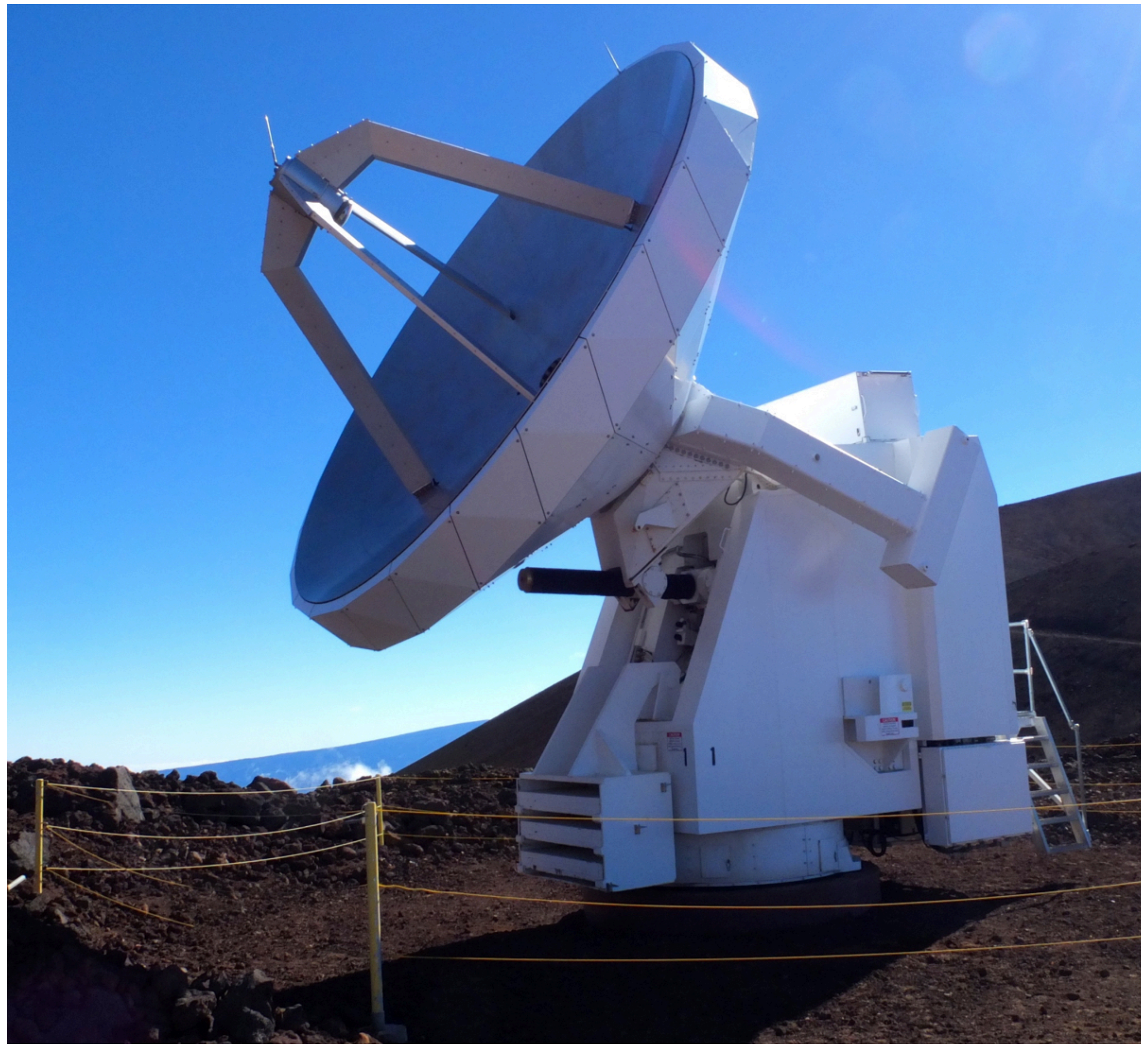
Regularized Maximum Likelihood Techniques for Interferometric Disk Imaging

Brianna Zawadzki
Penn State University

Radio Astronomy

Small single antenna
(lowest resolution)



Large single antenna
(better resolution)

Large array of many small antennas
(best resolution)*

*but now you have to deal with the special needs of interferometers

Needs of Interferometry

Image synthesis: process the Fourier visibilities from the interferometer to obtain sky brightness

➡ Must choose type of image synthesis

➡ Must make assumptions about unsampled spatial frequencies

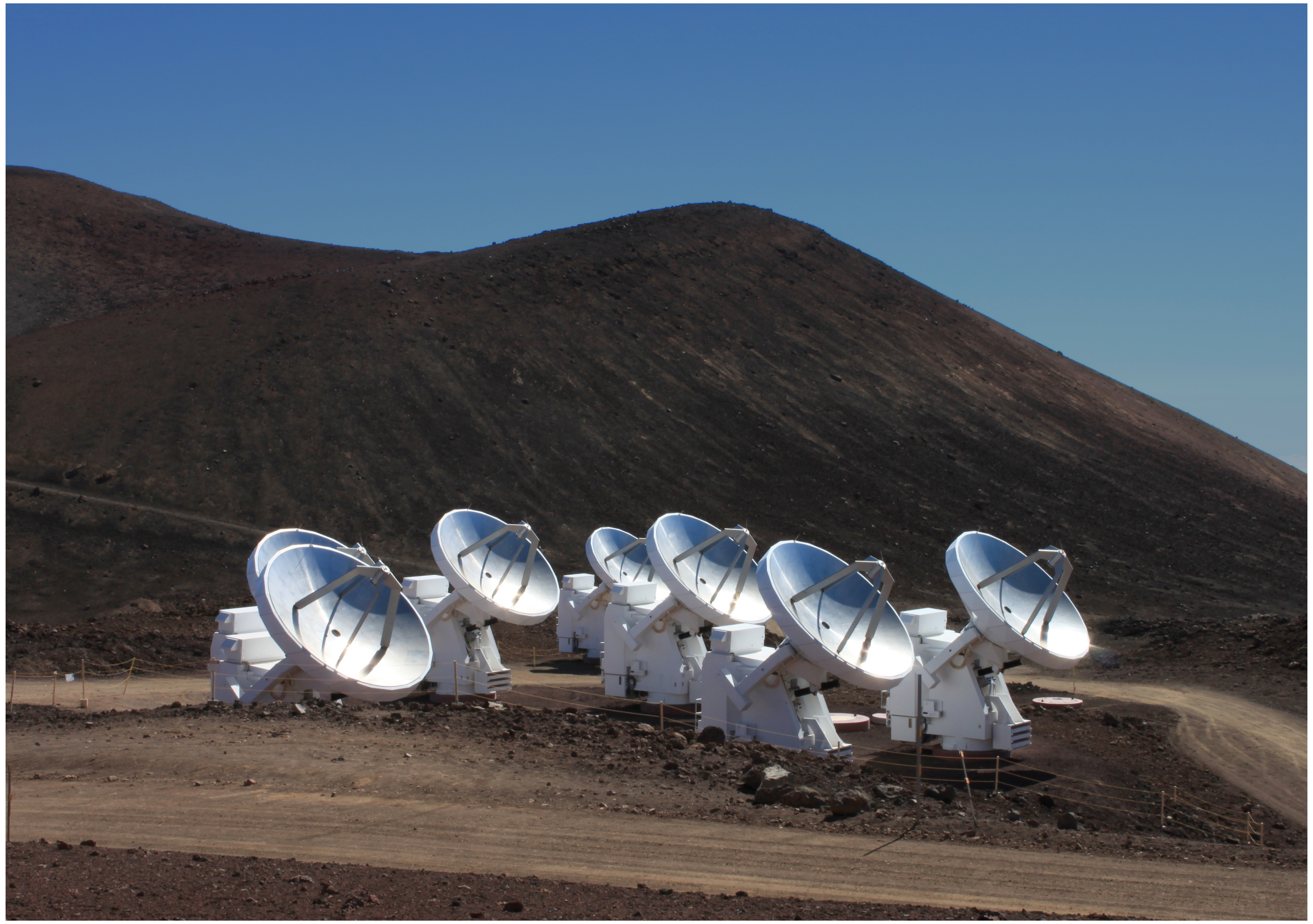
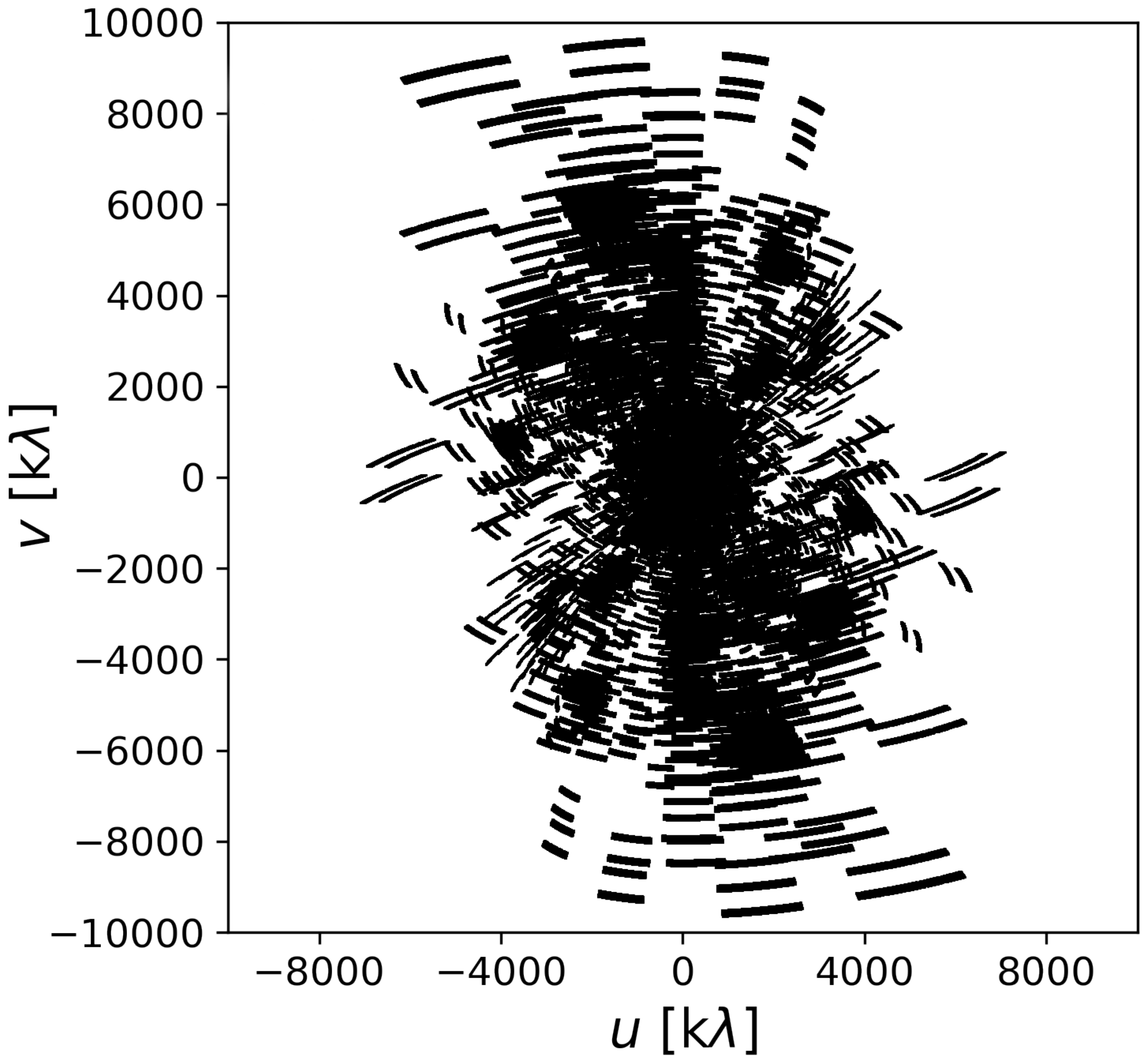
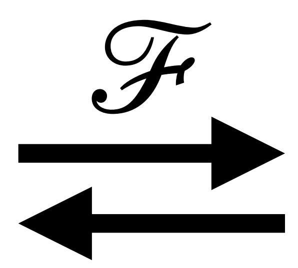
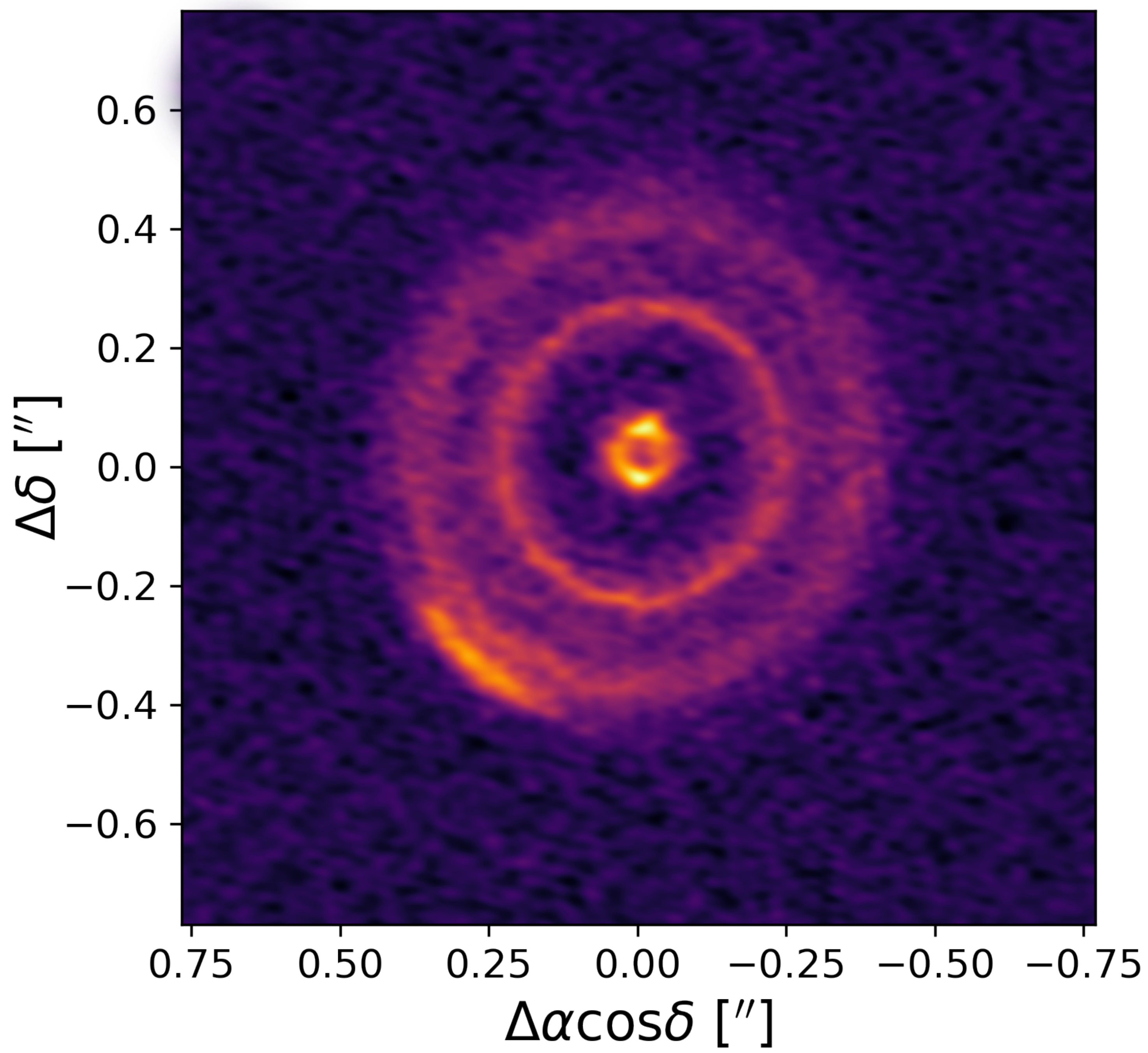


Photo by Glen Petitpas

Interferometric Image Processing



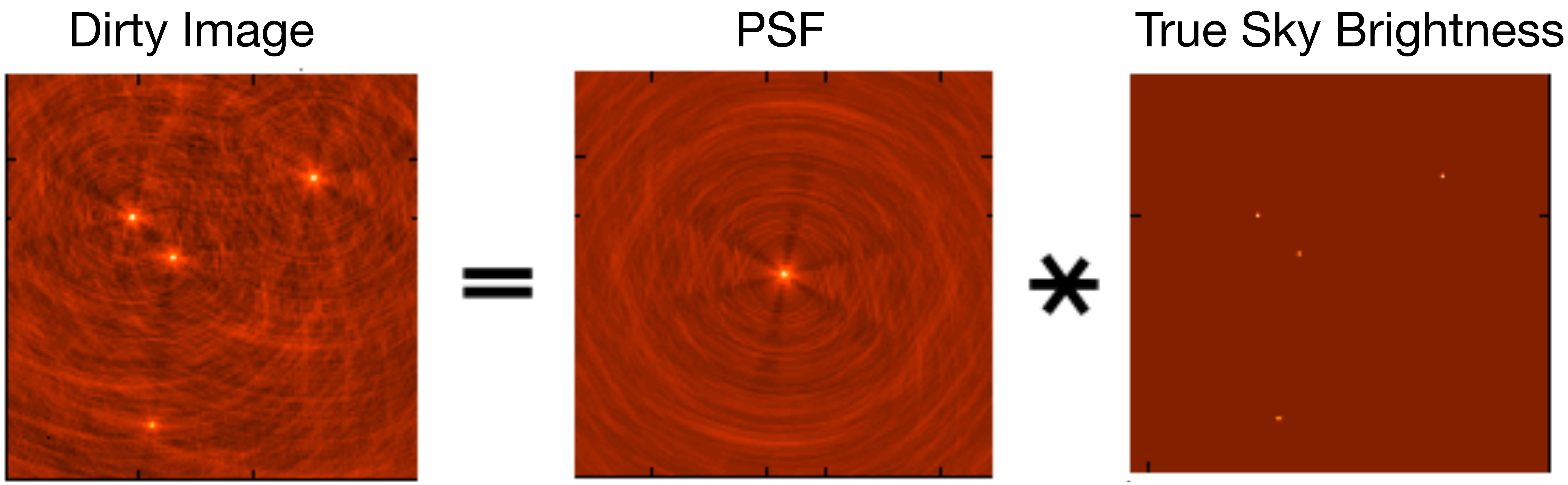




Incomplete sampling of visibilities (left) and corresponding dirty image with Briggs weighting (right), adapted from Zawadzki et al. (2023, submitted)

What assumptions can we make about the unsampled spatial frequencies?

CLEAN



https://casa.nrao.edu/casadocs-devel/stable/imaging/synthesis-imaging/deconvolution-algorithms

- Iteratively finds peaks in the dirty image and subtracts the PSF
- Procedural method for obtaining a cleaned image
- Cons: - Can be slow
  - Gaussian components may not lend themselves well to certain features (e.g. rings)

RML: Regularized Maximum Likelihood

- A forward-modeling approach to imaging
- We want to solve for the most likely image given the visibilities
  - Consider each pixel as a model parameter
  - Apply regularizers/priors to the model
- A true optimization algorithm: we write down some objective function and solve for it

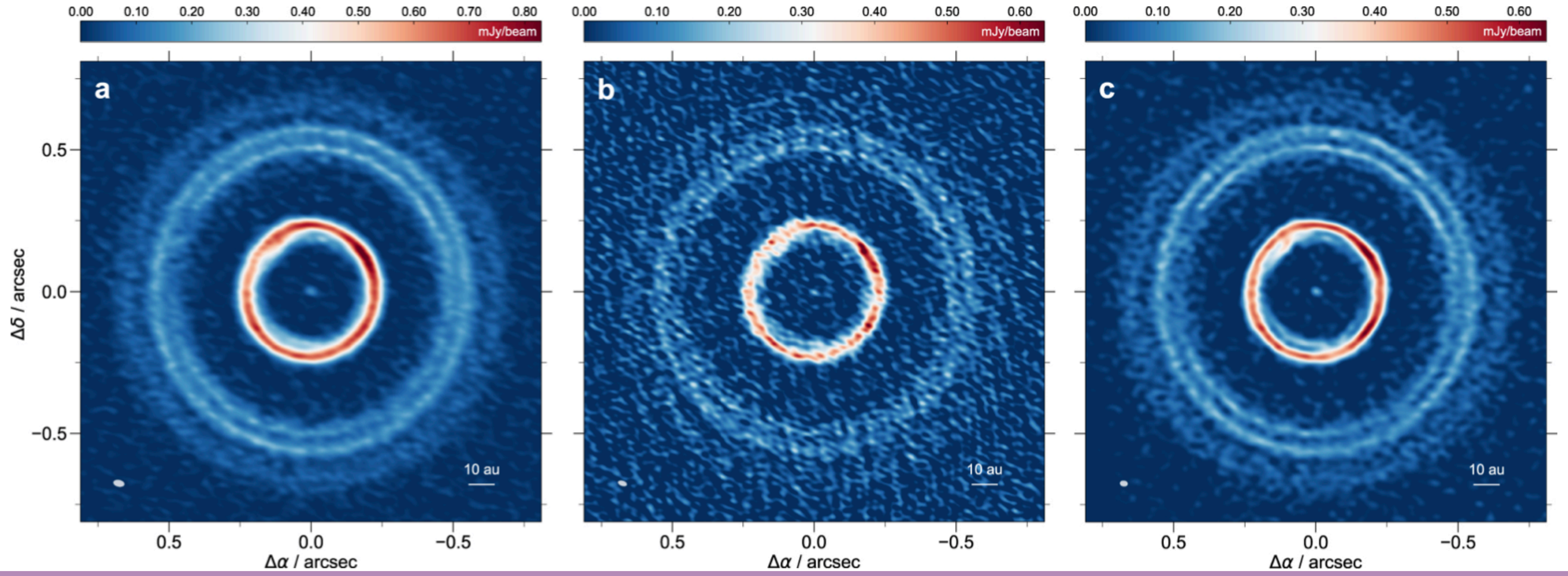


Image synthesis of the HD 169142 in dust continuum (figure from Perez et al. 2019)

Panels a) and b) show the CASA tclean image with Briggs and uniform weighting, respectively.

Panel c) shows RML imaging, which has a sensitivity comparable to panel a) and a spatial resolution comparable to panel b).

Optimizing with RML: Bayesian Perspective

$$p(\mathbf{I} \mid \mathbf{D}) \propto p(\mathbf{D} \mid \mathbf{I})p(\mathbf{I})$$

Posterior: what we're optimizing

Likelihood function: make assumptions about the data generating process (usually χ^2)

Priors: all additional constraints on the model

Optimizing with RML: Computing Perspective

$$L(\mathbf{I}) = L_{\mathrm{nll}}(\mathbf{I}) + \lambda_{\mathrm{A}} L_{\mathrm{A}}(\mathbf{I}) + \lambda_{\mathrm{B}} L_{\mathrm{B}}(\mathbf{I})$$

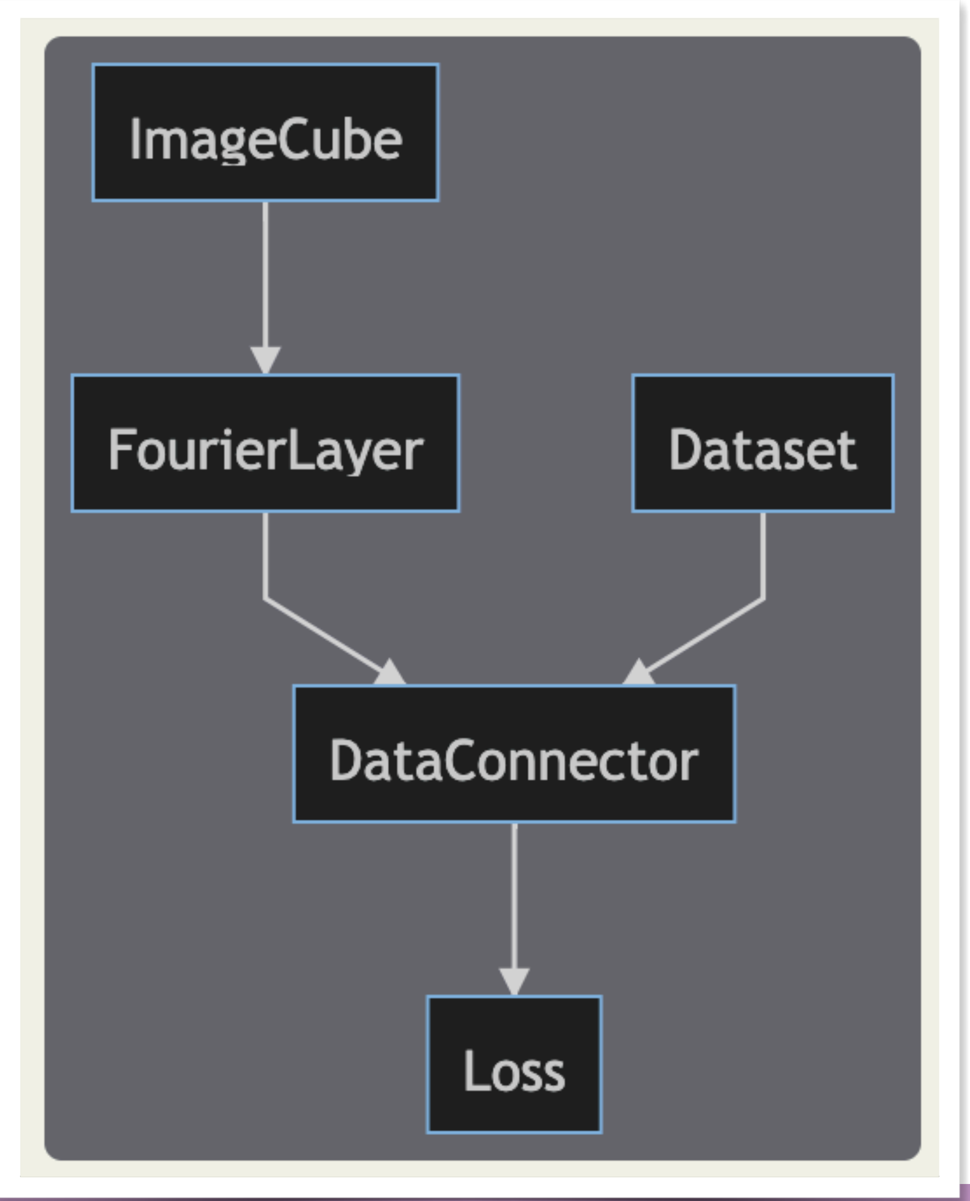
The loss function

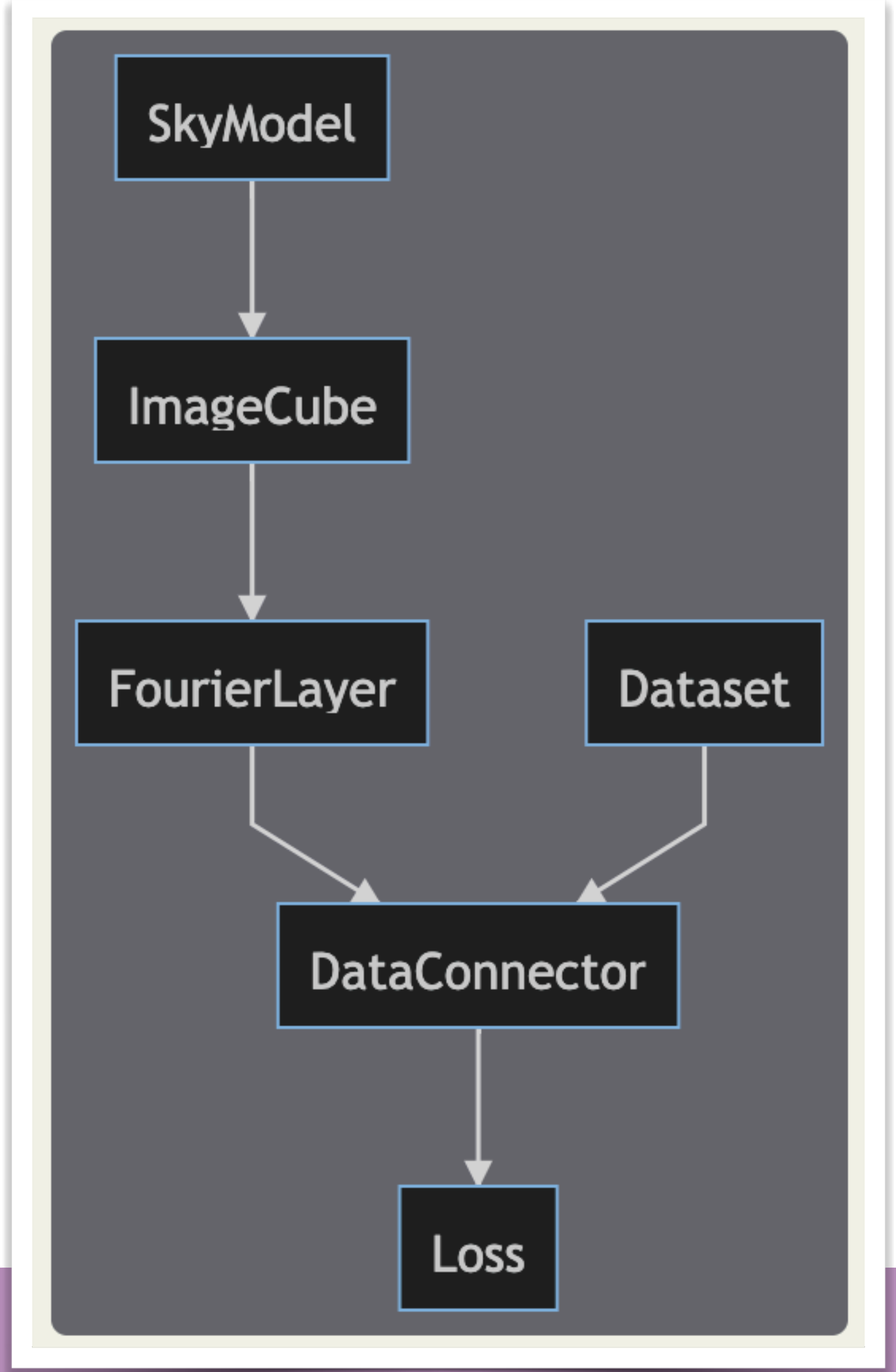
The negative log likelihood (usually χ^2)

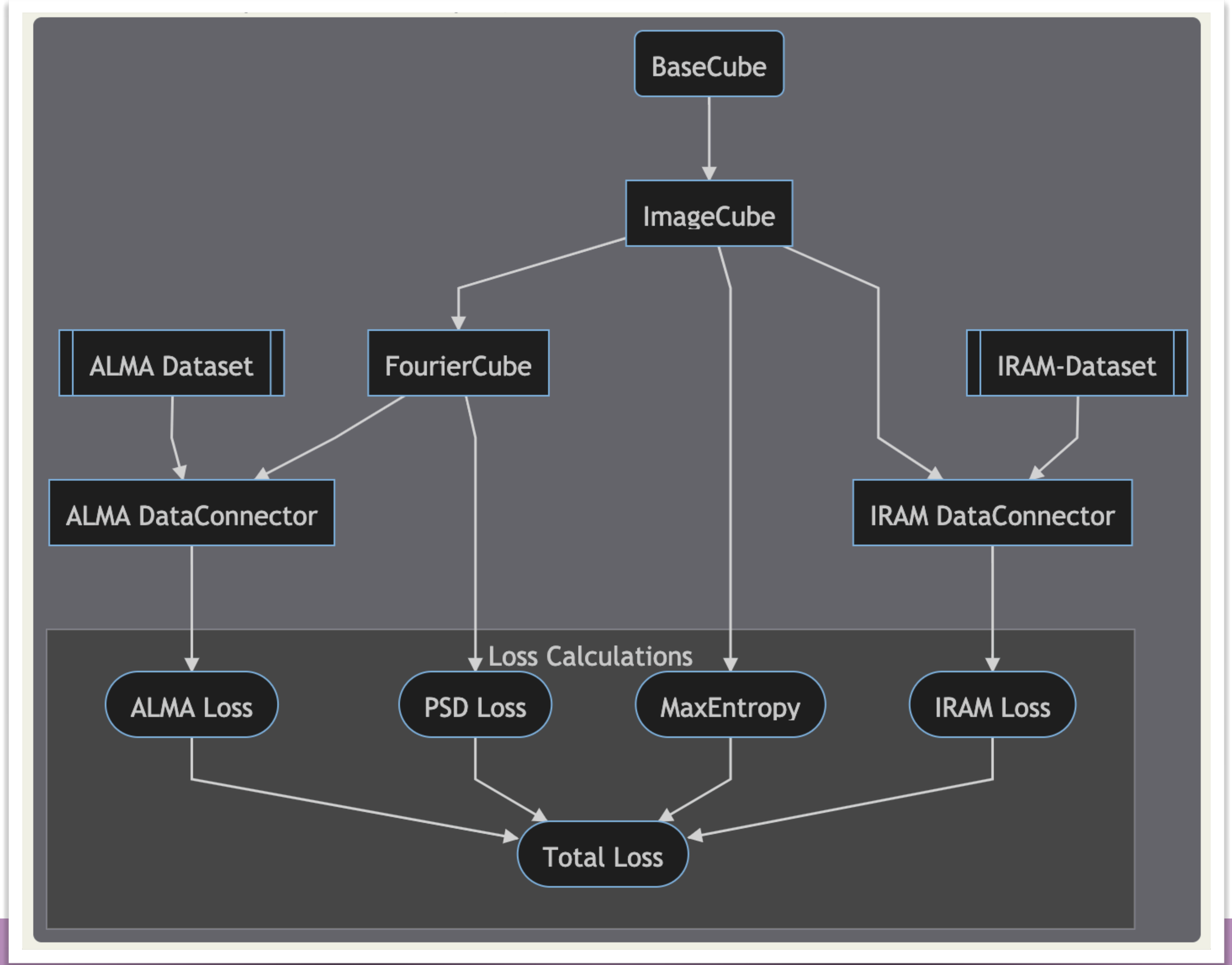
Regularizing terms add linearly and can have many functional forms

Million Points of Light

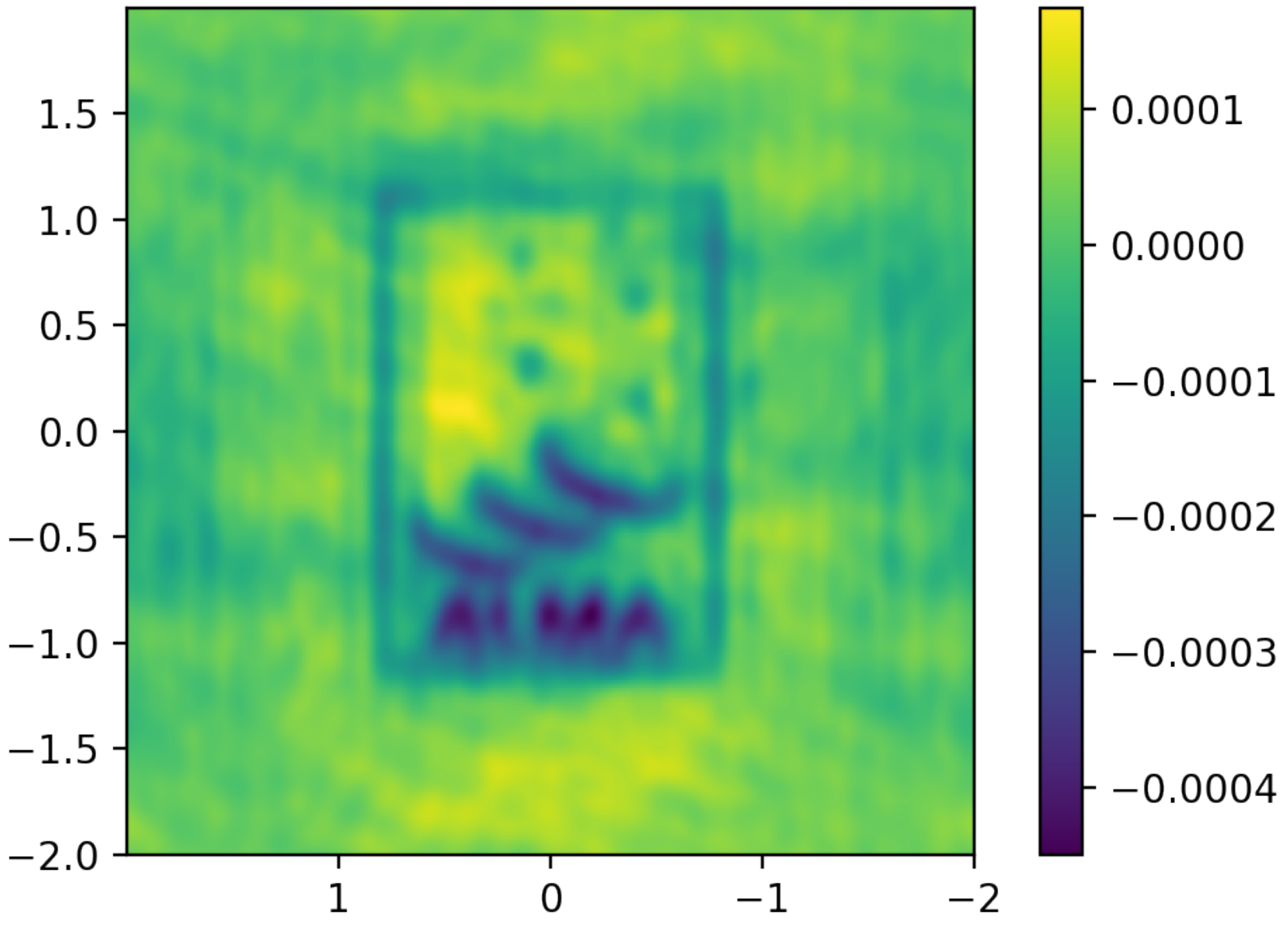
- Developing MPoL (https://mpol-dev.github.io/MPoL/)
  - Python package for RML based on PyTorch
  - Authors: Ian Czekala, Brianna Zawadzki, Ryan Loomis
  - RML frameworks are flexible with many possibilities



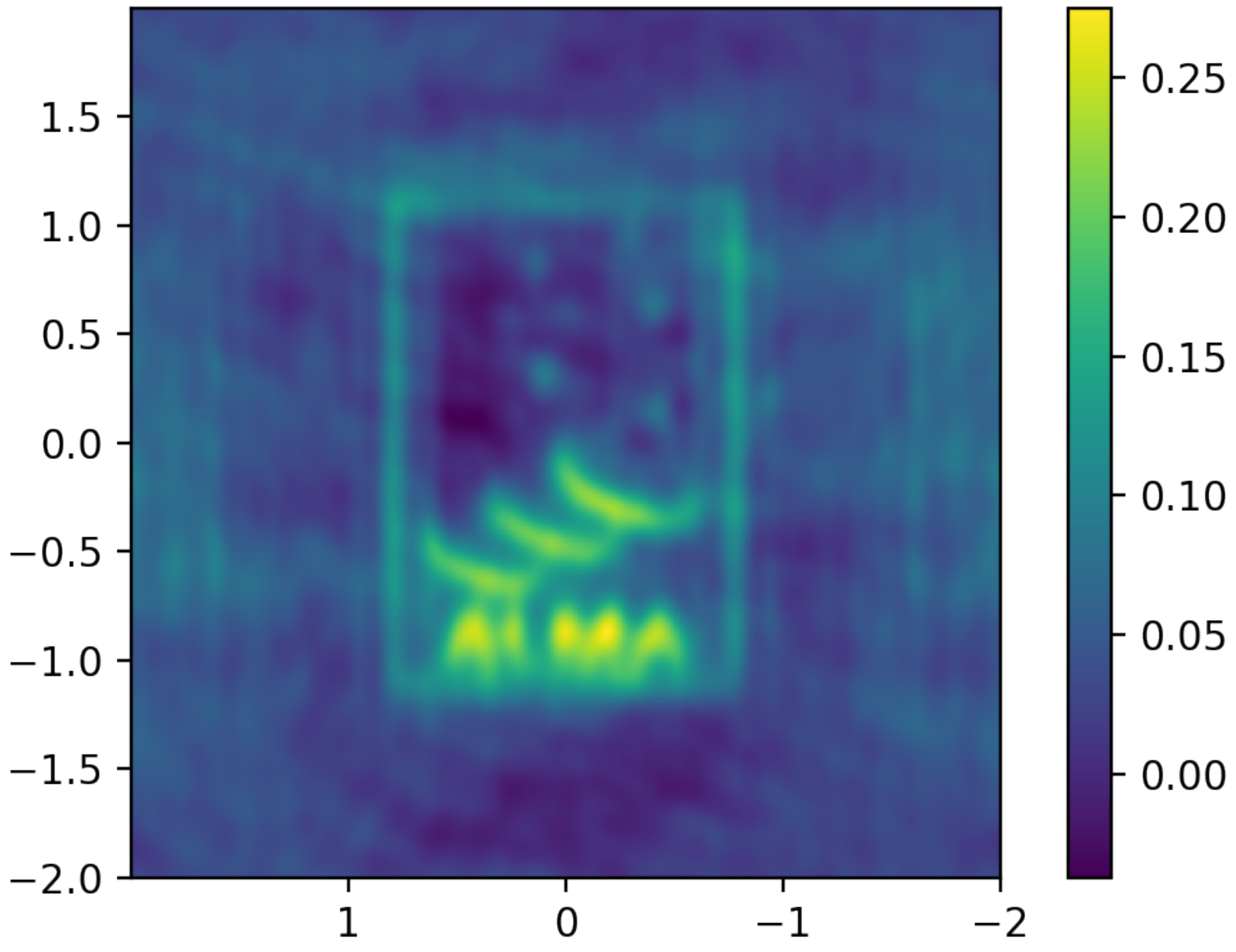




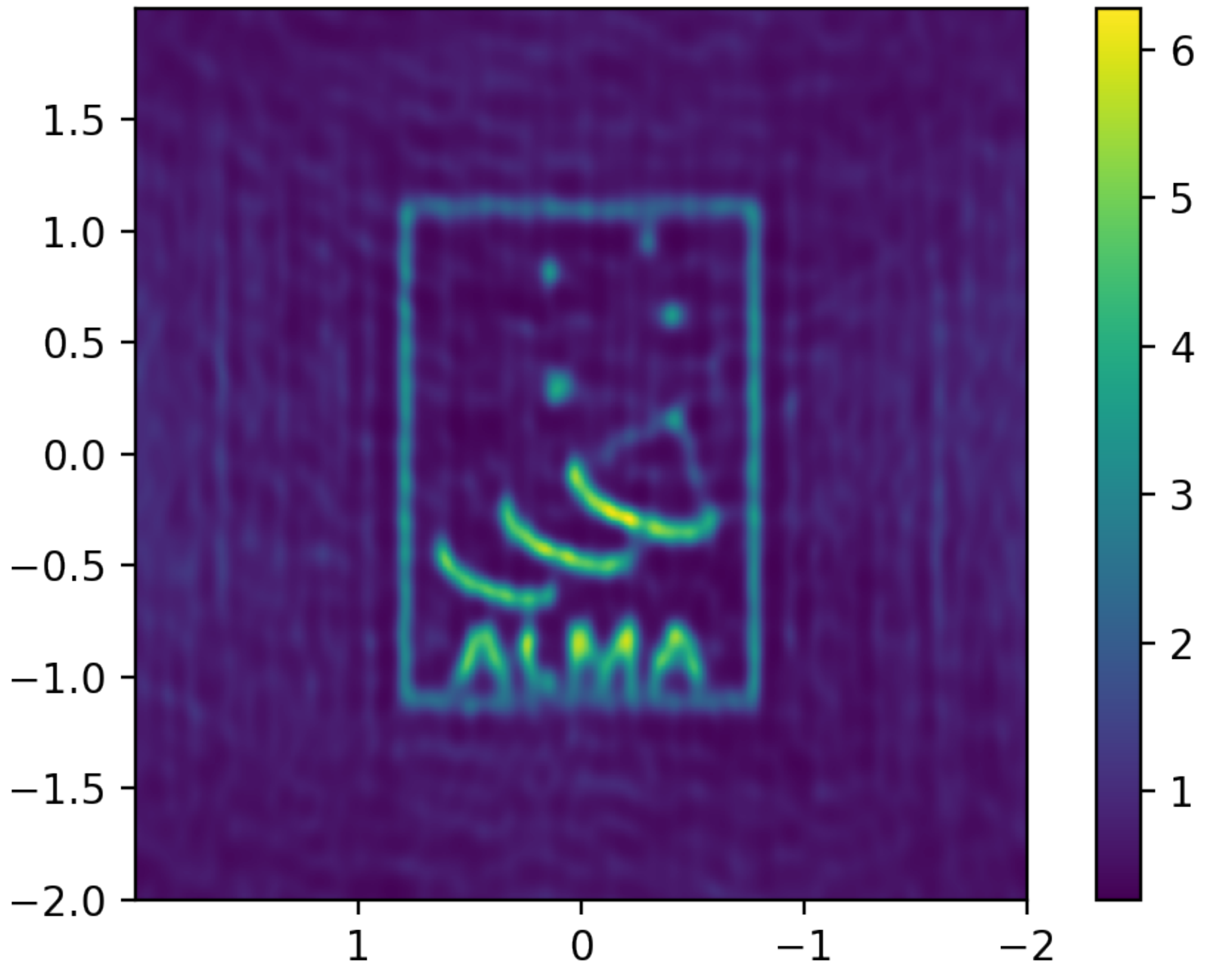
The RML Optimization Loop



Calculate gradients to be added to the base image



Base image after one iteration



Base image after 300 iterations

Development

- So far, we have implemented the following regularizers:
    - Positivity
    - Entropy
    - Sparsity
    - Total variation (TV)
    - Total squared variation (TSV)

Enforcing Positivity

- The flux of the observed source must be positive (or zero, if there is no flux)
- It follows that the intensity value of a given pixel must be positive

$$f_{\mathrm{ReLU}}(x) = \max(0, x)$$

$$f_{\mathrm{Softplus}}(x) = \frac{1}{\beta} * \log(1 + \exp(\beta * x))$$

If the pixel is positive, the value is unchanged
If the pixel is negative, the value becomes 0

Negative input values have a positive nonzero output
Little impact on large positive input values
Retain some information about the relative brightness of each pixel

Entropy

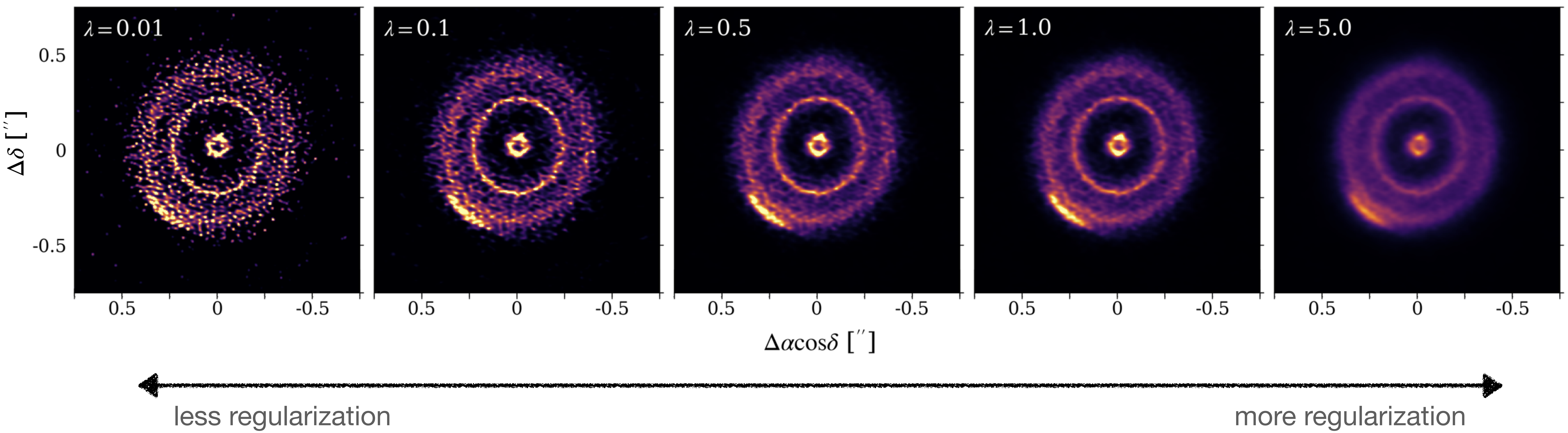
- Promotes images with similar pixel values to a given set of reference pixels
- Reference pixels p could be uniform or incorporate prior knowledge (e.g. assuming the source intensity is Gaussian)

$$L = \frac{1}{\sum_i I_i} \sum_i I_i \ln \frac{I_i}{p_i}$$

Functional form of the entropy regularizer

Entropy

- Promotes images with similar pixel values to a given set of reference pixels
- Reference pixels p could be uniform or incorporate prior knowledge (e.g. assuming the source intensity is Gaussian)



Sparsity

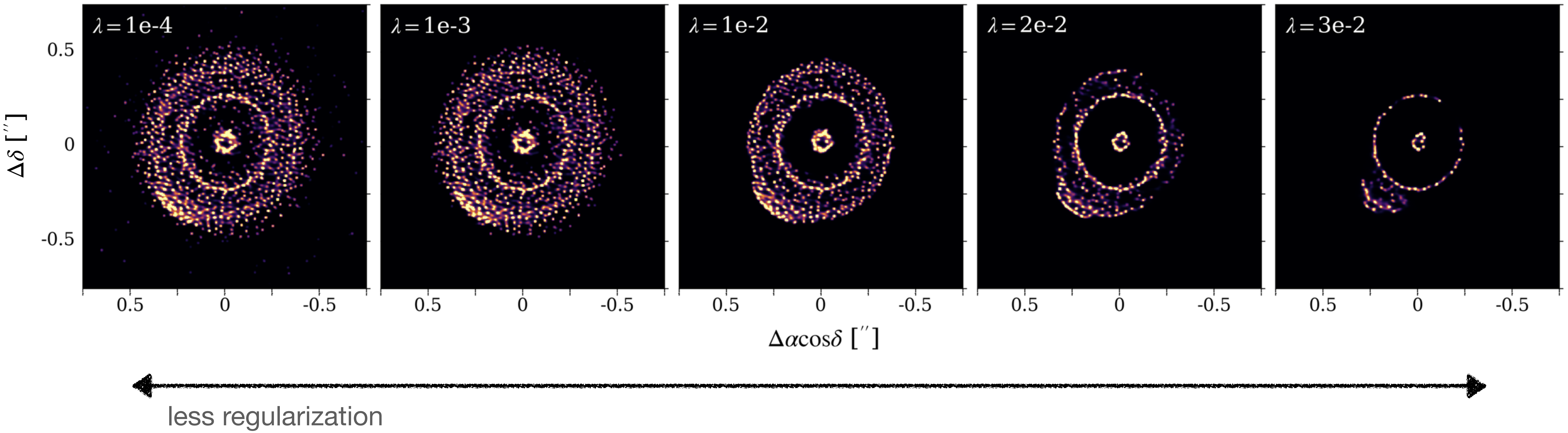
- Uses the L_1 norm to reduce the impact of unneeded pixels
- Promotes a final image that is a sparse collection of nonzero pixels

$$L = \sum_i \left| I_i \right|$$

Functional form of the sparsity

Sparsity

- Uses the L_1 norm to reduce the impact of unneeded pixels
- Promotes a final image that is a sparse collection of nonzero pixels



Total Variation (TV)

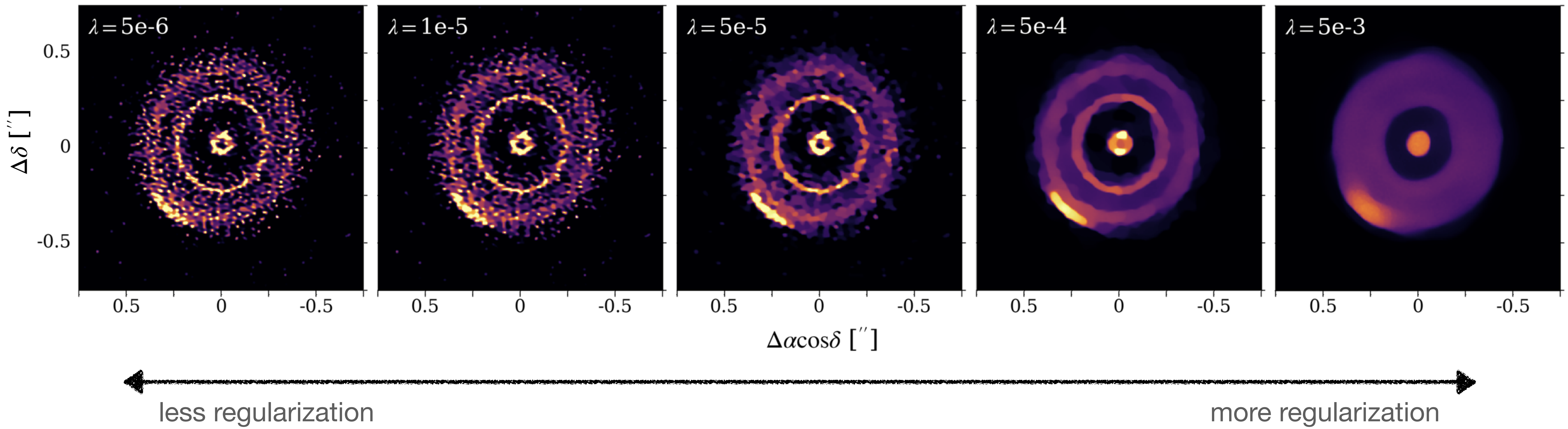
- Promotes images with sharp edges where significant changes in intensity are needed
- Otherwise promotes similarity/smoothness between adjacent pixels

$$L = \sum_{l,m,v} \sqrt{\left(I_{l+1,m,v} - I_{l,m,v}\right)^2 + \left(I_{l,m+1,v} - I_{l,m,v}\right)^2 + \epsilon}$$

Functional form of the TV regularizer

Total Variation (TV)

- Promotes images with sharp edges where significant changes in intensity are needed
- Otherwise promotes similarity/smoothness between adjacent pixels



Total Squared Variation (TSV)

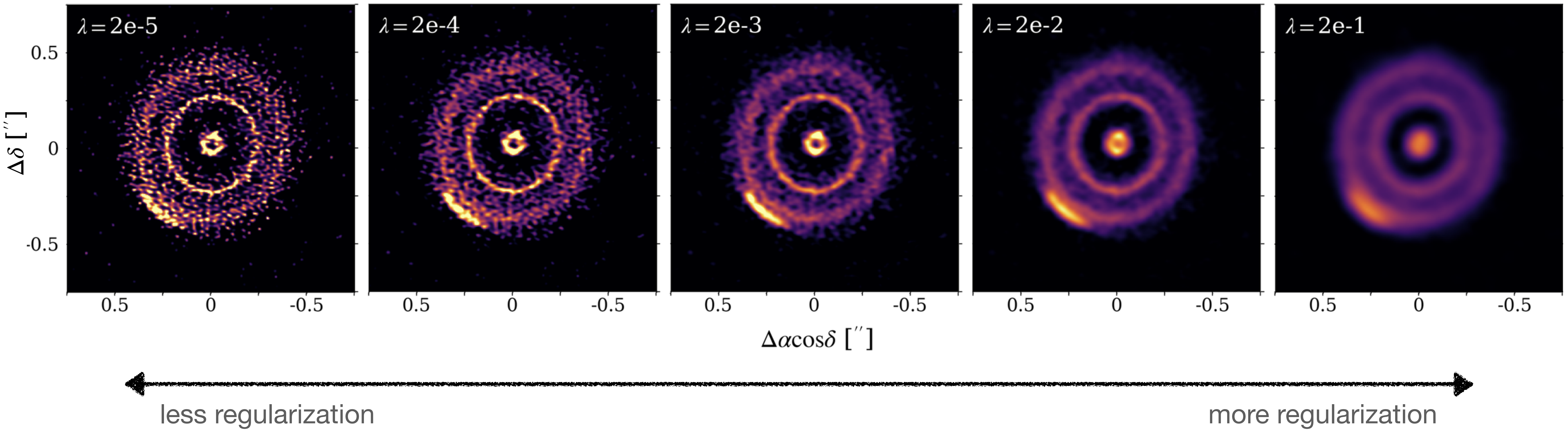
- A variant of the TV regularizer
- Edges are smoother with TSV than with TV

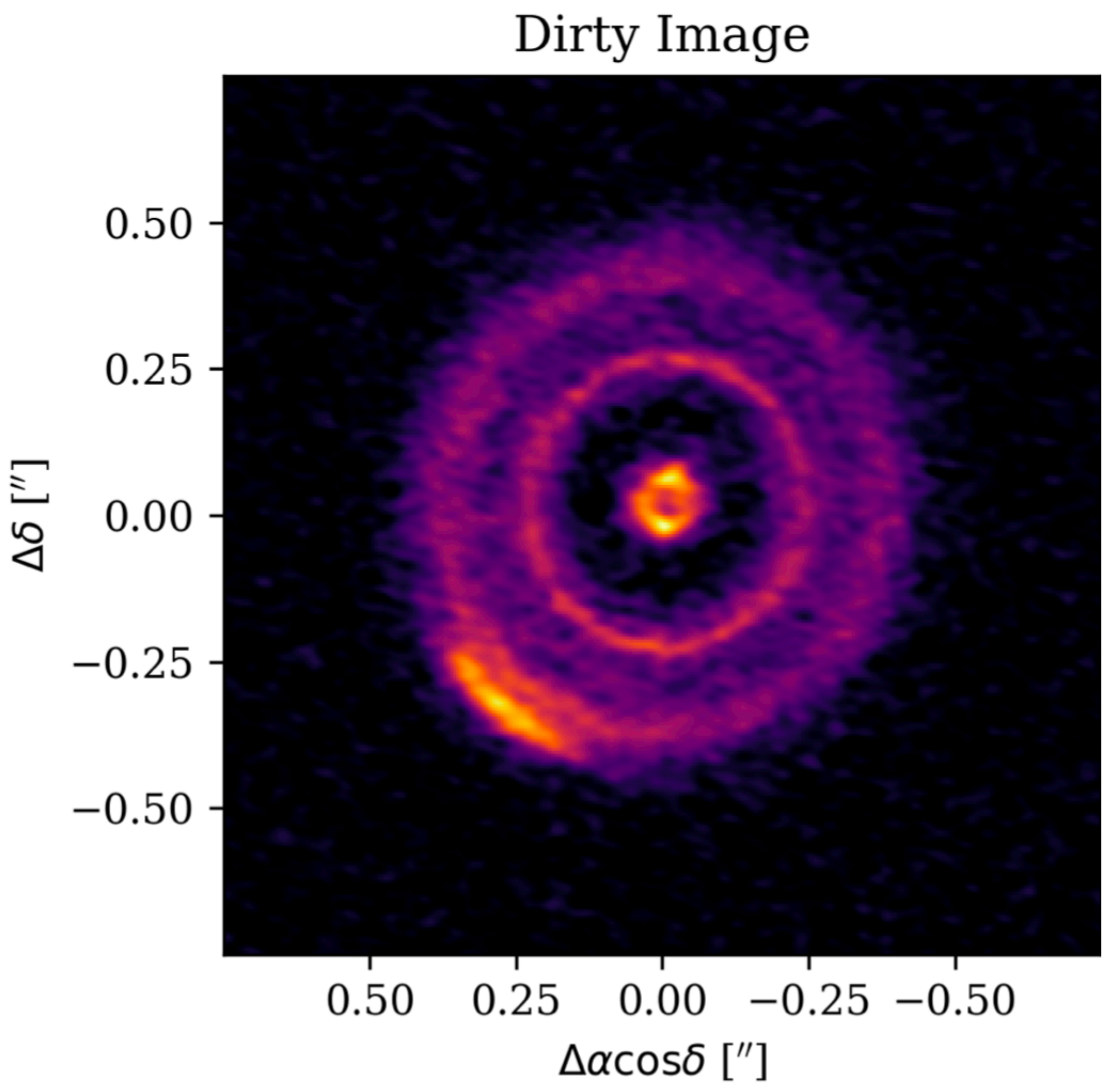
$$L = \sum_{l,m,v} \left(I_{l+1,m,v} - I_{l,m,v}\right)^2 + \left(I_{l,m+1,v} - I_{l,m,v}\right)^2$$

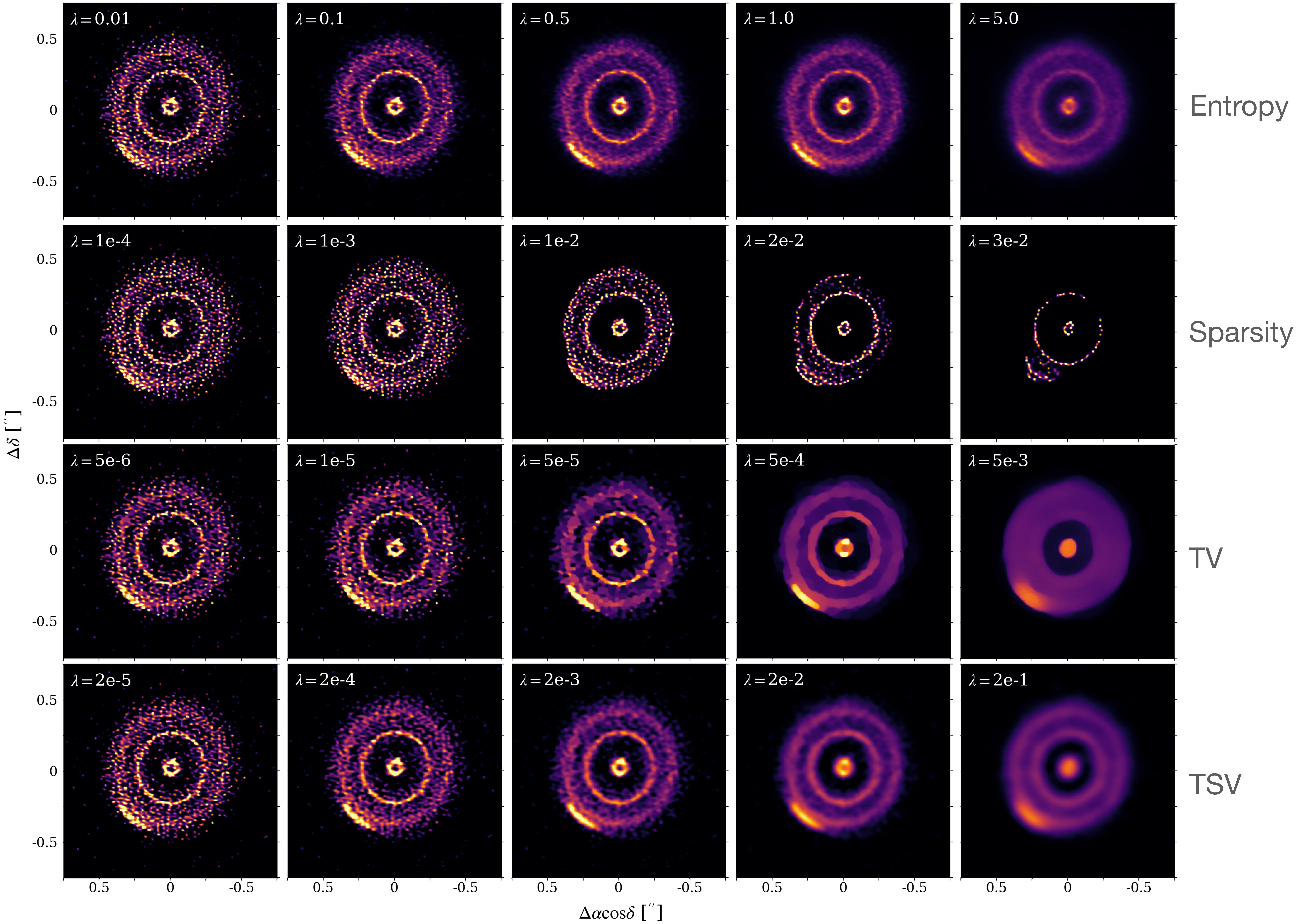
Functional form of the TSV regularizer

Total Squared Variation (TSV)

- A variant of the TV regularizer
- Edges are smoother with TSV than with TV

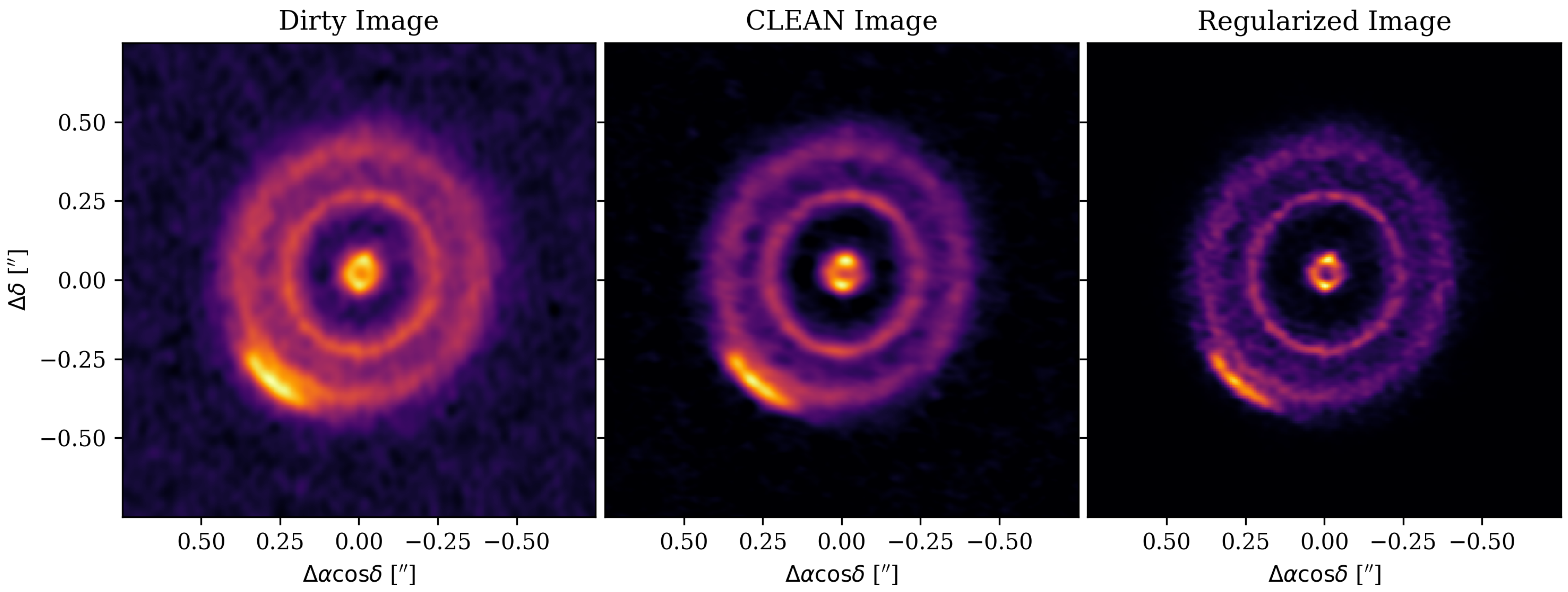






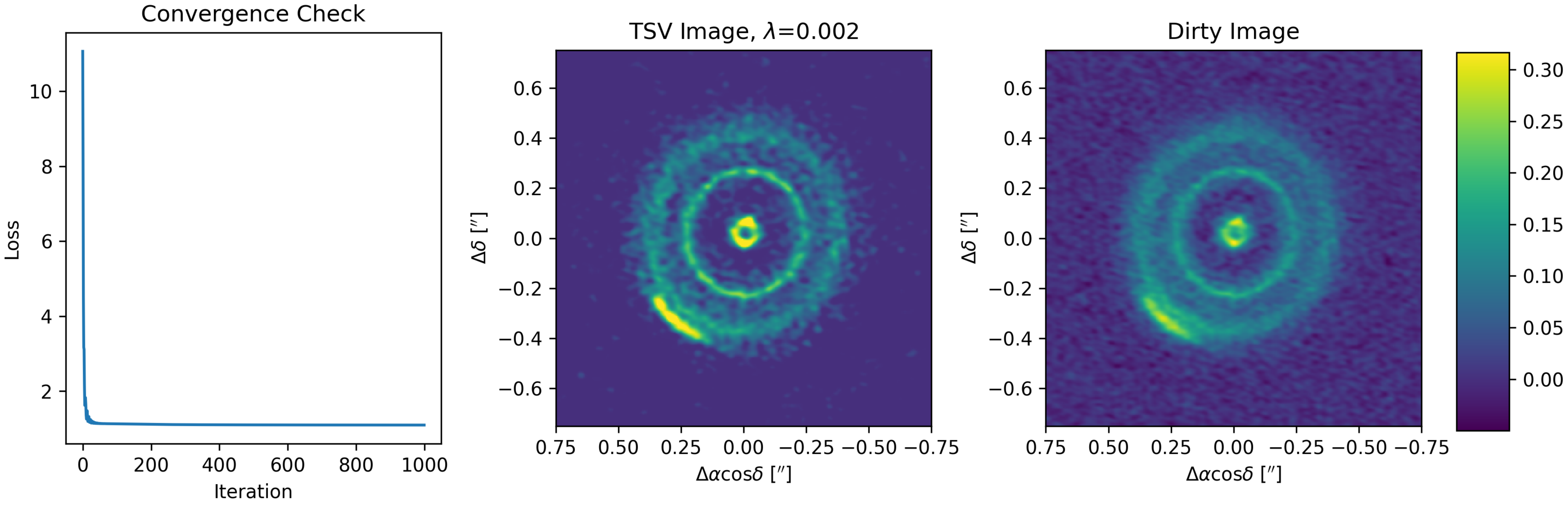
Adapted from Zawadzki et al. 2023a (submitted)

DSHARP data: HD 143006



Zawadzki et al. (2023, submitted)

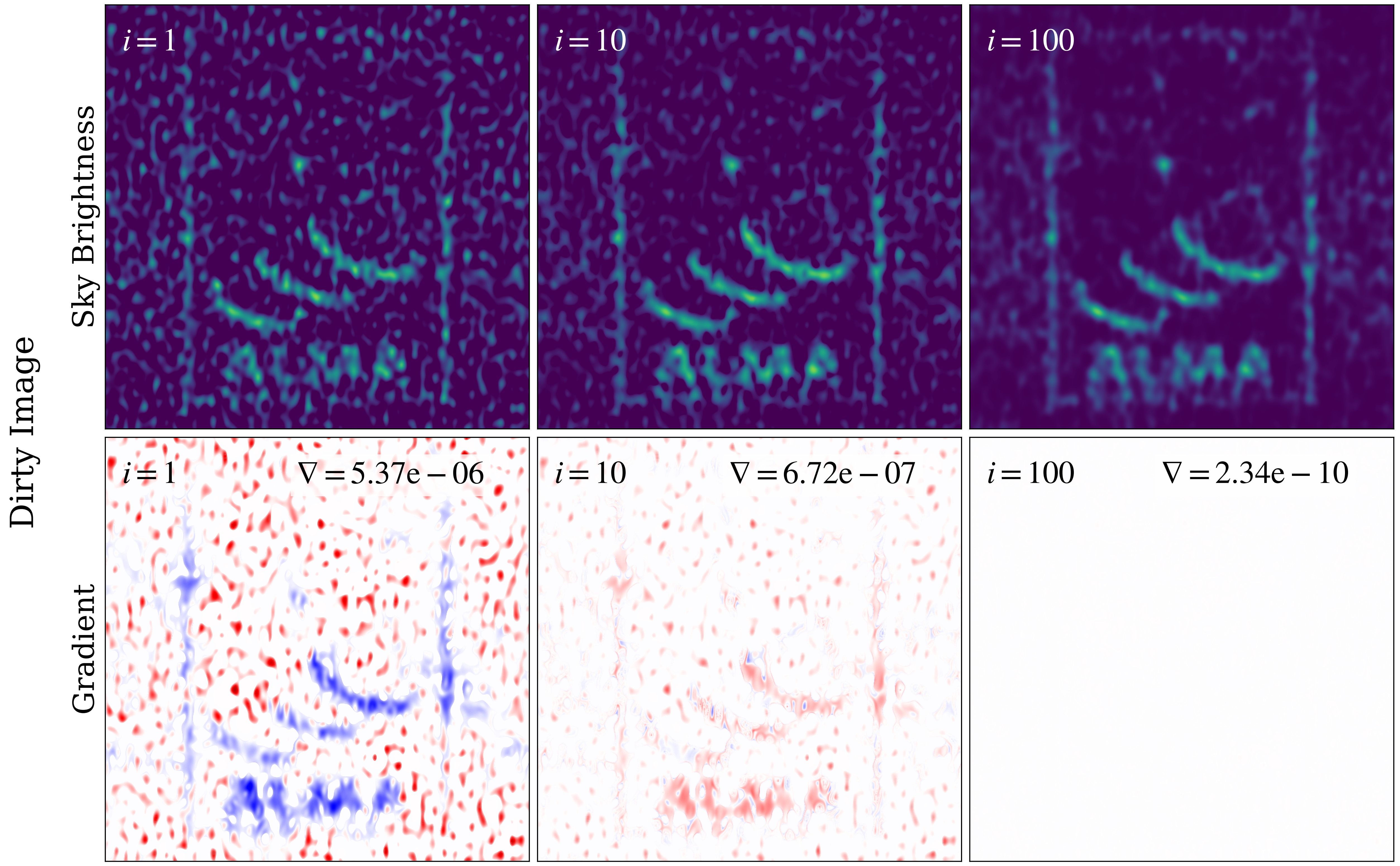
Checking Convergence

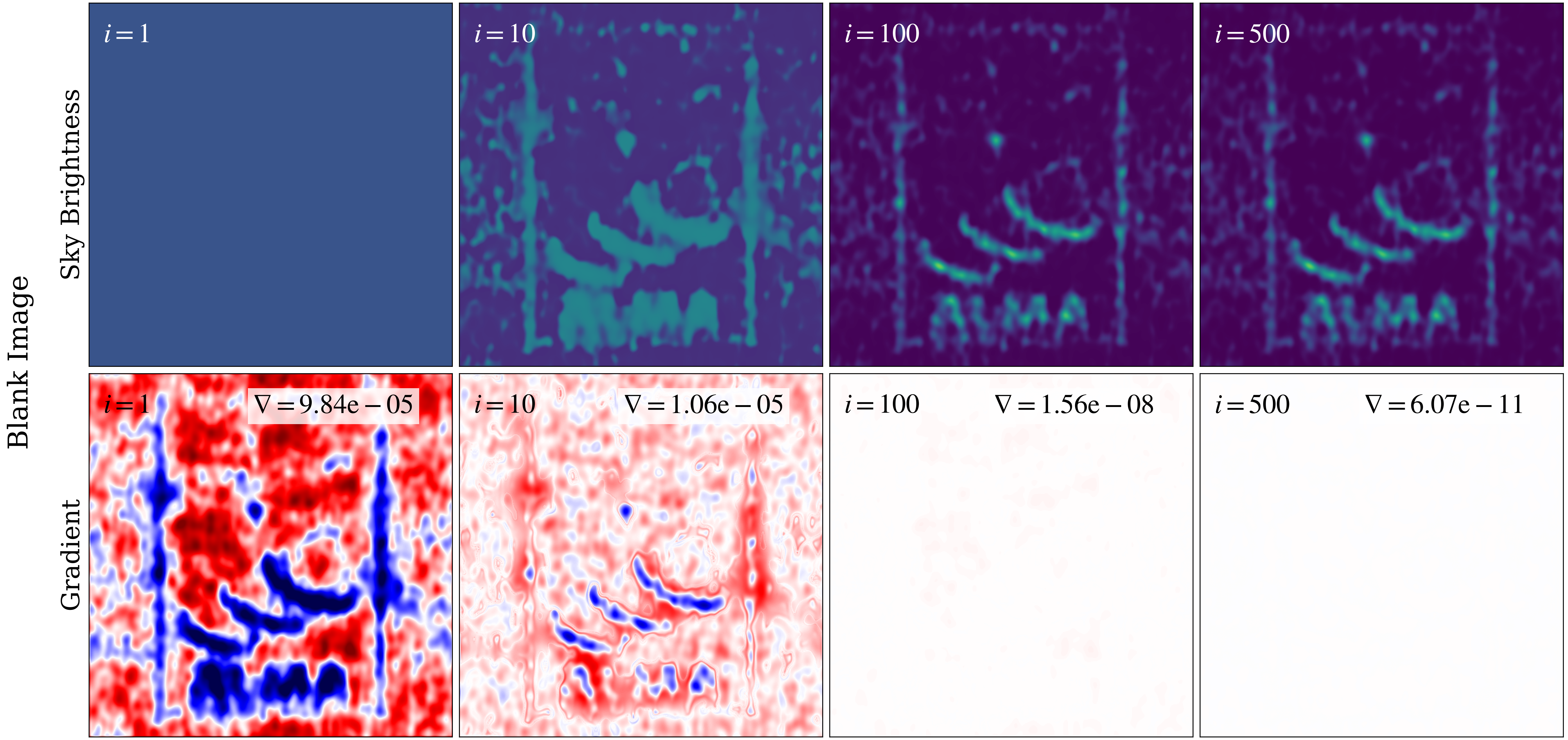


Imaging HD143006 with only TSV regularization. We check that the loss function has converged on a minimum to ensure the optimization process is finished.

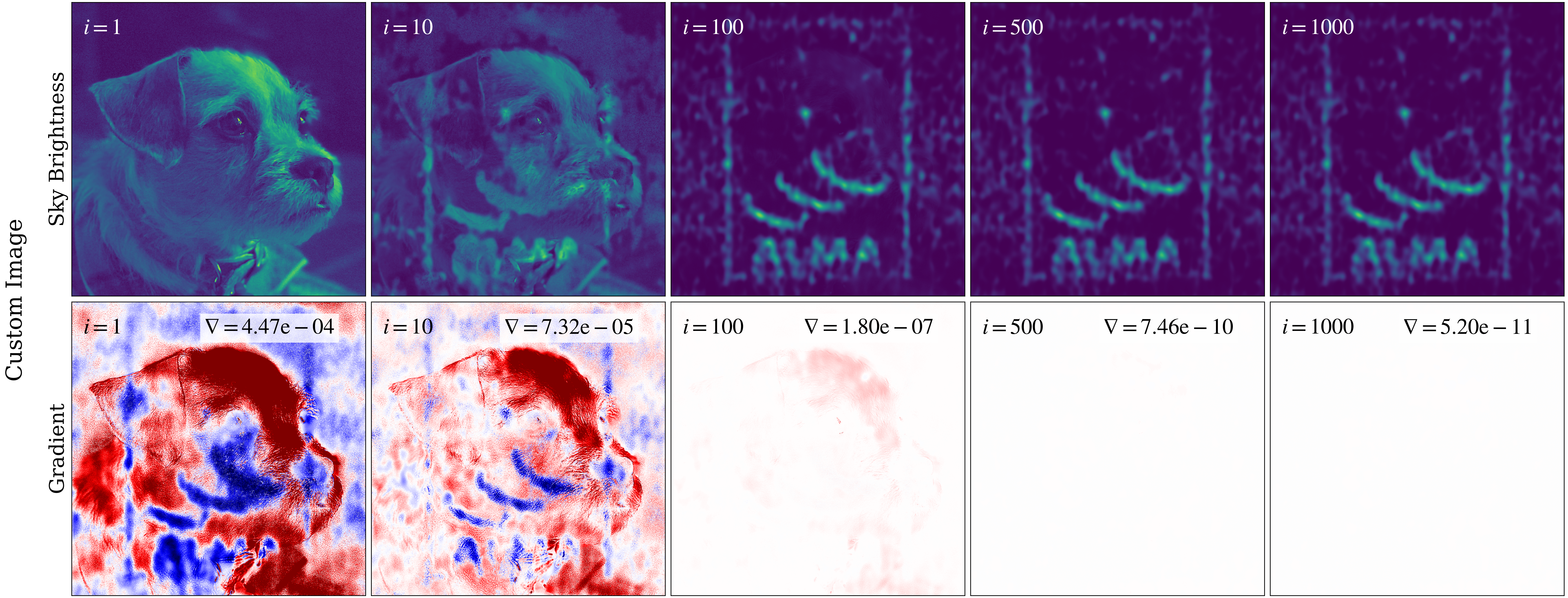
Optimization Speed

- Depends on number of pixels
- Depends on your starting image
- Typical times:
    - ~minutes on a CPU
    - ~seconds on a GPU

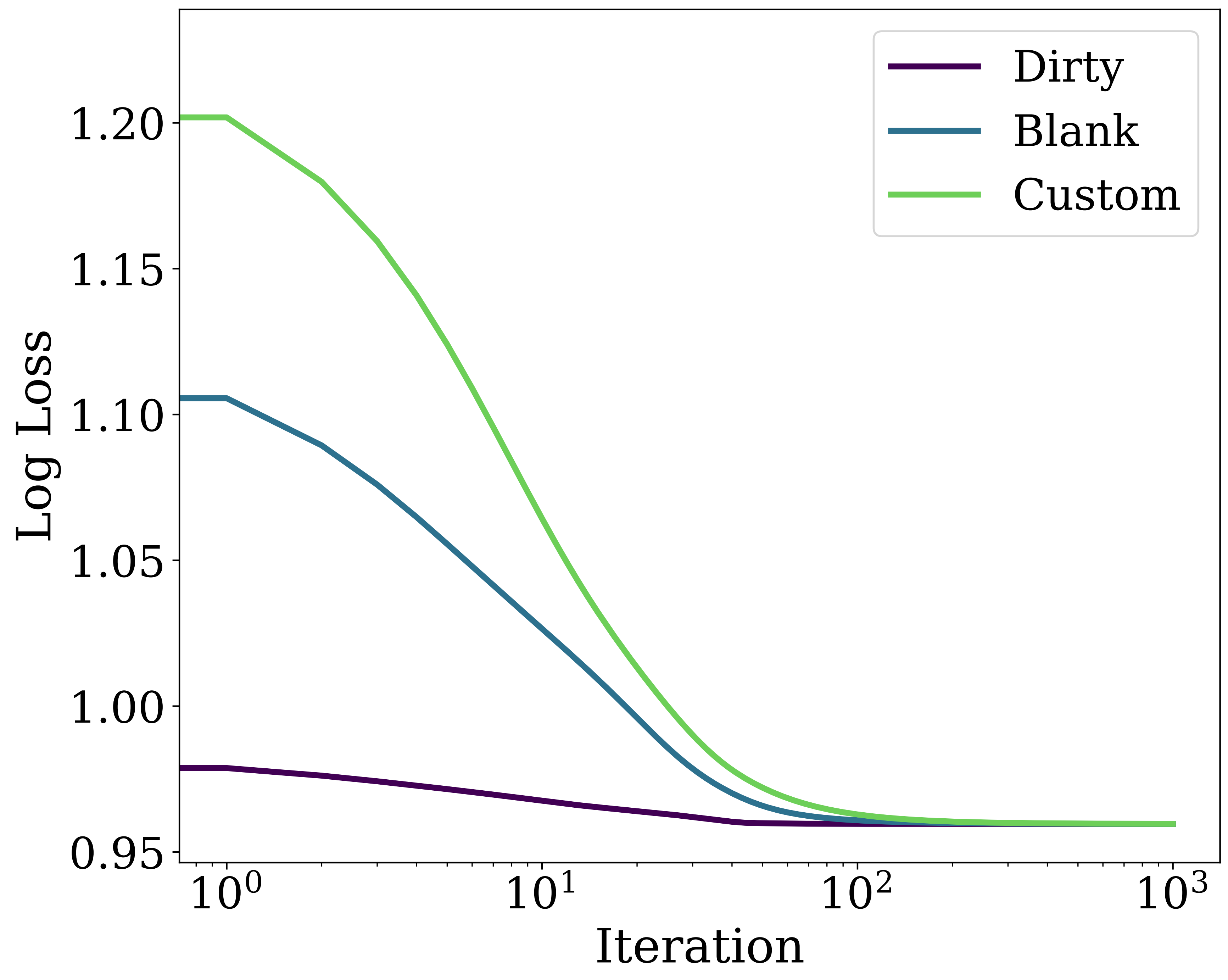




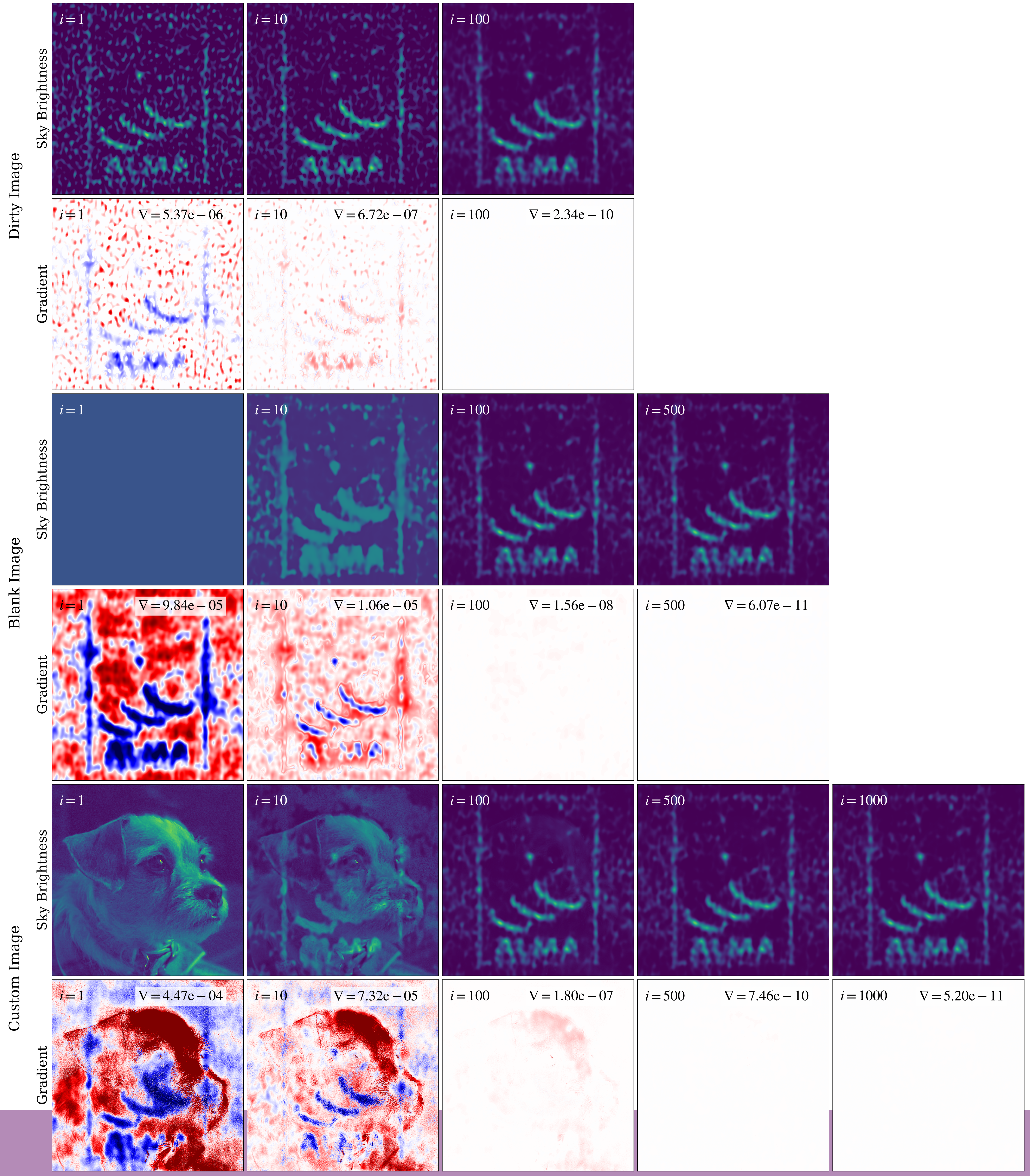
Blank Image
Sky Brightness
Gradient
i = 1
i = 10
i = 100
i = 500
i = 1 ∇ = 9.84e − 05
i = 10 ∇ = 1.06e − 05
i = 100 ∇ = 1.56e − 08
i = 500 ∇ = 6.07e − 11



Custom Image
Sky Brightness
Gradient
$i = 1$
$i = 10$
$i = 100$
$i = 500$
$i = 1000$
$i = 1$ $\nabla = 4.47e-04$
$i = 10$ $\nabla = 7.32e-05$
$i = 100$ $\nabla = 1.80e-07$
$i = 500$ $\nabla = 7.46e-10$
$i = 1000$ $\nabla = 5.20e-11$



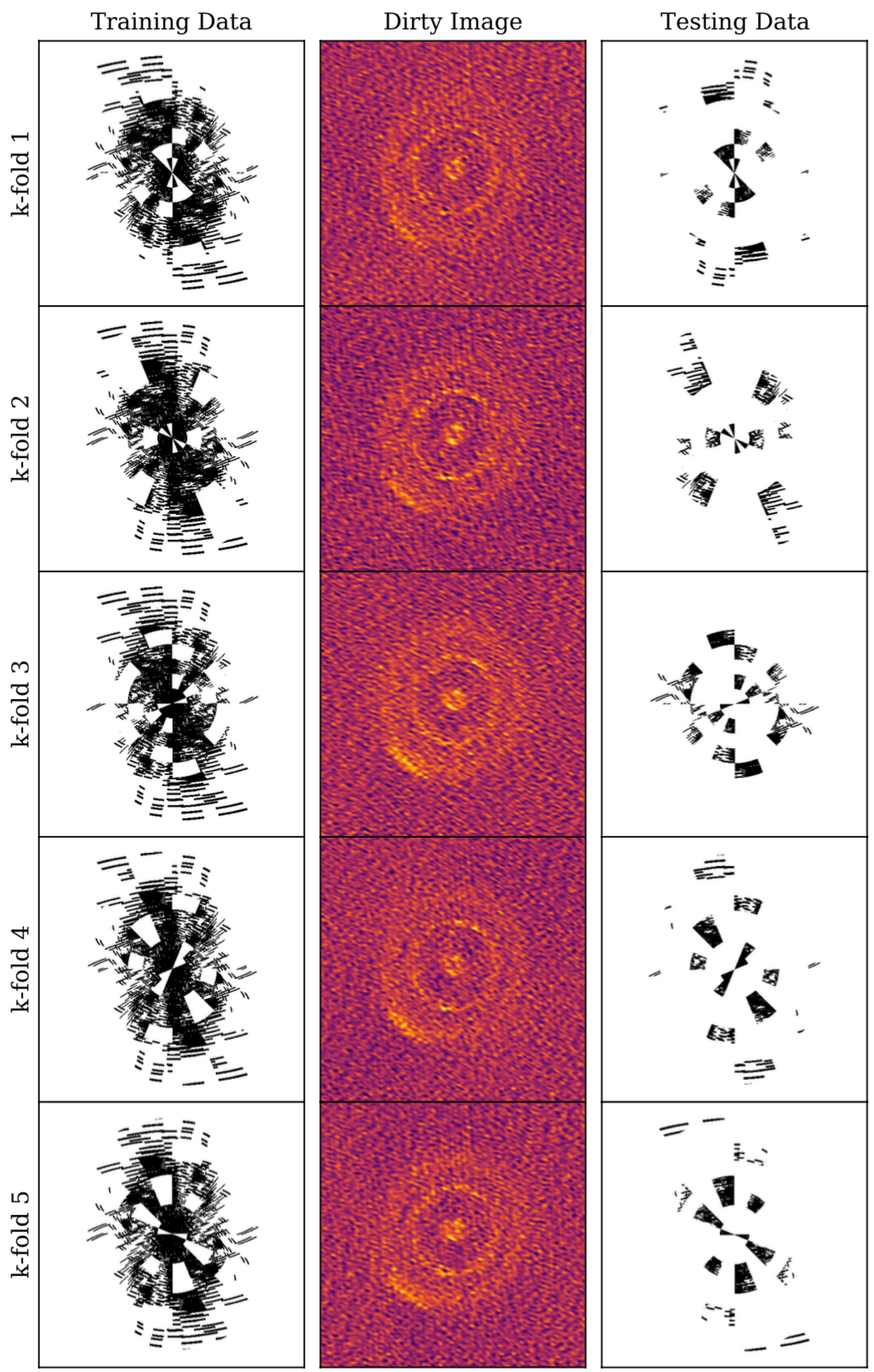
Dirty
Blank
Custom
Log Loss
1.20
1.15
1.10
1.05
1.00
0.95
10^0
10^1
10^2
10^3
Iteration



Dirty Image
Blank Image
Custom Image
Sky Brightness
Gradient
$i = 1$
$i = 10$
$i = 100$
$i = 500$
$i = 1000$
$\nabla = 5.37e-06$
$\nabla = 6.72e-07$
$\nabla = 2.34e-10$
$\nabla = 9.84e-05$
$\nabla = 1.06e-05$
$\nabla = 1.56e-08$
$\nabla = 6.07e-11$
$\nabla = 4.47e-04$
$\nabla = 7.32e-05$
$\nabla = 1.80e-07$
$\nabla = 7.46e-10$
$\nabla = 5.20e-11$

Cross-Validation

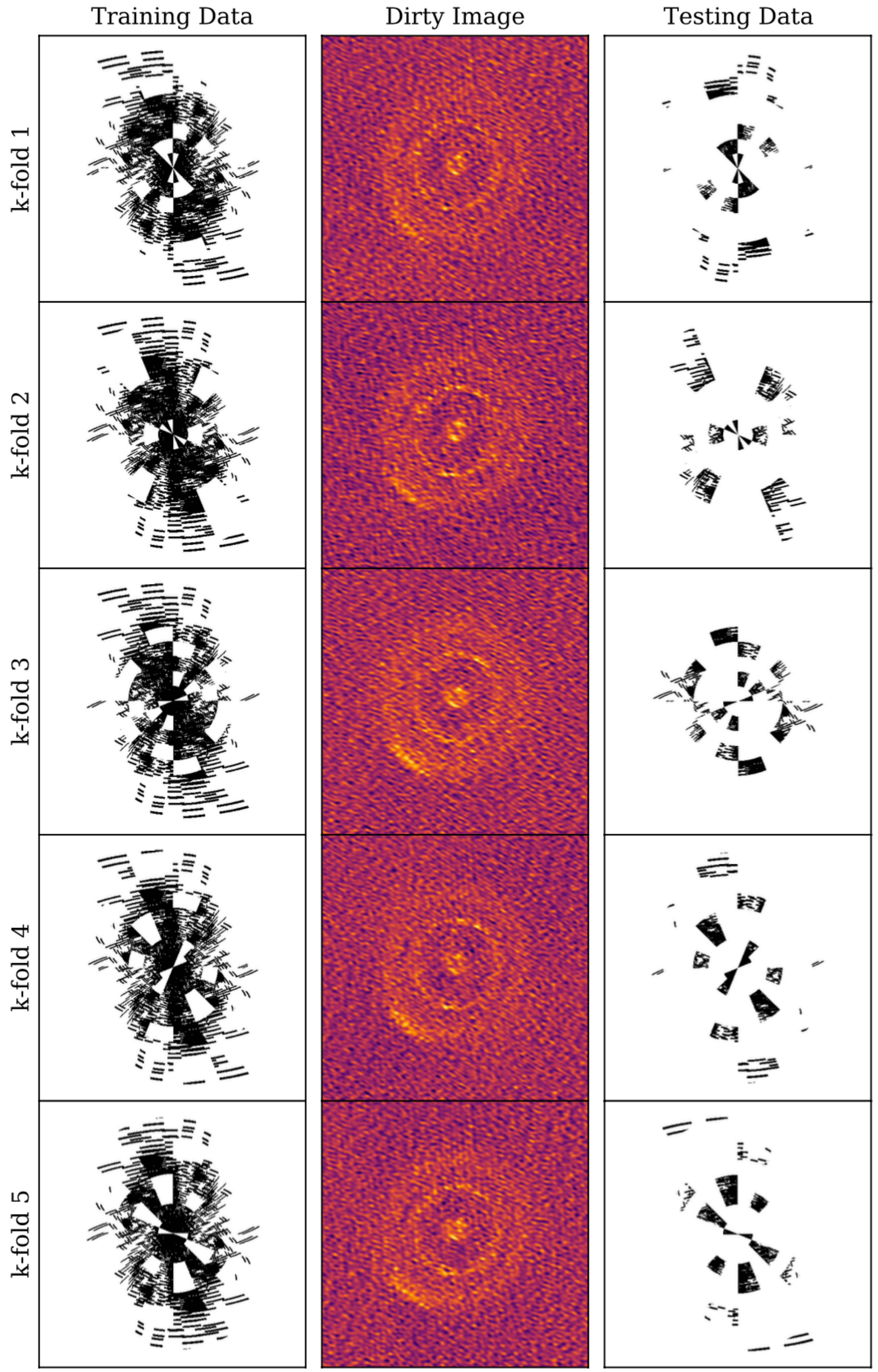
- Machine learning method for finding the model with the highest predictive power
- K-fold cross-validation:
  - split data into K chunks
  - use 1/K as the test dataset and train the model with the rest
  - compare model to test to get a cross-validation score



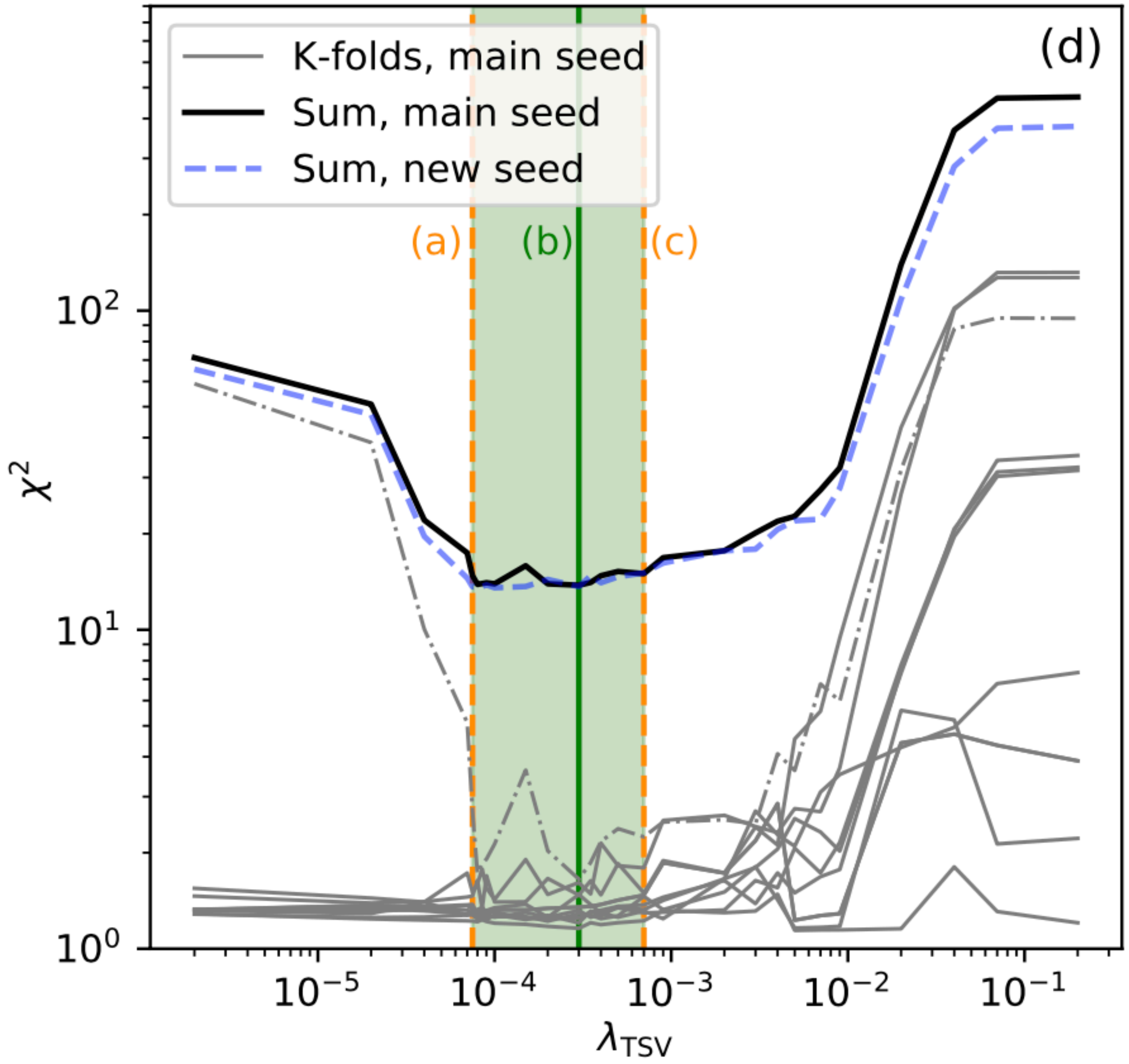
Adapted from Zawadzki et al. (2023, submitted)

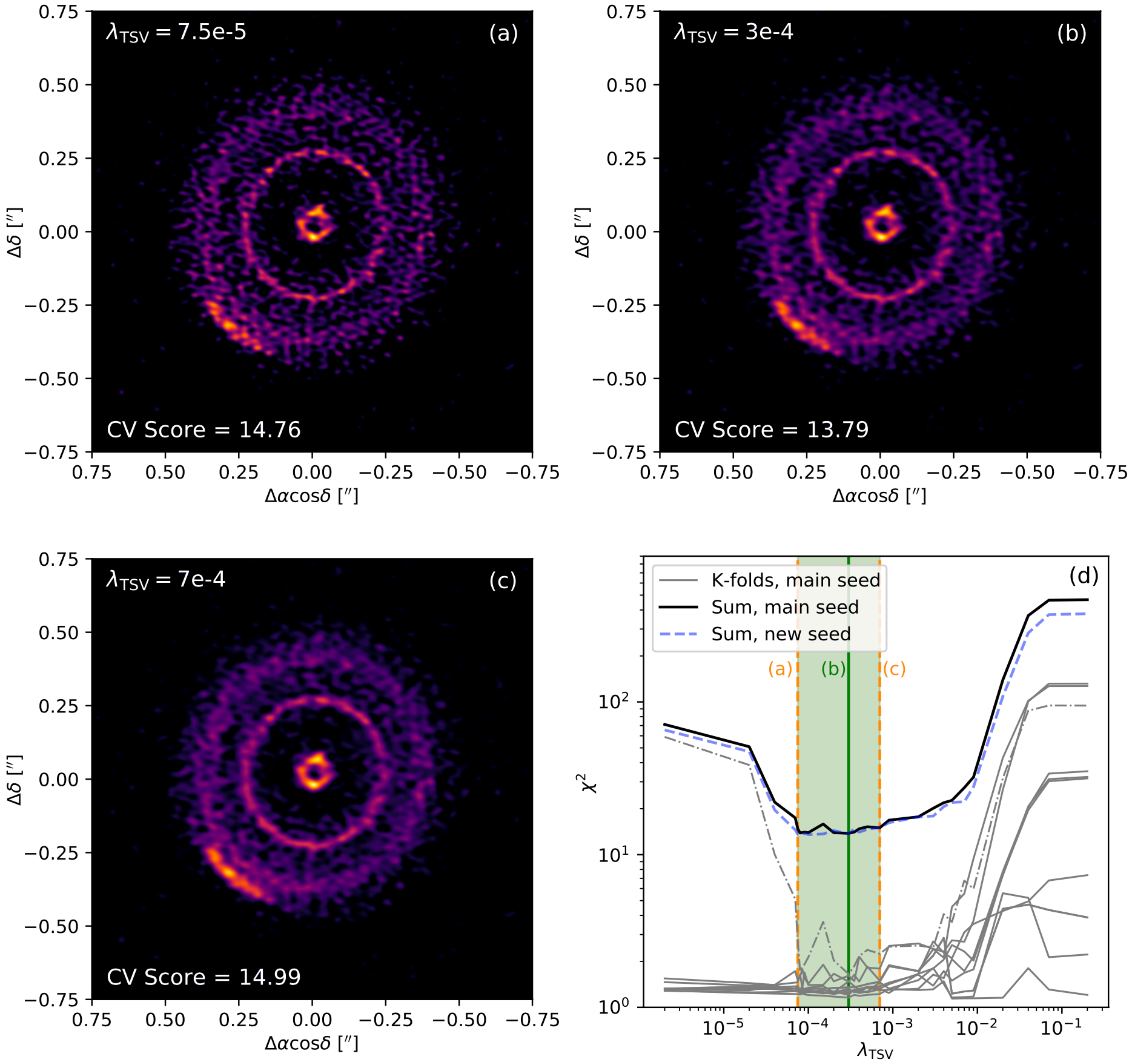
Hyperparameter Tuning

- Cross-validation can be used to determine the optimal λ values for each regularizer
- Minimizing the CV score yields a model with the best predictive power
- CV scores can be directly compared if CV setup and model parameterization remains constant



Adapted from Zawadzki et al. (2023, submitted)


K-folds, main seed
Sum, main seed
Sum, new seed
(d)
(a)
(b)
(c)
χ^2
10^2
10^1
10^0
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
λ_{TSV}

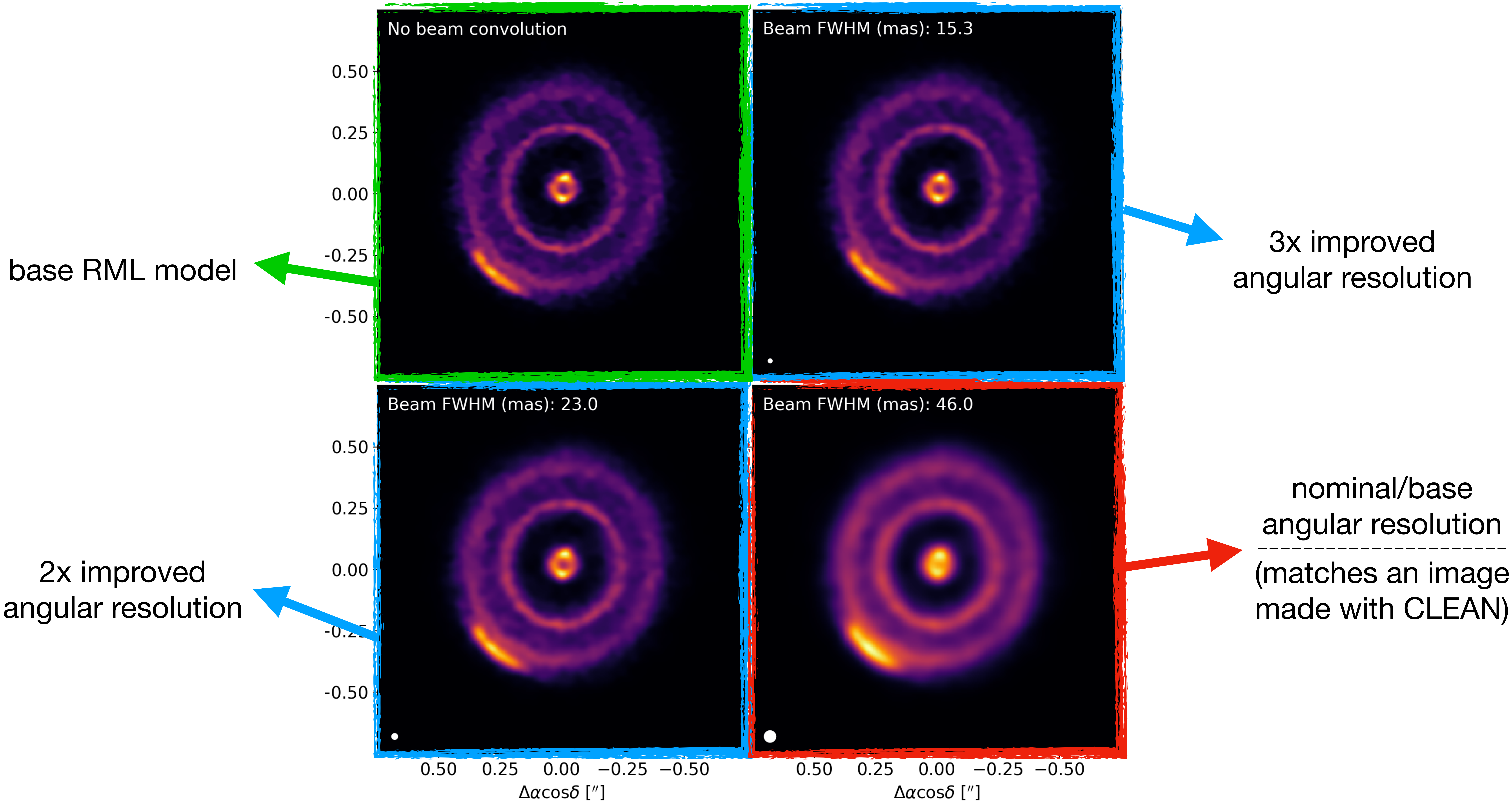


λ_{TSV} = 7.5e-5
(a)
CV Score = 14.76
λ_{TSV} = 3e-4
(b)
CV Score = 13.79
λ_{TSV} = 7e-4
(c)
CV Score = 14.99
$\Delta\delta$ ["]
$\Delta\alpha\cos\delta$ ["]
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
(d)
K-folds, main seed
Sum, main seed
Sum, new seed
(a)
(b)
(c)
χ^2
λ_{TSV}
10^0
10^1
10^2
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}

Cross-Validation Tips

- Selecting visibilities with the dartboard method tests how the model responds to data in different u-v space
- Selecting visibilities uniformly/randomly tests how the model responds to data in comparable u-v space
- Convergence is doubly important during CV

			Contribution to total CV score per K-fold									
iterations	time (s)	CV score	1	2	3	4	5	6	7	8	9	10
1000	123.5	62.50	1.88	1.67	45.15	3.06	2.49	1.30	1.18	1.20	2.02	2.55
3000	370.5	14.19	1.31	1.27	1.99	1.68	1.50	1.17	1.15	1.16	1.42	1.55



Adapted from Zawadzki et al. (2023a, submitted)

Future Work with MPoL

https://mpol-dev.github.io/MPoL/

MPoL

- MPoL is already functional, but still in development
- We want to expand to applications like
    - Spectral line data (+ new types of regularization)
    - Data from other telescopes (e.g. SMA)
    - New sources (more disks + other kinds of sources)