Denoising and Inpainting Techniques for Beam Profile Analysis

Glenn Anta Bucagu (BE-CSS-DSB)

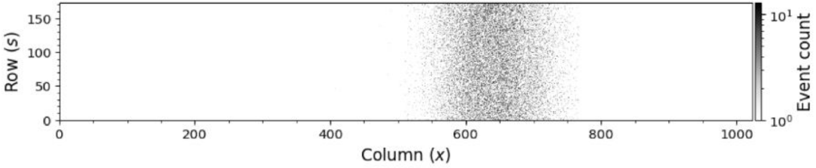
18th October 2023

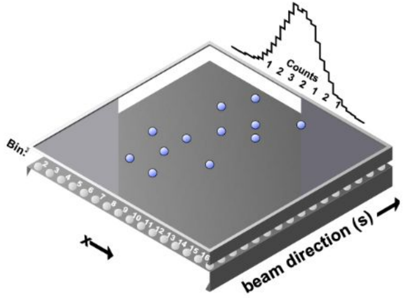
Overview

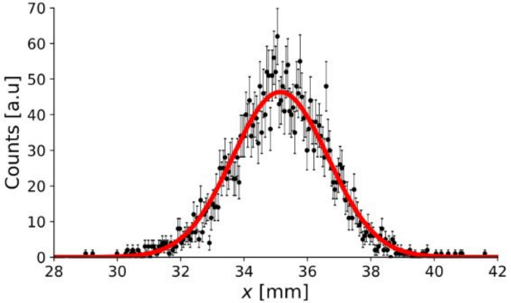
1. Review of PS-BGI and the problem to be solved
2. Short discussion on the previous approach used to tackle the problem
3. Discussion on our proposed approach
4. Next steps and possible improvements
5. Q & A

PS-BGI

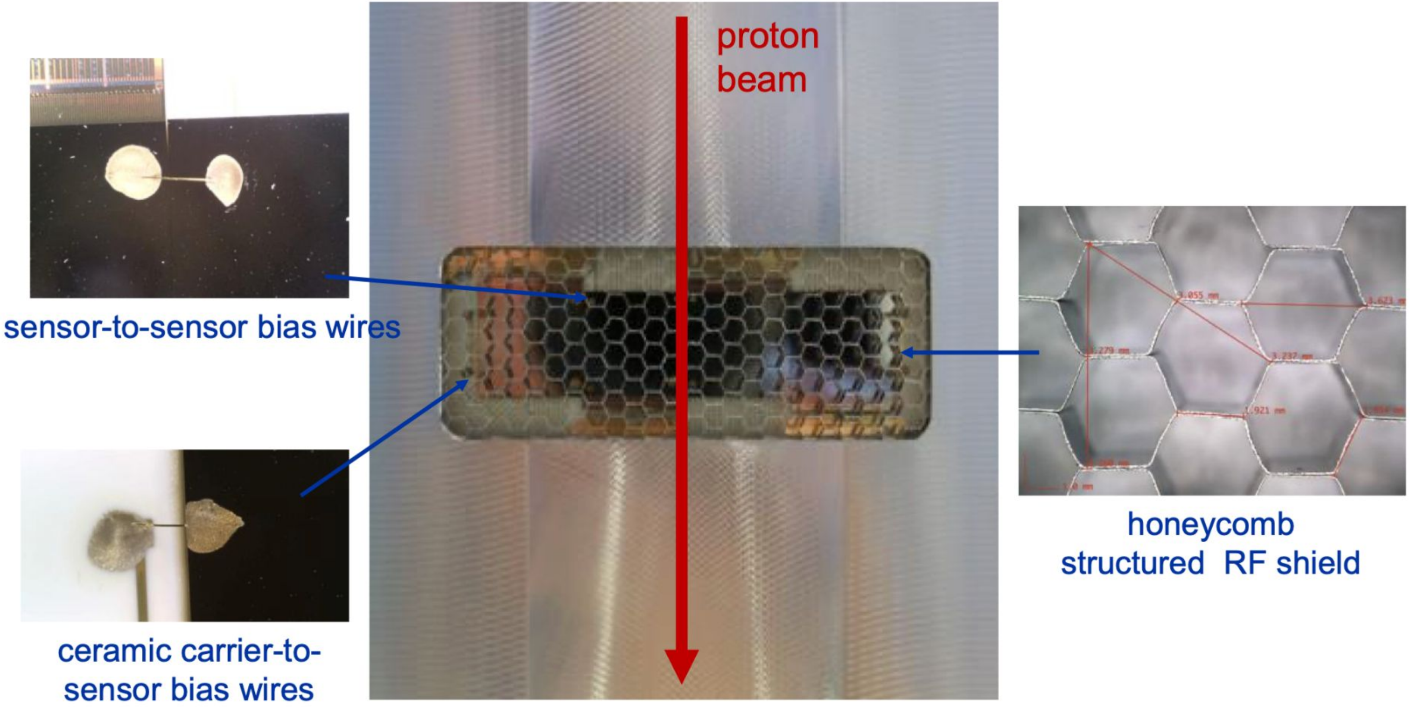
- Goal is to detect ionisation electrons to provide key information about the beam
- We expect the beam profile to follow a Gaussian distribution
- Standard deviation of this distribution is the key quantity



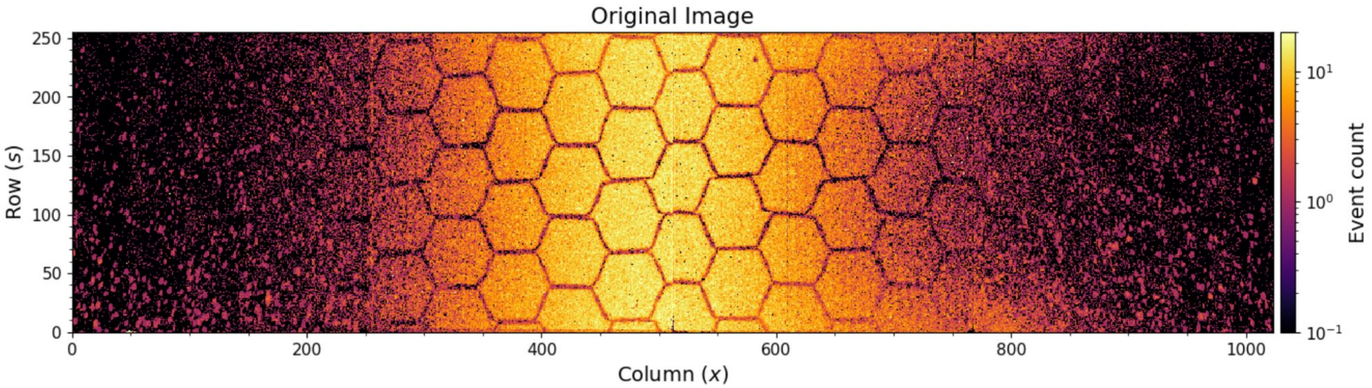




PS-BGI - Instruments



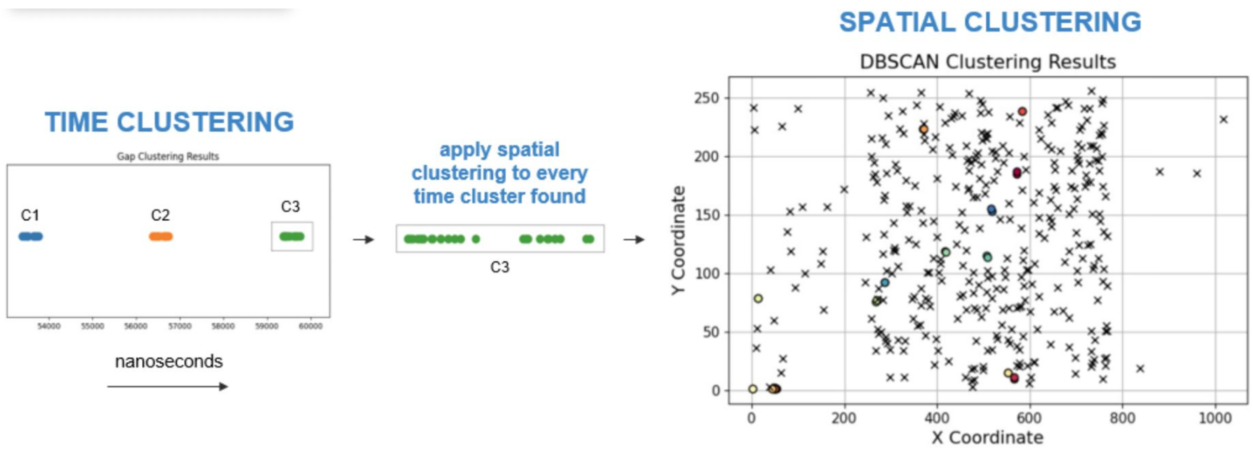
3 Key Issues

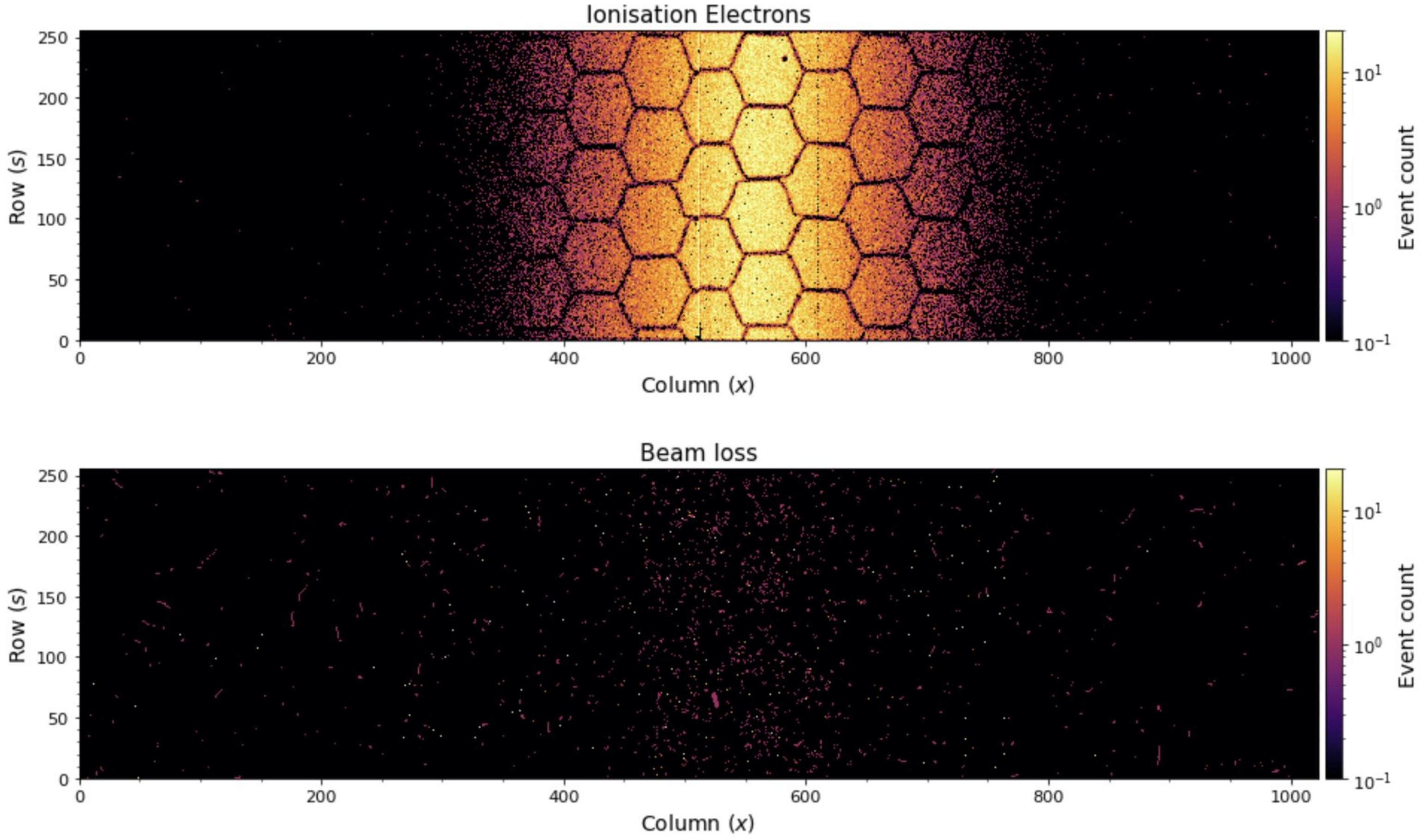


- Acquisitions can be viewed as a timeseries of pixel images (effectively a video). Not every timestamp is useful for analysis.
- If we consider a single pixel image (snapshot of the video) it comes with so-called beam (signal) and beam loss (noise)
- The honeycomb-shaped RF shield also masks certain pixels

Previous Approach: Spatial and Temporal DBSCAN Clustering

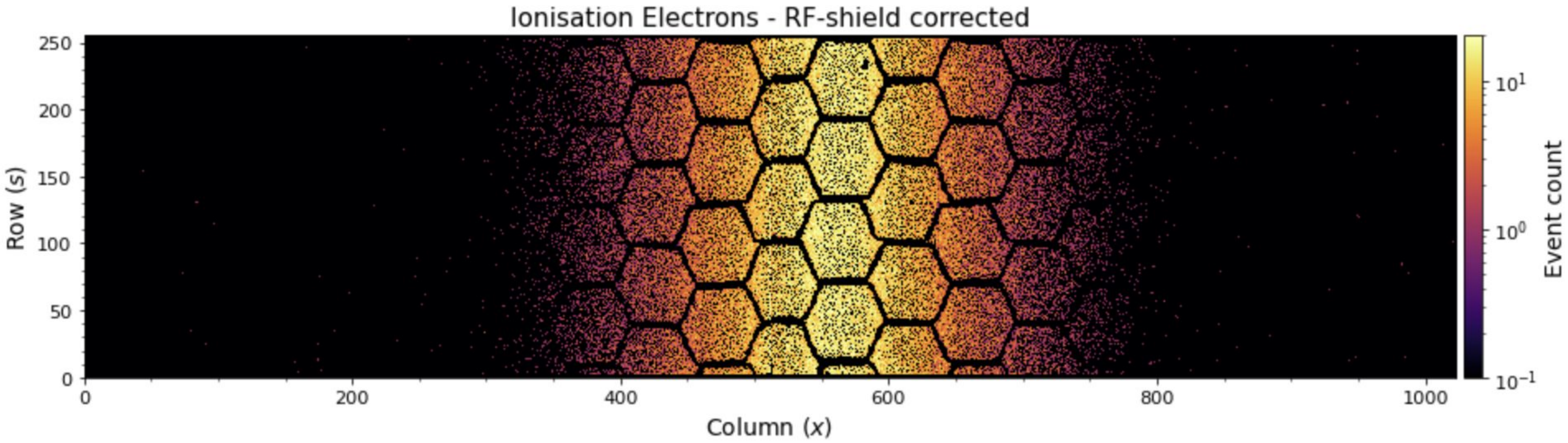
- Temporal clustering applied
- Spatial clustering + Time over Threshold (ToT) Filtering




Ionisation Electrons
Row (s)
Column (x)
Event count
Beam loss

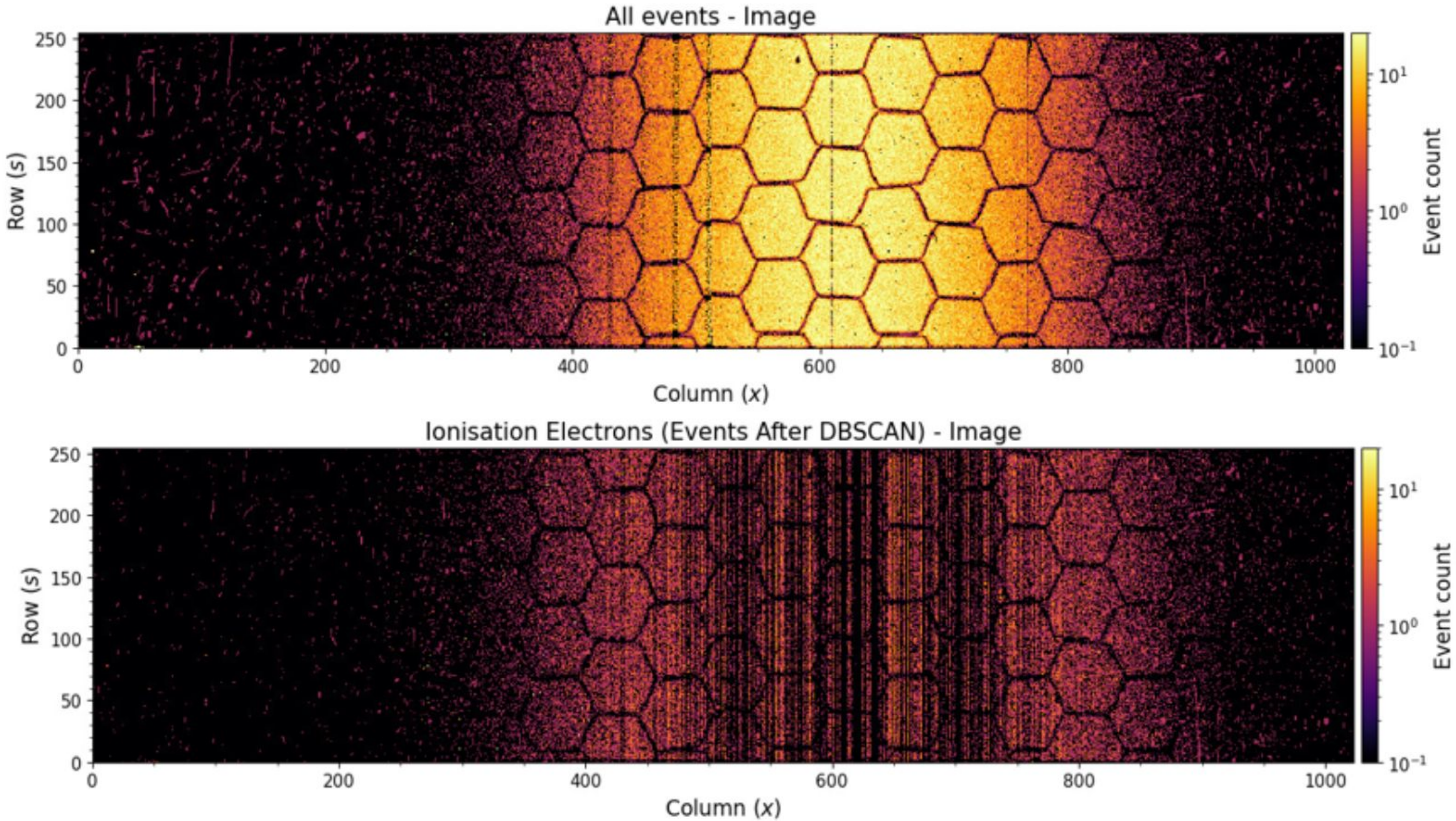
Previous Approach: RF-Shield Correction

- Identify the column with the most masked pixels (as a result of RF-shield)
- Pseudo-uniformly mask pixels across the entire pixel image such that each column has equal amount of masking


Ionisation Electrons - RF-shield corrected
Row (s)
Column (x)
Event count
0
50
100
150
200
250
0
200
400
600
800
1000
10^{1}
10^{0}
10^{-1}

Problems

- Previous method works but only by recording raw data and processing offline
- DBSCAN is slow: each acquisition takes 50-100 seconds even with multiprocessing on several CPUs
- Scikit-learn is not GPU-accelerated (we can rely on multiprocessing but there may be some overhead)
- DBSCAN (spatial) clustering is not always optimal
- ToT Filtering is arbitrary and should be tuned



All events - Image
Row (s)
Column (x)
Event count
Ionisation Electrons (Events After DBSCAN) - Image
Row (s)
Column (x)
Event count

Our Objectives and Constraints

- Provide an efficient method for spatial clustering
- Provide an efficient method to correct for the RF shield
- Both methods should be sample-efficient
- Process should be entirely automated and generalize to a variety of conditions (e.g. beam type, amount of noise, instrument type etc.)

A Two Stage Approach

- Ideally, we would like neural networks to process acquisitions end-to-end (i.e. perform denoising, RF shield correction).
- The Universal Approximation Theorem[1] guarantees that such a neural network exists (but it doesn't tell us how to set up the architecture!)
- We posit that such a neural network should be trained under a supervised setting. For this, we need (raw image, fully denoised and corrected image) pairs.
- To this end, we propose a two stage approach to process beam images

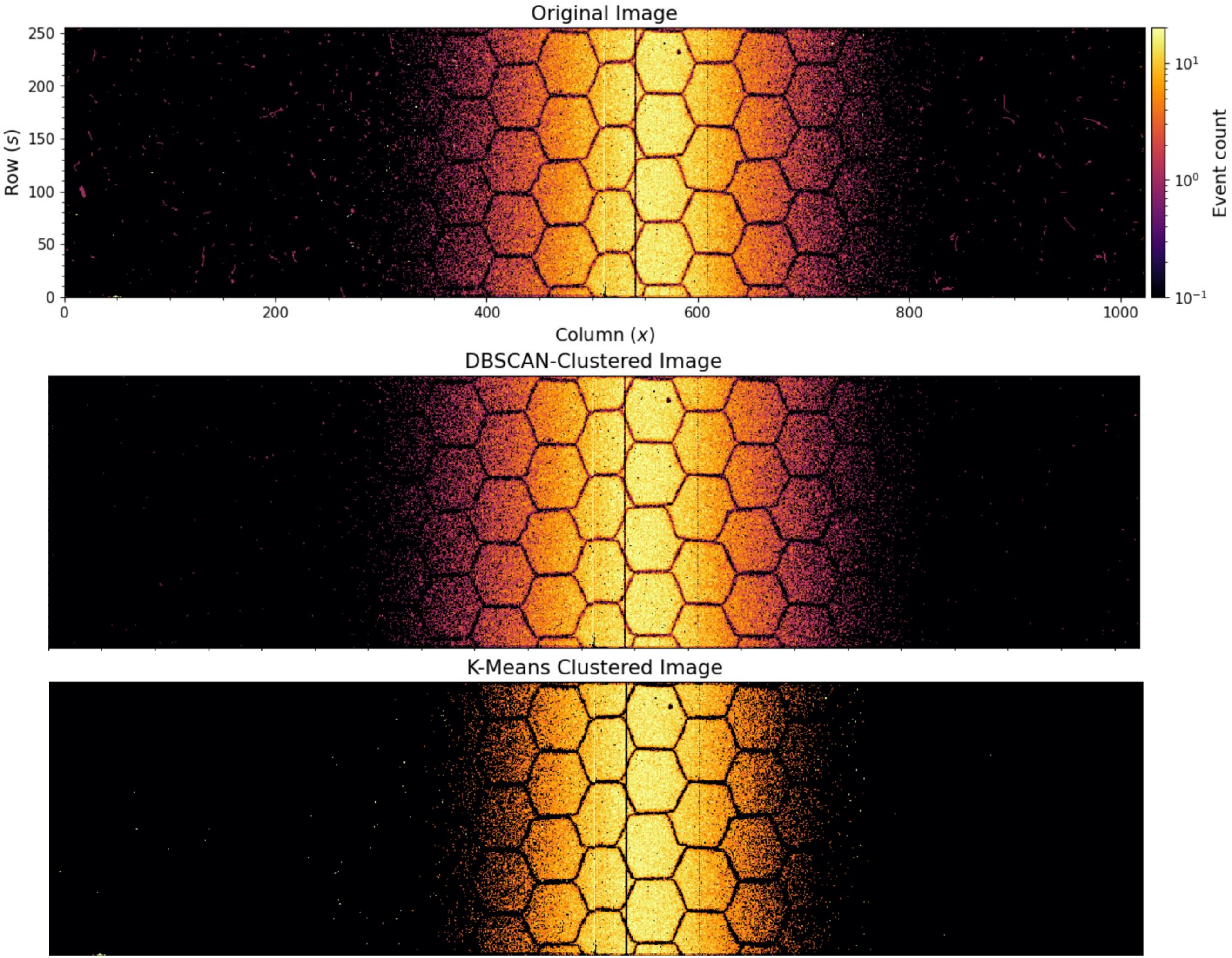
1. Hornik, Kurt; Stinchcombe, Maxwell; White, Halbert (1989). Multilayer Feedforward Networks are Universal Approximators. Neural Networks. Vol. 2. Pergamon Press. pp. 359–366

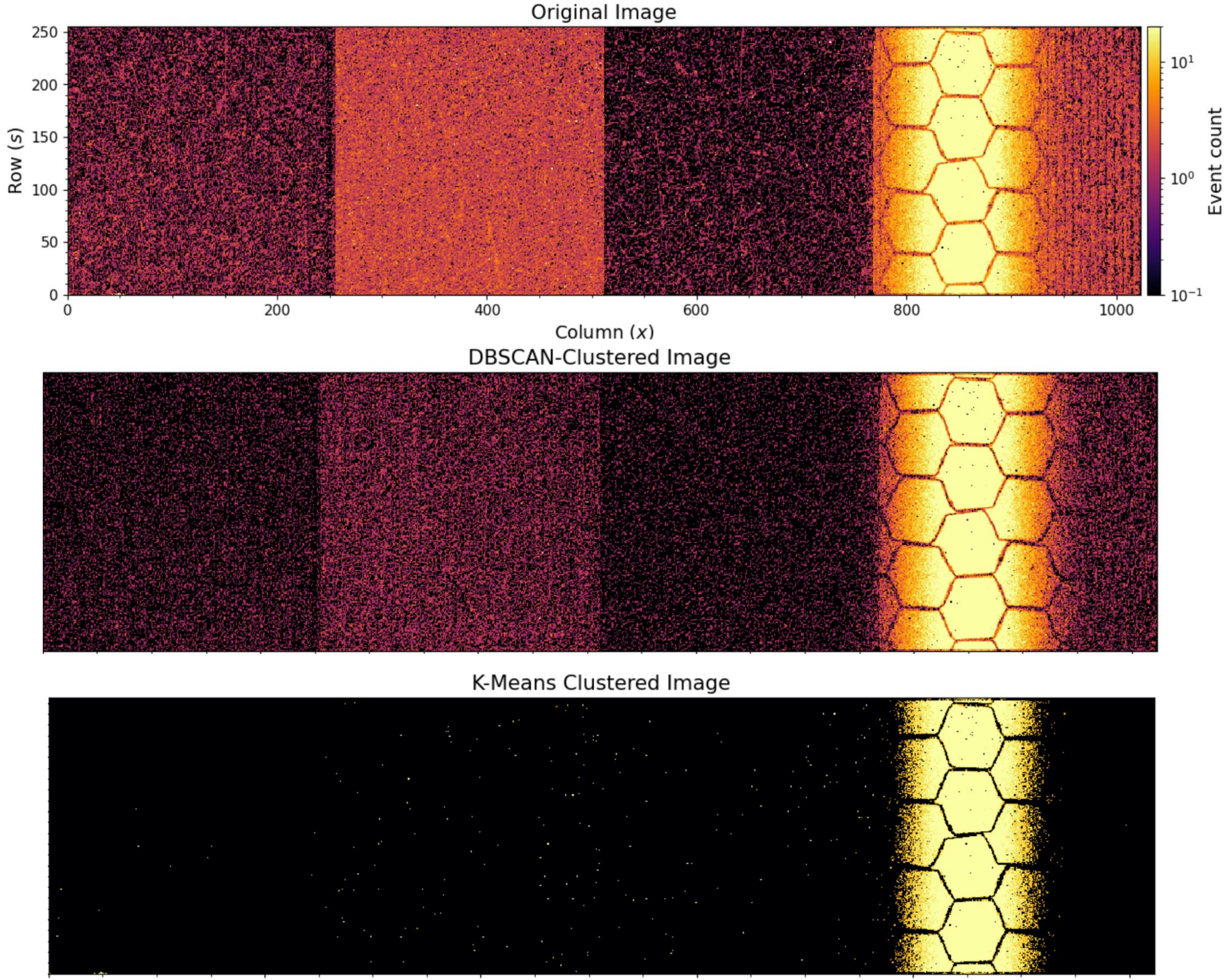
Stage 1: K-Means for Spatial Clustering

- The slowest operation in the pipeline is spatial clustering
- In clustering, **NO FREE LUNCH**; we select the clustering algorithm depending on the geometry of our space
- Replace DBSCAN with K-Means which is known to work well for general purpose flat geometries
- K-Means: Separate the pixel image into $K > 0$ groups such as to minimize the intra-group variance
- From the denoised image, extract a binary mask highlighting the RF-shield

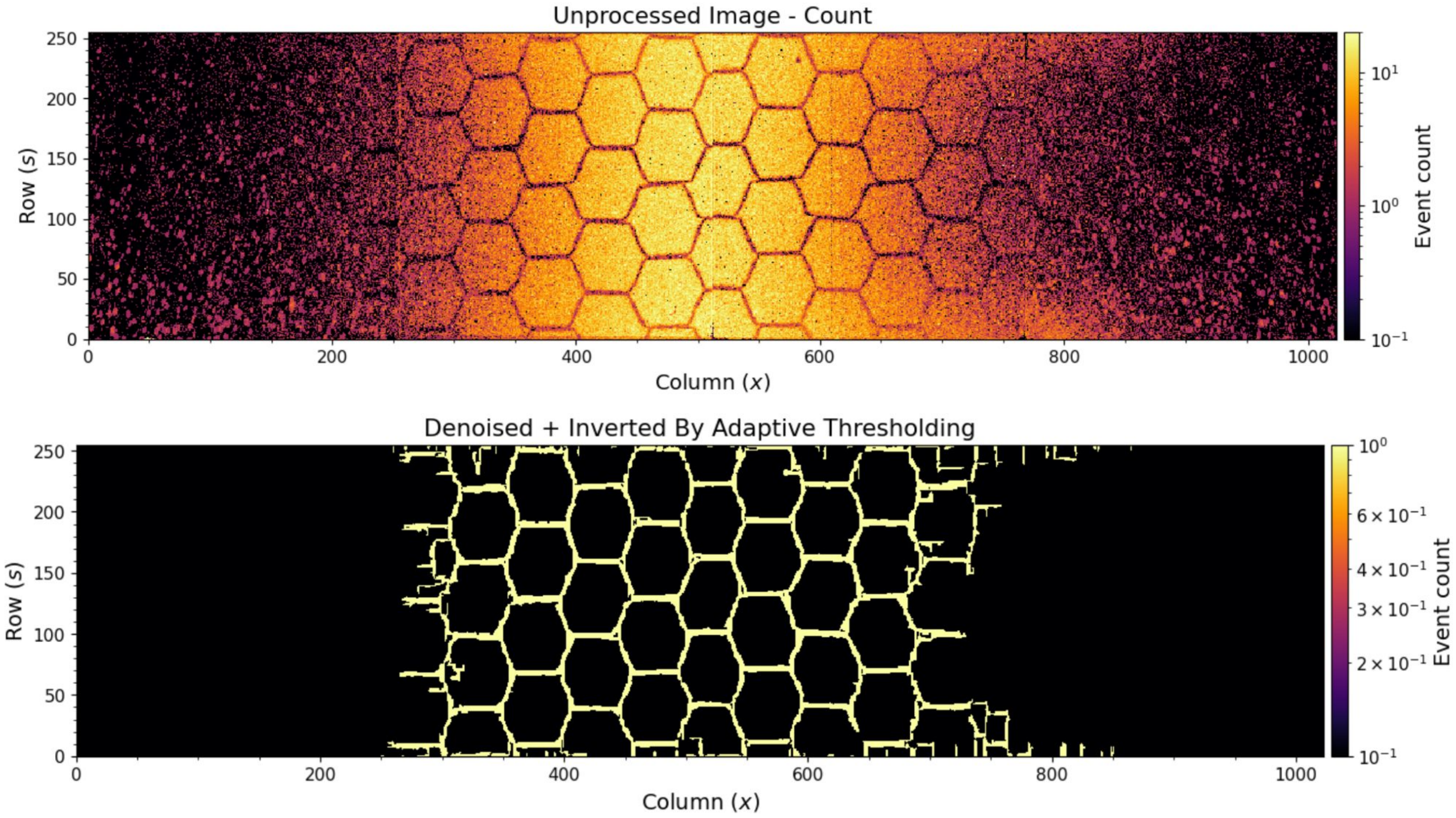
Advantages of K-Means Clustering

- In our case, for n pixels, K-Means has O(n) worst case time complexity vs $O(n^2)$ for DBSCAN
- DBSCAN also has greater space complexity (e.g. distance matrix needed)
- Single tunable parameter vs. two tunable hyperparameters for DBSCAN
- Highly sample-efficient (no “training” needed)


Original Image
Row (s)
Column (x)
Event count
DBSCAN-Clustered Image
K-Means Clustered Image


Original Image
Row (s)
Column (x)
Event count
DBSCAN-Clustered Image
K-Means Clustered Image

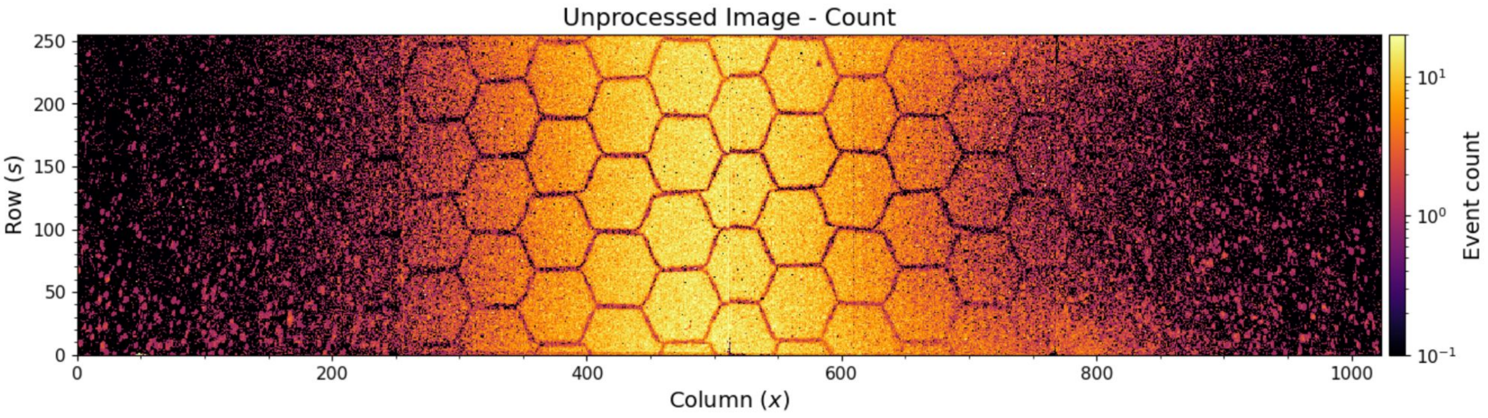
Extracting the RF Shield



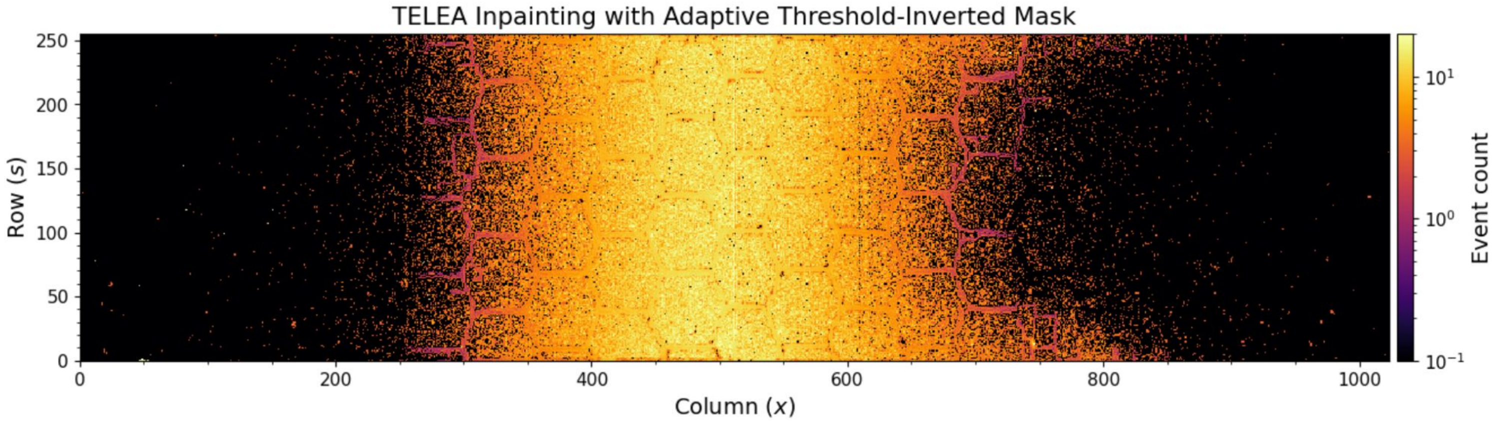
Stage 2: Inpainting the Denoised Image

- Given the denoised image and the binary mask, we can now inpaint.
- Inpainting = filling in missing parts of an image with plausible content. It's interpolation for images.
- For inpainting, we use the TELEA algorithm[2] which is efficiently implemented in OpenCV.
- On average, inpainting a single 256 x 1024 beam pixel image takes less than 0.1 seconds (single CPU).
- The two stage process gives us a good idea of "ground truth"

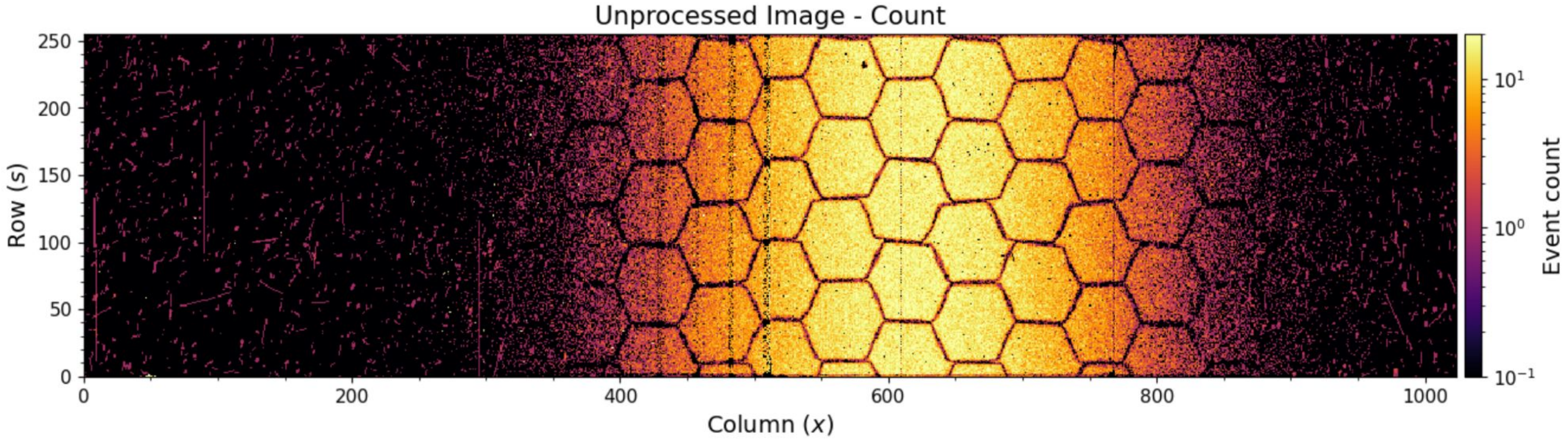
2. Telea, Alexandru. (2004). An Image Inpainting Technique Based on the Fast Marching Method. Journal of Graphics Tools. 9. 10.1080/10867651.2004.10487596.

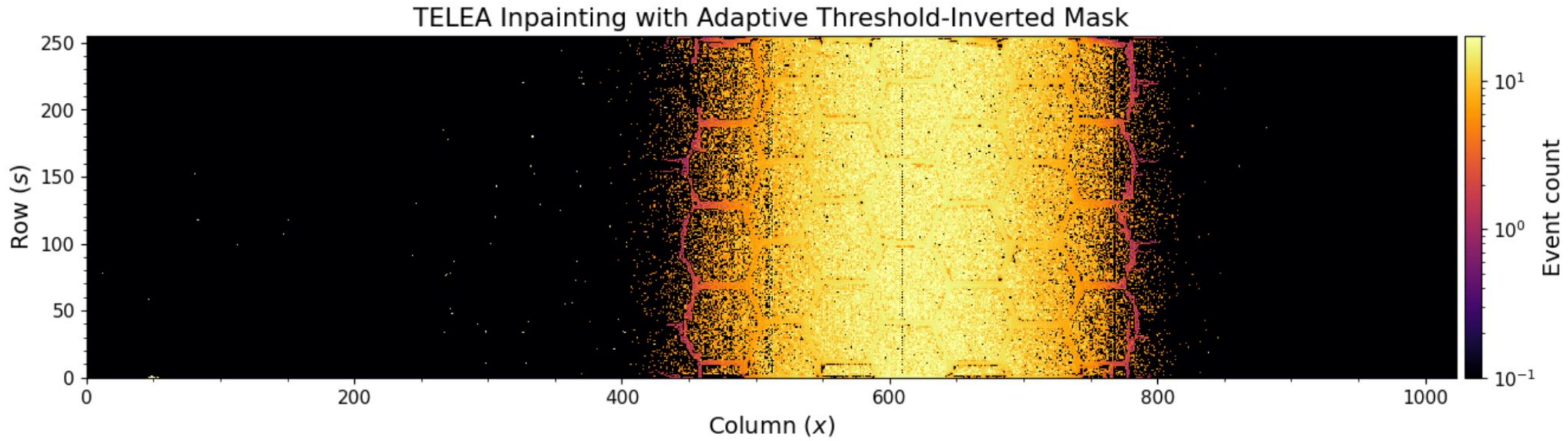


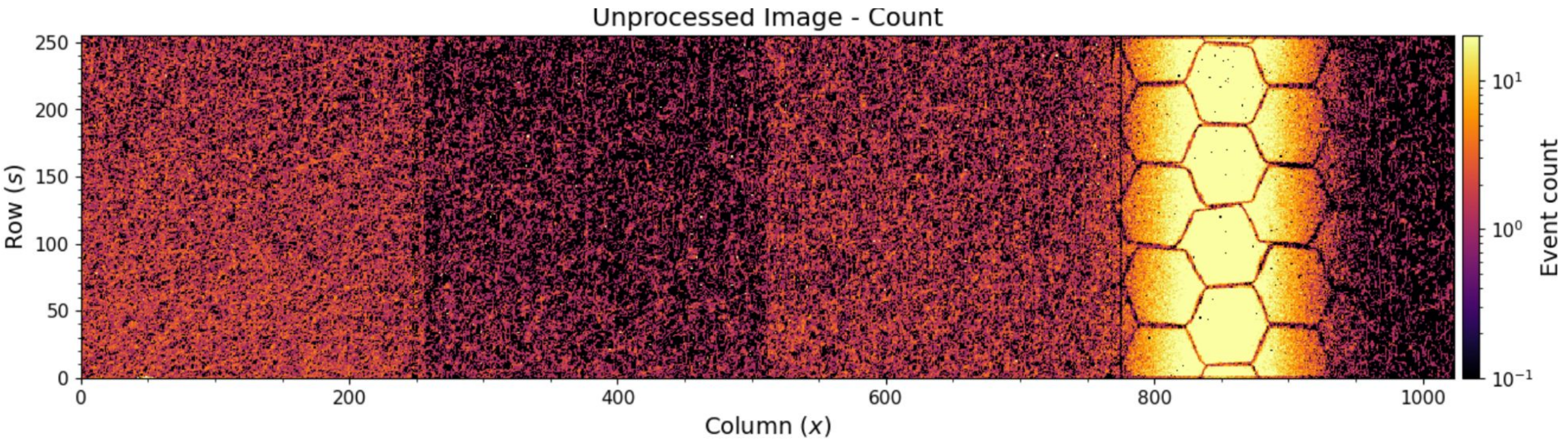
Unprocessed Image - Count
Row (s)
Column (x)
Event count

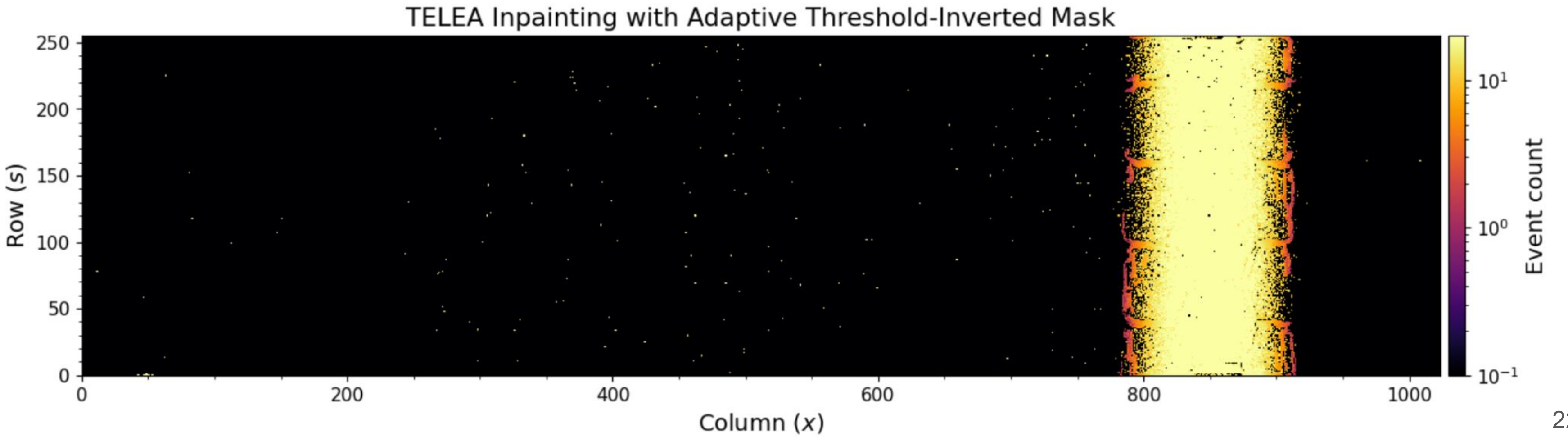


TELEA Inpainting with Adaptive Threshold-Inverted Mask
Row (s)
Column (x)
Event count

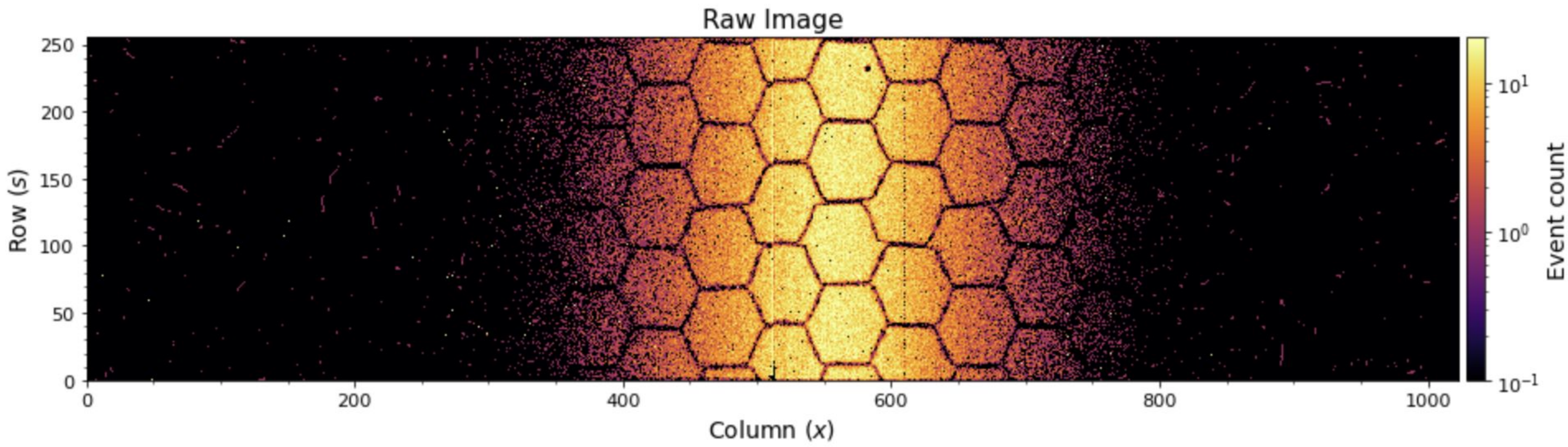

Unprocessed Image - Count
Row (s)
Column (x)
Event count
250
200
150
100
50
0
0
200
400
600
800
1000
10^1
10^0
10^{-1}

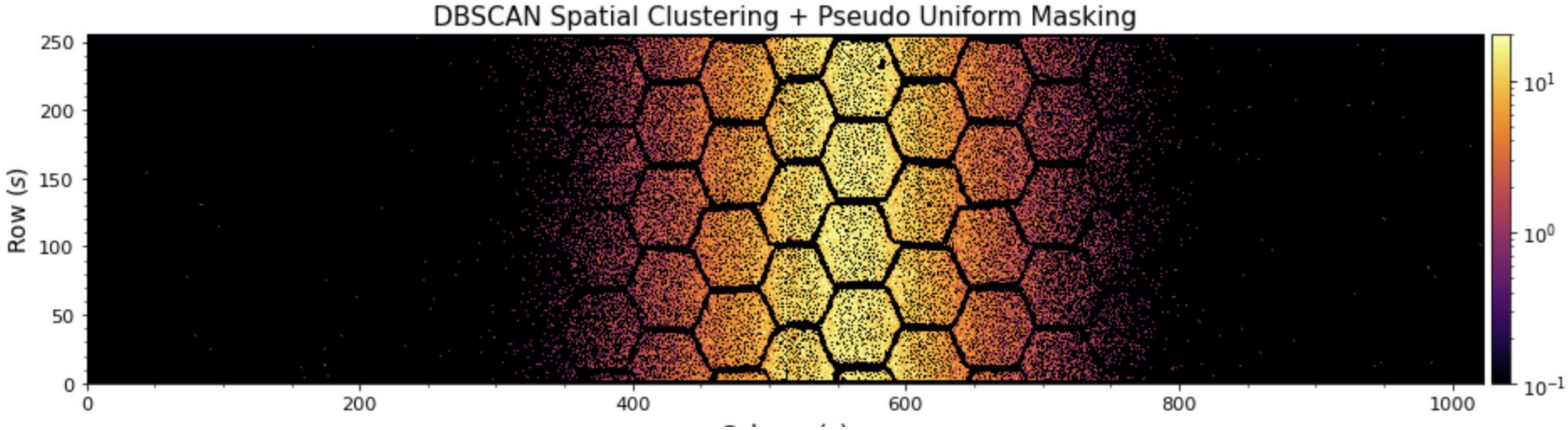

TELEA Inpainting with Adaptive Threshold-Inverted Mask
Row (s)
Column (x)
Event count
250
200
150
100
50
0
0
200
400
600
800
1000
10^1
10^0
10^{-1}

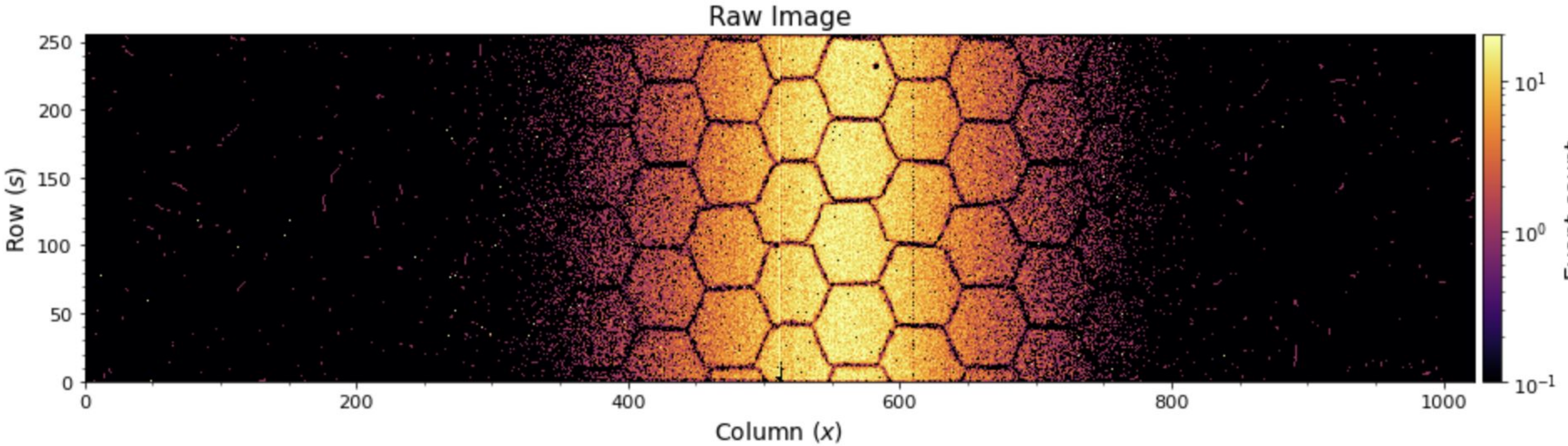

Unprocessed Image - Count
Row (s)
Column (x)
Event count
0
50
100
150
200
250
0
200
400
600
800
1000
10^{-1}
10^{0}
10^{1}

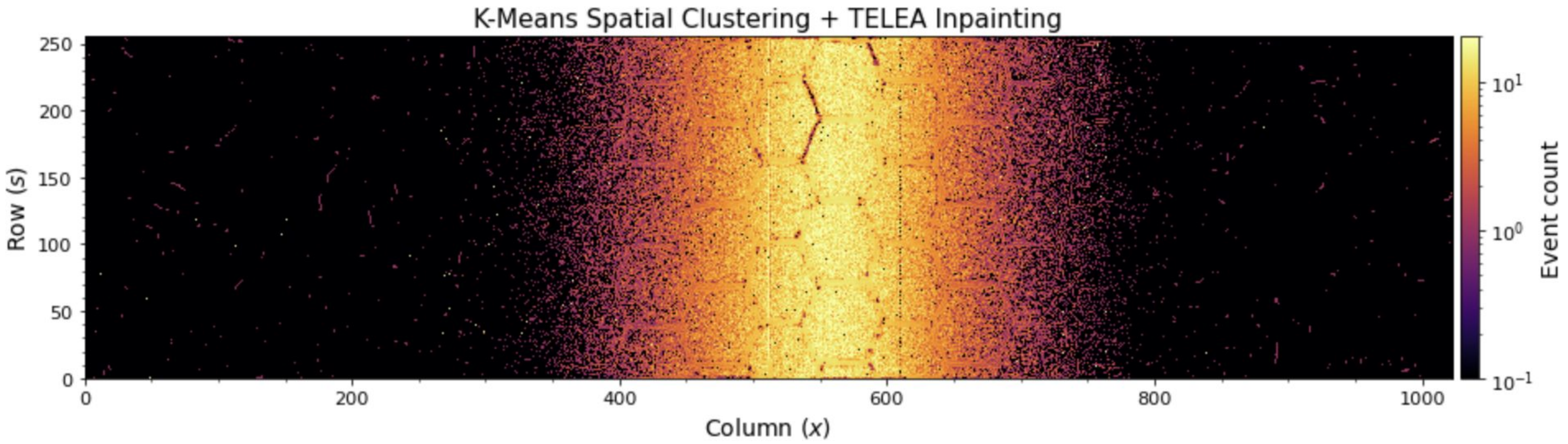

TELEA Inpainting with Adaptive Threshold-Inverted Mask
Row (s)
Column (x)
Event count
0
50
100
150
200
250
0
200
400
600
800
1000
10^{-1}
10^{0}
10^{1}

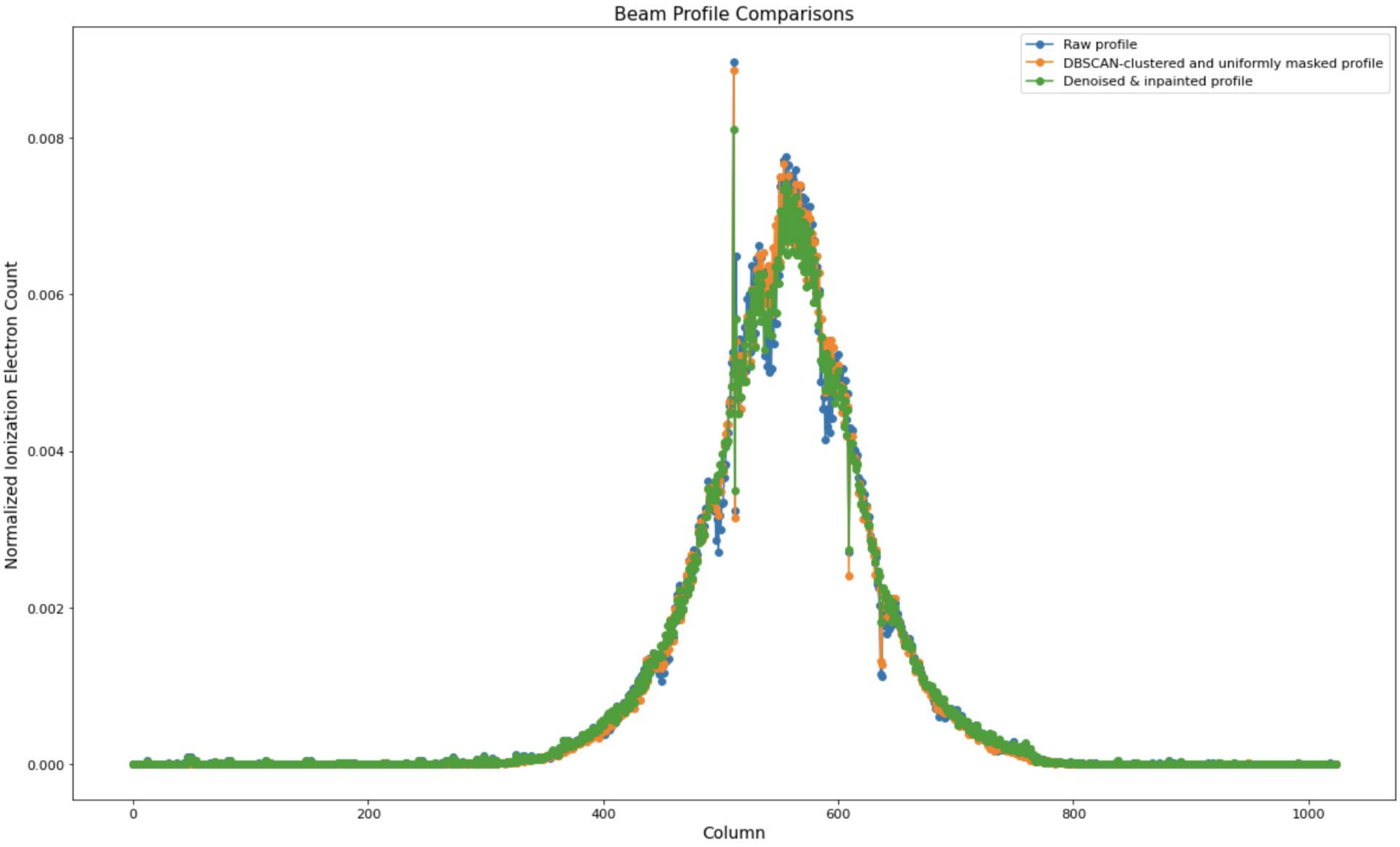
Comparing Beam Profiles

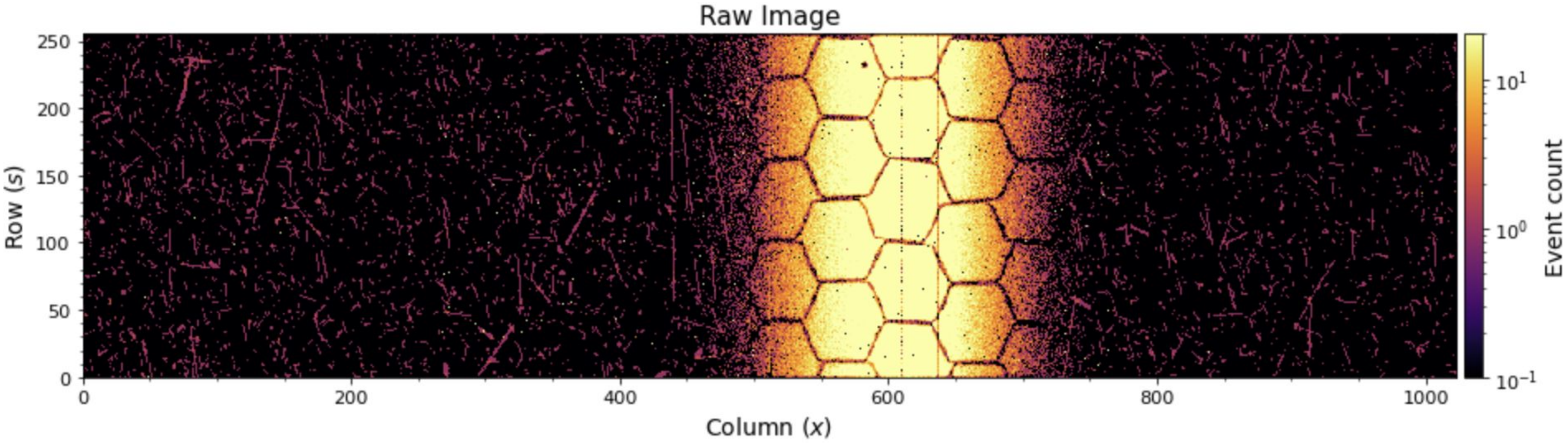

Raw Image
Row (s)
Column (x)
Event count
250
200
150
100
50
0
200
400
600
800
1000
10^1
10^0
10^{-1}

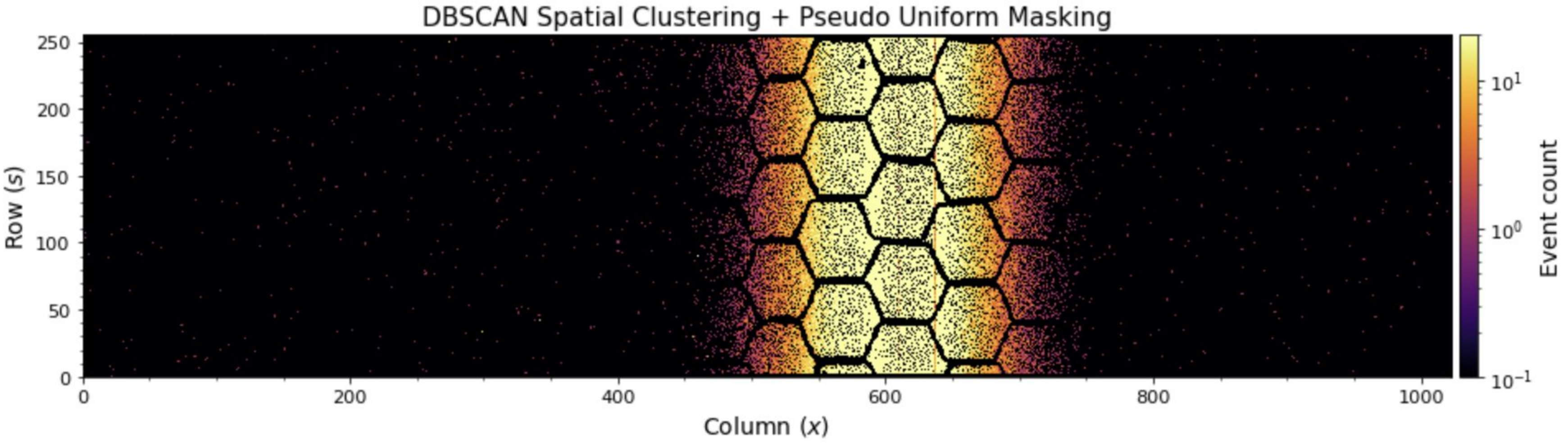

DBSCAN Spatial Clustering + Pseudo Uniform Masking
Row (s)
Event count
250
200
150
100
50
0
200
400
600
800
1000
10^1
10^0
10^{-1}

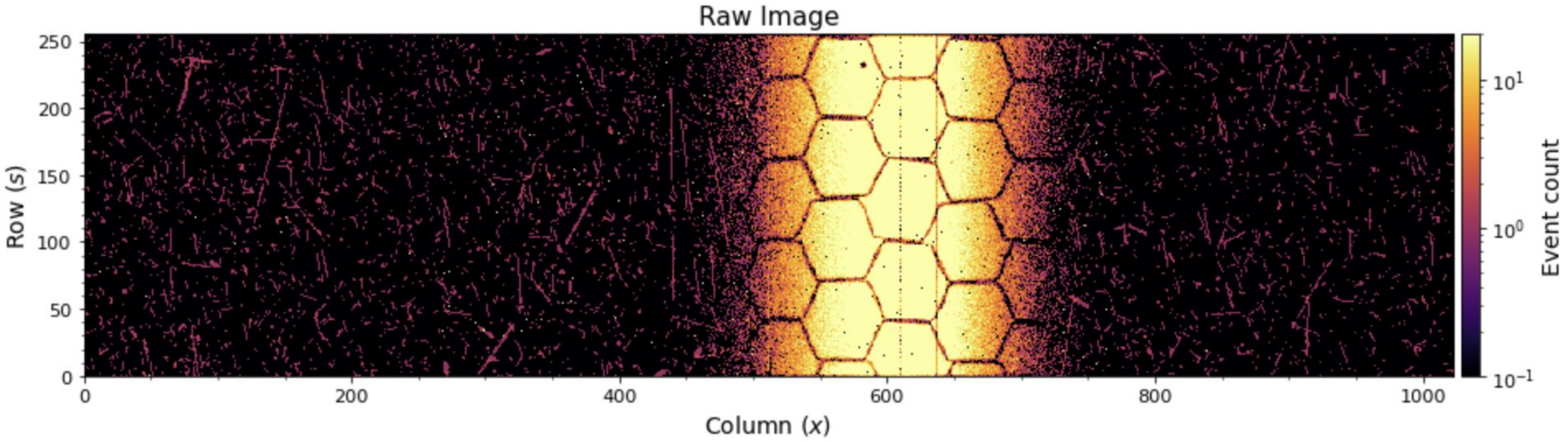

Raw Image
Row (s)
Column (x)
Event count

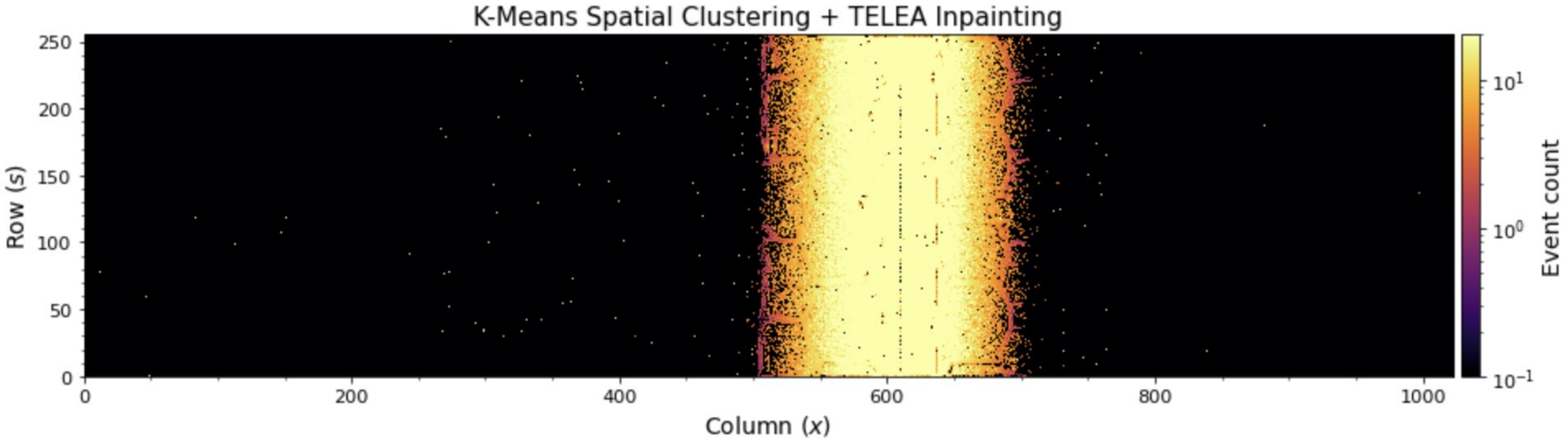

K-Means Spatial Clustering + TELEA Inpainting
Row (s)
Column (x)
Event count

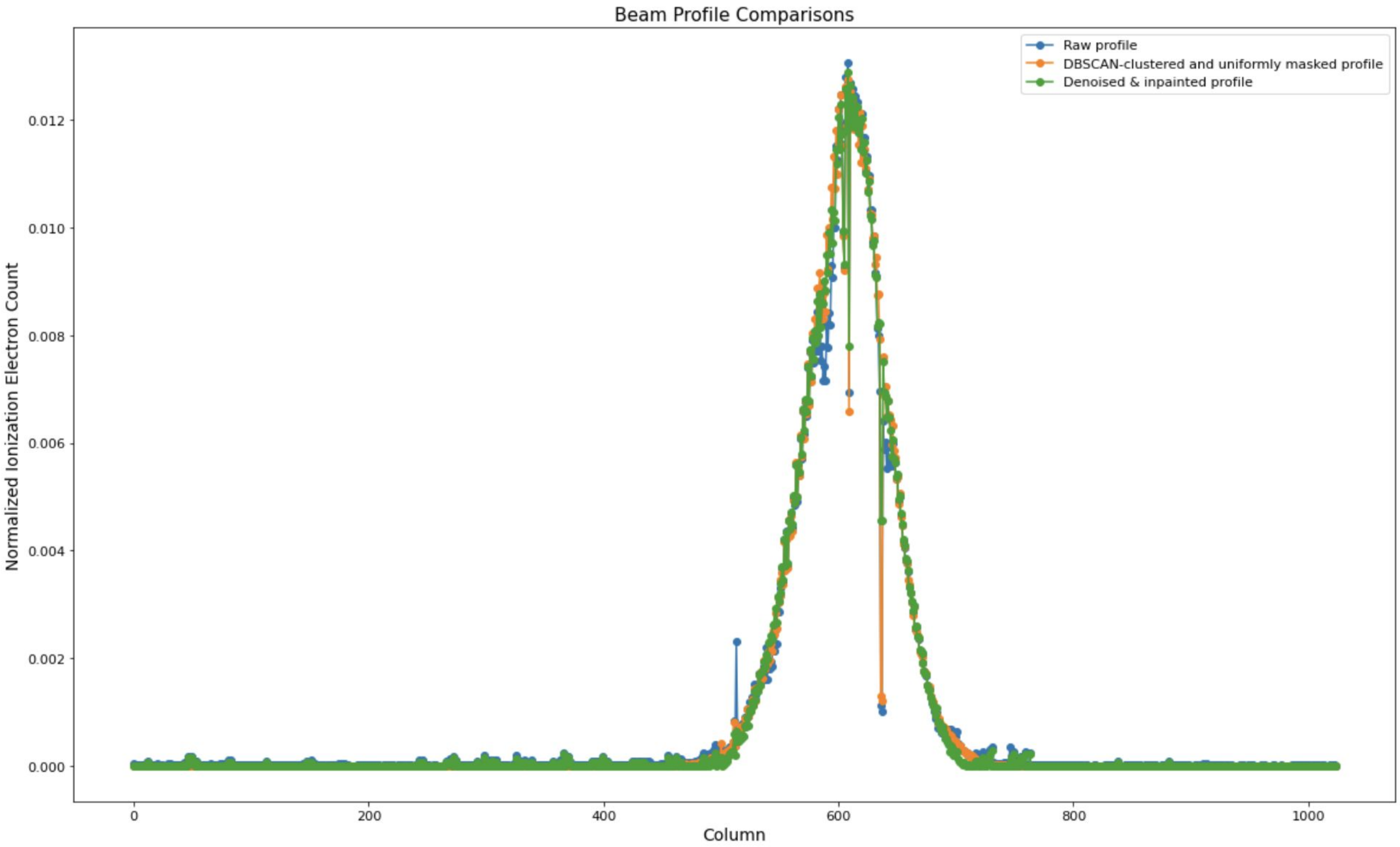

Beam Profile Comparisons
Raw profile
DBSCAN-clustered and uniformly masked profile
Denoised & inpainted profile
Normalized Ionization Electron Count
0.000
0.002
0.004
0.006
0.008
0
200
400
600
800
1000
Column


Raw Image
Row (s)
Column (x)
Event count
250
200
150
100
50
0
200
400
600
800
1000
10^1
10^0
10^{-1}

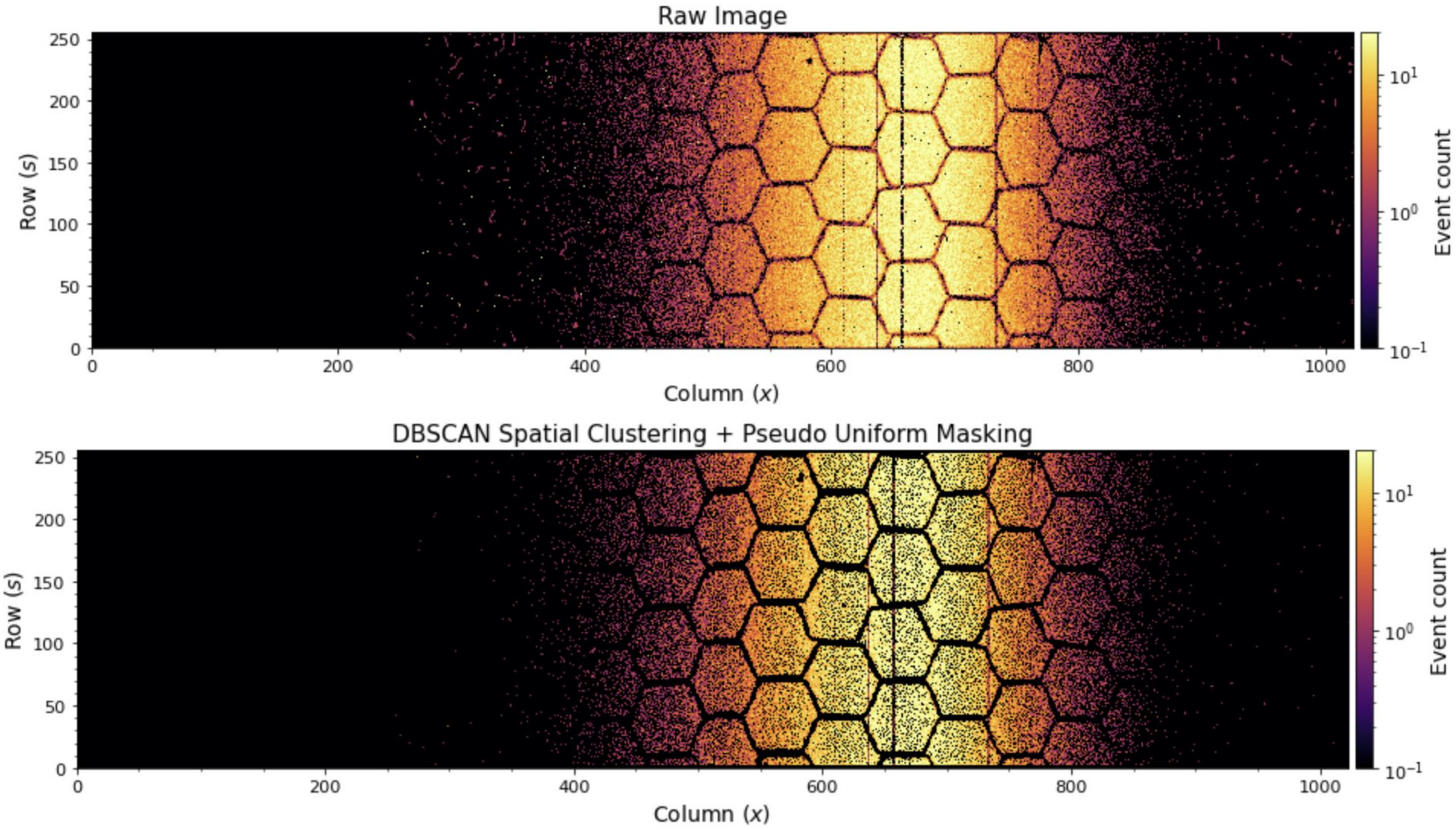

DBSCAN Spatial Clustering + Pseudo Uniform Masking
Row (s)
Column (x)
Event count
250
200
150
100
50
0
200
400
600
800
1000
10^1
10^0
10^{-1}

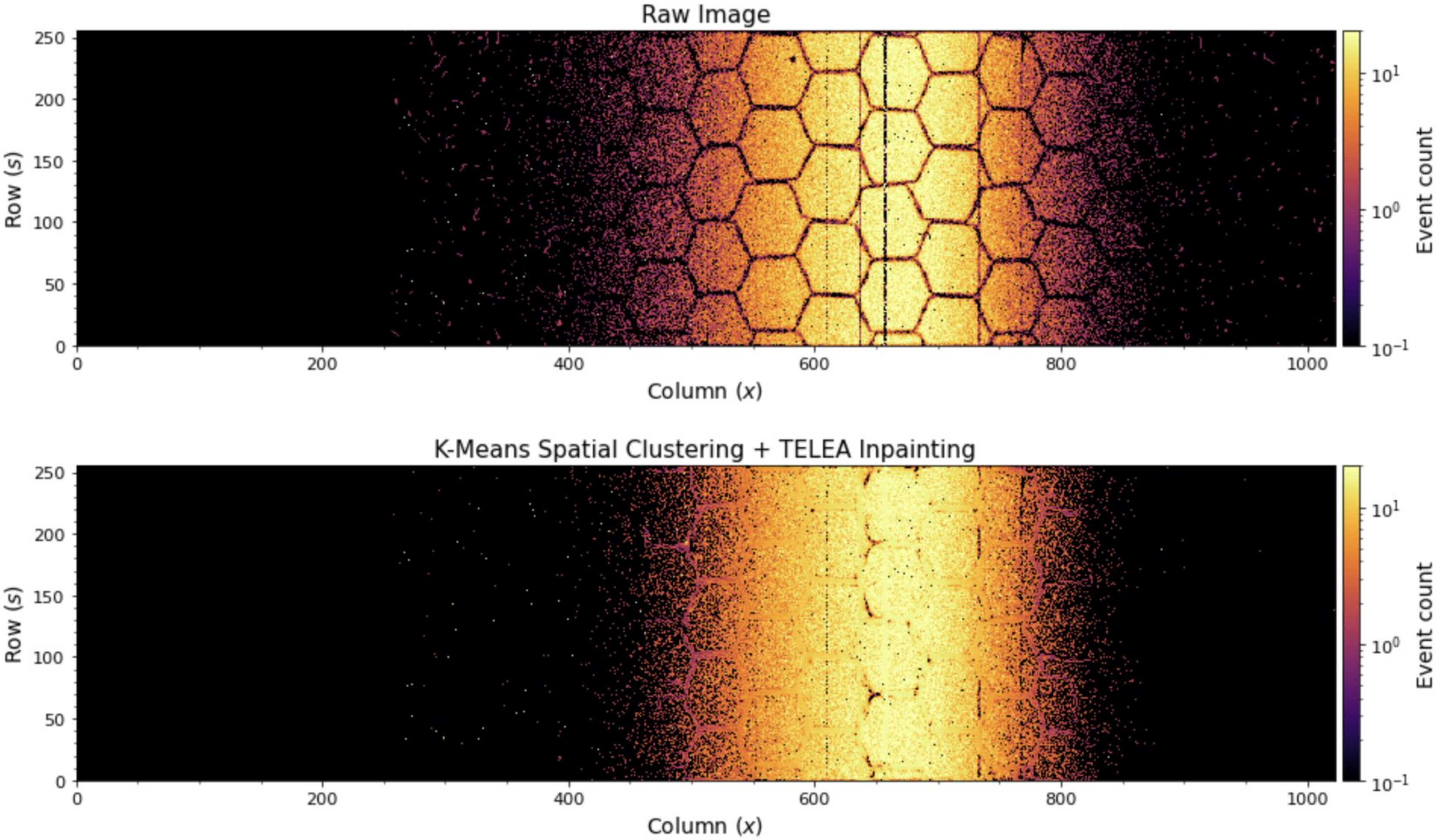

Raw Image
Row (s)
Column (x)
Event count
0
50
100
150
200
250
0
200
400
600
800
1000
10^{-1}
10^{0}
10^{1}

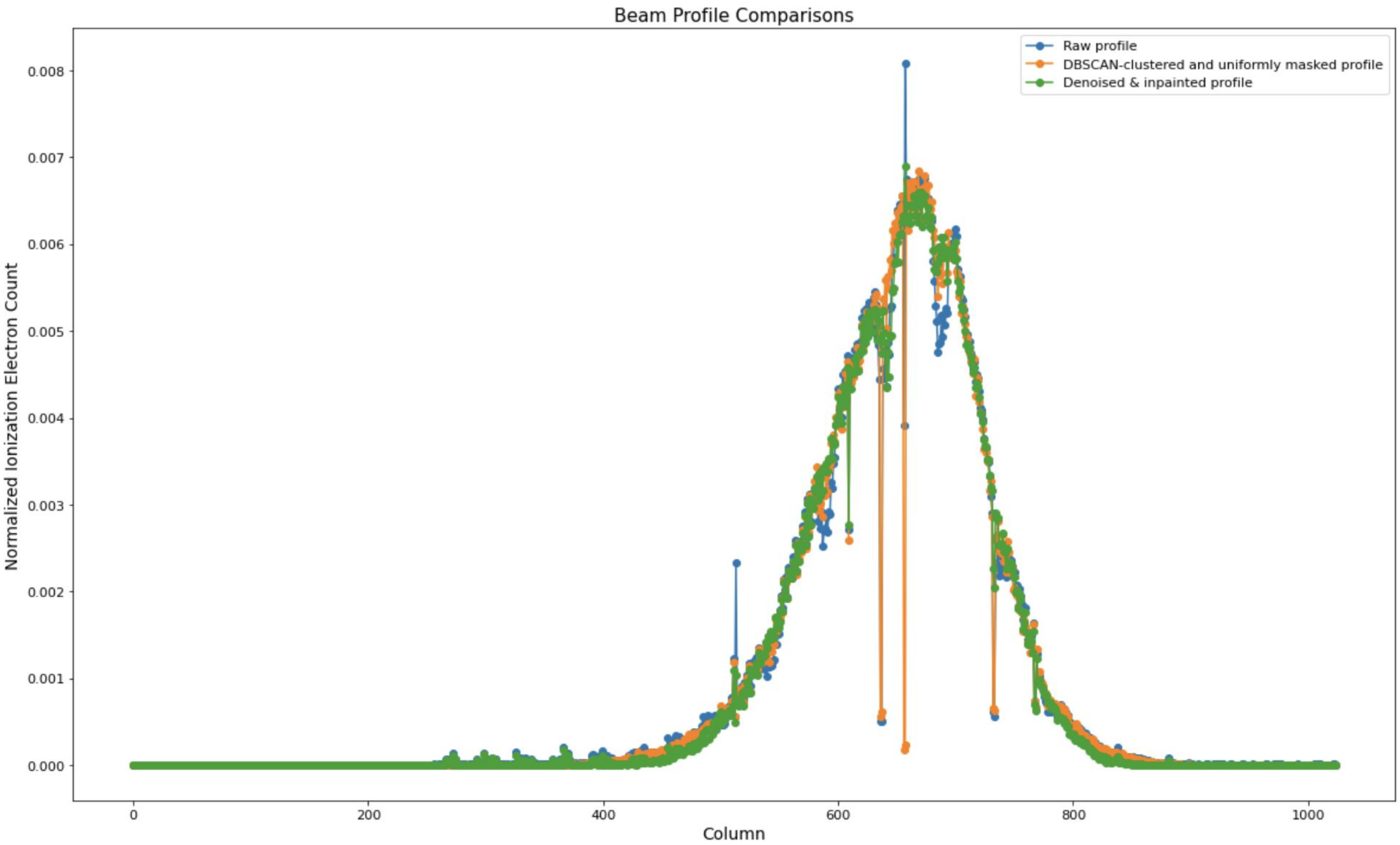

K-Means Spatial Clustering + TELEA Inpainting
Row (s)
Column (x)
Event count
0
50
100
150
200
250
0
200
400
600
800
1000
10^{-1}
10^{0}
10^{1}



Beam Profile Comparisons
Raw profile
DBSCAN-clustered and uniformly masked profile
Denoised & inpainted profile
Normalized Ionization Electron Count
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
200
400
600
800
1000
Column


Raw Image
Row (s)
Column (x)
Event count
0
50
100
150
200
250
200
400
600
800
1000
10^{1}
10^{0}
10^{-1}
DBSCAN Spatial Clustering + Pseudo Uniform Masking
Row (s)
Column (x)
Event count


Raw Image
Row (s)
Column (x)
Event count
K-Means Spatial Clustering + TELEA Inpainting
Row (s)
Column (x)
Event count



Beam Profile Comparisons
Raw profile
DBSCAN-clustered and uniformly masked profile
Denoised & inpainted profile
Normalized Ionization Electron Count
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
0
200
400
600
800
1000
Column

In Summary

- We proposed a two stage pipeline that efficiently performs spatial clustering and inpainting to correct the RF shield
- We have published this package on Acc-Py (bgi-denoise)
- We have deployed this tool via UCAP (in testing phase right now)
- Idea is that file is saved -> processed immediately -> profile(s) visualized all within attention span (ideally as fast as possible)

Next Steps

- Temporal, spatial clustering and RF-shield correction can be done by neural networks (e.g. autoencoder, RIDNet, Visual Transformer)
- Neural networks can be GPU-accelerated
- Such a network would need to be trained in a supervised manner
- Such a network would require large amounts of data
- Without any idea of “ground truth”, one could use the results of the two-stage pipeline as a “ground truth”