VERITAS

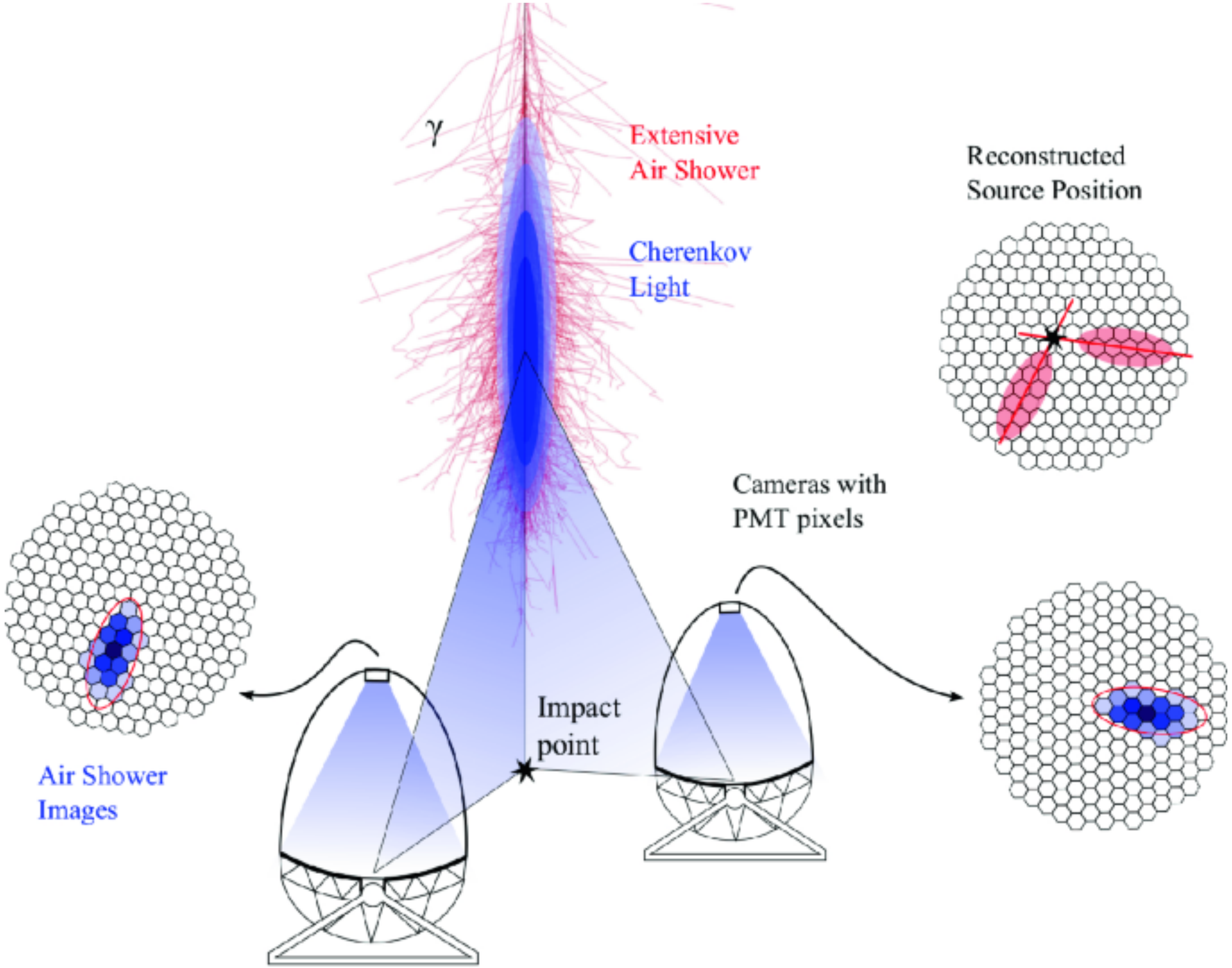
BARNARD COLLEGE
COLUMBIA UNIVERSITY

Universal Image Denoiser

Workshop on Machine Learning for Analysis of High-Energy Cosmic Particles

Ruo-Yu Shang

Reconstructing γ-ray Source Location

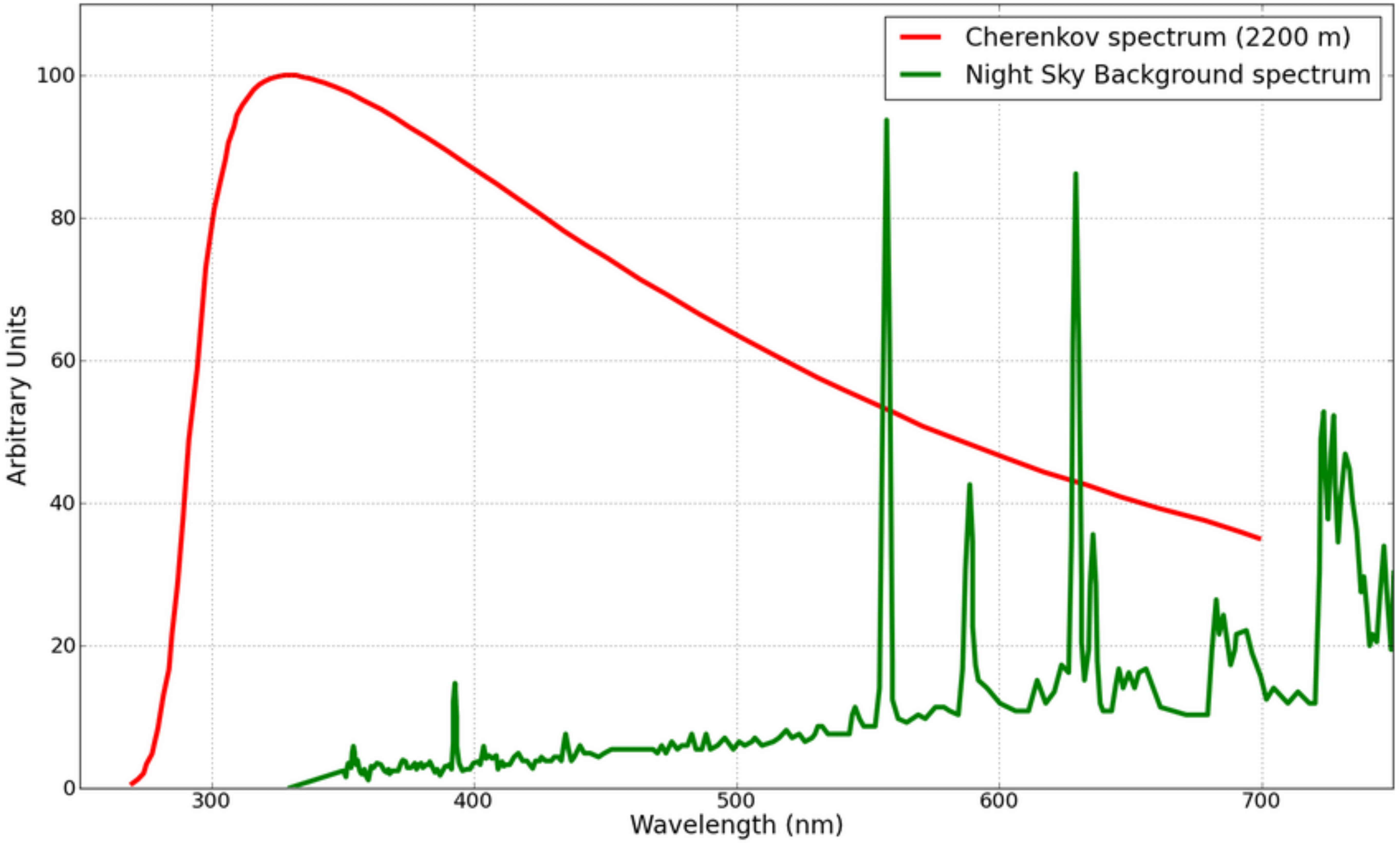


- Particle cascade in the air radiates Cherenkov light
- Telescope camera collects Cherenkov photons
- Source location can be found by following the trace of the shower image
- Require ≥ 2 telescopes for stereo reconstruction

Credit: Carreto Fidalgo, D. (2019). Cherenkov Telescopes and MAGIC

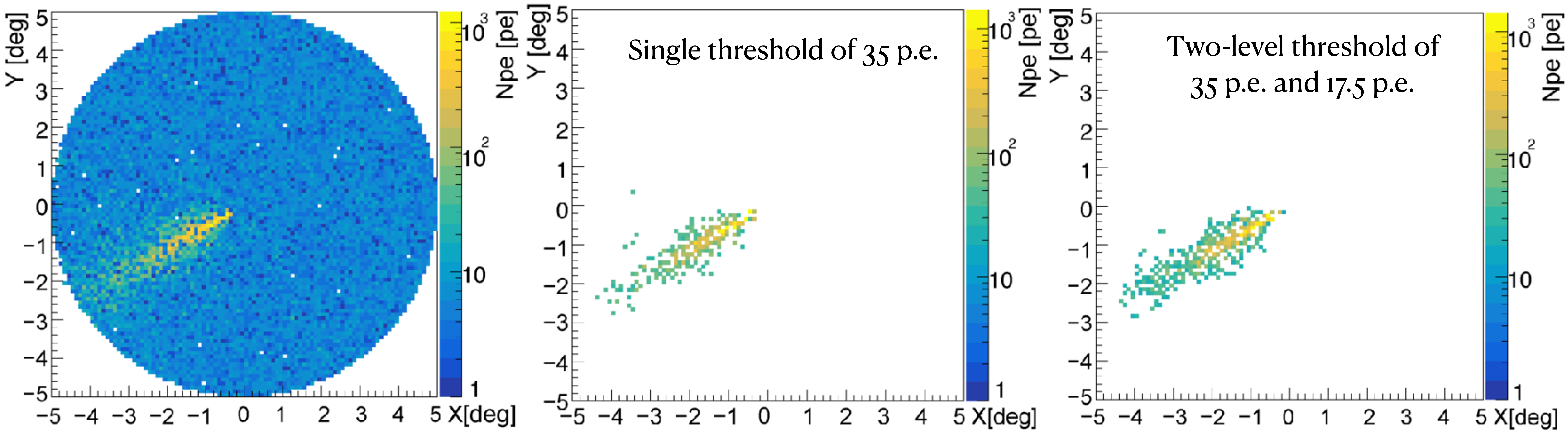
Night-sky background

arXiv:1308.1390



- The Cherenkov emission spectrum scales as λ^{-2} with atmospheric absorption cutting-off the spectrum at wavelengths below ~ 300 nm
- Background (green curve): Emission spectrum of the night sky background measured in La Palma.
- Sources of NSB: atomic airglow emission lines, Galactic plane, distant cities, etc

Conventional image cleaning: tail-cut



- Cherenkov image with no image cleaning (left), cleaned using the single-threshold cleaning method (middle, with a threshold of 35 p.e.), and cleaned using the **tail-cut** clean method (right, with thresholds of 35 p.e. and 17.5 p.e.)
- Today we will see how we can improve cleaning with neural network models

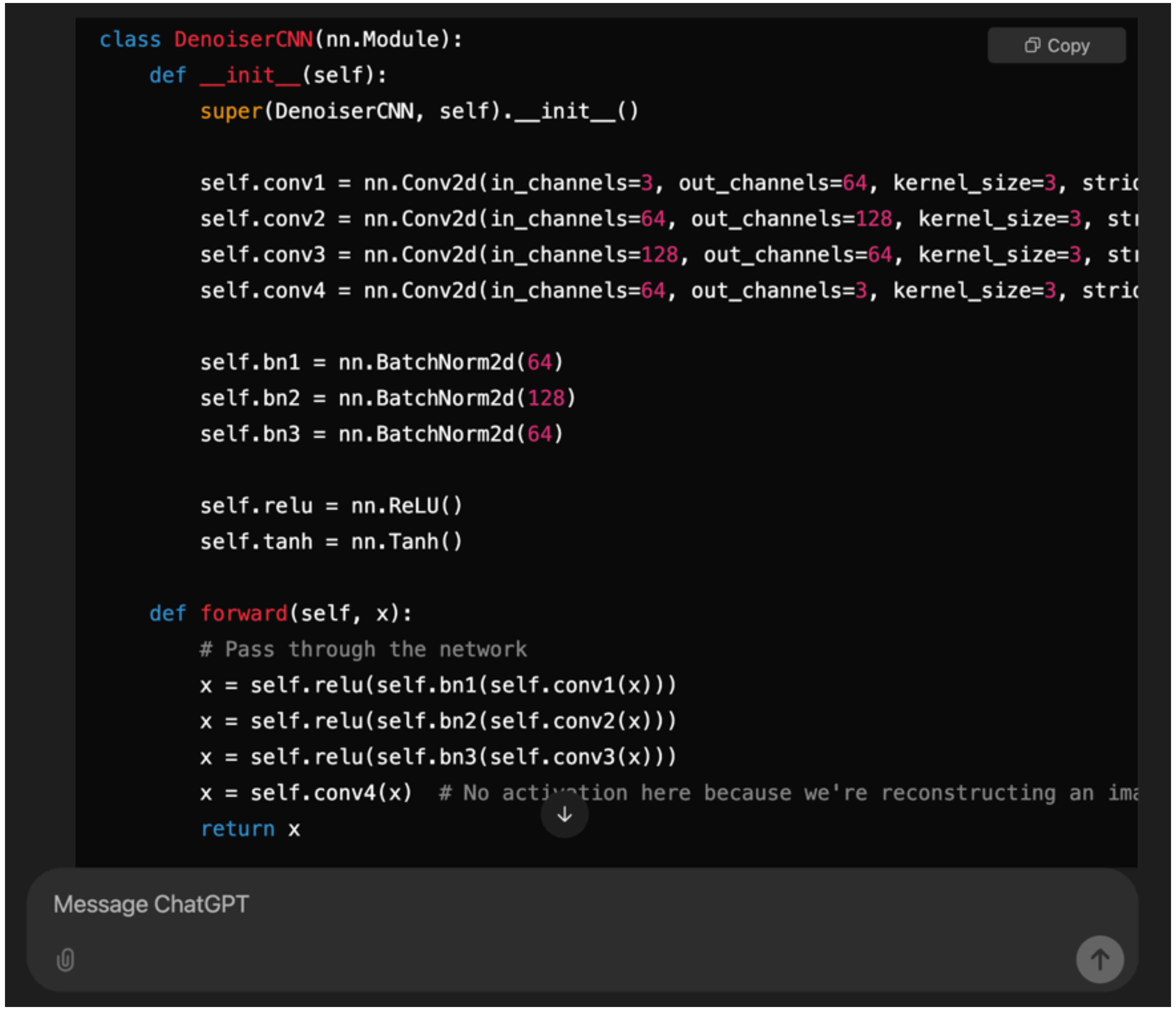
Tail-cut cleaning isn't efficient

noisy image
truth image
tailcut ceaning
True image
Tail-cut cleaning
Y (m)
X (m)
1.0
0.5
0.0
−0.5
−1.0
Unknown Frame

In the case of large background fluctuations, tail-cut would discard pixels with signal strengths comparable to the fluctuations

Convolutional Neural Network Denoiser

- Me: Please give me an example of convolution neural network denoiser in PyTorch :)
- ChatGPT: Certainly! Below is an example of a simple Convolutional Neural Network (CNN) denoiser implemented in PyTorch. This network is designed to remove noise from images by learning the mapping between noisy and clean images.



```python
class DenoiserCNN(nn.Module):
    def __init__(self):
        super(DenoiserCNN, self).__init__()

        self.conv1 = nn.Conv2d(in_channels=3, out_channels=64, kernel_size=3, stric
        self.conv2 = nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, st
        self.conv3 = nn.Conv2d(in_channels=128, out_channels=64, kernel_size=3, st
        self.conv4 = nn.Conv2d(in_channels=64, out_channels=3, kernel_size=3, stric

        self.bn1 = nn.BatchNorm2d(64)
        self.bn2 = nn.BatchNorm2d(128)
        self.bn3 = nn.BatchNorm2d(64)

        self.relu = nn.ReLU()
        self.tanh = nn.Tanh()

    def forward(self, x):
        # Pass through the network
        x = self.relu(self.bn1(self.conv1(x)))
        x = self.relu(self.bn2(self.conv2(x)))
        x = self.relu(self.bn3(self.conv3(x)))
        x = self.conv4(x)  # No activation here because we're reconstructing an ima
        return x
```

Tests with Toy Simulation

noisy image
truth image
denoised image
True image
CNN de-noiser
1.0
0.5
0.0
−0.5
−1.0
Y (m)
X (m)
Unknown Frame

- The CNN denoiser is trained with randomly generated 2D-ellipse images with injected Gaussian noise
- CNN is impressive at removing noise in the images generated with the toy simulations

Tests with non-Gaussian noise

noisy image
truth image
denoised image
Y (m)
X (m)
1.0
0.5
0.0
−0.5
−1.0
Unknown Frame
True image
CNN de-noiser

- Non-Gaussian noise = non-central chi-square function with d.o.f. = 2
- The CNN denoiser trained with Gaussian noise fails to de-noise the image with non-Gaussian noise

Expectation value of true image

- Suppose a noisy observation of an image, $y = x + z$, where x is the original image drawn from $p(x)$ and z is sample of Gaussian white noise
- Given a noisy observation, the mean-square-error (MSE) estimate of the true signal
$$\hat{x}(y) = \int x p(x|y) dx = \int x \frac{p(y|x)p(x)}{p(y)} dx$$
- The observed density is a convolution of the prior with the noise
$$p(y) = \int p(y|x)p(x)dx = \int g(y-x)p(x)dx$$

Implicit prior

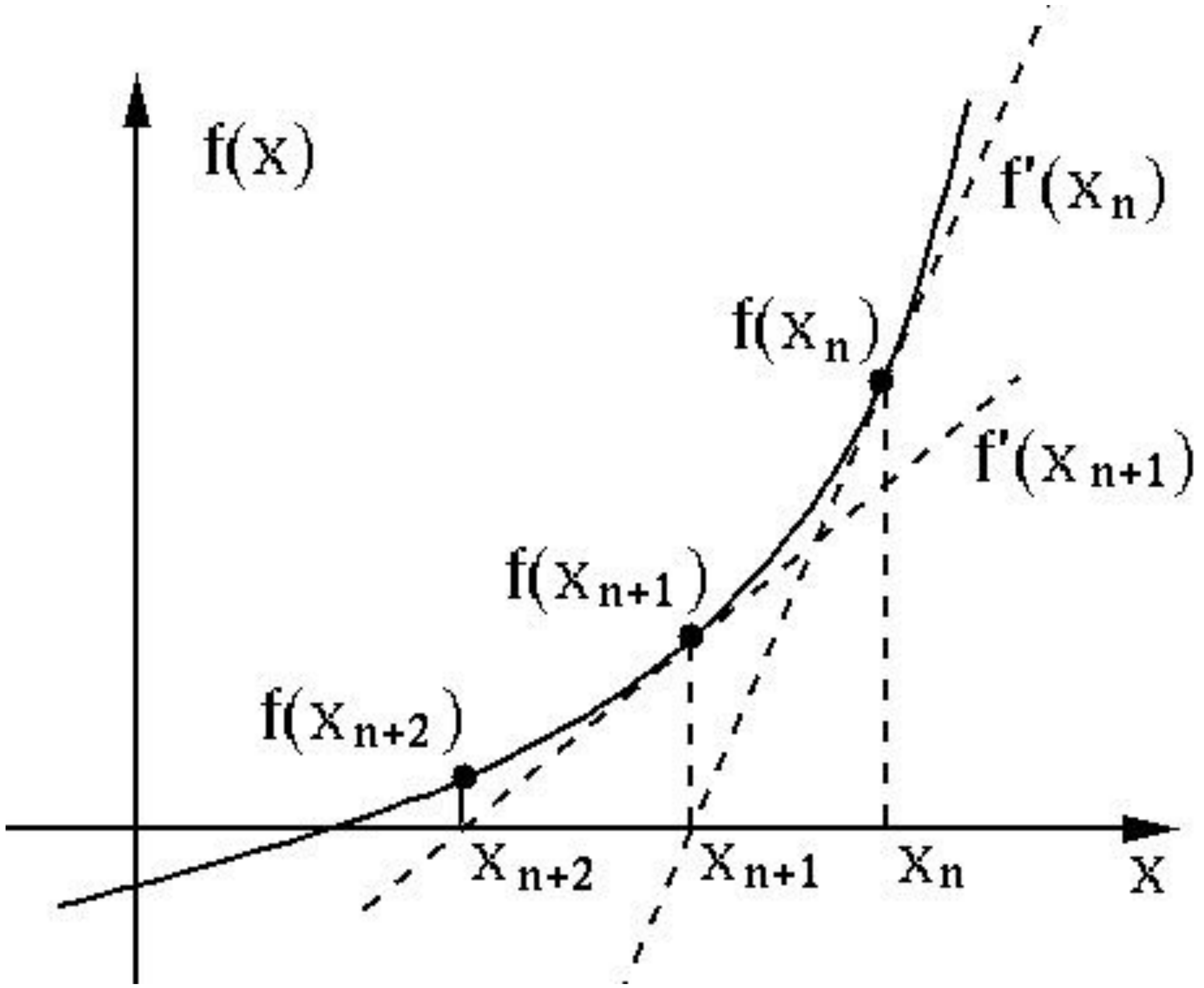
Ref: Advances in Neural Information Processing Systems 34 (2021): 13242-13254

- The gradient

$$\nabla_y p(y) = \nabla_y \int g(y-x)p(x)dx = \frac{1}{\sigma^2}\int (x-y)p(y\,|\,x)p(x)dx = \frac{1}{\sigma^2}\int (x-y)p(x\,|\,y)p(y)dx$$

- Multiplying both sides by $\sigma^2/p(y)$, we get $\sigma^2\dfrac{\nabla_y p(y)}{p(y)} = \sigma^2\,\nabla_y \log p(y) = \hat{x} - y$
- Note that
  - the relevant density is not the prior, $p(x)$, but the noisy observation density, $p(y)$;
  - the gradient is computed on the log density

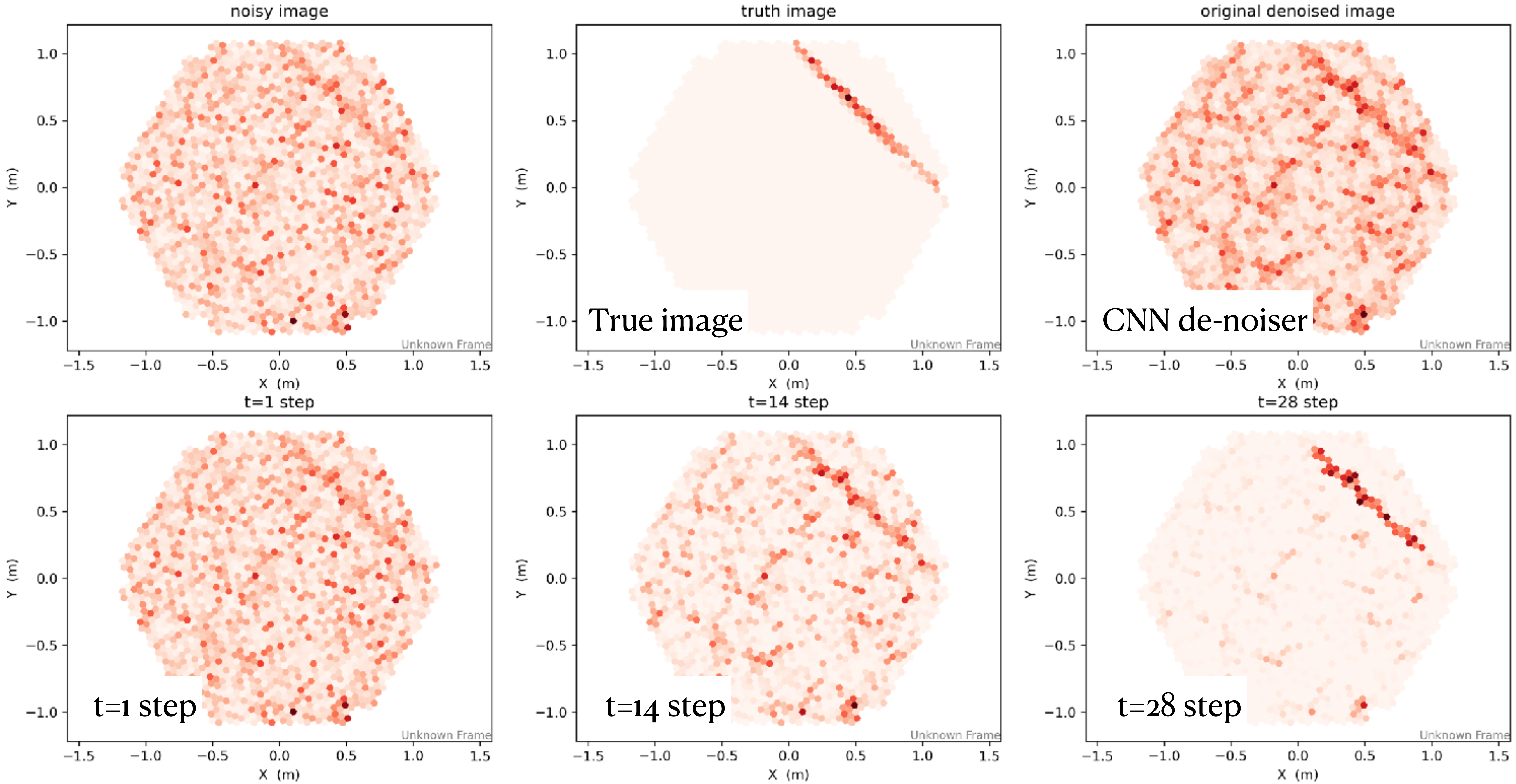
Newton’s root finding method



The Algorithm

- Let $\hat{x}(y)$ be the denoiser function, and $f(y) = \hat{x}(y) - y$ is the residual which is proportional to $\nabla_y \log p(y)$
- We start with a noisy image y_0, and the denoiser tells us to make a full step of $f(y_0)$
- Instead, we will only take a fraction step $hf(y_0)$, where $h = 0.2$
- We will arrive at a new image $y_1 = y_0 + hf(y_0)$, and we repeat the process with $y_2 = y_1 + hf(y_1)$
- In each step, we apply a tail-cut cleaning with a pre-defined threshold, and replace the denoised pixels inside the mask with the original pixels, and also calculate noise RMS outside the mask
- We continue the process $y_{t+1} = y_t + hf(y_t)$ until the the background $\mathrm{RMS}(t)/\mathrm{RMS}(t_0) < 1/5$

Follow the Gradient $\nabla_y \log p(y)$



Denoising example: SCT camera

noisy image
original denoiser
universal denoiser
CNN de-noiser
Universal de-noiser
CameraFrame
y (m)
x (m)
0.25
0.20
0.15
0.10
0.05
0.00
−0.30
−0.25
−0.20
−0.15
−0.10
−0.05
0.00

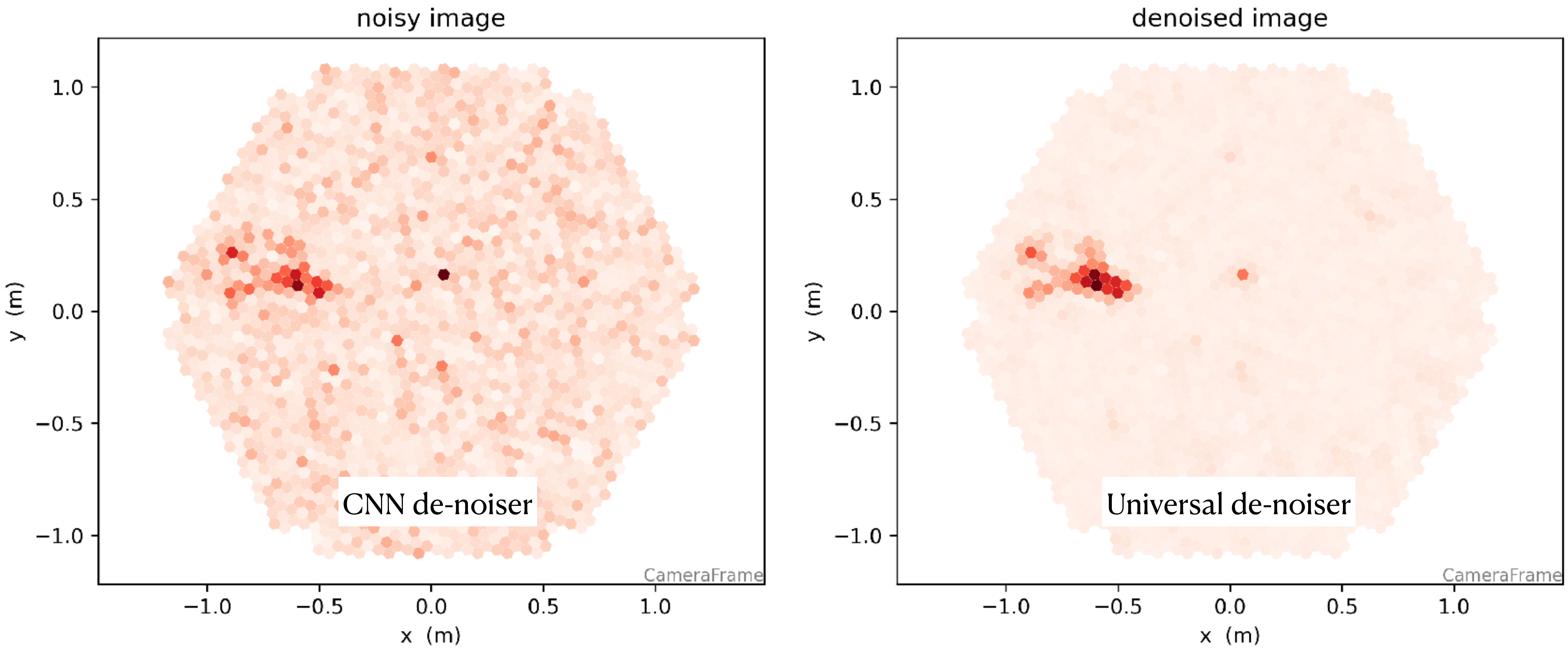
Denoising example: Flash camera

noisy image
original denoiser
universal denoiser
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
y (m)
−1.2
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
x (m)
CameraFrame

CNN de-noiser

Universal de-noiser

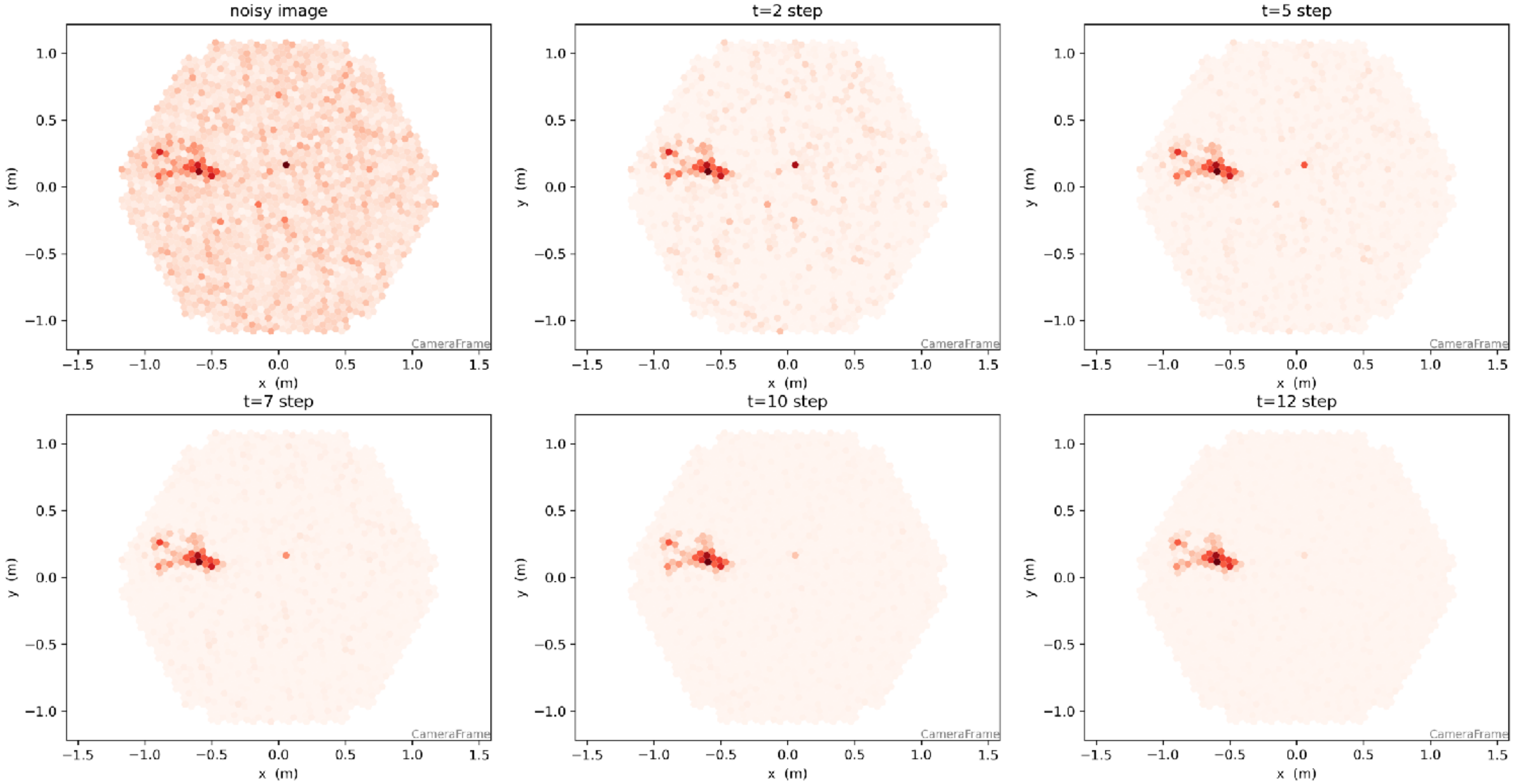
Denoising example: LST camera



CNN de-noiser

Universal de-noiser

Denoising example: LST camera



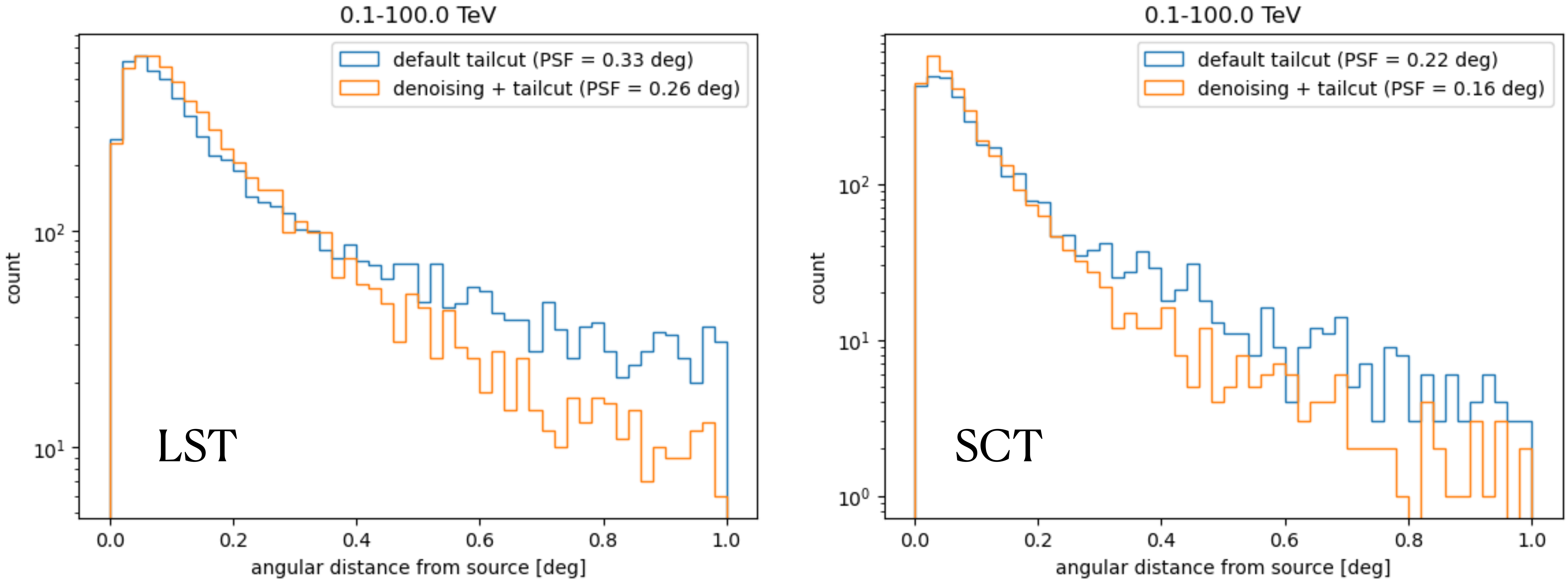
Summary

- Applying tailcut cleaning to original noisy images is inefficient, leading to worse angular reconstruction
- Based on the paper "Linear Inverse Problems using the Prior Implicit in a Denoiser", we develop an algorithm that allows us to use a simple toy-model-trained denoiser to clean images corrupted by arbitrary noise
- We also plan to test this method with VERITAS real data
  - Metrics: gamma-ray angular resolution, reconstruction efficiency, detection significance

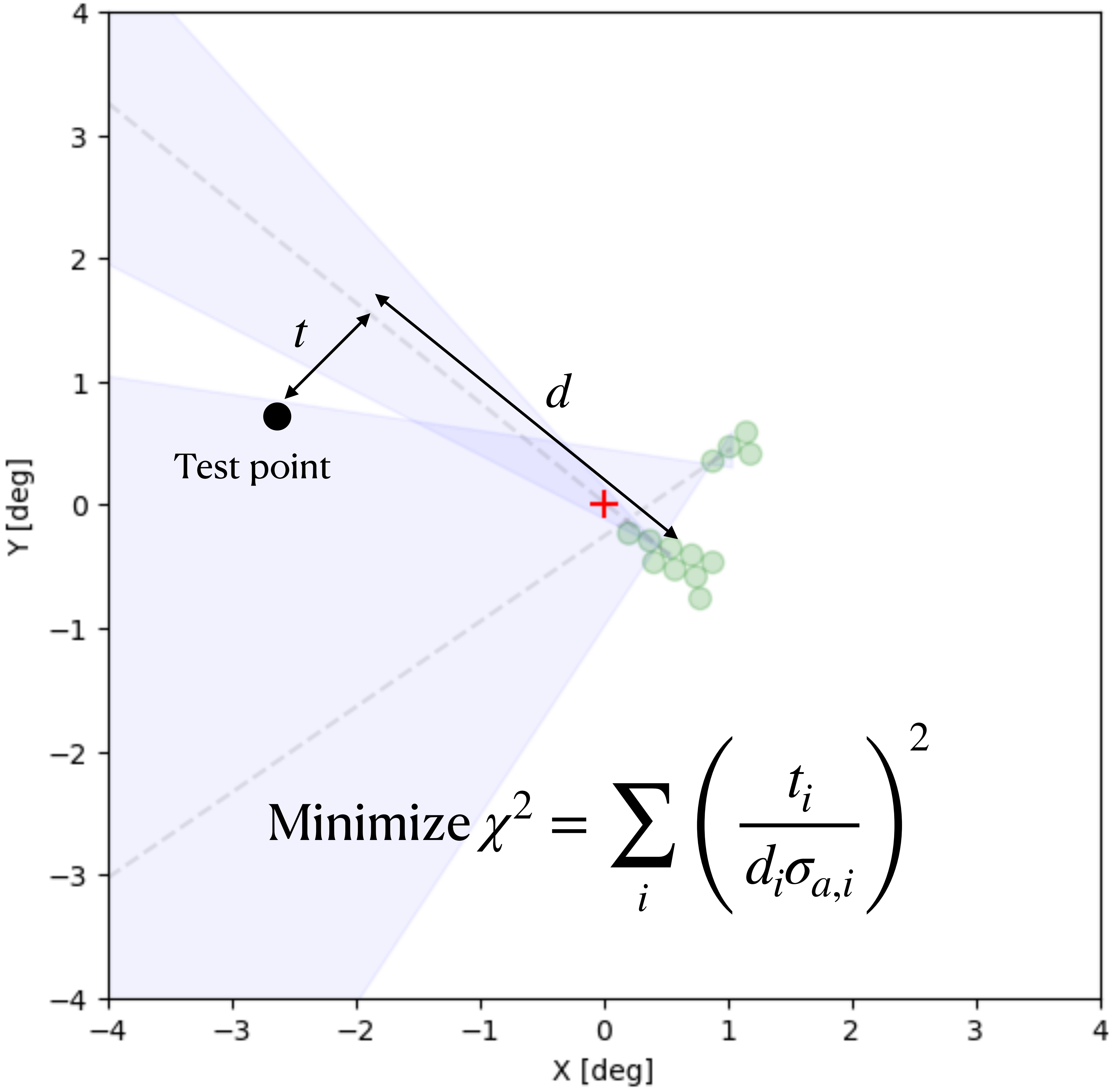
Backup

Gamma-ray angular resolution

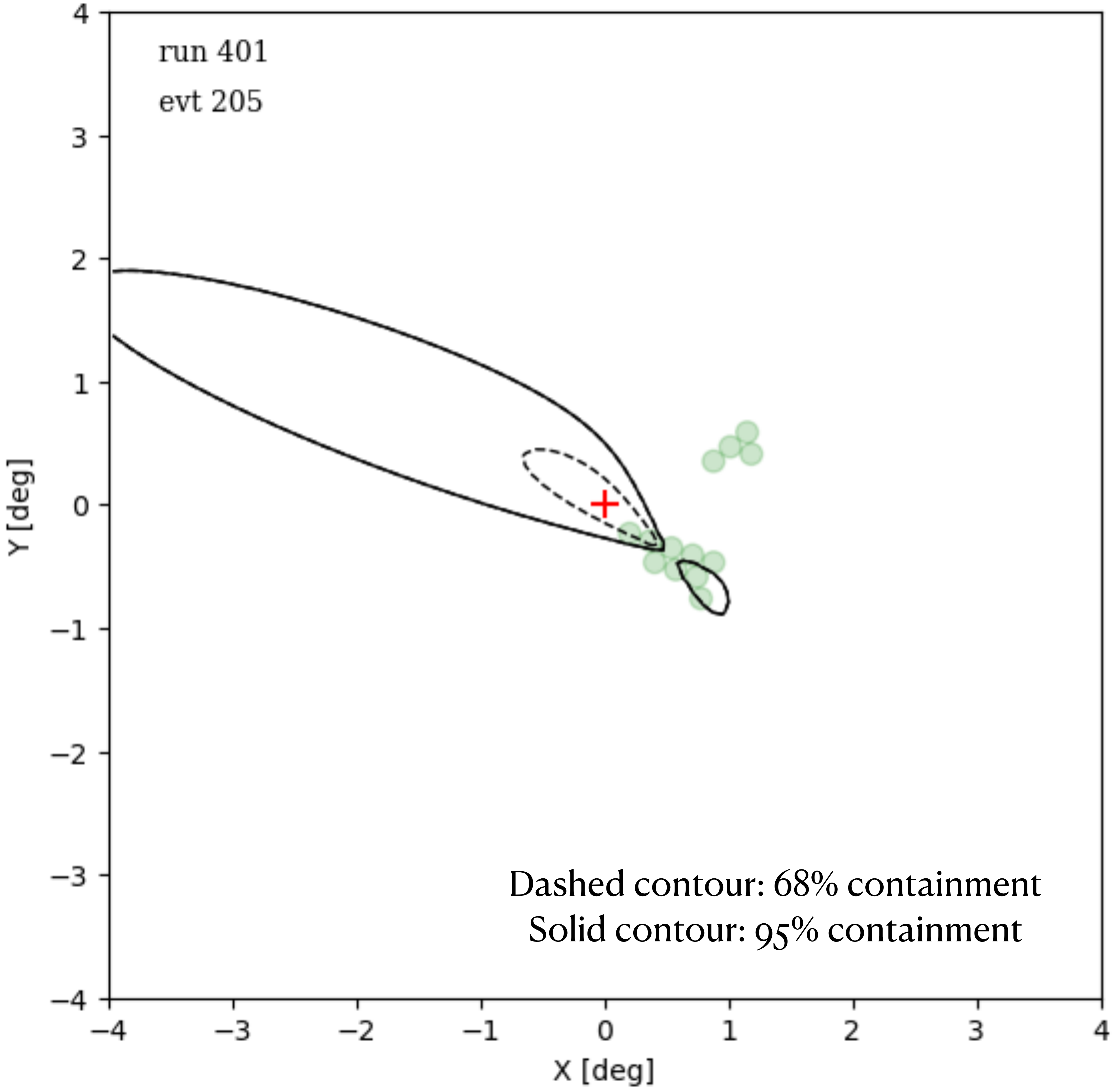
Optimized default cleaning: picture threshold 3.5 p.e., boundary threshold 2.5 p.e.



Source Location Estimation & Uncertainty



$$\text{Minimize } \chi^2 = \sum_i \left(\frac{t_i}{d_i \sigma_{a,i}} \right)^2$$



Dashed contour: 68% containment
Solid contour: 95% containment

Truth image

noisy image
truth image
universal denoiser
−0.05
−0.10
−0.15
−0.20
−0.25
y (m)
0.00
0.05
0.10
0.15
0.20
0.25
x (m)
CameraFrame

Truth image

noisy image
truth image
universal denoiser
y (m)
x (m)
0.20
0.18
0.16
0.14
0.12
0.10
0.08
0.04
0.06
0.08
0.10
0.12
0.14
0.16
0.18
CameraFrame

Truth image

noisy image
truth image
universal denoiser
−0.10
−0.15
−0.20
−0.25
−0.30
−0.35
−0.40
y (m)
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
x (m)
CameraFrame