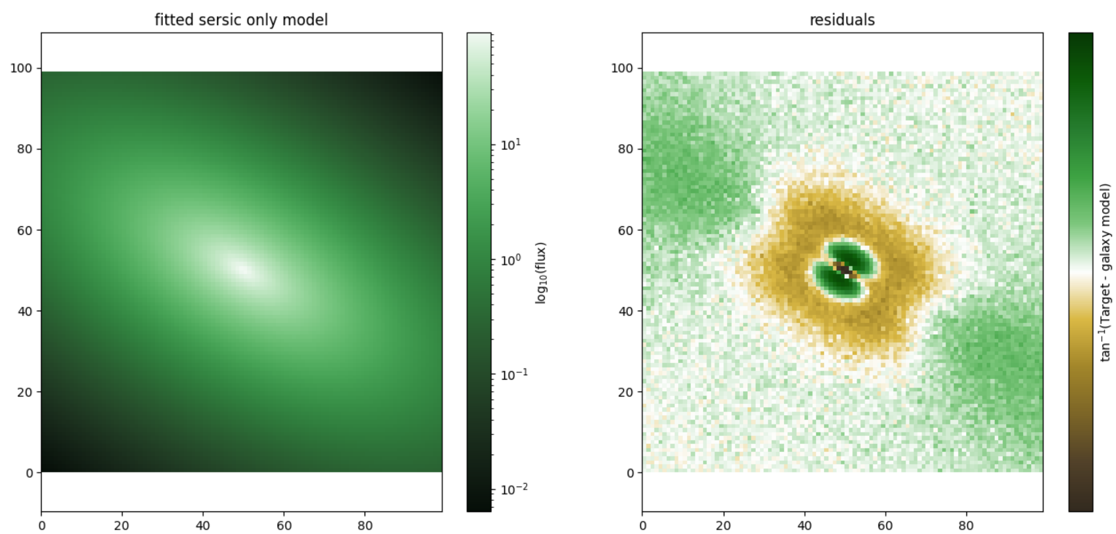
```python
true_model = ap.models.AutoPhot_Model(
    name = "true model",
    model_type = "sersic galaxy model",
    target = target,
    parameters = {
        "center": [50.1,50.1],
        "q": 0.4,
        "PA": np.pi/3,
        "n": 2,
        "Re": 25,
        "Ie": 1,
    },
    psf_mode = "full",
)

# use the true model to make some data
sample = true_model()
torch.manual_seed(61803398)
target.data = sample.data + torch.normal(torch.zeros_like(sample.data), 0.1)
target.variance = 0.01*torch.ones_like(sample.data)

fig, ax = plt.subplots(1,2, figsize = (16,7))
ap.plots.model_image(fig, ax[0], true_model)
ap.plots.target_image(fig, ax[1], target)
ax[0].set_title("true sersic+psf model")
ax[1].set_title("mock observed data")
plt.show()
```



[3]:
```python
# Now we will try and fit the data using just a plain sersic

# Here we set up a sersic model for the galaxy
plain_galaxy_model = ap.models.AutoPhot_Model(
    name = "galaxy model",
    model_type = "sersic galaxy model",
    target = target,
)

# Let AutoPhot determine its own intial parameters, so it has to start with␣
 ↪whatever it decides automatically,
# just like a real fit.
plain_galaxy_model.initialize()

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

```
L: 1.0
---------init---------
LM loss: 243.27554987704144
L: 1.0
---------iter---------
LM loss: 43.69233849093099
accept
L: 0.1111111111111111
---------iter---------
LM loss: 23.553020518271587
accept
L: 0.012345679012345678
---------iter---------
LM loss: 23.001317881106633
accept
L: 0.0013717421124828531
---------iter---------
LM loss: 22.997477806531908
accept
L: 0.00015241579027587256
---------iter---------
LM loss: 22.99742308876948
accept
L: 1.6935087808430286e-05
---------iter---------
LM loss: 22.997422269494322
accept
success
```

[4]:
```python
# The shape of the residuals here shows that there is still missing information;
↪ this is of course
# from the missing PSF convolution to blur the model. In fact, the shape of␣
↪those residuals is very
# commonly seen in real observed data (ground based) when it is fit without␣
↪accounting for PSF blurring.
fig, ax = plt.subplots(1,2, figsize = (16,7))
ap.plots.model_image(fig, ax[0], plain_galaxy_model)
ap.plots.residual_image(fig, ax[1], plain_galaxy_model)
ax[0].set_title("fitted sersic only model")
ax[1].set_title("residuals")
plt.show()
```



[5]:
```python
# Now we will try and fit the data with a sersic model and a "live" psf

# Here we set up a sersic model for the galaxy
live_galaxy_model = ap.models.AutoPhot_Model(
    name = "galaxy model",
    model_type = "sersic galaxy model",
    target = target,
    psf_mode = "full",
)

# Here we create a moffat model for the PSF. Note that this is just a regular␣
↪AutoPhot model that we have chosen
# to be a moffat, really any model can be used. To make it suitable as a PSF we␣
↪will need to apply some very
# specific settings. The window for the model is now just used to set the size␣
↪of the PSF (51 by 51 pixels).
```

```python
# The "center" must be put at the center of the window (note that this is a bit␣
↪harder when pixelscale != 1).
# The "I0" value doesn't matter for a PSF (which will be normalized␣
↪internally), we pick I0 = 1 just for
# numerical stability. The "psf_upscale" parameter is set to 1 just to indicate␣
↪we are not doing any super
# resolution fitting this time.
live_psf_model = ap.models.AutoPhot_Model(
    name = "psf",
    model_type = "moffat star model",
    target = target,
    window = [[0,51],[0,51]],
    parameters = {
        "n": 1., # True value is 2.
        "Rd": 2., # True value is 3.
        "center": {"value": [25.5,25.5], "locked":True},
        "I0": {"value": 1., "locked":True},
    },
    psf_upscale = 1.,
)

# Here we bind the PSF model to the galaxy model, this will add the psf_model␣
↪parameters to the galaxy_model
# object since it now depends on those.
live_galaxy_model.set_aux_psf(live_psf_model)

live_galaxy_model.initialize()

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

```
L: 1.0
---------init---------
LM loss: 1310.4333831440476
L: 1.0
---------iter---------
LM loss: 1274.9370099118728
reject
L: 11.0
---------iter---------
LM loss: 1242.8076703995657
accept
L: 1.2222222222222223
---------iter---------
LM loss: 642.0780399853535
accept
```

```
L: 0.1358024691358025
---------iter---------
LM loss: 13.90136061065249
accept
L: 0.015089163237311388
---------iter---------
LM loss: 2.3188079403598376
accept
L: 0.0016765736930345987
---------iter---------
LM loss: 1.0132489669127749
accept
L: 0.00018628596589273318
---------iter---------
LM loss: 0.9950764393407693
accept
L: 2.069844065474813e-05
---------iter---------
LM loss: 0.9950758944881941
accept
L: 2.299826739416459e-06
---------iter---------
LM loss: 0.9950758944880296
accept
success
```

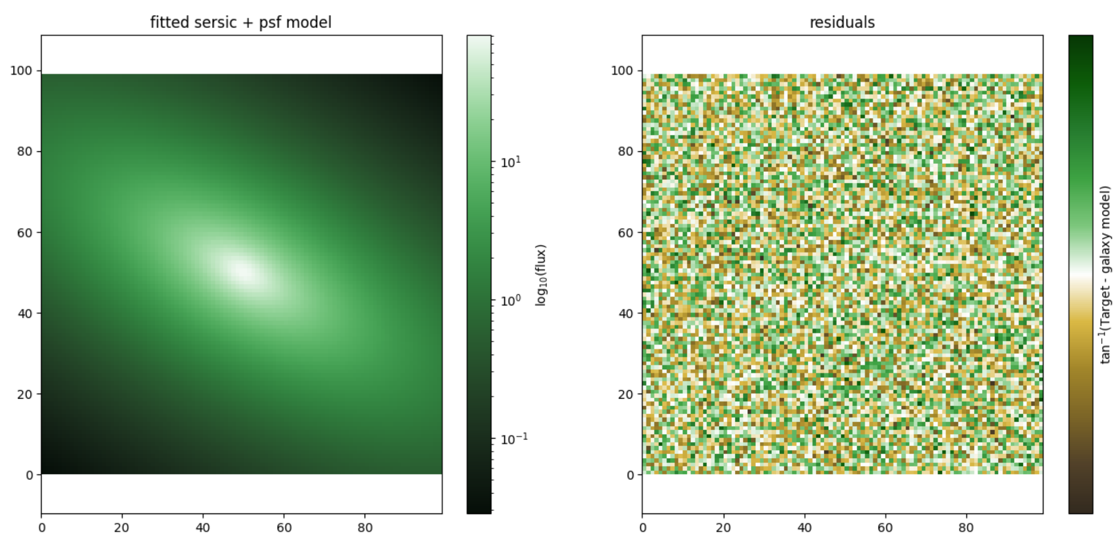
[6]:

```
print("fitted n for moffat PSF: ", live_galaxy_model["psf:n"].value.item(), "we␣
↪were hoping to get 2!")
print("fitted Rd for moffat PSF: ", live_galaxy_model["psf:Rd"].value.item(),␣
↪"we were hoping to get 3!")
print(live_galaxy_model.parameters)
```

```
fitted n for moffat PSF:  2.005028505314216 we were hoping to get 2!
fitted Rd for moffat PSF:  3.006722733274736 we were hoping to get 3!
Parameter Group: galaxy model
center: [50.101554956337935, 50.09966390213996] +- [0.0031671451985719103,
0.002369698988150343] [arcsec]
q: 0.39993709191888094 +- 0.0014871924704202982 [b/a, (0.0, 1.0)]
PA: 1.0473284923386428 +- 0.00035866954713633216 [radians, (0.0,
3.141592653589793), cyclic]
n: 1.9989995405555154 +- 0.011379395536874637 [none, (0.36, 8.0)]
Re: 25.004126033672932 +- 0.04107068969655657 [arcsec, (0.0, 'None')]
Ie: 0.9999812816800422 +- 0.0014146244310768471 [log10(flux/arcsec^2)]
center: [25.5, 25.5] +- None [arcsec, locked]
n: 2.005028505314216 +- 0.024164100687721932 [none, (0.1, 10.0)]
Rd: 3.006722733274736 +- 0.023778575943862092 [arcsec, (0.0, 'None')]
I0: 1.0 +- None [log10(flux/arcsec^2), locked]
```

This is truly remarkable! With no stars available we were still able to extract an accurate PSF from the image! To be fair, this example is essentially perfect for this kind of fitting and we knew the true model types (sersic and moffat) from the start. Still, this is a powerful capability in certain scenarios. For many applications (e.g. weak lensing) it is essential to get the absolute best PSF model possible. Here we have shown that not only stars, but galaxies in the field can be useful tools for measuring the PSF!

[7]:
```
fig, ax = plt.subplots(1,2, figsize = (16,7))
ap.plots.model_image(fig, ax[0], live_galaxy_model)
ap.plots.residual_image(fig, ax[1], live_galaxy_model)
ax[0].set_title("fitted sersic + psf model")
ax[1].set_title("residuals")
plt.show()
```



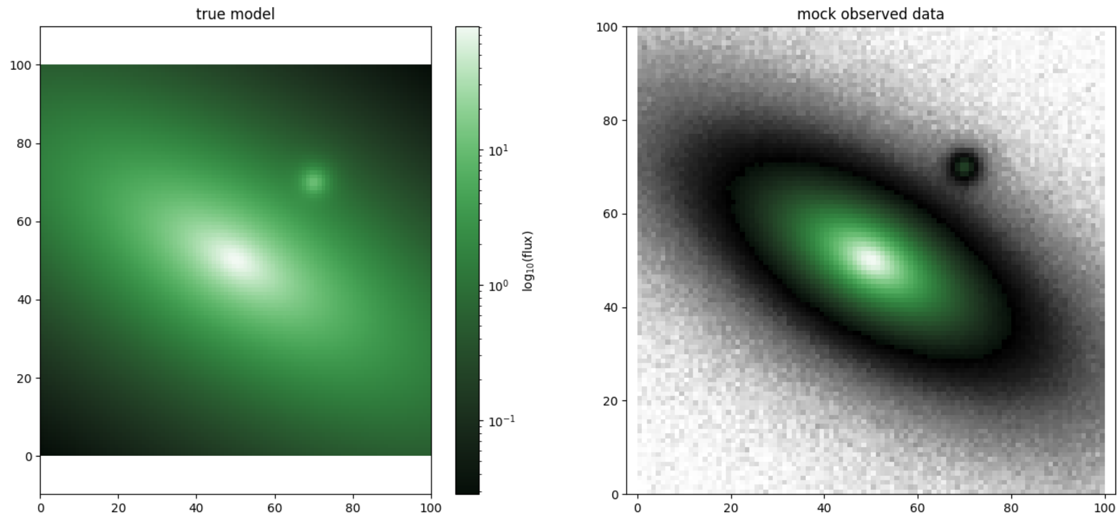
There are regions of parameter space that are degenerate and so even in this idealized scenario the PSF model can get stuck. If you rerun the notebook with different random number seeds for pytorch you may find some where the optimizer "fails by immobility" this is when it get's stuck in the parameter space and can't find any way to improve the likelihood. In fact most of these "fail" fits do return really good values for the PSF model, so keep in mind that the "fail" flag only means the possibility of a truly failed fit. Unfortunatly, detecting convergence is hard.

2.1 PSF fitting with a bad star

Fitting a PSF to a galaxy is perhaps not the most stable way to get a good model. However, there is a very common situation where this kind of fitting is quite helpful. Consider the scenario that there is a star, but it is not very bright and it is right next to a galaxy. Now we need to model the galaxy and the star simultaneously, but the galaxy should be convolved with the PSF for the fit to be stable (otherwise you'll have to do several iterations to converge). If there were many stars you could perhaps just stack a bunch of them and hope the average is close enough, but in this case we don't have many to work with so we need to squeeze out as much statistical power as possible.

[8]:
```
# Lets make some data that we need to fit

true_psf2 = ap.utils.initialize.moffat_psf(
    2., # n                                  !!!!! Take note, we want to get n =␣
 ↪2. !!!!!!
    3., # Rd                                 !!!!! Take note, we want to get Rd =␣
 ↪3.!!!!!!
    51, # pixels
    1.  # pixelscale
)

target2 = ap.image.Target_Image(
    data = torch.zeros(100,100),
    pixelscale = 1.,
    psf = true_psf,
)

true_galaxy2 = ap.models.AutoPhot_Model(
    name = "true galaxy",
    model_type = "sersic galaxy model",
    target = target2,
    parameters = {
        "center": [50.1,50.1],
        "q": 0.4,
        "PA": np.pi/3,
        "n": 2,
        "Re": 25,
        "Ie": 1,
    },
    psf_mode = "full",
)
true_star2 = ap.models.AutoPhot_Model(
    name = "true star",
    model_type = "moffat star model",
    target = target2,
    parameters = {
        "center": [70,70],
        "n": 2,
        "Rd": 3,
        "I0": 1.,
    },
)
true_model2 = ap.models.AutoPhot_Model(
    name = "true model",
    model_type = "group model",
    target = target2,
    models = [true_galaxy2, true_star2],
```

```
)

# use the true model to make some data
sample = true_model2()
torch.manual_seed(1618033988)
target2.data = sample.data + torch.normal(torch.zeros_like(sample.data), 0.1)
target2.variance = 0.01 * torch.ones_like(sample.data)

fig, ax = plt.subplots(1,2, figsize = (16,7))
ap.plots.model_image(fig, ax[0], true_model2)
ap.plots.target_image(fig, ax[1], target2)
ax[0].set_title("true model")
ax[1].set_title("mock observed data")
plt.show()
```



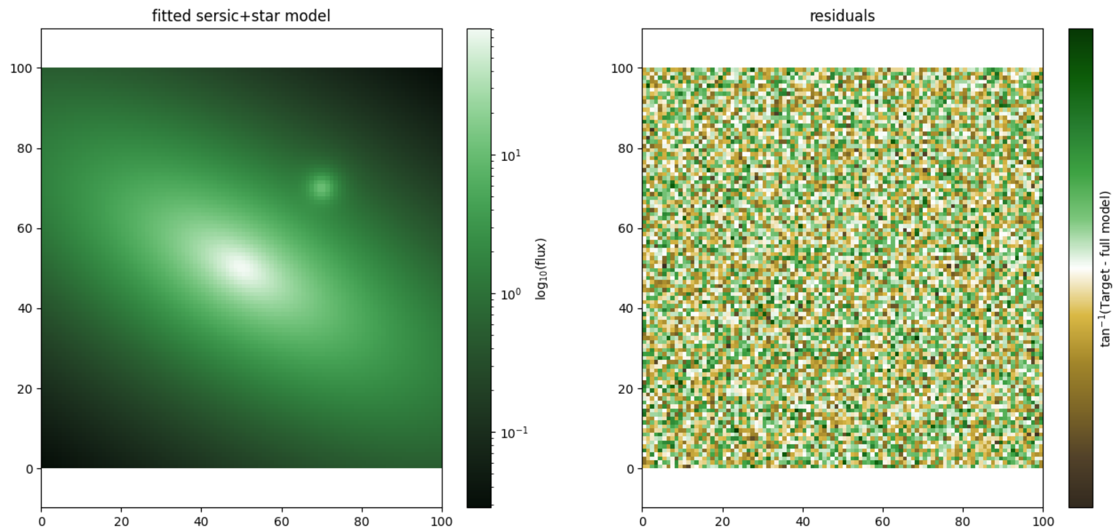
[9]:
```
# Now we will try and fit the data

# Here we set up a sersic model for the galaxy
galaxy_model2 = ap.models.AutoPhot_Model(
    name = "galaxy model",
    model_type = "sersic galaxy model",
    target = target,
    psf_mode = "full",
)

psf_model2 = ap.models.AutoPhot_Model(
    name = "psf",
    model_type = "moffat star model",
    target = target2,
```

```python
    window = [[0,51],[0,51]],
    parameters = {
        "n": 1., # True value is 2.
        "Rd": 2., # True value is 3.
        "center": {"value": [25.5,25.5], "locked":True},
        "I0": {"value": 1., "locked":True},
    },
    psf_upscale = 1.,
)

# Here we bind the PSF model to the galaxy model, this will add the psf_model␣
 ↪parameters to the galaxy_model
# object since it now depends on those.
galaxy_model2.set_aux_psf(psf_model2)

# Let AutoPhot determine its own intial parameters, so it has to start with␣
 ↪whatever it decides automatically,
# just like a real fit.
galaxy_model2.initialize()

star_model2 = ap.models.AutoPhot_Model(
    name = "star model",
    model_type = "moffat star model",
    target = target2,
    parameters = {
        "center": [72,68], # start the star in roughly the right location
    },
)

star_model2.initialize()

star_model2.add_equality_constraint(psf_model2, ["n", "Rd"])

full_model2 = ap.models.AutoPhot_Model(
    name = "full model",
    model_type = "group model",
    models = [galaxy_model2, star_model2],
    target = target2,
)

result = ap.fit.LM(full_model2, verbose = 1).fit()
print(result.message)
```

```
L: 1.0
---------init---------
LM loss: 1289.223033554867
L: 1.0
```

```
---------iter---------
LM loss: 1334.8376503263753
reject
L: 11.0
---------iter---------
LM loss: 1224.8487253437497
accept
L: 1.2222222222222223
---------iter---------
LM loss: 643.5736326433428
accept
L: 0.1358024691358025
---------iter---------
LM loss: 10178.661454199417
reject
L: 1.4938271604938274
---------iter---------
LM loss: 18.25025355497835
accept
L: 0.16598079561042525
---------iter---------
LM loss: 11.1150916162103
accept
L: 0.018442310623380583
---------iter---------
LM loss: 3.912909600013935
accept
L: 0.0020491456248200647
---------iter---------
LM loss: 2.1570678687668674
accept
L: 0.00022768284720222294
---------iter---------
LM loss: 1.9060252075483866
accept
L: 2.5298094133581044e-05
---------iter---------
LM loss: 1.9057736082672139
accept
L: 2.8108993481756715e-06
---------iter---------
LM loss: 1.9057736037124882
accept
L: 3.123221497972968e-07
---------iter---------
LM loss: 1.905773603712473
accept
success
```

[10]:
```python
fig, ax = plt.subplots(1,2, figsize = (16,7))
ap.plots.model_image(fig, ax[0], full_model2)
ap.plots.residual_image(fig, ax[1], full_model2)
ax[0].set_title("fitted sersic+star model")
ax[1].set_title("residuals")
plt.show()
```



[11]:
```python
print("fitted n for moffat PSF: ", galaxy_model2["psf:n"].value.item(), "we were hoping to get 2!")
print("fitted Rd for moffat PSF: ", galaxy_model2["psf:Rd"].value.item(), "we were hoping to get 3!")

print("---Note that we can just as well look at the star model parameters since they are the same---")
print("fitted n for moffat PSF: ", star_model2["n"].value.item(), "we were hoping to get 2!")
print("fitted Rd for moffat PSF: ", star_model2["Rd"].value.item(), "we were hoping to get 3!")
```

```
fitted n for moffat PSF:  1.9736268259836525 we were hoping to get 2!
fitted Rd for moffat PSF:  2.981495511551701 we were hoping to get 3!
---Note that we can just as well look at the star model parameters since they
are the same---
fitted n for moffat PSF:  1.9736268259836525 we were hoping to get 2!
fitted Rd for moffat PSF:  2.981495511551701 we were hoping to get 3!
```

Note that the fitted moffat parameters aren't much better than they were earlier when we just fit the galaxy alone. This shows us that extended objects have plenty of constraining power when it comes to PSF fitting, all this information has previously been left on the table! It makes sense that the galaxy dominates the PSF fit here, while the star is very simple to fit, it has much less light than the galaxy in this scenario so the S/N for the galaxy dominates. The reason this works really well is of course that the true data is in fact a sersic model, so we are working in a very idealized scenario. Real world galaxies are not necessarily well described by a sersic, so it is worthwhile to be cautious when doing this kind of fitting. Always make sure the results make sense before storming ahead with galaxy based PSF models, that said the payoff can be well worth it.

3 PSF fitting for faint stars

Sometimes there are stars available, but they are faint and it is hard to see how a reliable fit could be obtained. We have already seen how faint stars next to galaxies are still viable for PSF fitting. Now we will consider the case of isolated but faint stars. The trick here is that we have a second high resolution image, perhaps in a different band. To perform this fitting we will link up the two bands using joint modelling to constrain the star centers, this will constrain some of the parameters making it easier to fit a PSF model.

[12]:
```
# Coming soon
```

4 PSF fitting for saturated stars

A saturated star is a bright star, and it's just begging to be used for modelling a PSF. There's just one catch, the highest signal to noise region is completely messed up and can't be used! Traditionally these stars are either ignored, or a two stage fit is performed to get an "inner psf" and an "outer psf" which are then merged. Why not fit the inner and outer PSFs all at once! This can be done with AutoPhot using parameter constraints and masking.

[13]:
```
# Coming soon
```