Deconvolution of images in the presence of Poisson noise

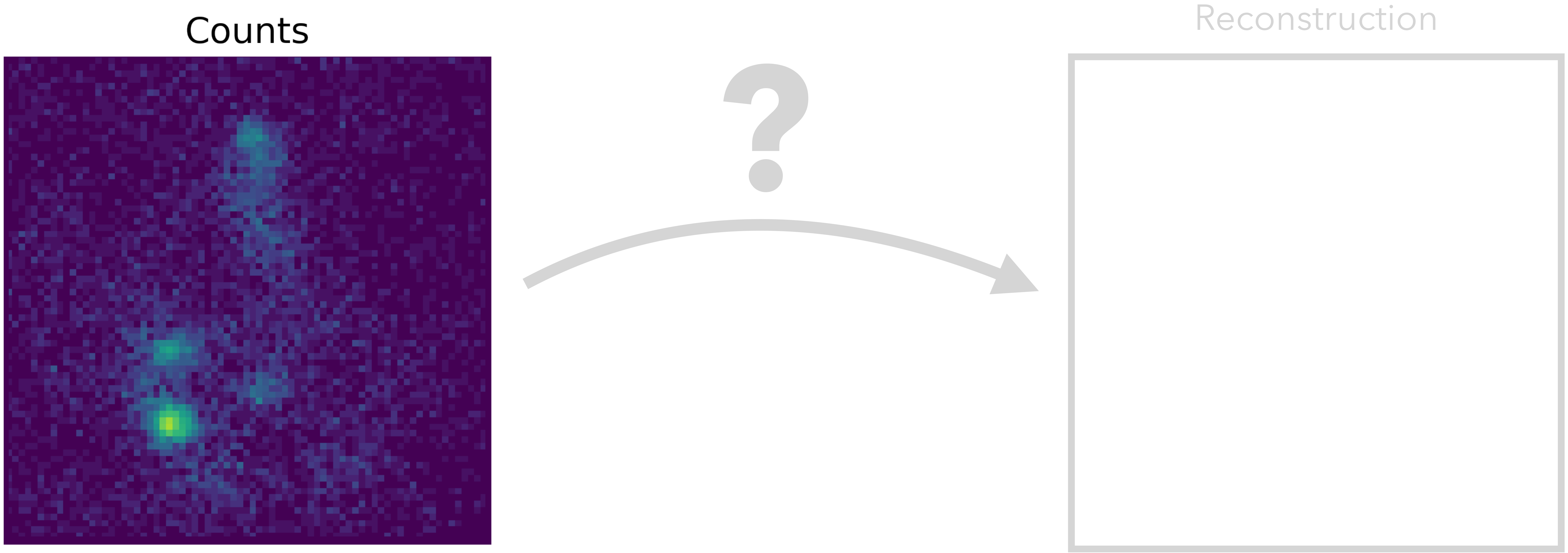
August 2nd, 2022

RISE-CHASC Workshop, Cambridge MA

Axel Donath

Aneta Siemiginowska, Vinay Kashyap, DvD

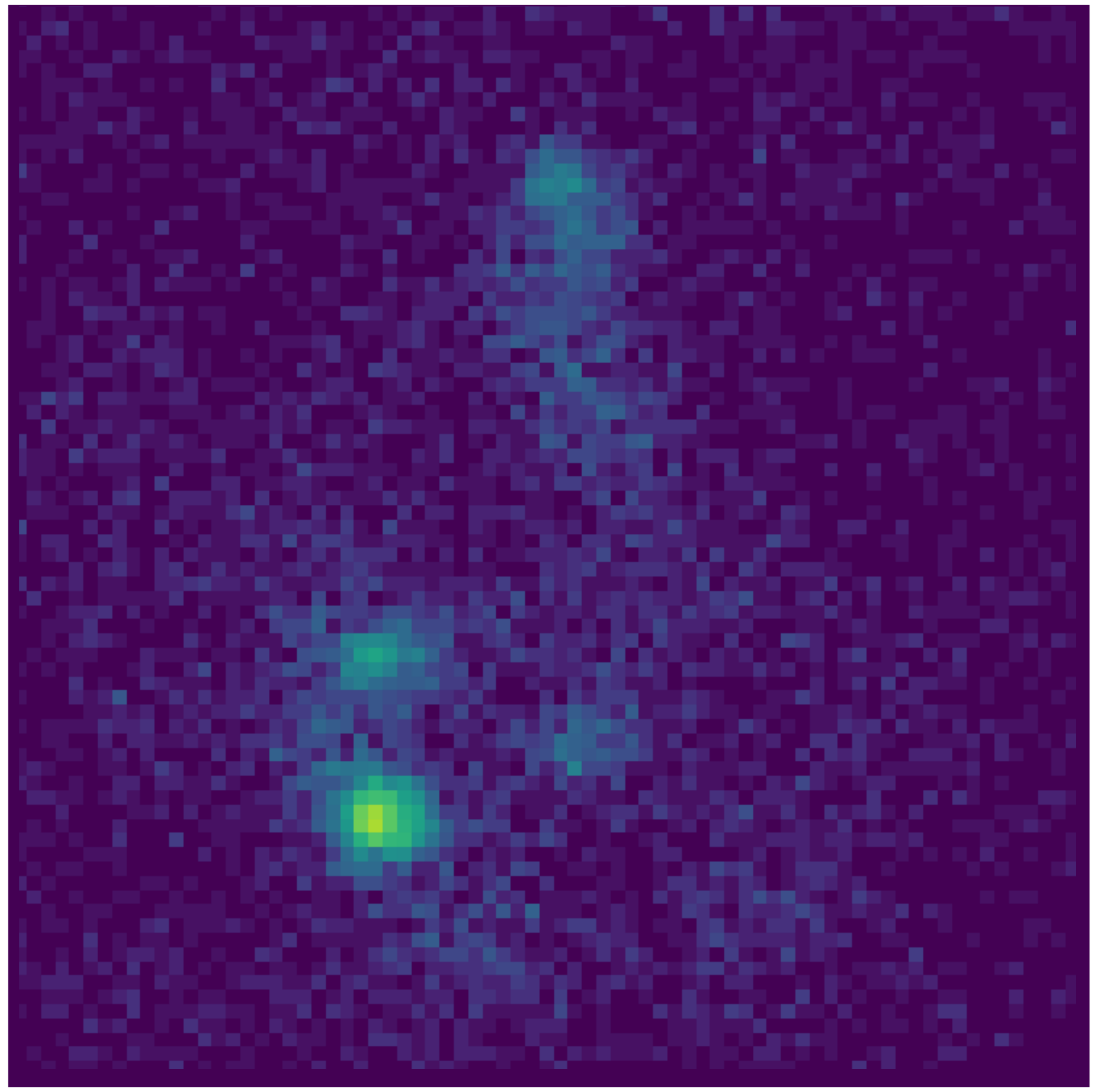
The problem of "unblind" deconvolution



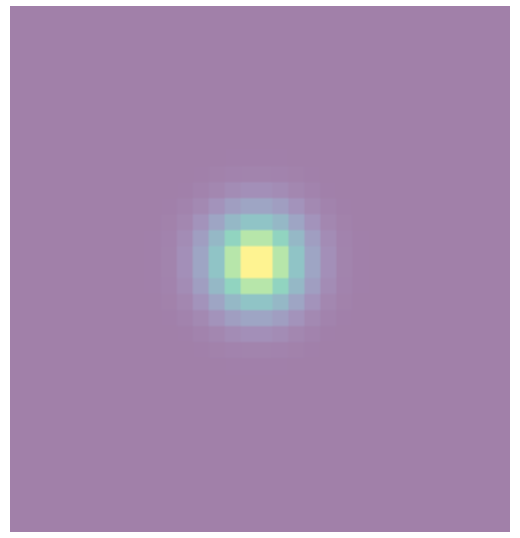
Low counts astronomical images…

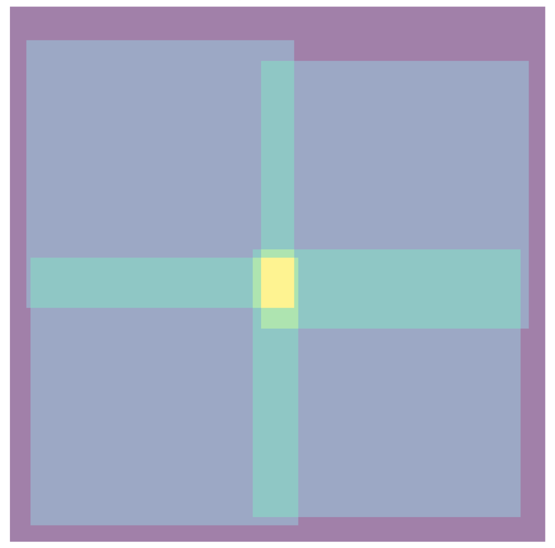
The problem of "unblind" deconvolution

Counts



Low counts astronomical images…



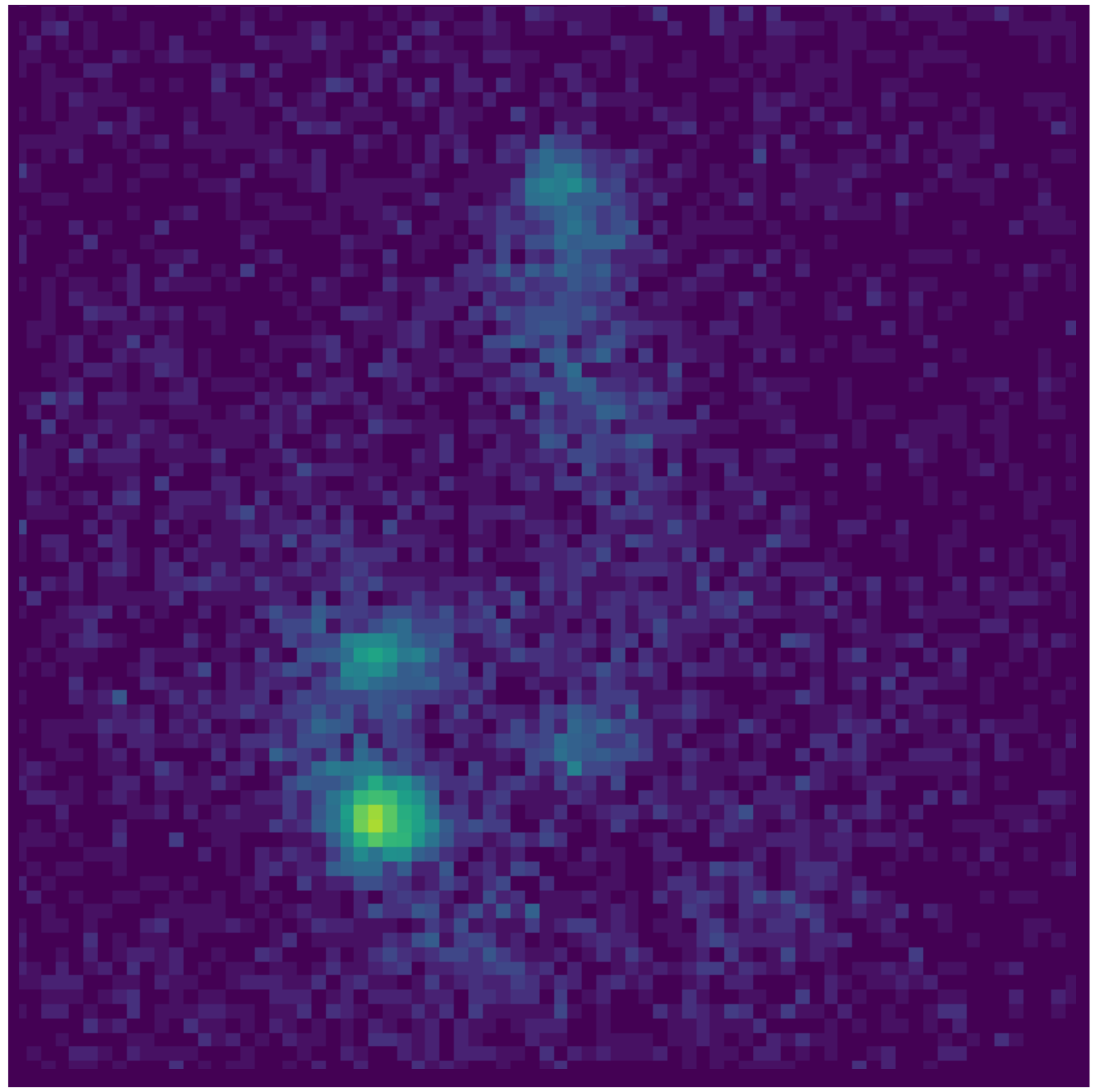


"Unblind": PSF and exposure are known or can be simulated

Reconstruction

The problem of “unblind” deconvolution

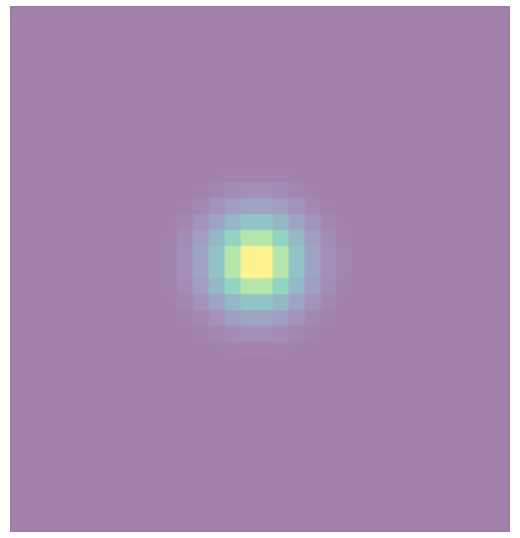
Counts

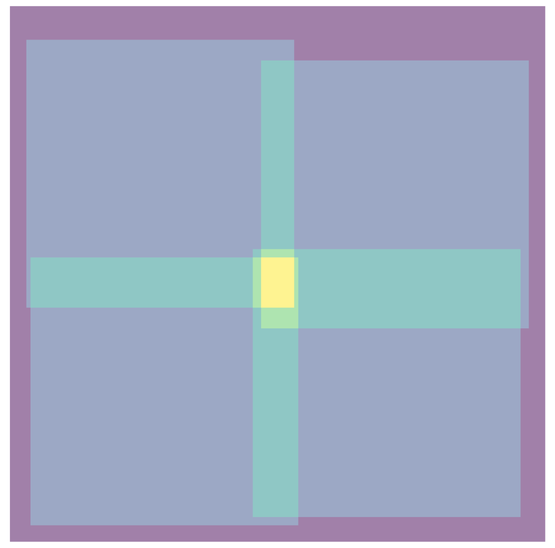


Low counts astronomical images…

$$\mathscr{L}(\mathbf{d}\,|\,\lambda) = \prod_i^N \frac{\lambda_i^{d_i} e^{-d_i}}{\ } \qquad \lambda = \mathbf{x} \circledast \mathrm{PSF}$$

$$\lambda_i = \sum_k \left(e_i \cdot (x_i \; b_i)\right) p_{i-k}$$





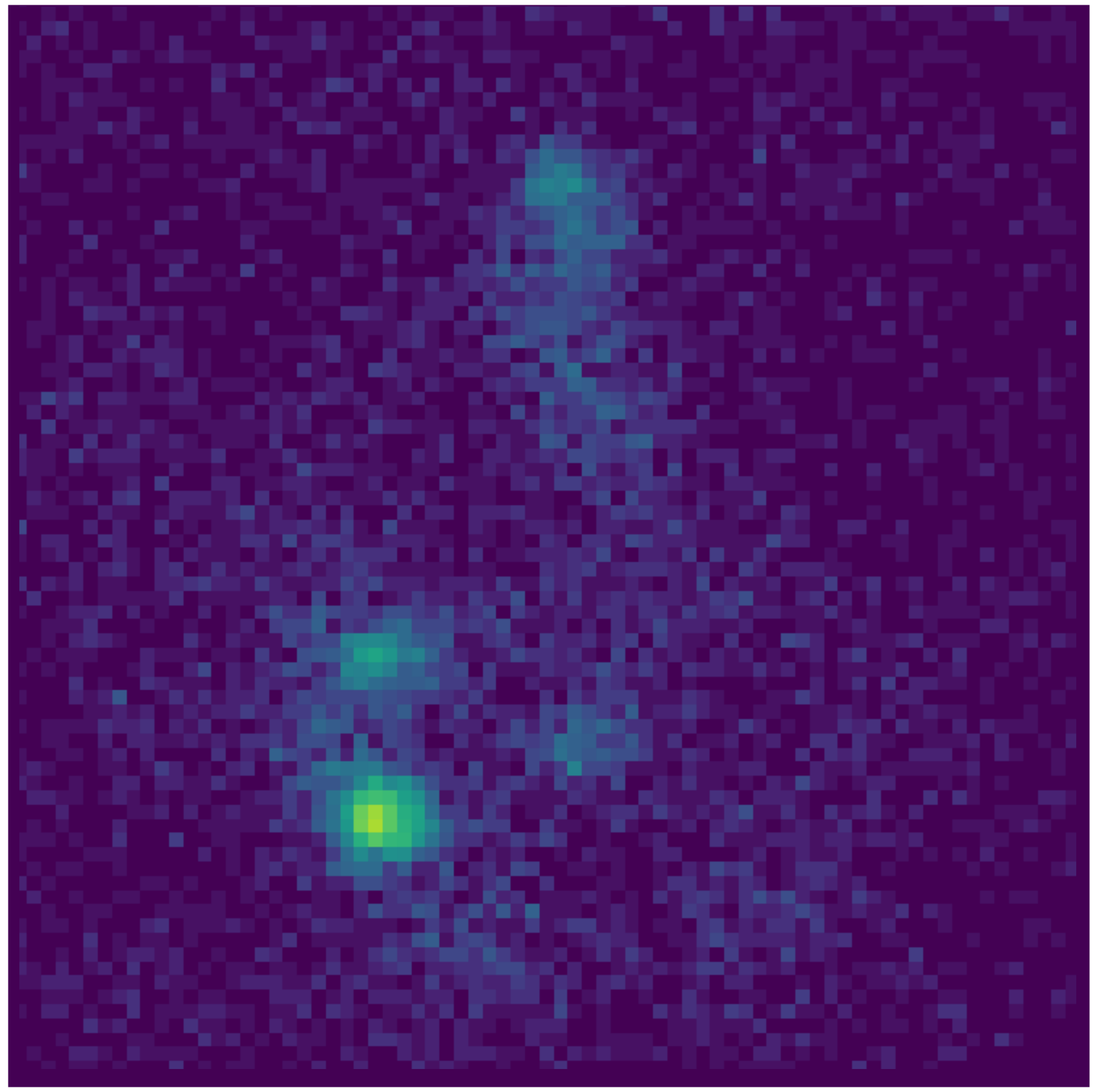
“Unblind”: PSF and exposure are known or can be simulated

Reconstruction

Needs to be reconstructed using statistical methods

The problem of “unblind” deconvolution

Counts

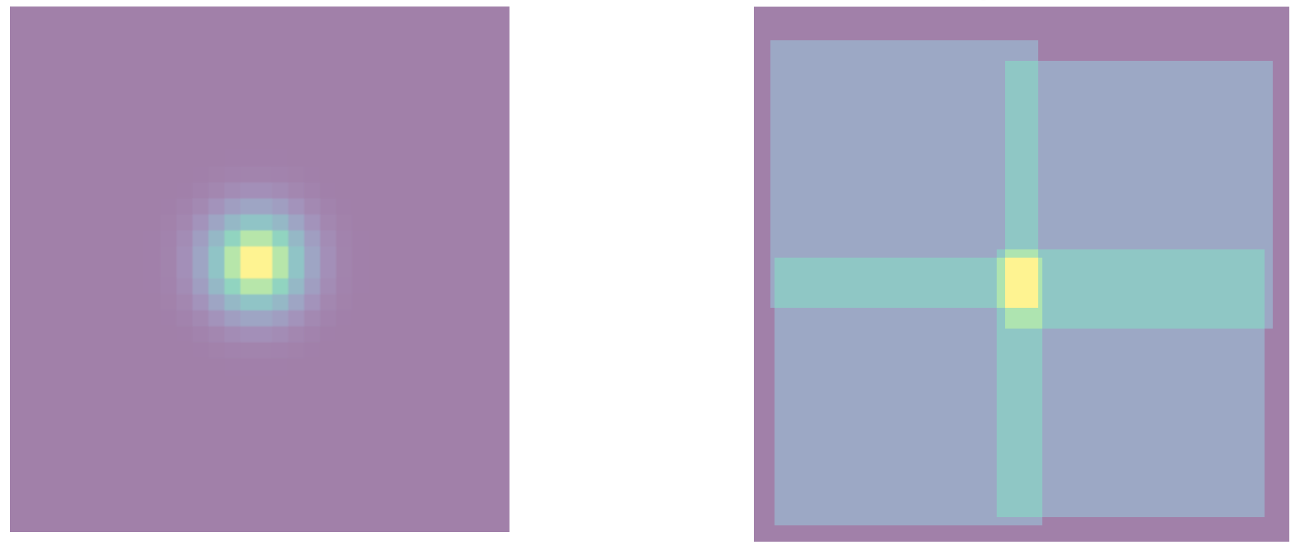


Low counts astronomical images…

$$\mathscr{L}(\mathbf{d}\,|\,\lambda) = \prod_i^N \lambda_i^{d_i} e^{-d_i}$$

$$\lambda = \mathbf{x} \circledast \mathrm{PSF}$$

$$\lambda_i = \sum_k \left(e_i \cdot (x_i \bullet b_i)\right) p_{i-k}$$



“Unblind”: PSF and exposure are known or can be simulated

Reconstruction

Needs to be reconstructed using statistical methods

“Ill-posed inference problem”

Some math…

For the counts image assume per pixel **Poisson** likelihood:

$$\mathscr{L}\left(\mathbf{d} \,|\, \lambda\right) = \prod_{i}^{N} \frac{\lambda_i^{d_i} \mathrm{e}^{-d_i}}{d_i!}$$

Some math...

For the counts image assume per pixel **Poisson** likelihood:

$$\mathscr{L}\left(\mathbf{d} \,|\, \lambda\right) = \prod_{i}^{N} \frac{\lambda_i^{d_i} \mathrm{e}^{-d_i}}{d_i!}$$

As usual: take the **negative log-likelihood**, in Astronomy often call "Cash" statistics

$$\mathscr{C}\left(\mathbf{d} \,|\, \lambda\right) = \sum_{i}^{N} (\lambda_i - d_i \log \lambda_i)$$

Some math…

For the counts image assume per pixel **Poisson** likelihood:

$$\mathscr{L}\left(\mathbf{d}\,|\,\lambda\right) = \prod_i^N \frac{\lambda_i^{d_i} e^{-d_i}}{d_i!}$$

As usual: take the **negative log-likelihood**, in Astronomy often call "Cash" statistics

$$\mathscr{C}\left(\mathbf{d}\,|\,\lambda\right) = \sum_i^N \left(\lambda_i - d_i \log \lambda_i\right)$$

λ are the "model counts"

$$\lambda = \mathbf{x} \circledast \mathrm{PSF}$$

x is the reconstructed image we are looking for. Consider each pixel x_i as independent parameter in the model…

Some math…

For the counts image assume per pixel **Poisson** likelihood:

$$\mathscr{L}\left(\mathbf{d}\,|\,\lambda\right) = \prod_{i}^{N} \frac{\lambda_i^{d_i} \mathrm{e}^{-d_i}}{d_i!}$$

As usual: take the **negative log-likelihood**, in Astronomy often call "Cash" statistics

$$\mathscr{C}\left(\mathbf{d}\,|\,\lambda\right) = \sum_{i}^{N} (\lambda_i - d_i \log \lambda_i)$$

λ are the "model counts"

$$\lambda = \mathbf{x} \circledast \mathrm{PSF}$$

x is the reconstructed image we are looking for. Consider each pixel x_i as independent parameter in the model…

E.g. could be solved by **"Gradient Descent"**…

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \cdot \frac{\partial \mathscr{C}\left(\mathbf{d}\,|\,\mathbf{x}\right)}{\partial x_i}$$

Some math…

For the counts image assume per pixel **Poisson** likelihood:

$$\mathscr{L}\left(\mathbf{d}\,|\,\lambda\right) = \prod_{i}^{N} \frac{\lambda_i^{d_i} \mathrm{e}^{-d_i}}{d_i!}$$

λ are the "model counts"

$$\lambda = \mathbf{x} \circledast \mathrm{PSF}$$

x is the reconstructed image we are looking for. Consider each pixel x_i as independent parameter in the model…

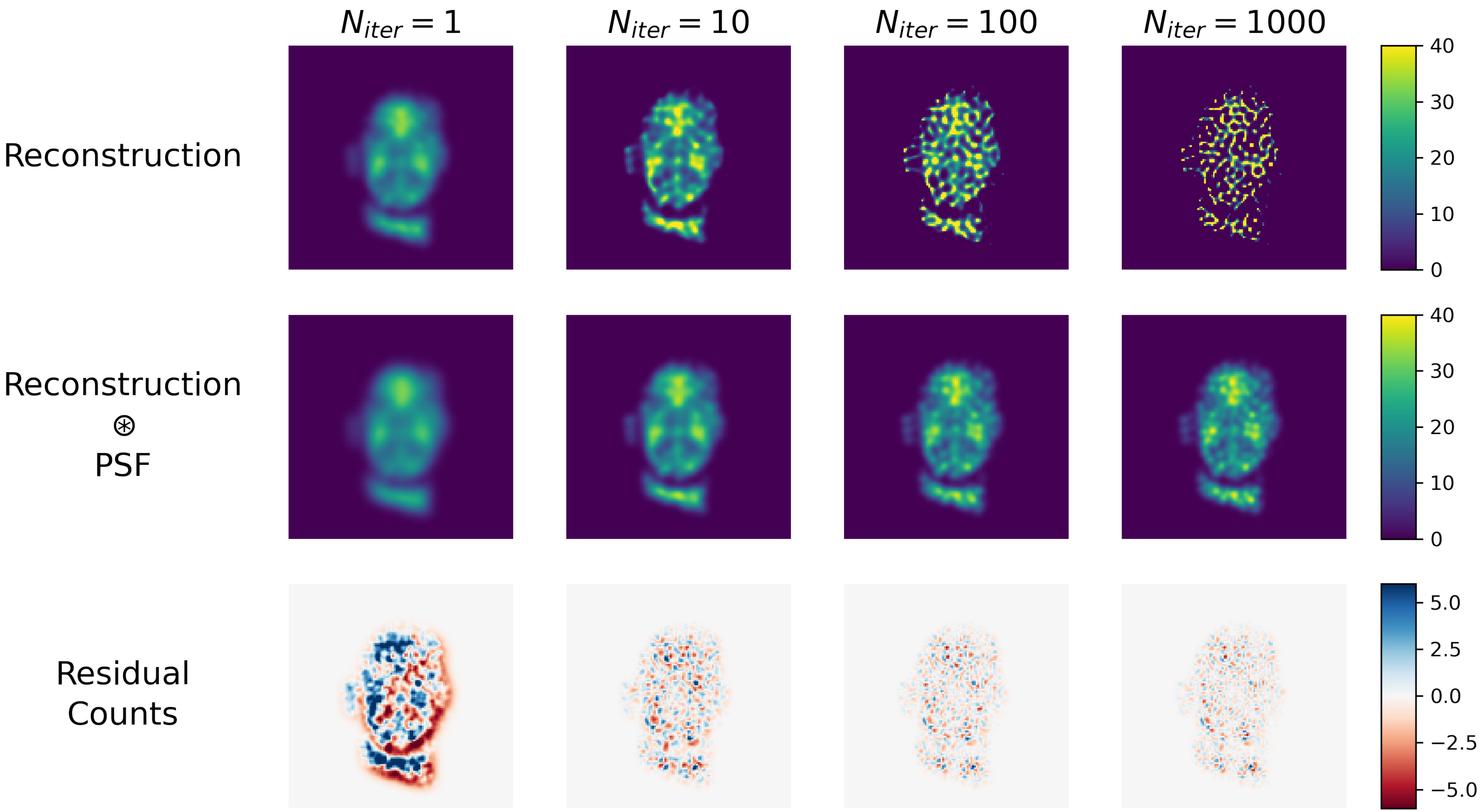
As usual: take the **negative log-likelihood**, in Astronomy often call "Cash" statistics

$$\mathscr{C}\left(\mathbf{d}\,|\,\lambda\right) = \sum_{i}^{N} \left(\lambda_i - d_i \log \lambda_i\right)$$

…or using **Expectation Maximisation (EM)** Proposed by [Richardson 1972] & [Lucy 1974]

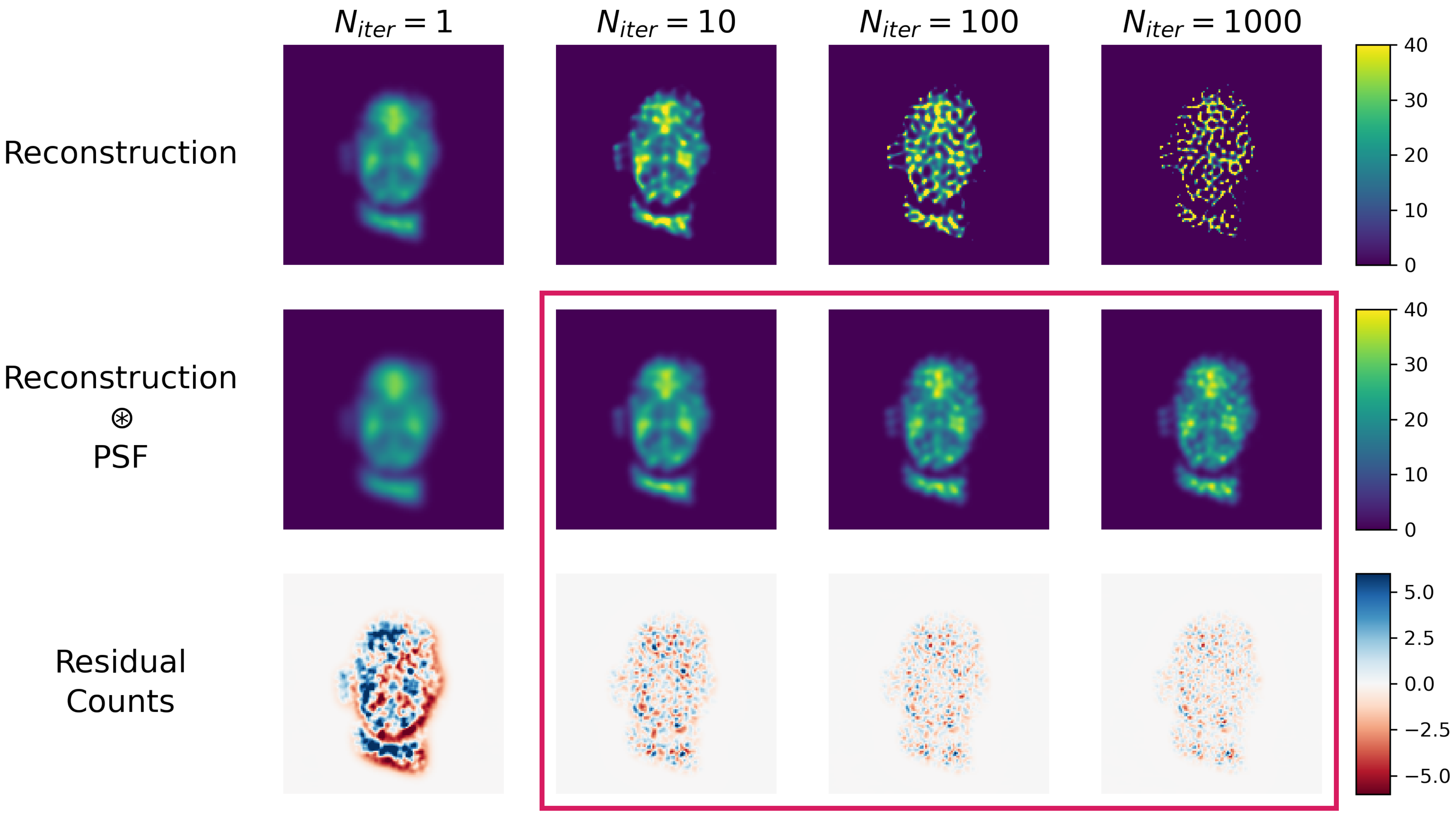
$$\mathbf{x}_{n+1} = \mathbf{x}_n \frac{\mathbf{d}}{\mathbf{x}_n \circledast \mathrm{PSF}} \circledast \mathrm{PSF}^{\mathrm{T}}$$

RL reconstruction quality



Uses [skimage.restoration.richardson_lucy]

RL reconstruction quality



All show good residuals and model counts. But reconstructions very different...

Developed in the CHASC group ~20 yrs ago…

LIRA method

(L)ow counts (I)mage (R)econstruction and (A)nalysis

Objective function **extended by a log-prior term,** only depending on λ :

$$\mathscr{L}\left(\lambda\,|\,\mathbf{d}\right) = \mathscr{C}\left(\mathbf{d}\,|\,\lambda\right) + \mathscr{P}(\mathbf{x})$$

Log-Posterior

Log-Likelihood

LIRA method

(L)ow counts (I)mage (R)econstruction and (A)nalysis

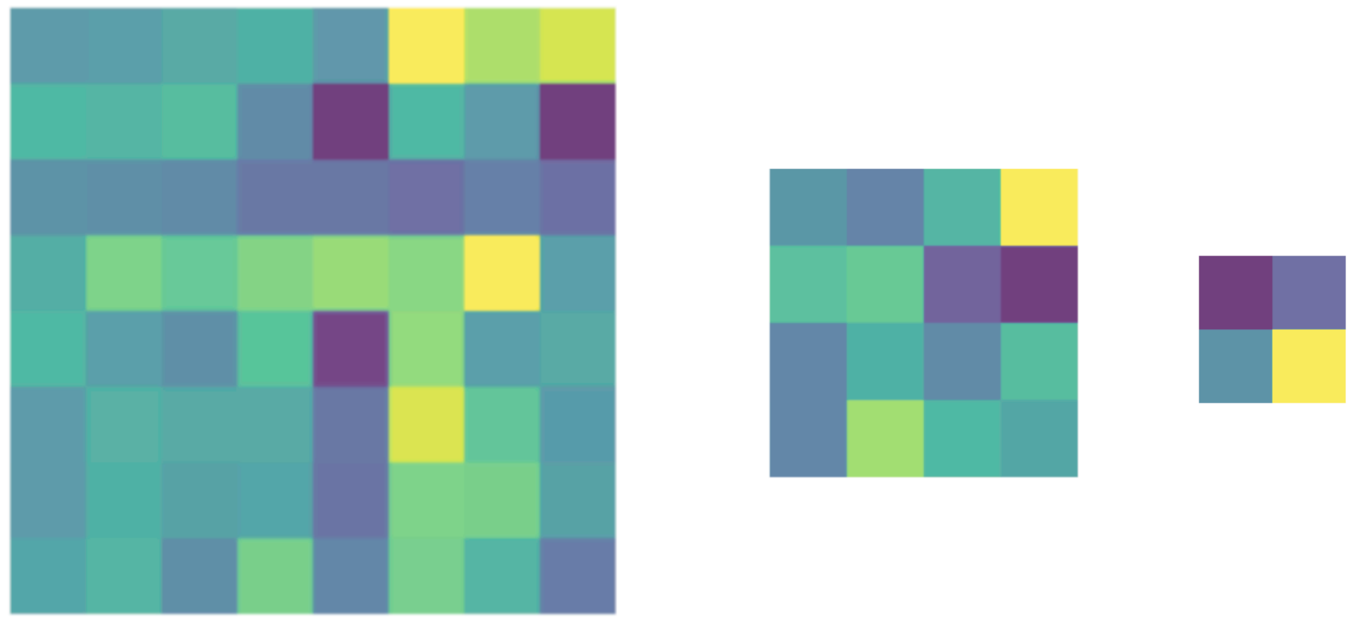
Objective function **extended by a log-prior term,** only depending on λ :

$$\mathscr{L}\left(\lambda \,|\, \mathbf{d}\right) = \mathscr{C}\left(\mathbf{d} \,|\, \lambda\right) + \mathscr{P}(\mathbf{x})$$

Log-Posterior

Log-Likelihood

Log-Prior



A special "multi-scale"prior to achieve **smoothness on multiple scales.**
Initial idea and implementation by [Esch et al. 2004].

LIRA method

(L)ow counts (I)mage (R)econstruction and (A)nalysis

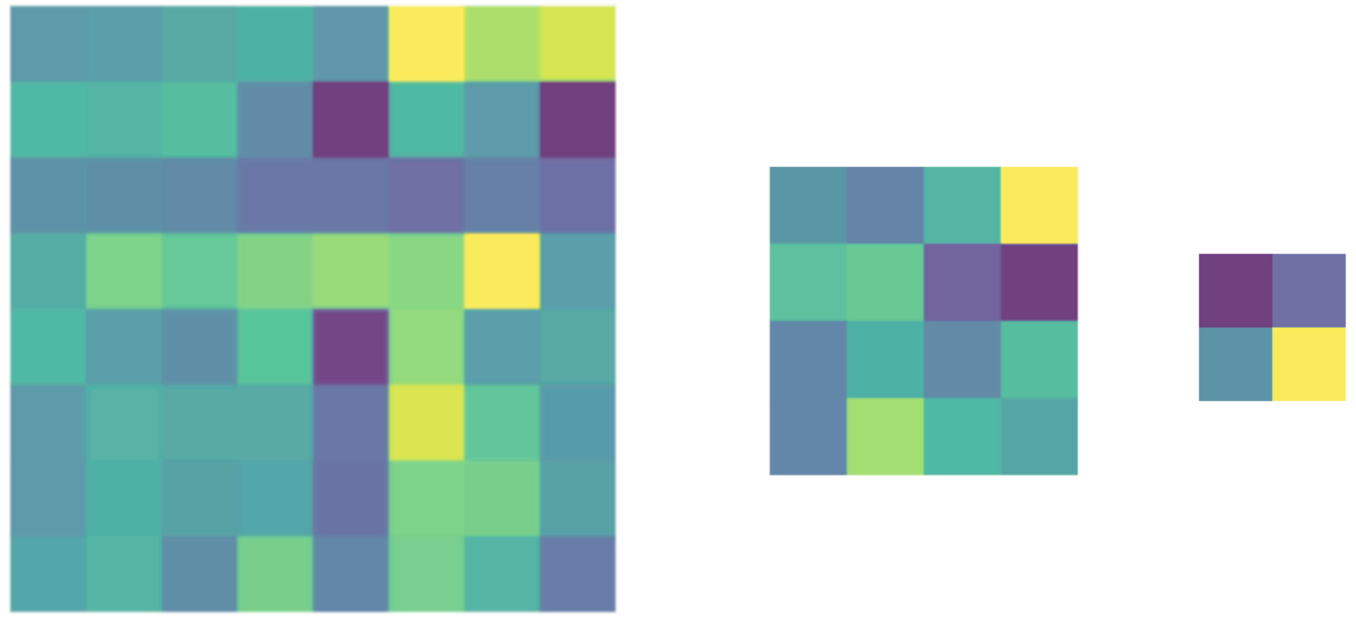
Objective function **extended by a log-prior term,** only depending on λ :

$$\mathscr{L}\left(\lambda \mid \mathbf{d}\right) = \mathscr{C}\left(\mathbf{d} \mid \lambda\right) + \mathscr{P}(\mathbf{x})$$

Log-Posterior

Log-Likelihood

Log-Prior



A special "multi-scale"prior to achieve **smoothness on multiple scales.**
Initial idea and implementation by [Esch et al. 2004].

Model counts extend by **exposure e** and optional **baseline ("background") component b** by [Connors et al. 2011]

$$\lambda = [(\mathbf{x} + \mathbf{b}) \cdot \mathbf{e}] \circledast \mathrm{PSF}$$

LIRA method

(L)ow counts (I)mage (R)econstruction and (A)nalysis

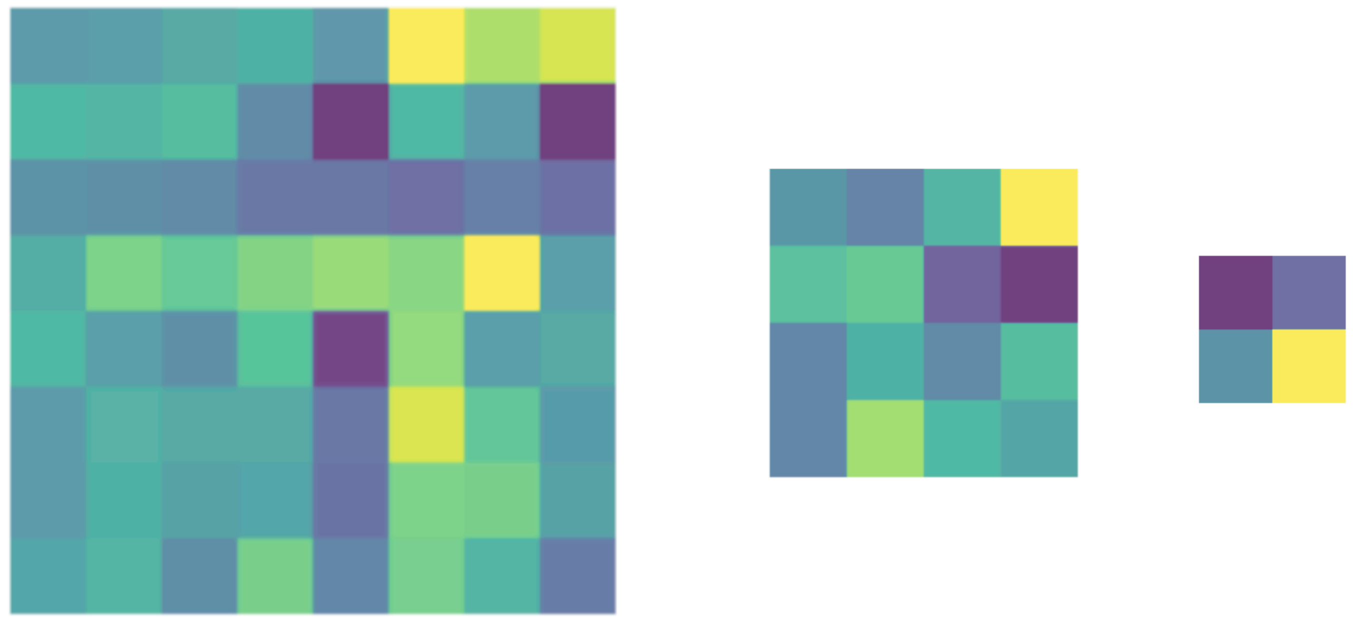
Objective function **extended by a log-prior term,** only depending on λ :

$$\mathscr{L}\left(\lambda\,|\,\mathbf{d}\right) = \mathscr{C}\left(\mathbf{d}\,|\,\lambda\right) + \mathscr{P}(\mathbf{x})$$

Log-Posterior

Log-Likelihood

Log-Prior



A special "multi-scale"prior to achieve **smoothness on multiple scales.** Initial idea and implementation by [Esch et al. 2004].

Model counts extend by **exposure e** and optional **baseline ("background") component b** by [Connors et al. 2011]

$$\lambda = [(\mathbf{x} + \mathbf{b}) \cdot \mathbf{e}] \circledast \mathrm{PSF}$$

Change from EM to **MCMC sampling** from the log-posterior. Basically **sampling a series of images**…

Pylira Python package

- The LIRA method was **initially implemented as an R package** by [Connors et al. 2011]. Original code available at https://github.com/astrostat/LIRA
- Meanwhile Python has become the favored language of choice for astronomical data analysis.
- **Pylira is a Python wrapper** around the initial C implementation implemented via [pybind11] and based on the [Astropy affiliated package template]
- Additional dependencies are Numpy, Scipy, Astropy, Matplotlib, …
- Github: https://github.com/astrostat/pylira
- Docs: https://pylira.readthedocs.io/en/latest/

A short code example…

```python
import numpy as np
from pylira import LIRADeconvolver
from pylira.data import point_source_gauss_psf

random_state = np.RandomState(372)

# load built in test dataset
data = point_source_gauss_psf()

data["flux_init"] = random_state.gamma(10, size=(32, 32))

# sklearn inspired API, take config on init
deconvolve = LIRADeconvolver(
    alpha_init=np.ones(5),
    n_iter_max=5_000,
    n_burn_in=500,
    random_state=random_state,
)

# run algorithm
result = deconvolve.run(data=data)

# serialise to FITS
result.write(“pylira-result.fits”)
```

[From Simple Point Source Tutorial]

A short code example…

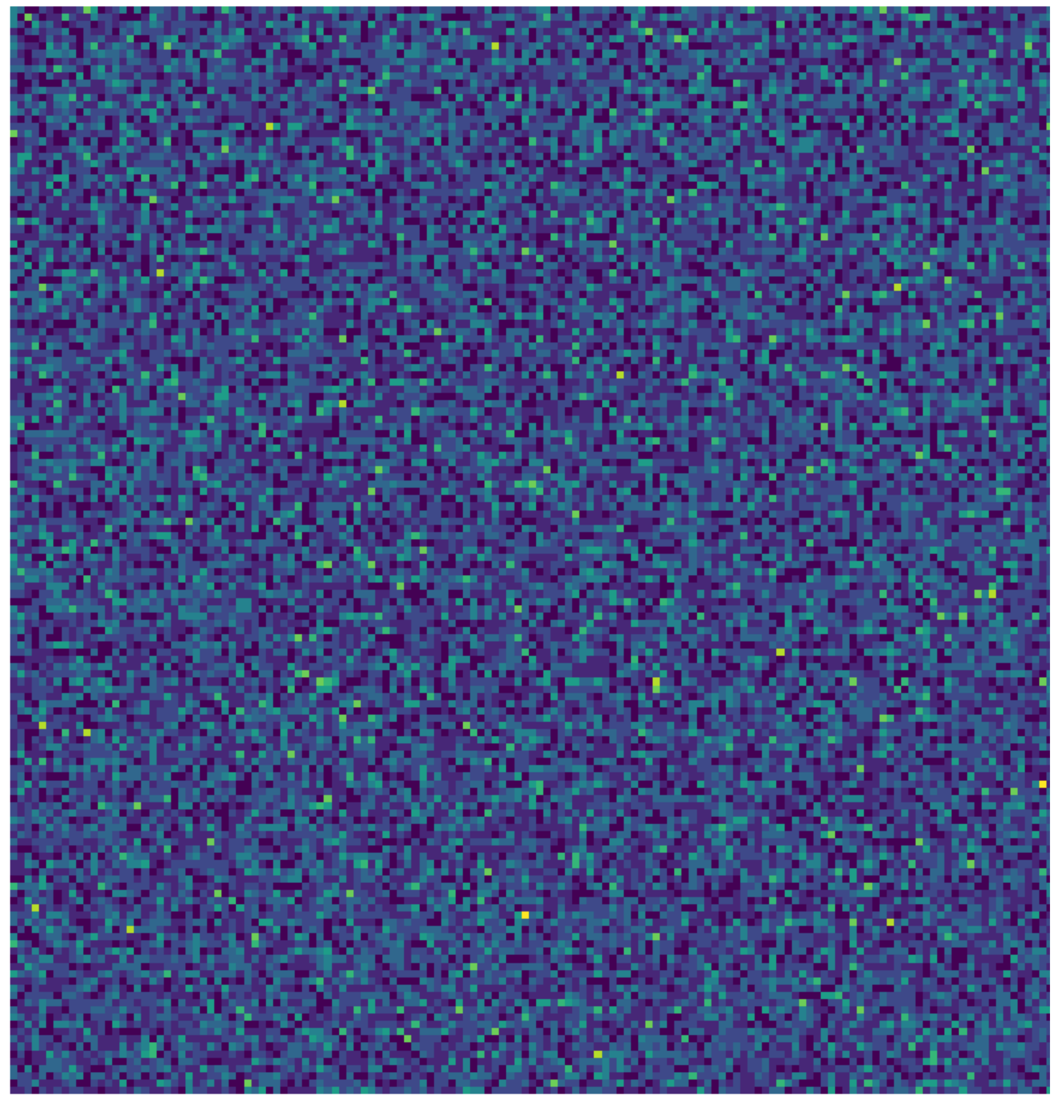
```python
import numpy as np
from pylira import LIRADeconvolver
from pylira.data import point_source_gauss_psf

random_state = np.RandomState(372)

# load built in test dataset
data = point_source_gauss_psf()

data["flux_init"] = random_state.gamma(10, size=(32, 32))

# sklearn inspired API, take config on init
deconvolve = LIRADeconvolver(
    alpha_init=np.ones(5),
    n_iter_max=5_000,
    n_burn_in=500,
    random_state=random_state,
)

# run algorithm
result = deconvolve.run(data=data)

# serialise to FITS
result.write(“pylira-result.fits”)
```

How much responsibility is on the CHASC members to provide implementations documentation and nice users APIs for statistical methods?

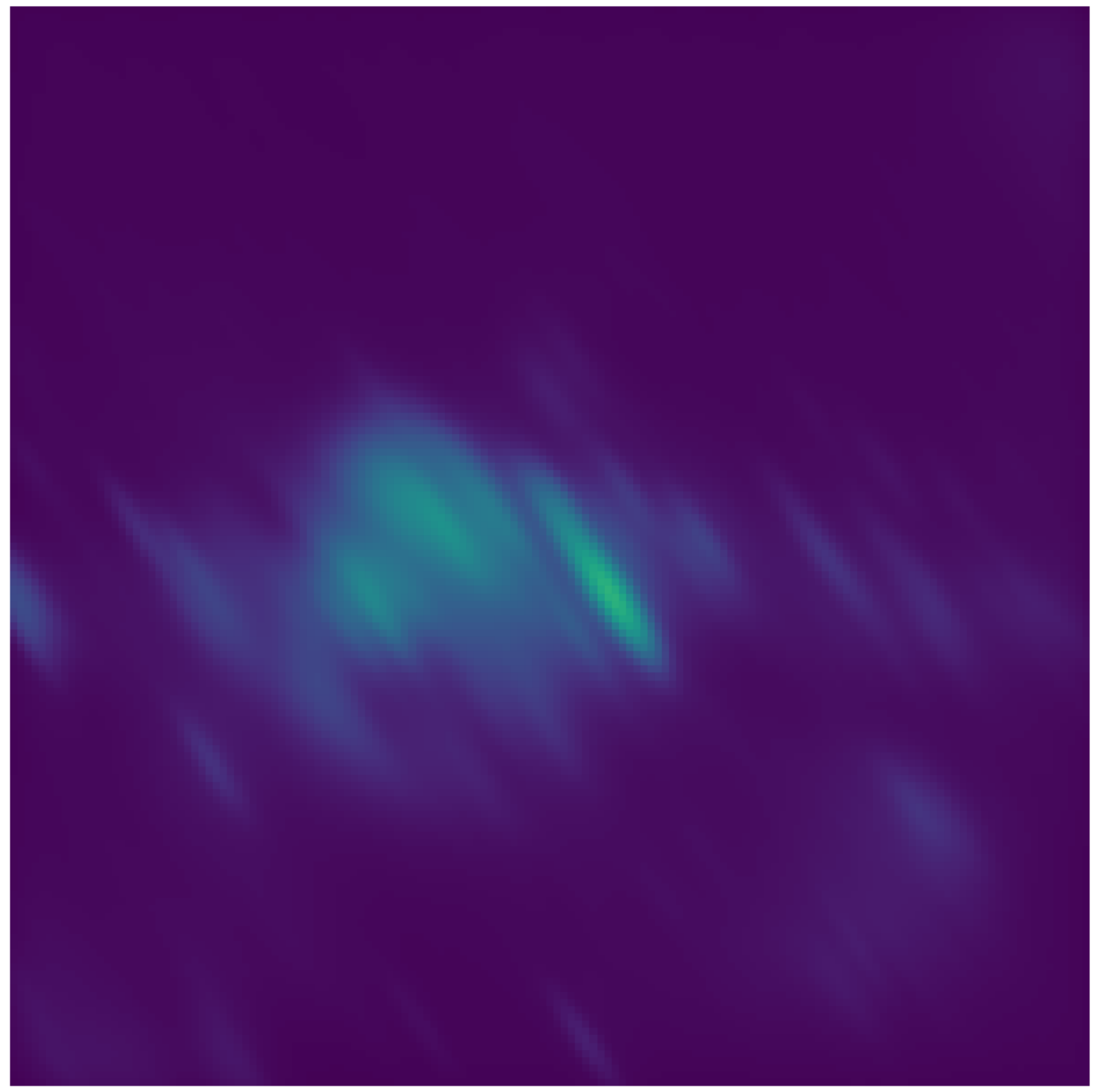
[From Simple Point Source Tutorial]

Patch "prior"

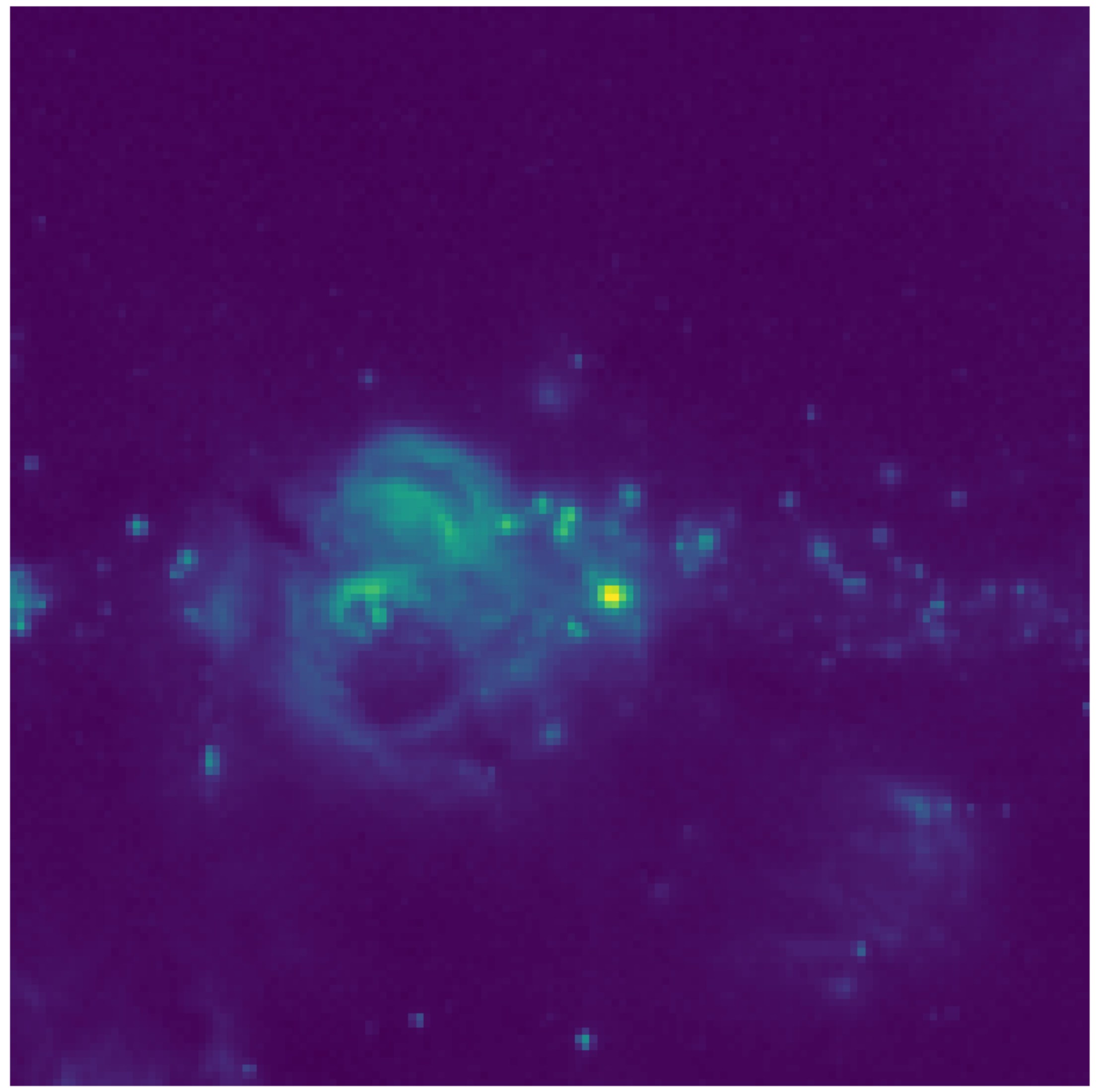
Basic motivation



An unlikely image



A more likely image

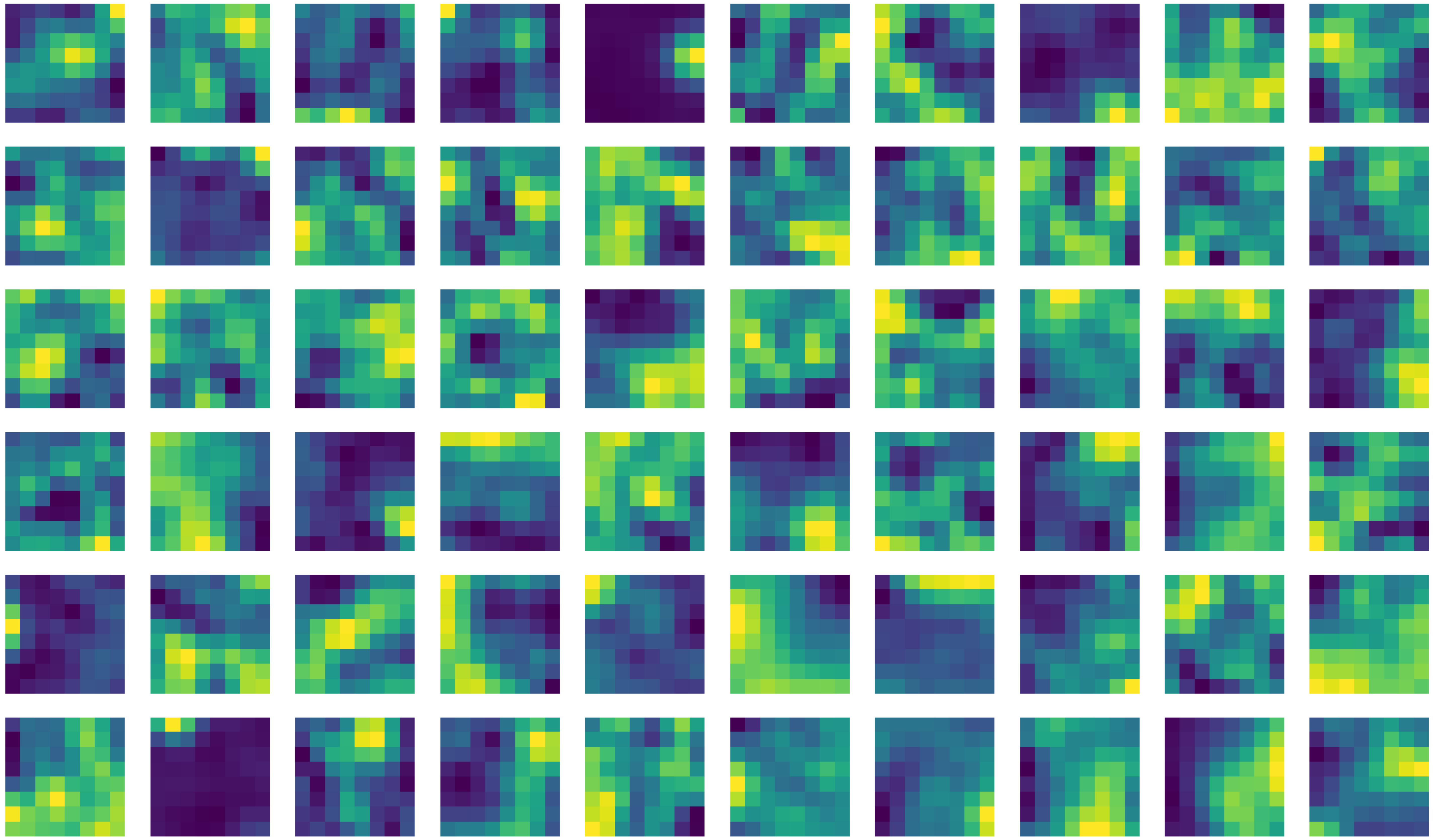


A likely image

- Intuitively we humans have a good understanding of what an actual astronomical image should look like, because we learned it from seeing many images
- Learning a full image PDF in a "deep learning way" is hard, there is not enough training data, we have no ground truth etc.
- But…

Patch "prior"

Main idea



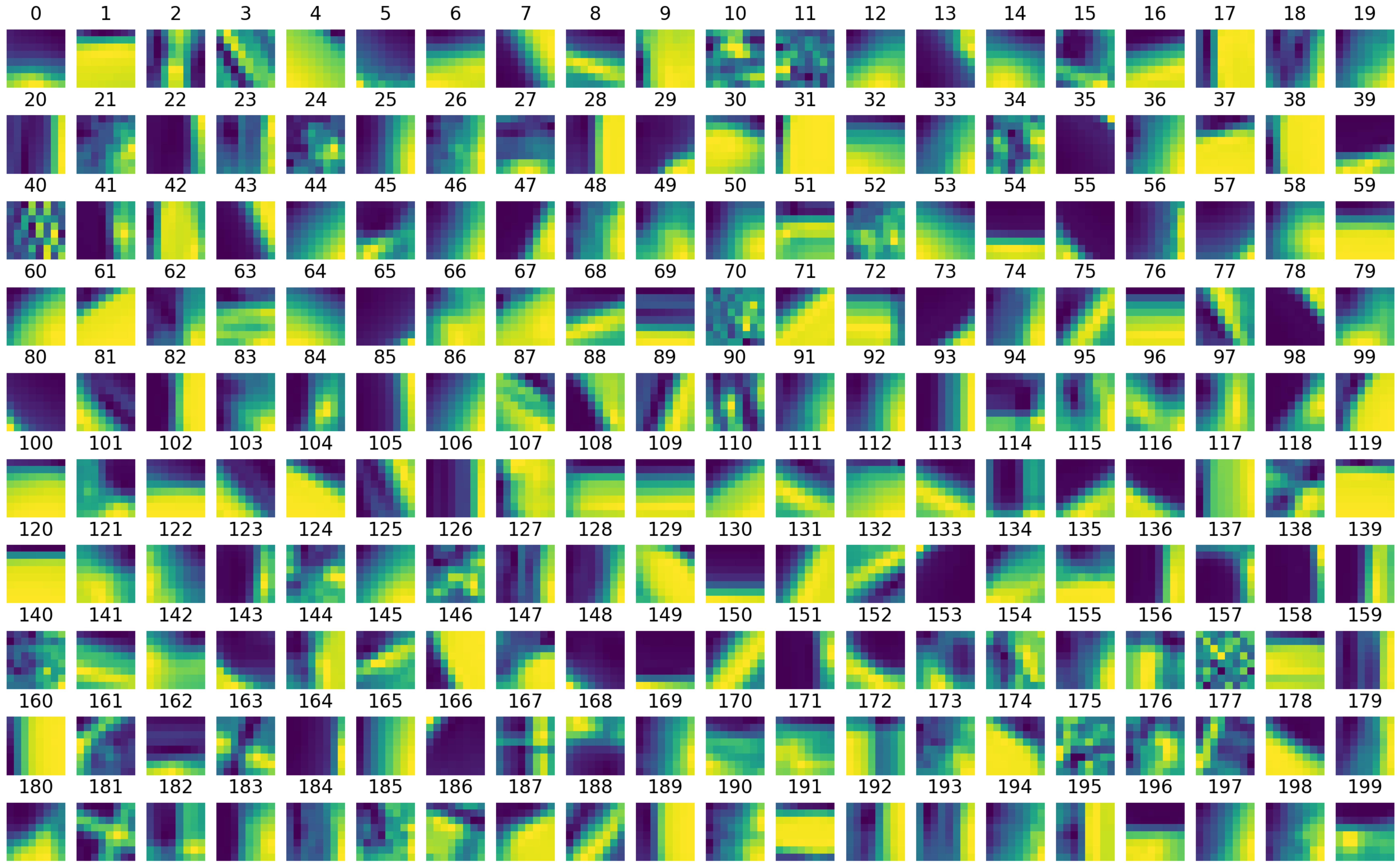
Some example patches from an astronomical image…

- Split images(s) into "patches" of a given size, e.g. 8x8 pixels
- Learn a 64 dimensional Gaussian Mixture Model (GMM) with N=200 components
- One can compute the likelihood for a GMM and train them using EM

$$\mathscr{P}(\mathbf{x}) = \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k N(x_i; \mu_k, \sigma_k^2)\right)$$

- Initial idea by [Zoran et al. 2011]
- Also used in EHT reconstruction "CHIRP" algorithm [Bouman et al. 2016]

GMM "Eigenimages"



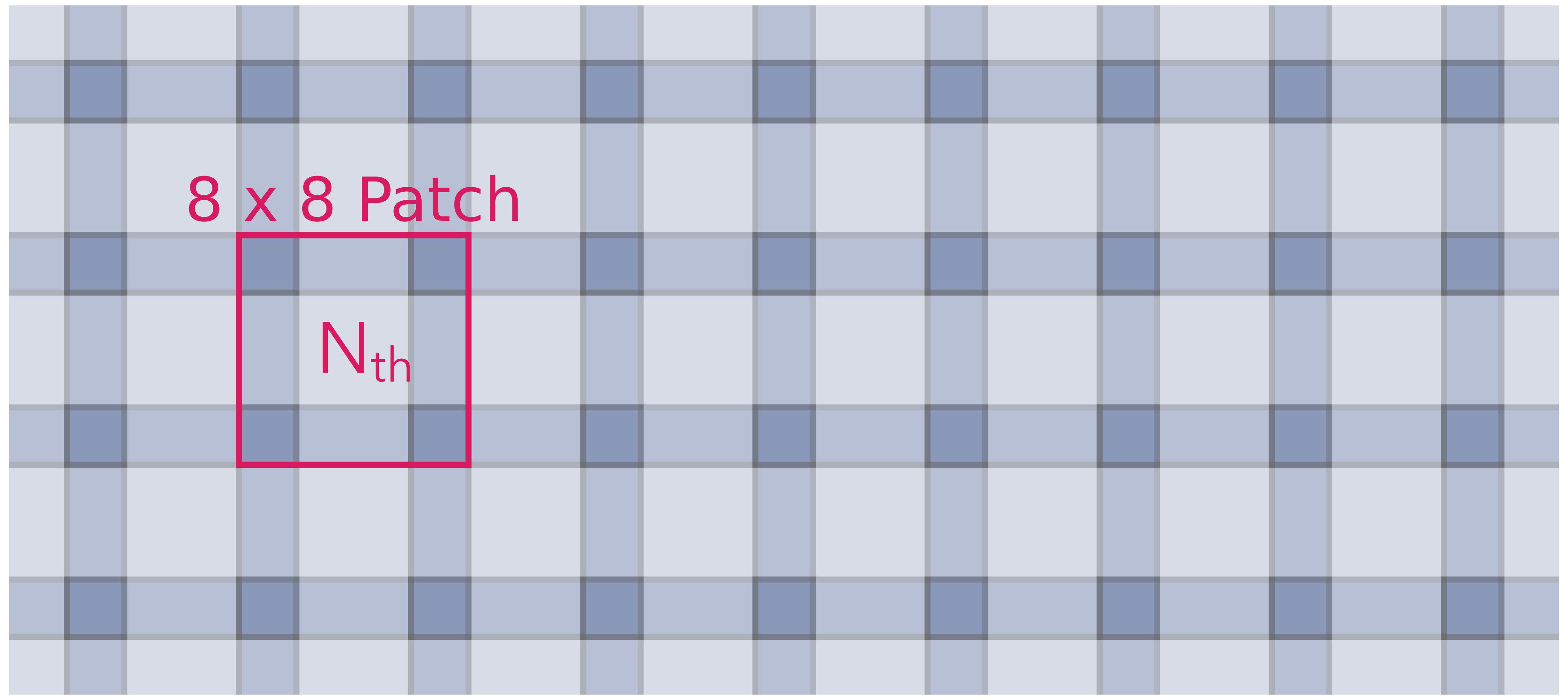
- GMM as clustering algorithm
- Patches are grouped in different "base" structures such as edges, curves, lines, etc.

GMM by [Zoran et al. 2011], trained on "real world" images

Patch prior

Reconstruction

- In each iteration split the image (which is reconstructed into overlapping patches



- For each patch evaluate the GMM and chose the component with the highest log-likelihood

- Sum up these “best” log-likelihood values for all patches in the image to compute the total log-prior value

$$\mathscr{P}(\mathbf{x}) = \sum_i \log p_{\hat{k},GMM}(\mathbf{P}_i \mathbf{x})$$

Where $\mathbf{P}_i$ is the matrix that extracts the i-th patch from $\mathbf{x}$

- Optimize the total log-posterior to achieve a Maximum A Posteriori (MAP) estimate

Jolideco

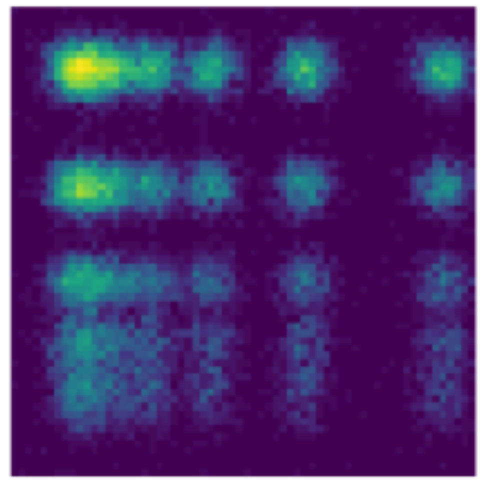
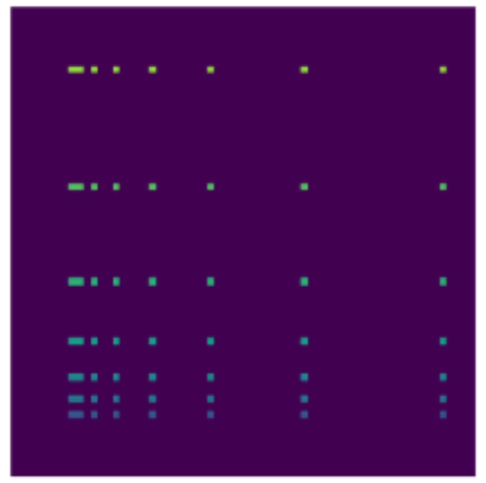
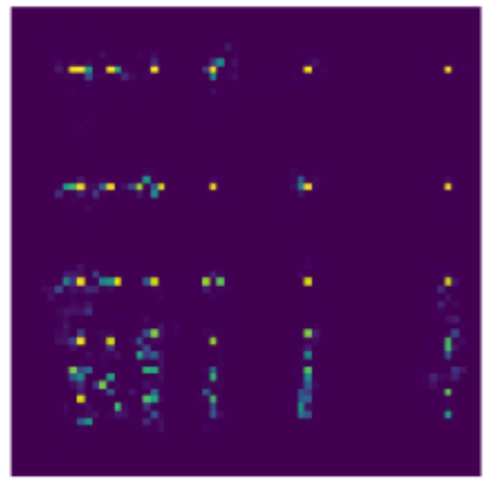
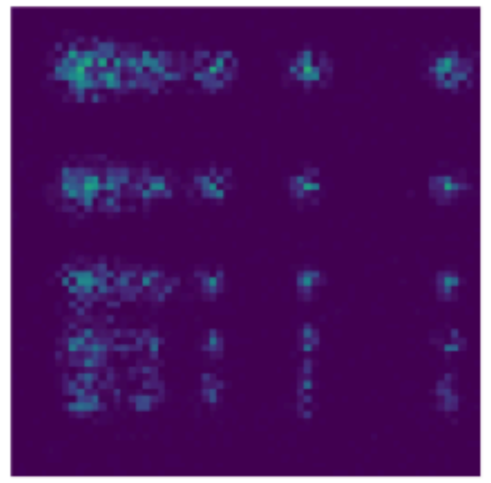
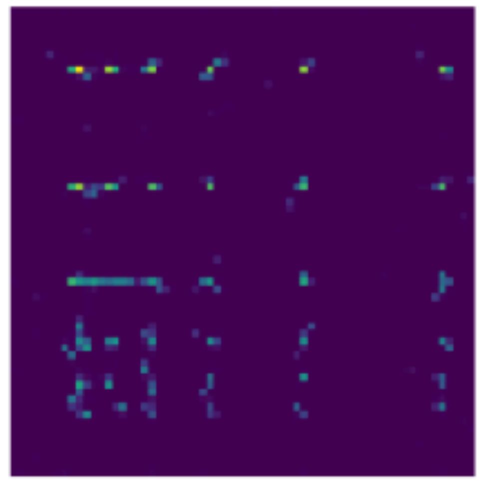
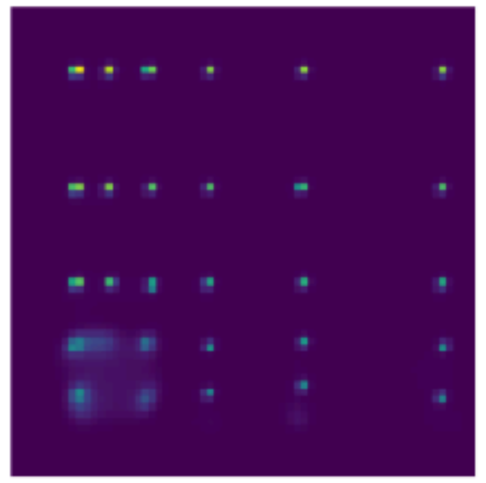
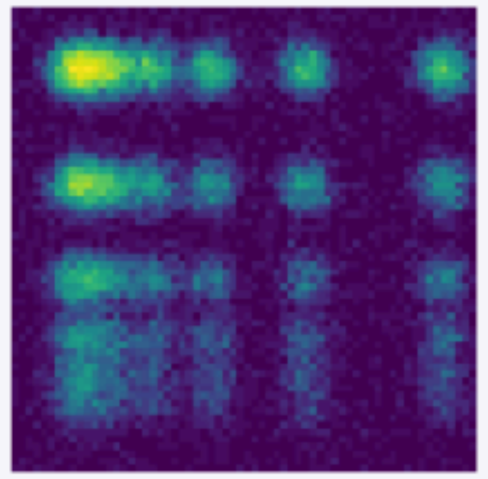
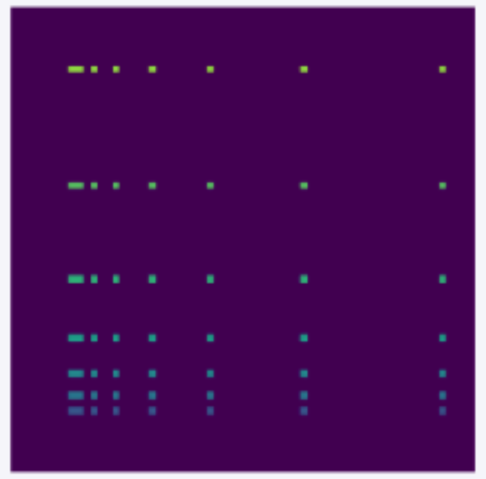
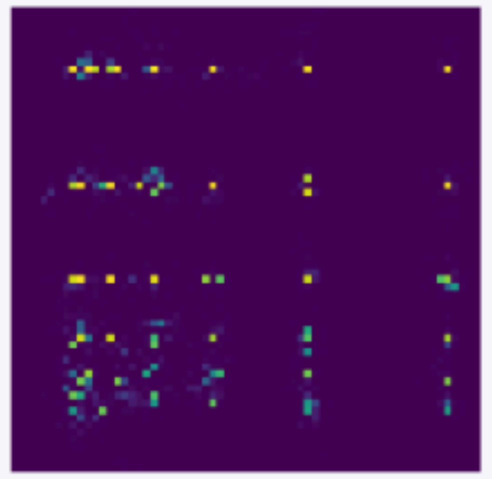
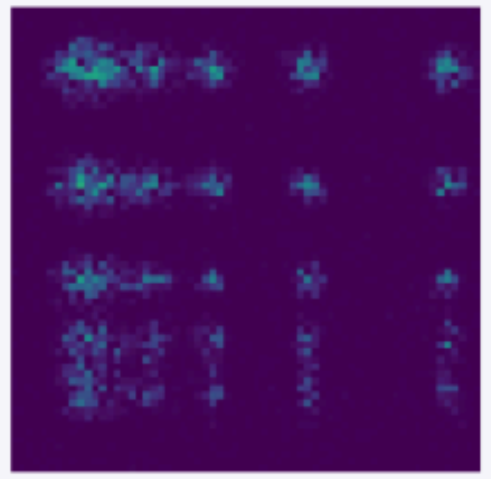
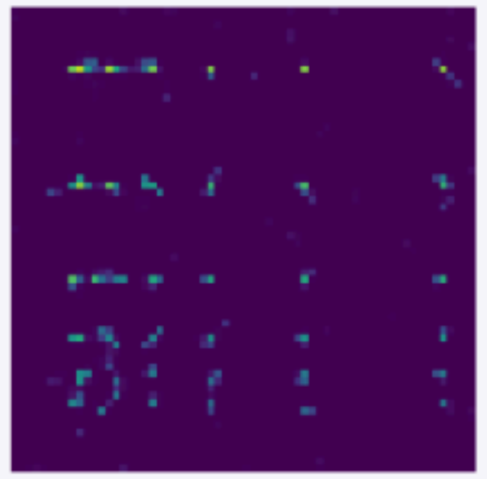
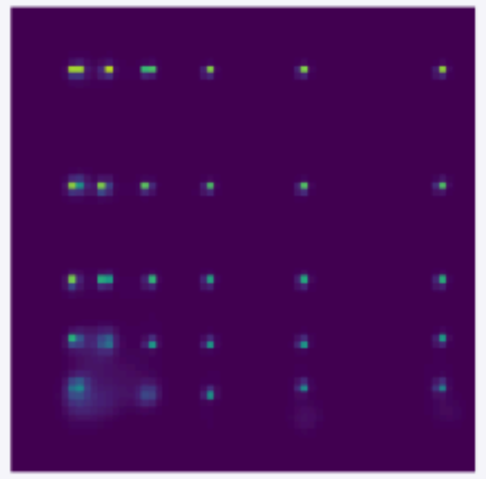
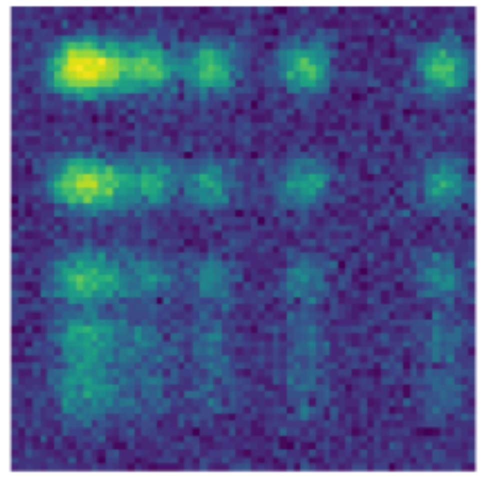
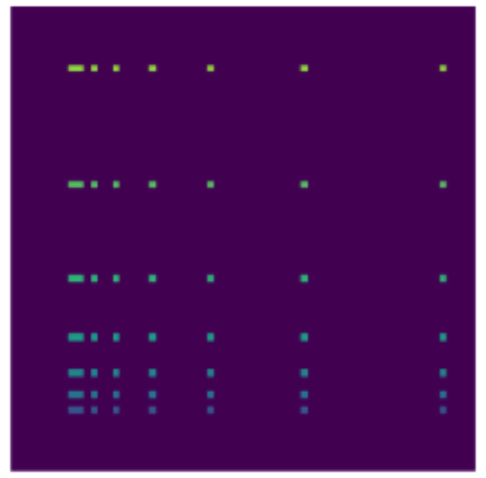
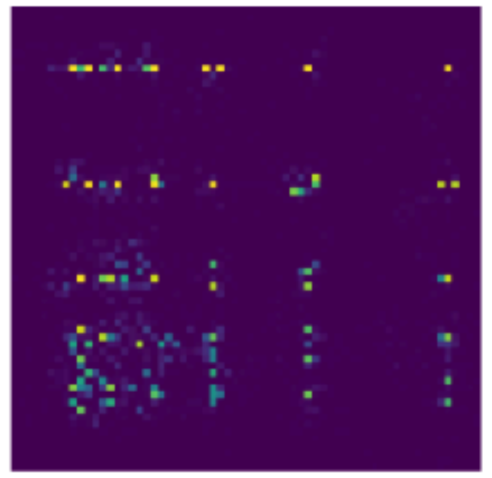
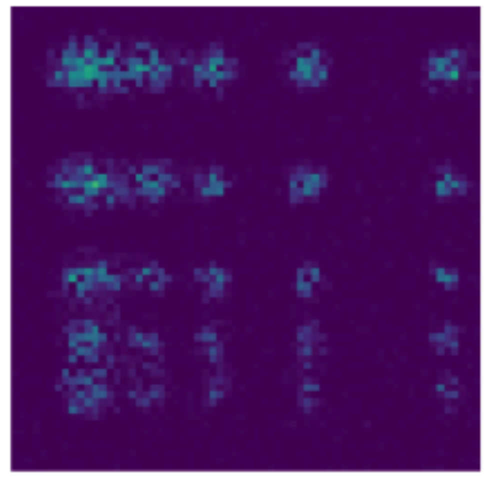
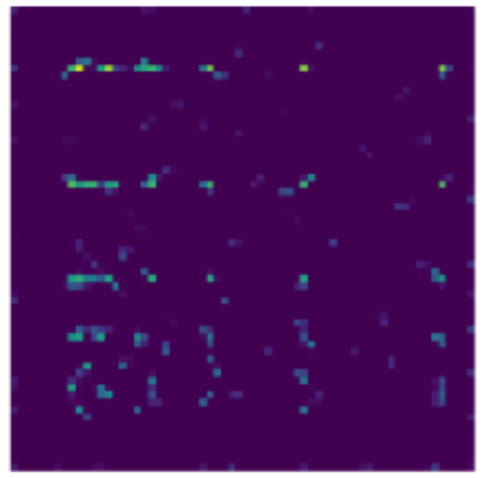
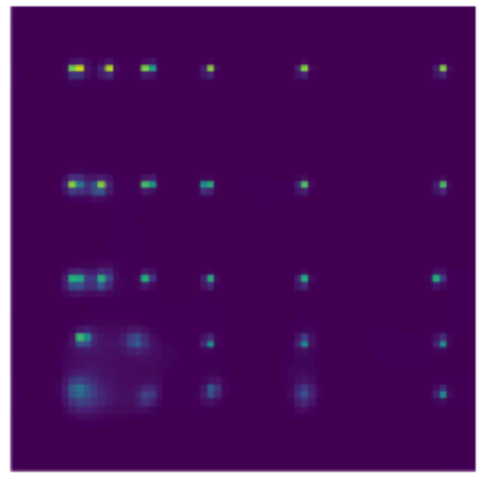
- Python based framework for (Jo)int (Li)kelihood (Deco)nvolution in presence of Poisson noise
- "Joint" refers to an extended likelihood function, with M independent observations:

$$\mathscr{L}\left(\mathbf{d_m} \mid \mathbf{x}\right) = \sum_{m}^{M} \mathscr{C}\left(\mathbf{d_m} \mid \mathbf{x}\right) - \beta \cdot \mathscr{P}(\mathbf{x})$$

- Implements the Poisson likelihood function with flexible prior, such as a uniform prior (basically equivalent to RL) and patch priors
- Optionally can reconstruct an upsampled image X
- Based on [Pytorch] for MAP estimation. Pytorch does automatic gradients (e.g. can compute the gradient of the GMM prior). It also allows for "scalability" by moving computations to the GPU
- Not yet public code…

Jolideco Comparison

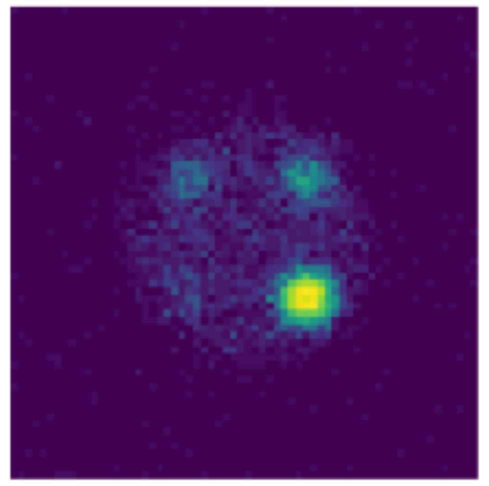
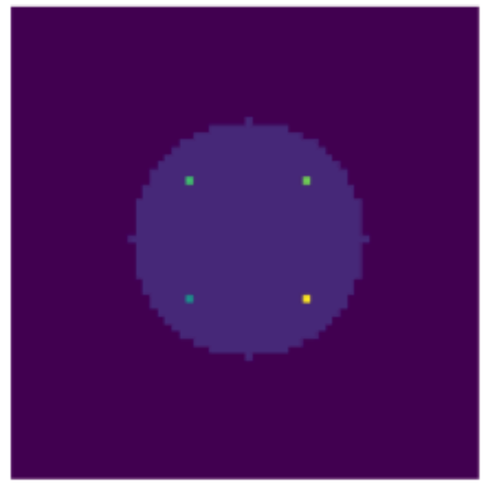
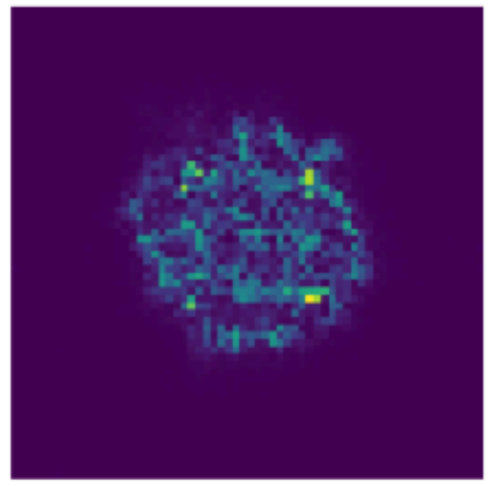
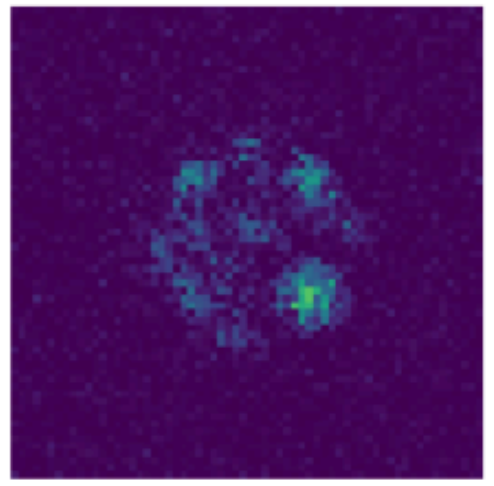
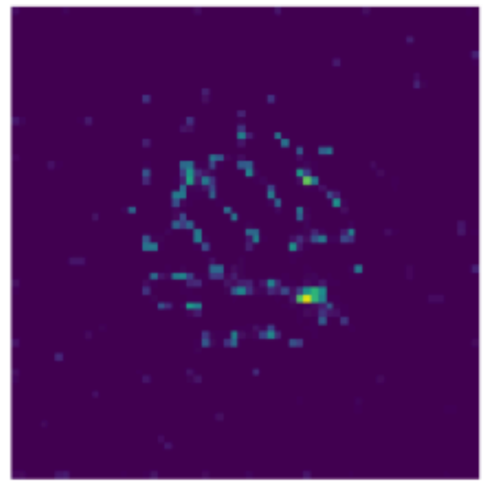
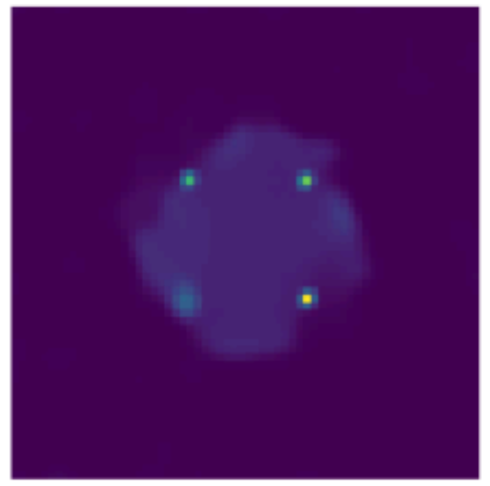
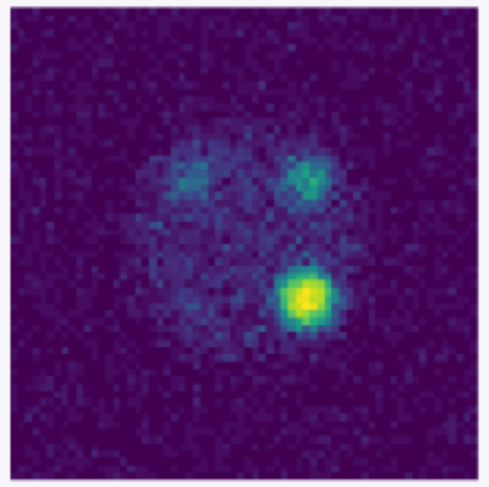
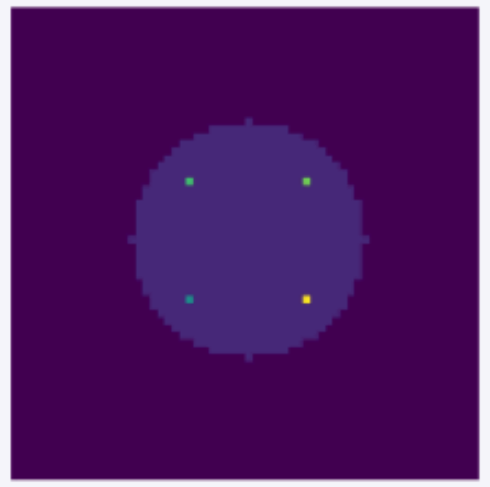
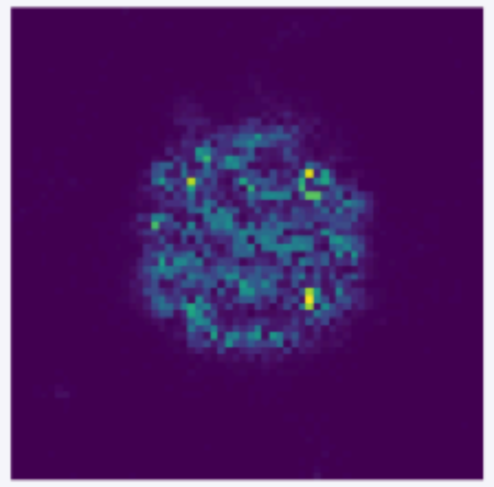
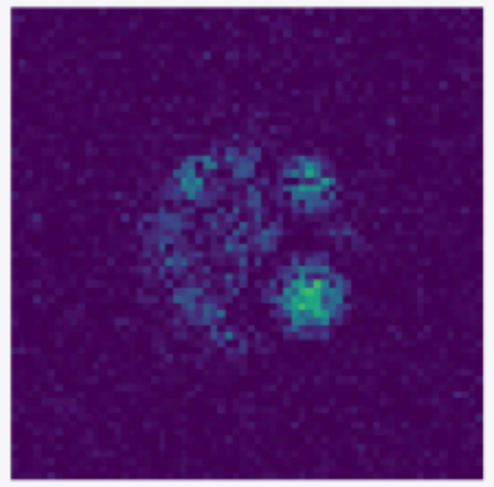
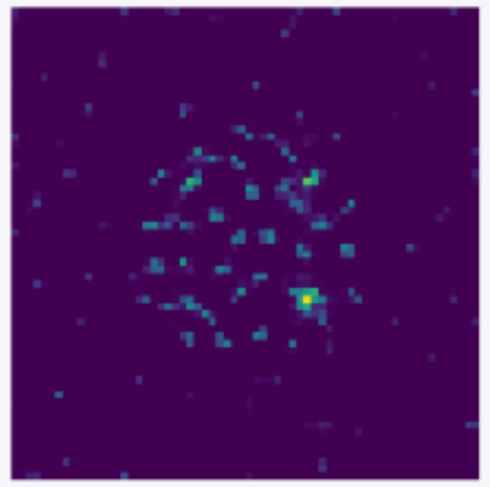
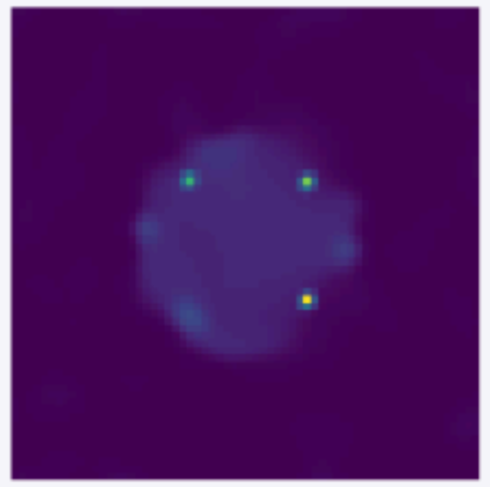
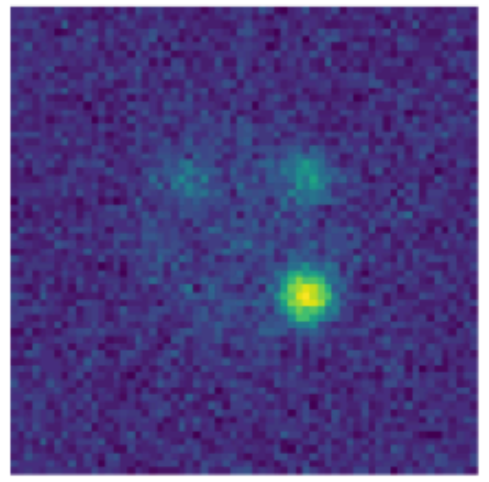
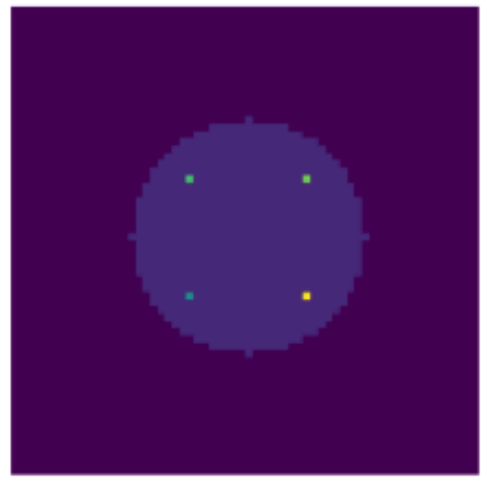
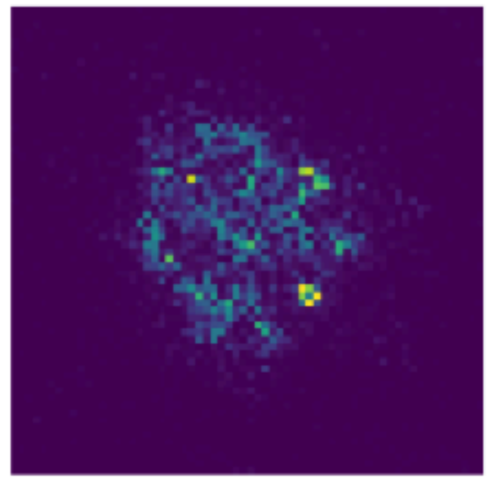
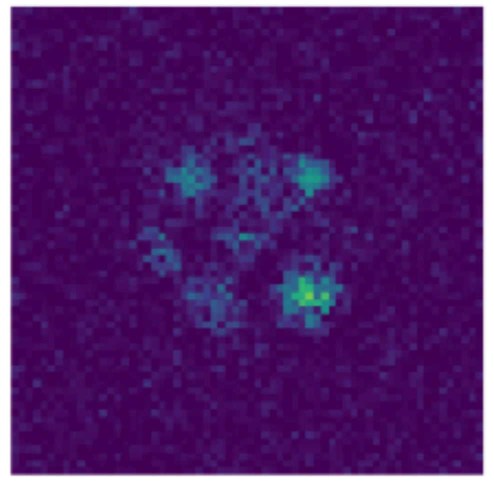
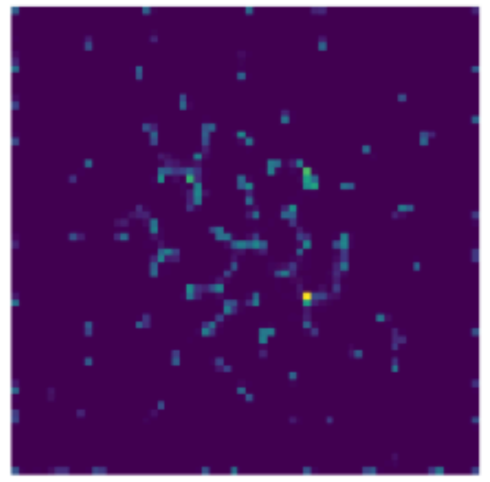
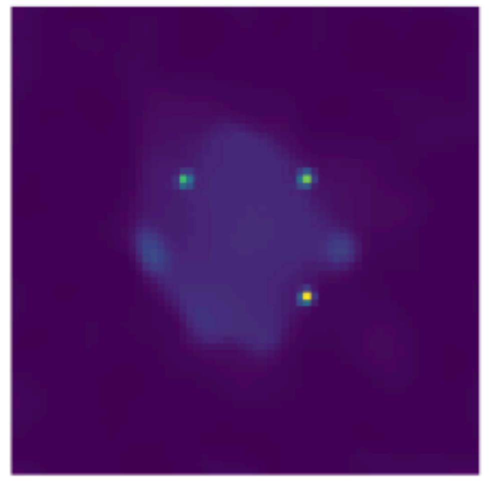
bg1 ≙ 0.01 ct / pixel
bg2 ≙ 0.1 ct / pixel
bg3 ≙ 1 ct / pixel

Bkg. Level	Counts	Ground Truth	Pylira	Jolideco Uniform Prior N=10	Jolideco Uniform Prior N=1000	Jolideco Patch Prior
bg1						
bg2						
bg3						

Thanks Vinay for the test datasets!

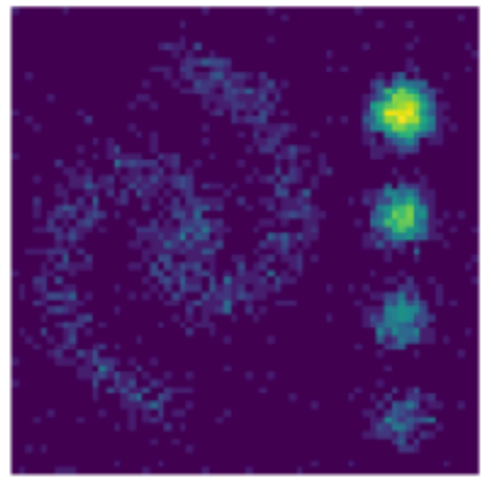
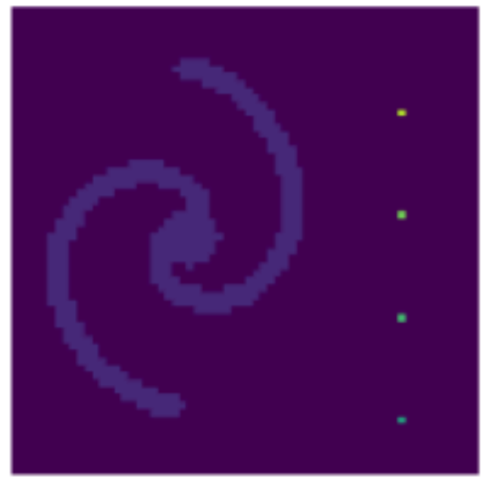
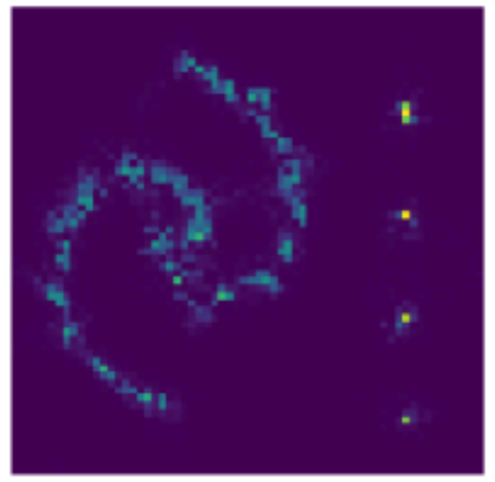
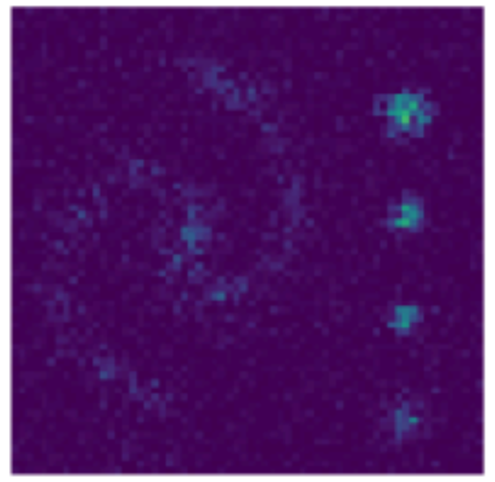
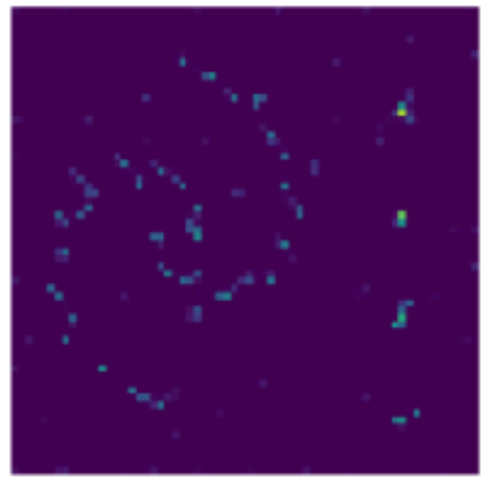
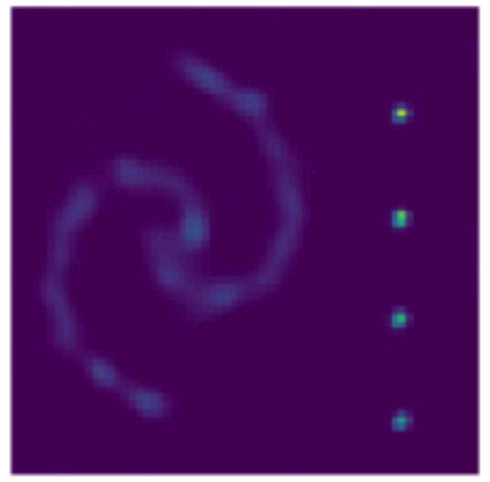
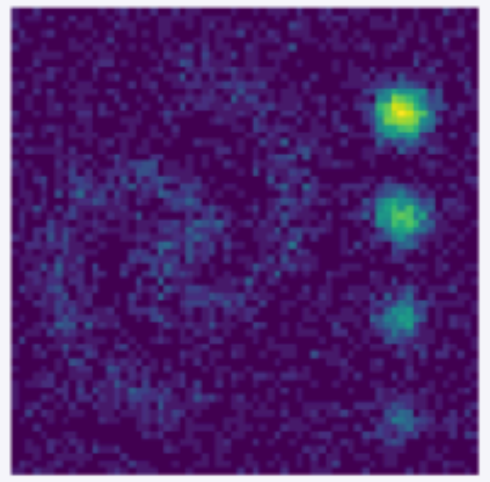
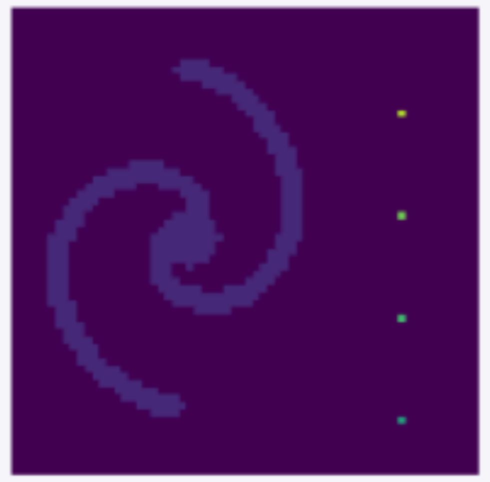
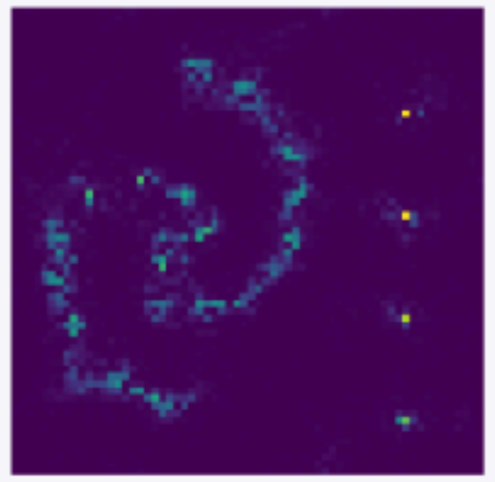
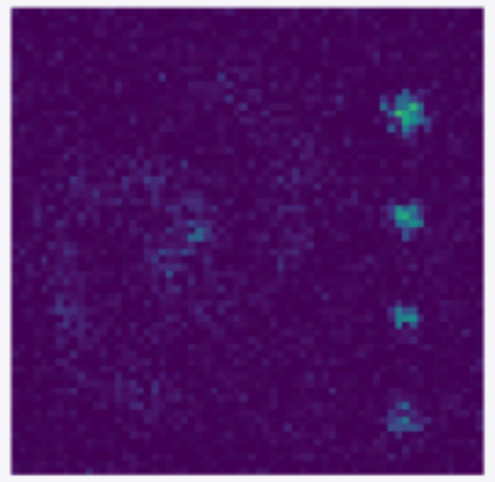
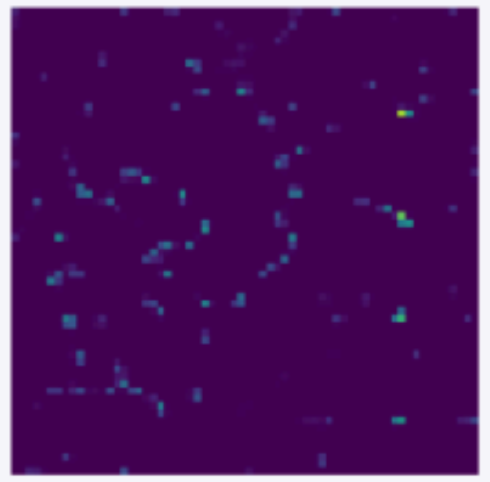
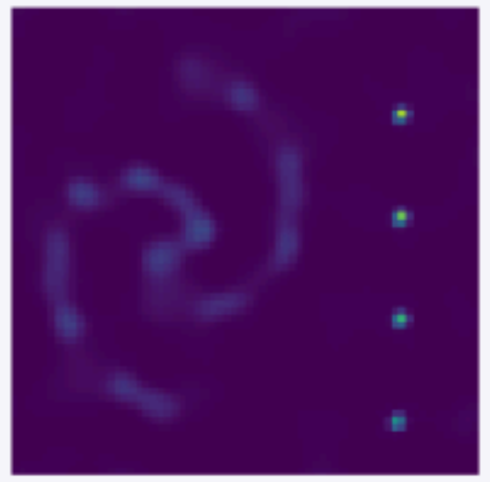
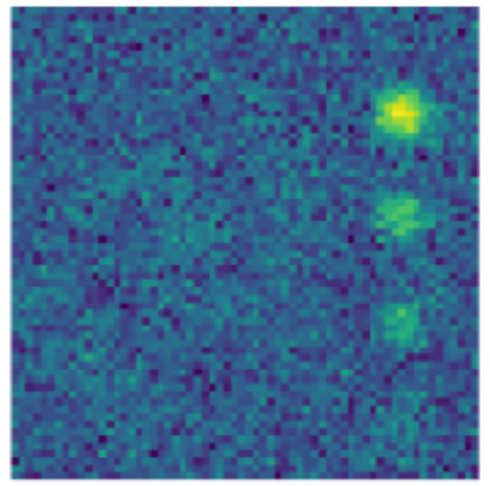
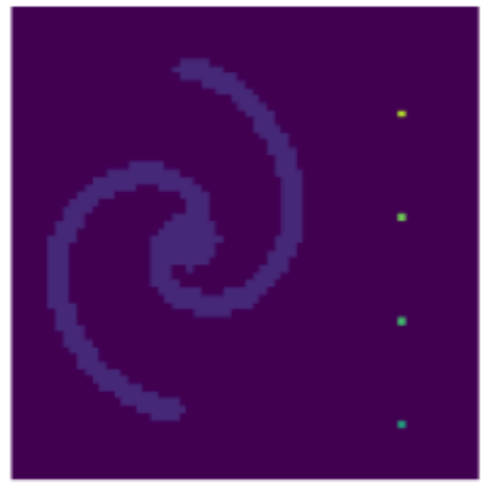
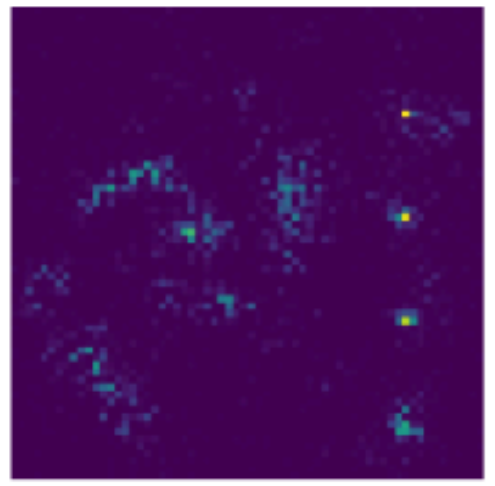
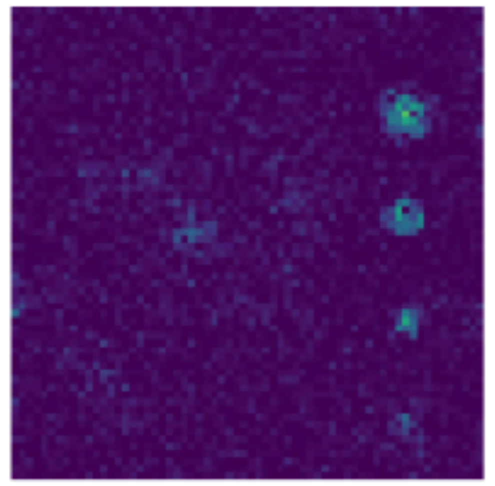
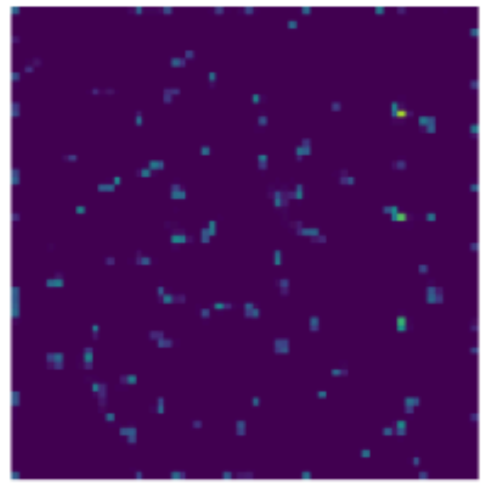
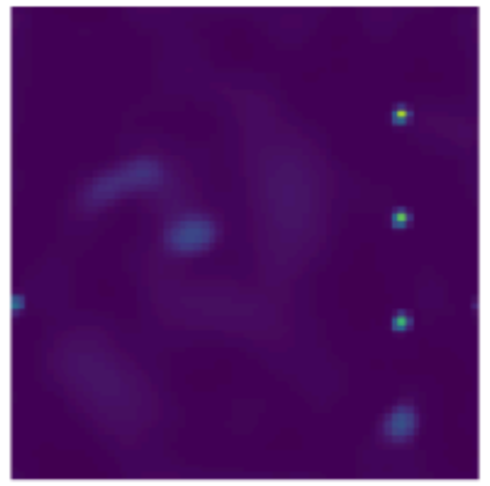
Jolideco Comparison

bg1 $\hat{=}$ 0.01 ct / pixel
bg2 $\hat{=}$ 0.1 ct / pixel
bg3 $\hat{=}$ 1 ct / pixel

Bkg. Level	Counts	Ground Truth	Pylira	Jolideco Uniform Prior N=10	Jolideco Uniform Prior N=1000	Jolideco Patch Prior
bg1						
bg2						
bg3						

Jolideco Comparison

bg1 ≙ 0.01 ct / pixel
bg2 ≙ 0.1 ct / pixel
bg3 ≙ 1 ct / pixel

Bkg. Level	Counts	Ground Truth	Pylira	Jolideco Uniform Prior N=10	Jolideco Uniform Prior N=1000	Jolideco Patch Prior
bg1						
bg2						
bg3						

Open questions / ToDos

- The GMM based patch prior is a very flexible prior that seems to deal with extended sources well, much less “decomposition” into point sources
- Started to “experiment” with combining different observations. E.g. combine good angular resolution but bad statistics with bad angular resolution and good statistics. Does it improve the reconstruction, if so in which case?
- What is a good way to quantify the reconstruction quality? Is there a single metric?
- Release test datasets as a benchmark / challenge?
- There is currently a user defined parameter β, can we get rid of it?
- Could imagine learning from different images and creating a “library” of patch priors adapted to certain analysis scenarios, e.g. Galactic Sources, AGN jets, point sources
- Change from MAP to MC sampling method, to get distributions and error estimates
- How to deal with image dynamics? The prior is needed for weak extended sources in an image, but not for a bright point source. Can this be improved?
- Treat point sources as independent model component?