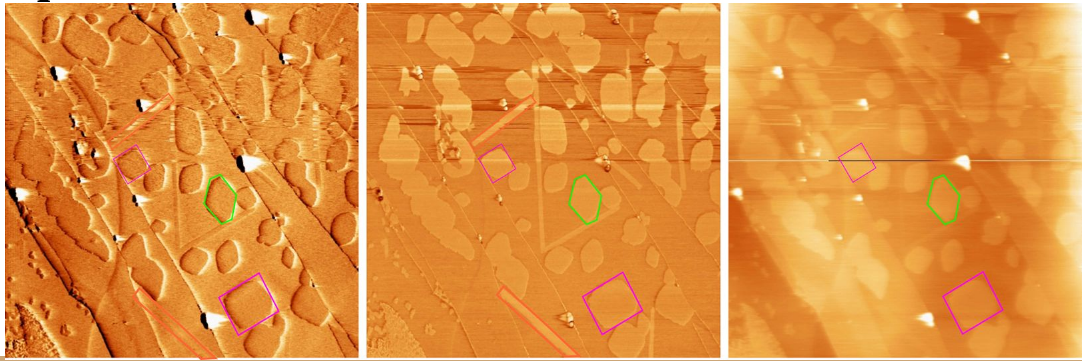
Enhancement and Feature Analysis of AFM Images

-CHM392A (UGPII)

Aniket Sandhan
Aakriti Gupta
Reeteswaroop Singh

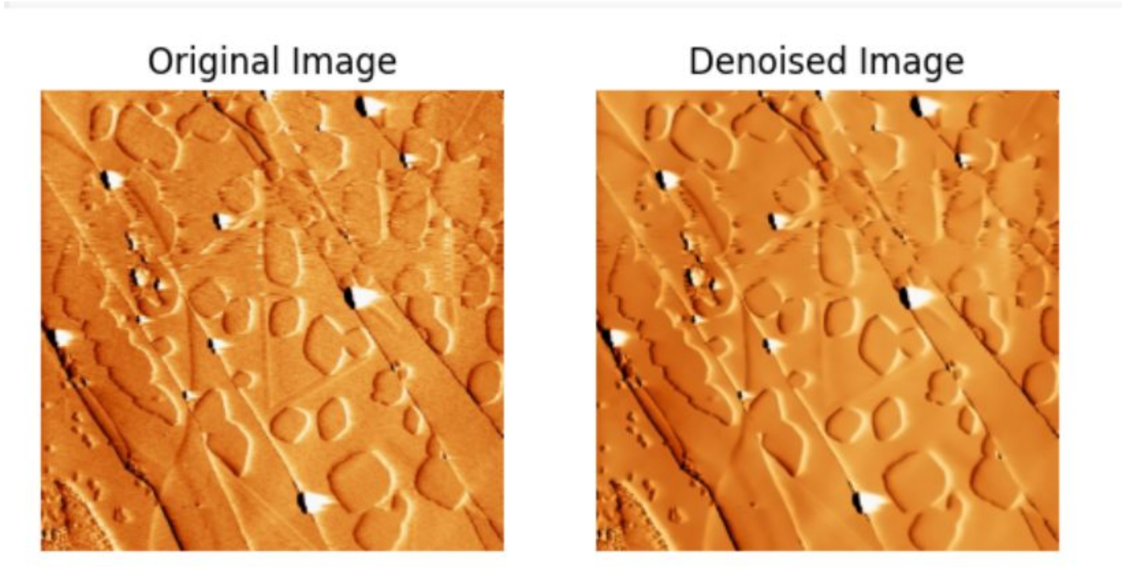
Problem statement

Different types of assembles islands of a molecule are observed in the AFM images. They are depicted by different color as follows: Hexagonal islands (green), Oblique islands (magenta), 1D islands (orange). We measure three channels in the AFM images (phase, amplitude and topography). Though topography is very useful, due to large height variations, the image quality is not very good. So, we dependent mostly on phase and amplitude images, where the islands are more clearly visible. In some cases, phase contrast can also distinguish different types of arrangements of molecules on surface. What we are looking for are the removal of noise, line like noise along the width of the image and recognising shapes of islands present.



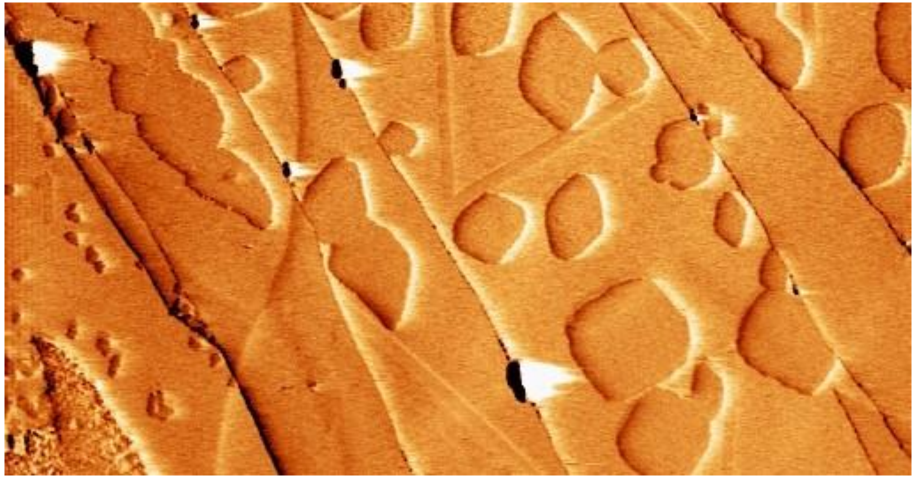
fastNlMeansDenoisingColored()

- Used fastNlMeansDenoisingColored which worked fine on Amplitude Images only.



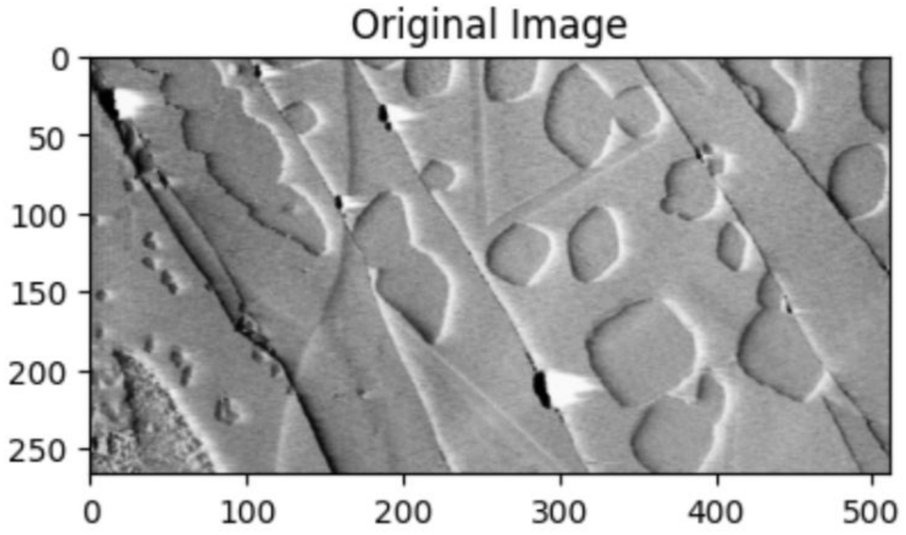
Edge Detection

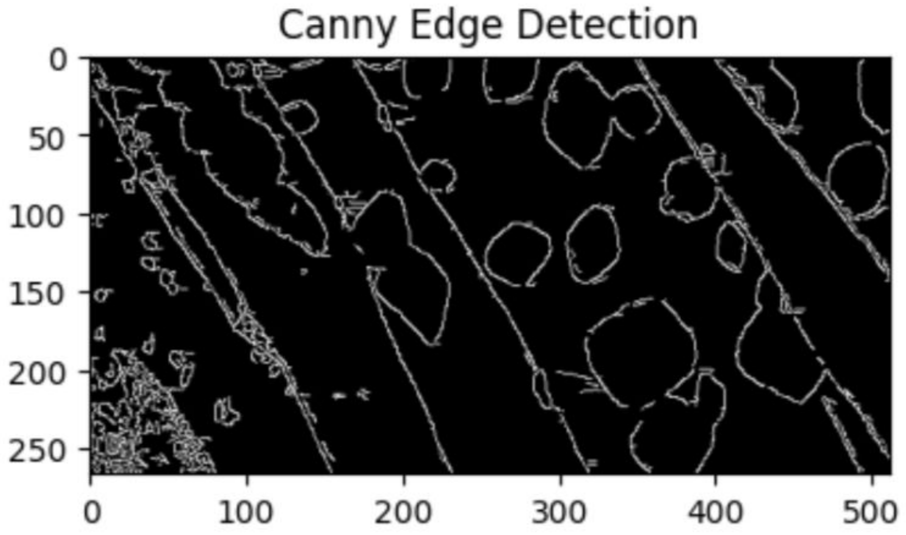
- Applied Canny Edge Detection on cropped denoised ‘1_amplitude’ image (below).



- After settings and experimenting with different sets of threshold values the output we got is in the upcoming slide.

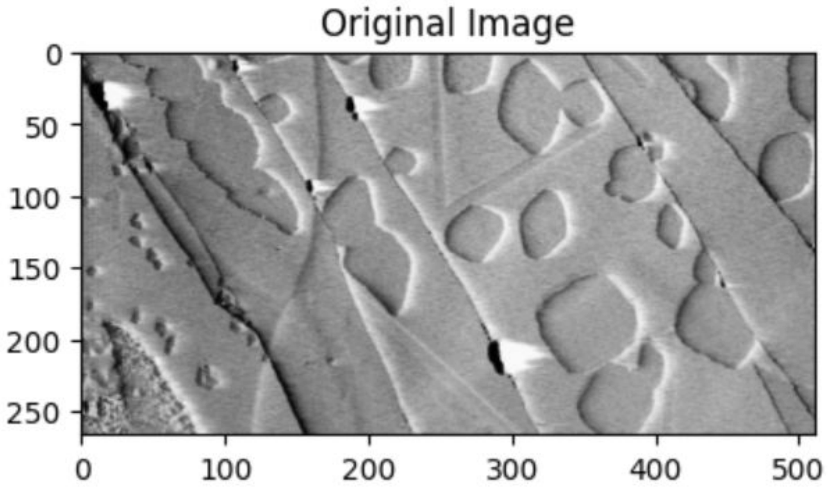
Canny Edge Detection Output

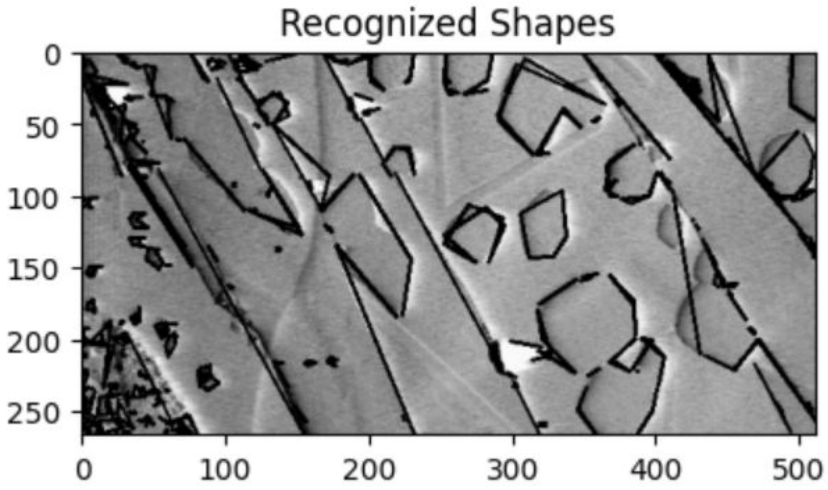




Contour Detection

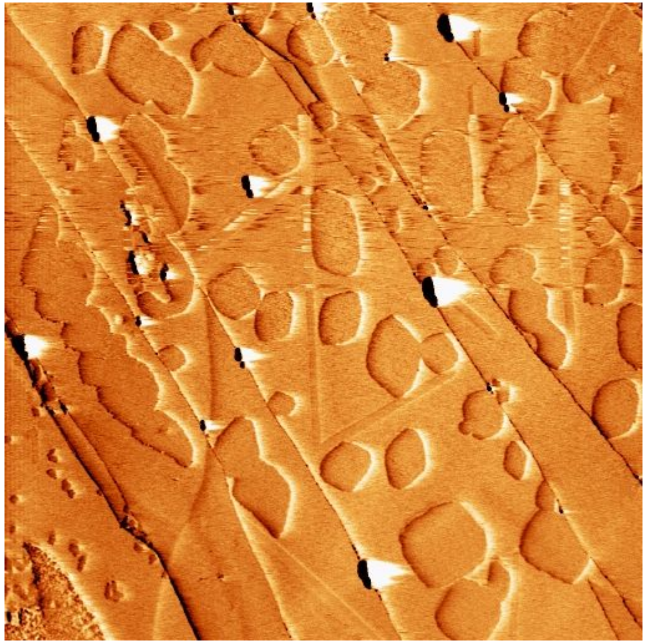
- Used Contour Detection and extracted number of hexagonal and oblique islands present.
- The output gave the sharp boundaries required to know the relative orientation and shapes.

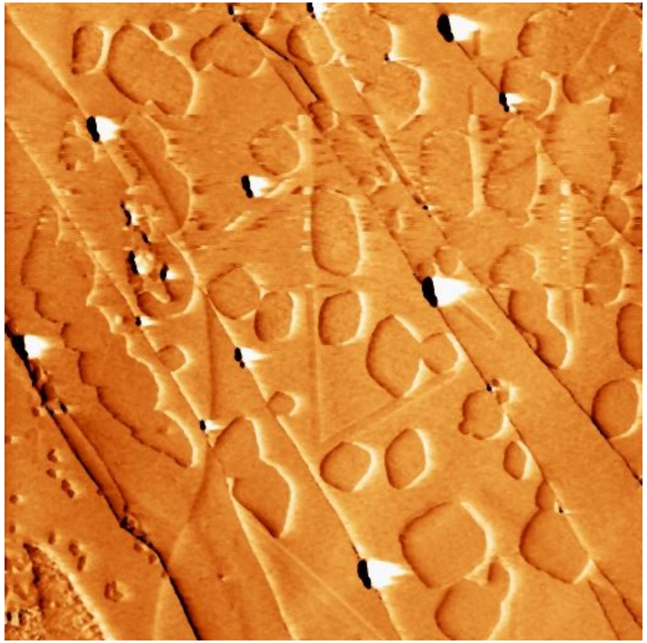




```
Number of Hexagon Islands:  31
Number of Oblique Islands:  180
```

Median Filter (Amplitude Image)





Fourier Transform

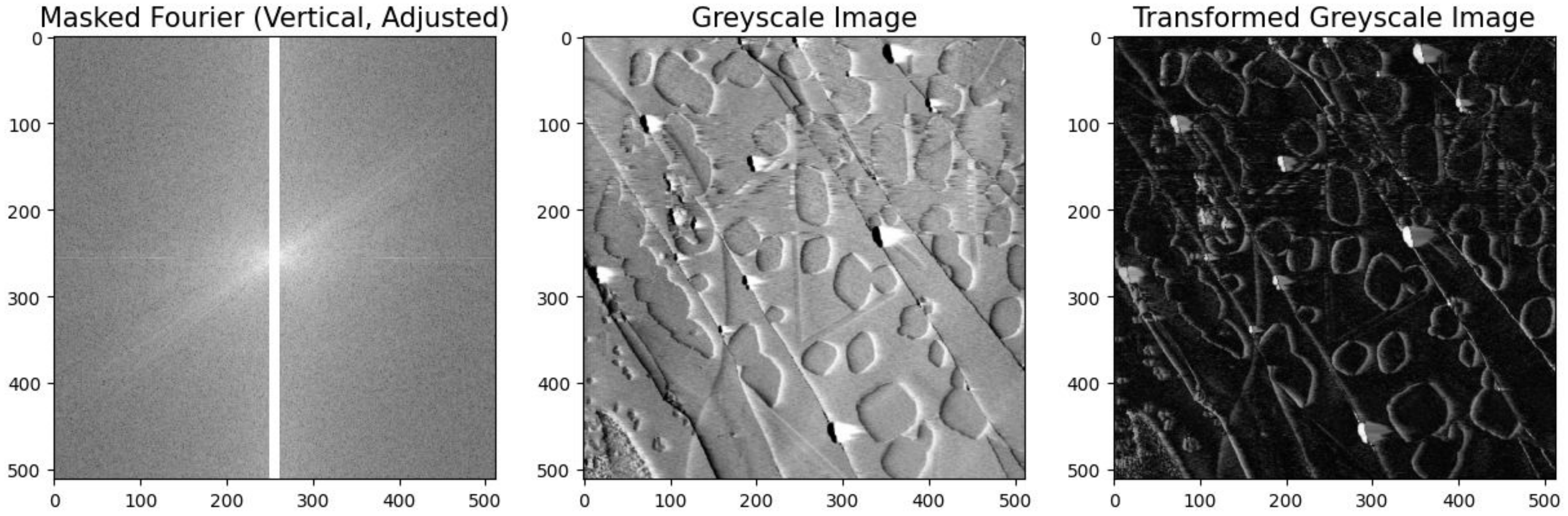
- A physical process can be described either in the *time domain* or *frequency domain,* which can be represented as a function of time *t*, i.e., *x(t)* and a function of *frequency, f* or *angular frequency,ω* (*ω*=2*πf*), i.e., *X(ω)*, respectively. The two representations of the the functions can transfer back and forth by means of the *Fourier transform* equations,
- Forward Fourier Transform: **Analysis Equation**

$$X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt$$

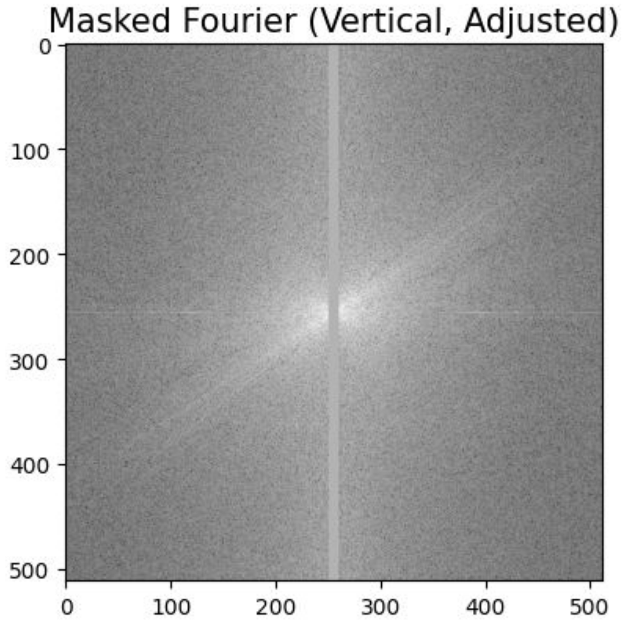
- Inverse Fourier Transform: **Synthesis Equation**

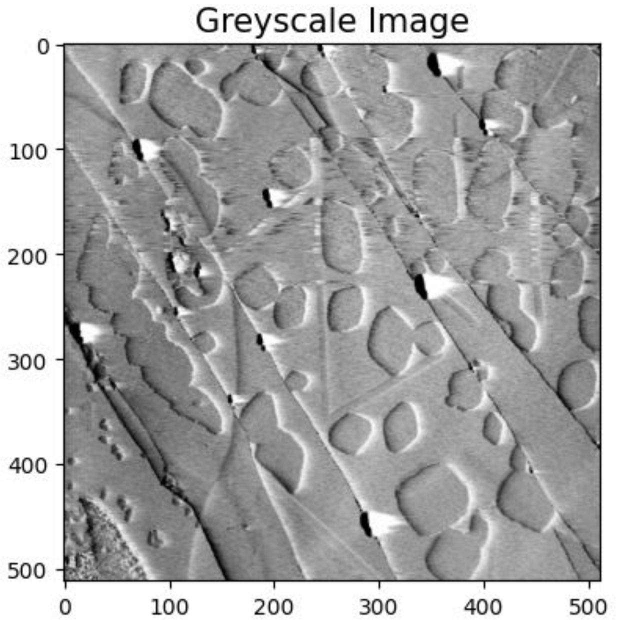
$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega)e^{j\omega t}d\omega$$

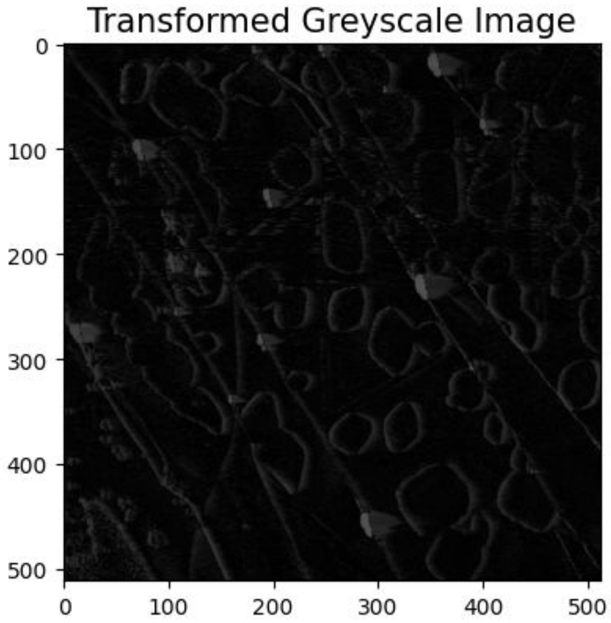
Fourier Transform with Sharp Vertical Mask



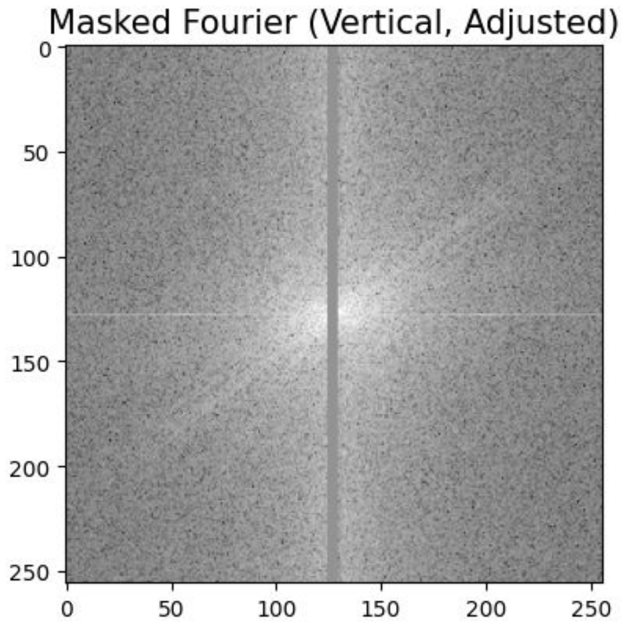
Fourier Transform with Smooth Vertical Mask

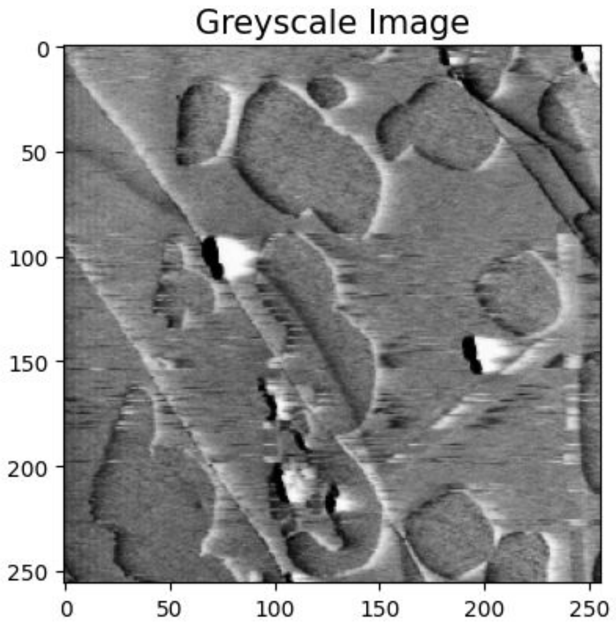


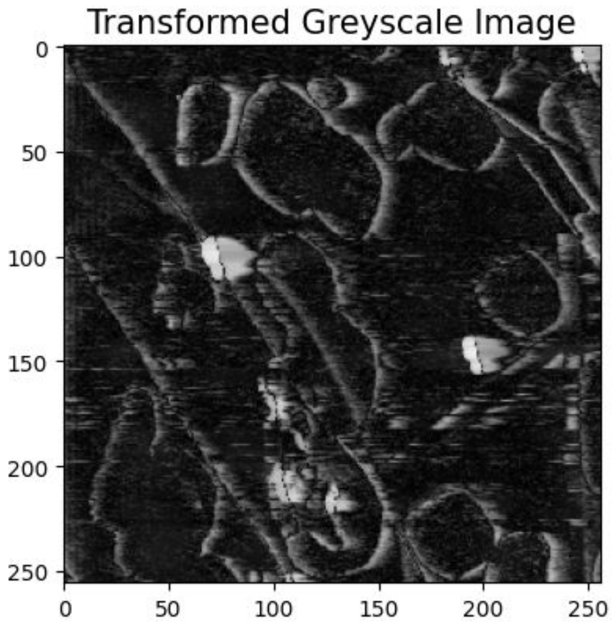




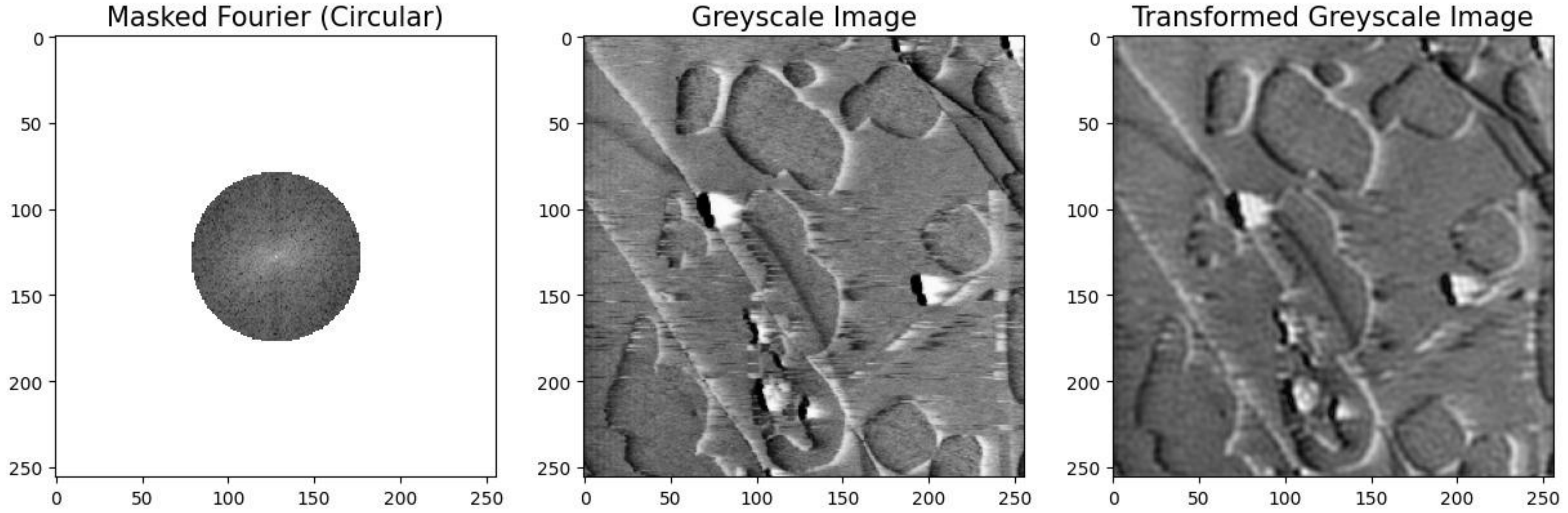
Fourier Transform in Patches (Vertical Mask)

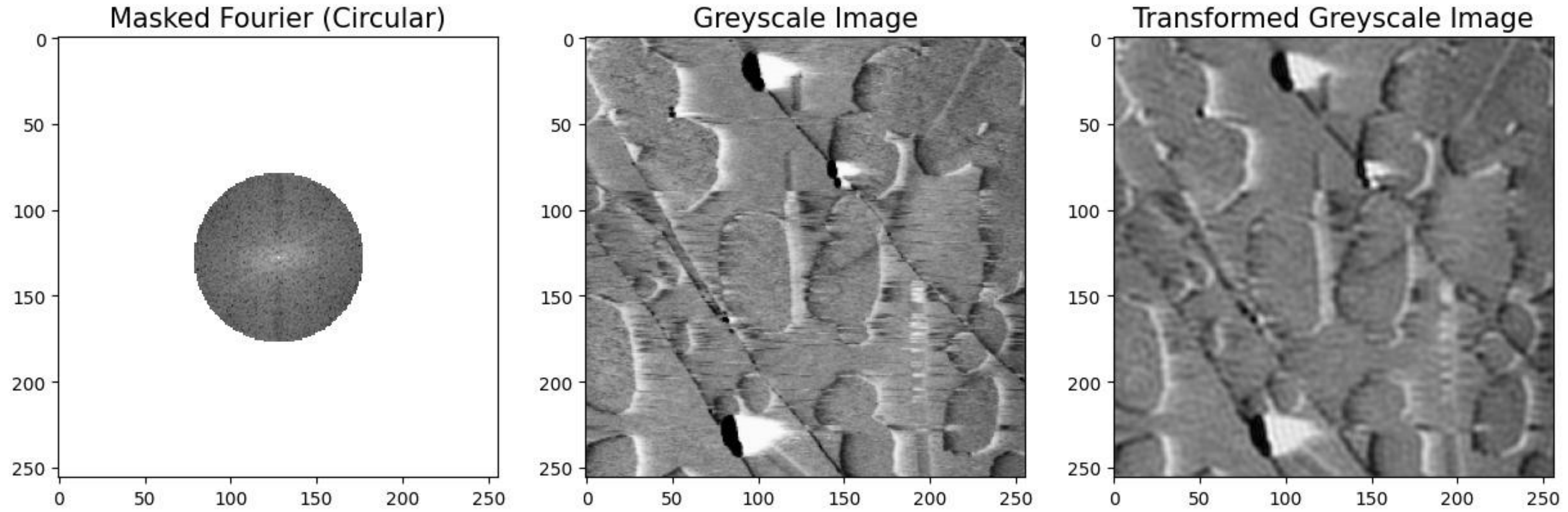


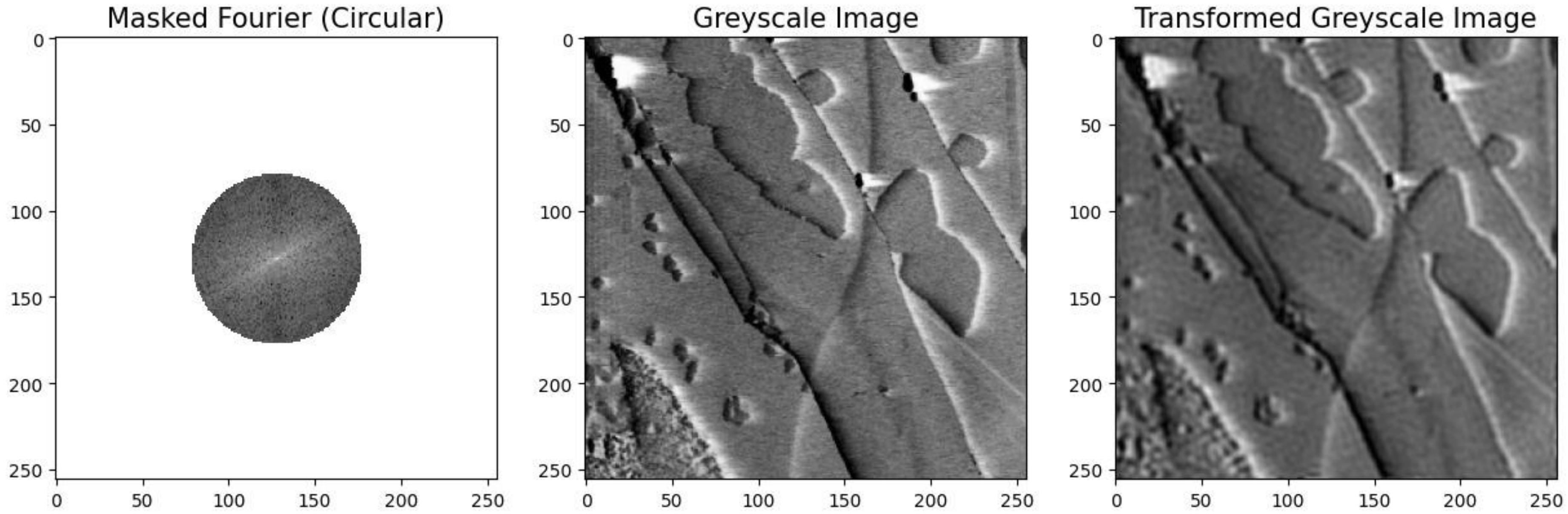


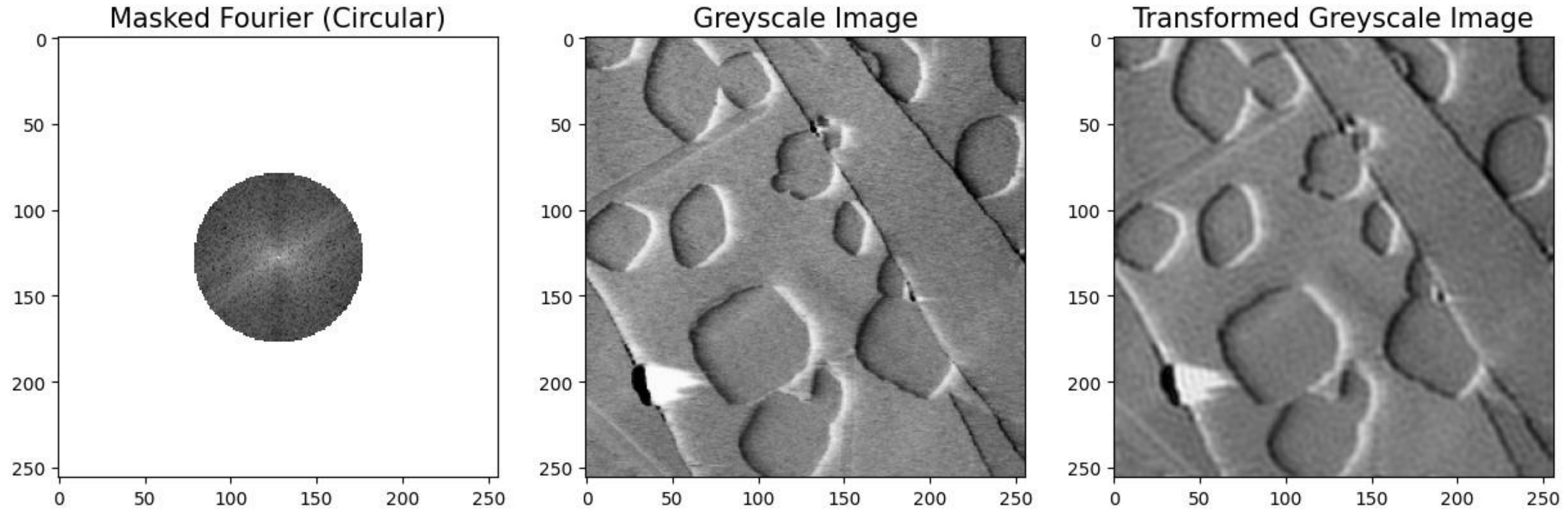


Fourier Transform in Patches (Low Pass Filter)




Masked Fourier (Circular)
Greyscale Image
Transformed Greyscale Image
0
50
100
150
200
250


Masked Fourier (Circular)
Greyscale Image
Transformed Greyscale Image
0
50
100
150
200
250


Masked Fourier (Circular)
Greyscale Image
Transformed Greyscale Image
0
50
100
150
200
250

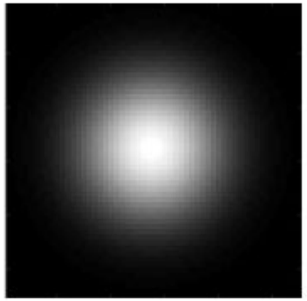
Bilateral Filter

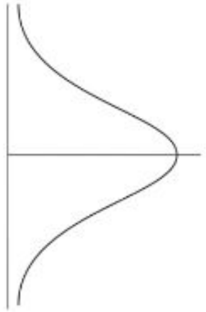
- Bilateral filtering is a technique to smooth images while **preserving edges**.

- In bilateral filter, each pixel is replaced by a **weighted average of its neighbors**.

- The bilateral filter has two key parameters: the **spatial domain parameter**, which controls the spatial extent of the filter, and the **intensity domain parameter**, which controls the influence of intensity differences on the filter weights. Adjusting these parameters allows for fine-tuning the filter's behavior based on the specific characteristics of the image and the desired level of denoising.

$$BF[I]_p = \underbrace{\frac{1}{W_p}}_{\text{Normalization Factor}} \sum_{q \epsilon S} \underbrace{G_{\sigma_s}(||p-q||)}_{\text{Space Weight}} \underbrace{G_{\sigma_r}(|I_p - I_q|)}_{\text{Range Weight}} I_q$$

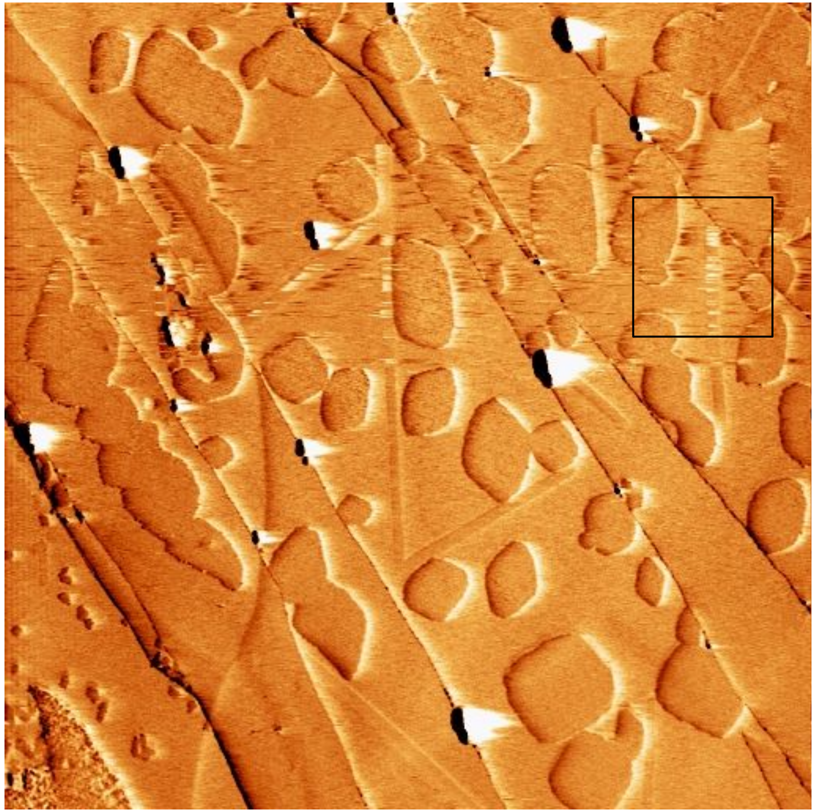


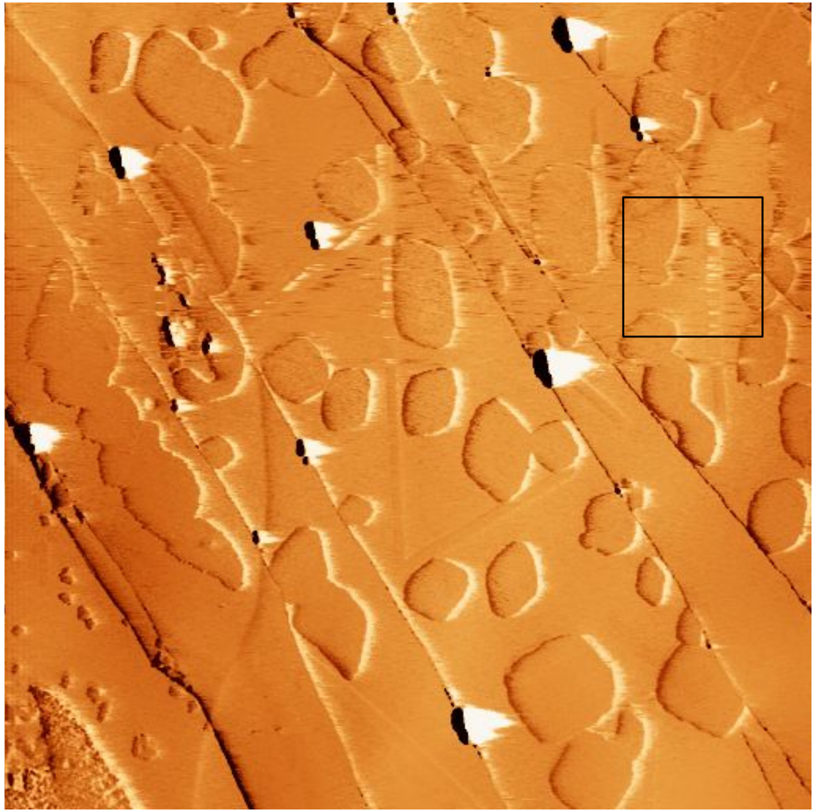


OpenCV has a function called **bilateralFilter()** with the following arguments:

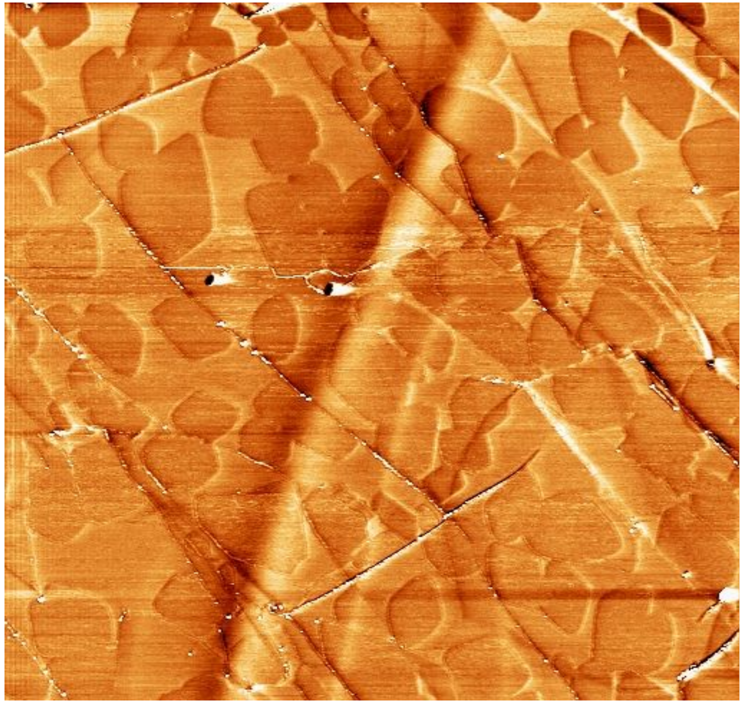
1. **d:** Diameter of each pixel neighborhood.
2. **sigmaColor:** Value of in the color space. The greater the value, the colors farther to each other will start to get mixed.
3. **sigmaSpace:** Value of in the coordinate space. The greater its value, the more further pixels will mix together, given that their colors lie within the sigmaColor range.

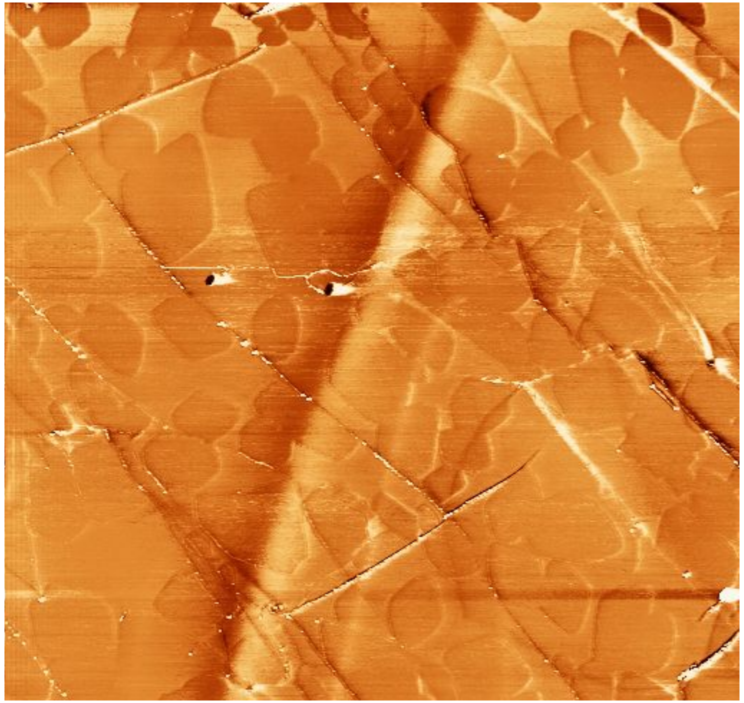
Bilateral Filter (Amplitude image)



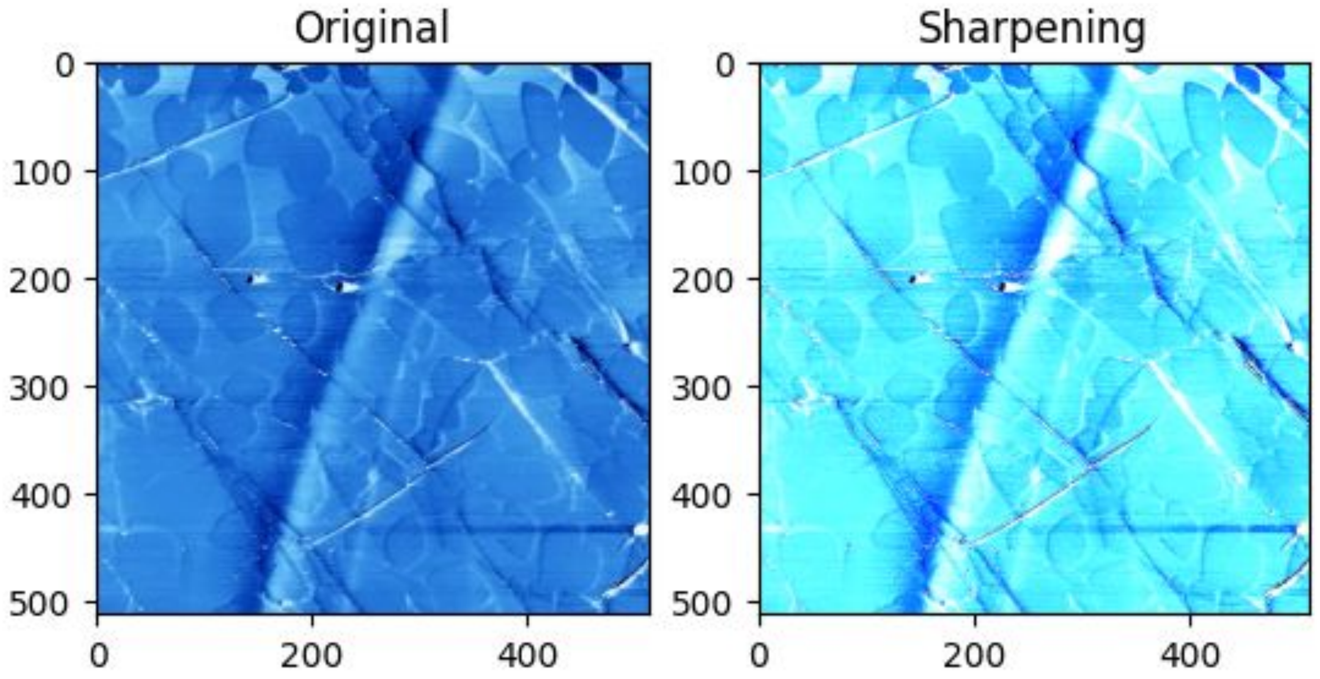


Bilateral Filter (Phase Image)

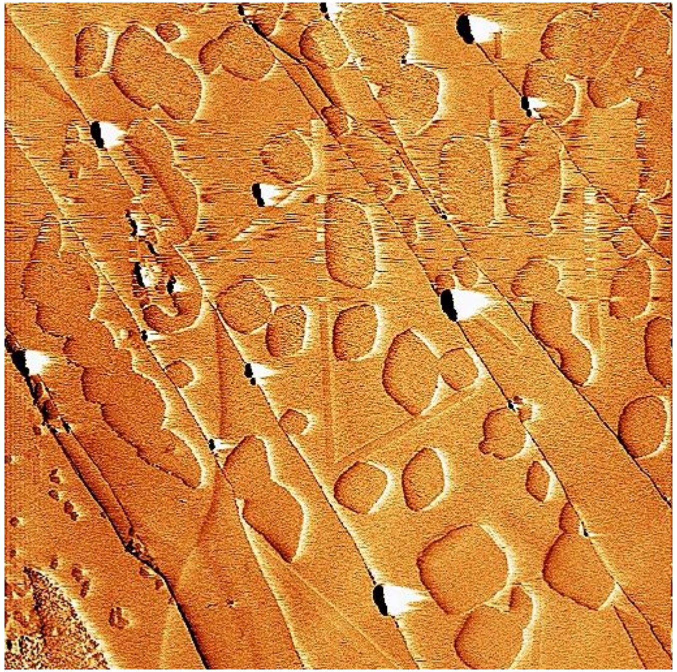


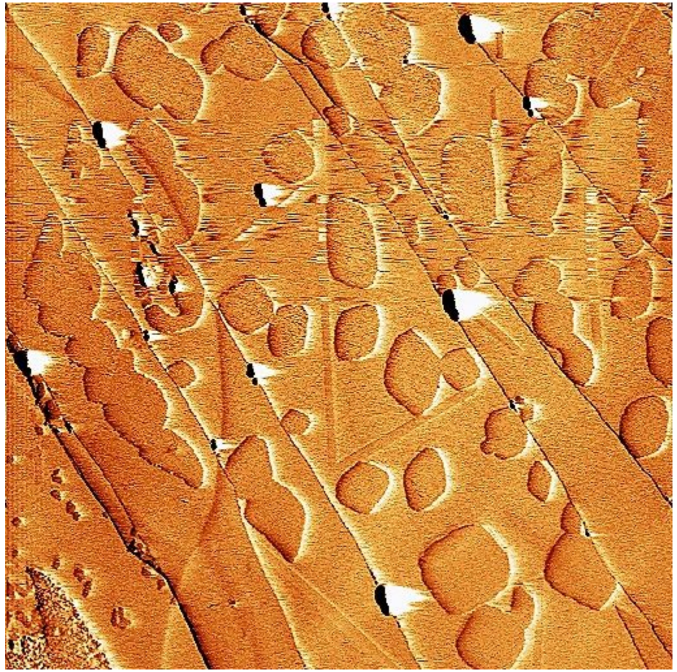


Bilateral Filter —› Edge Sharpening



Edge Sharpening —› Bilateral Filter





BM3D (Block Matching & 3D Filtering)

Architecture of the Algorithm

The algorithm is divided in two major steps:

1. The first step estimates the denoised image using hard thresholding during the collaborative filtering. Parameters in this step are denoted by the exponent hard

2. The second step is based both on the original noisy image, and on the basic estimate obtained in the first step. It uses Wiener filtering. The second step is therefore denoted by the exponent wiener.

BM3D

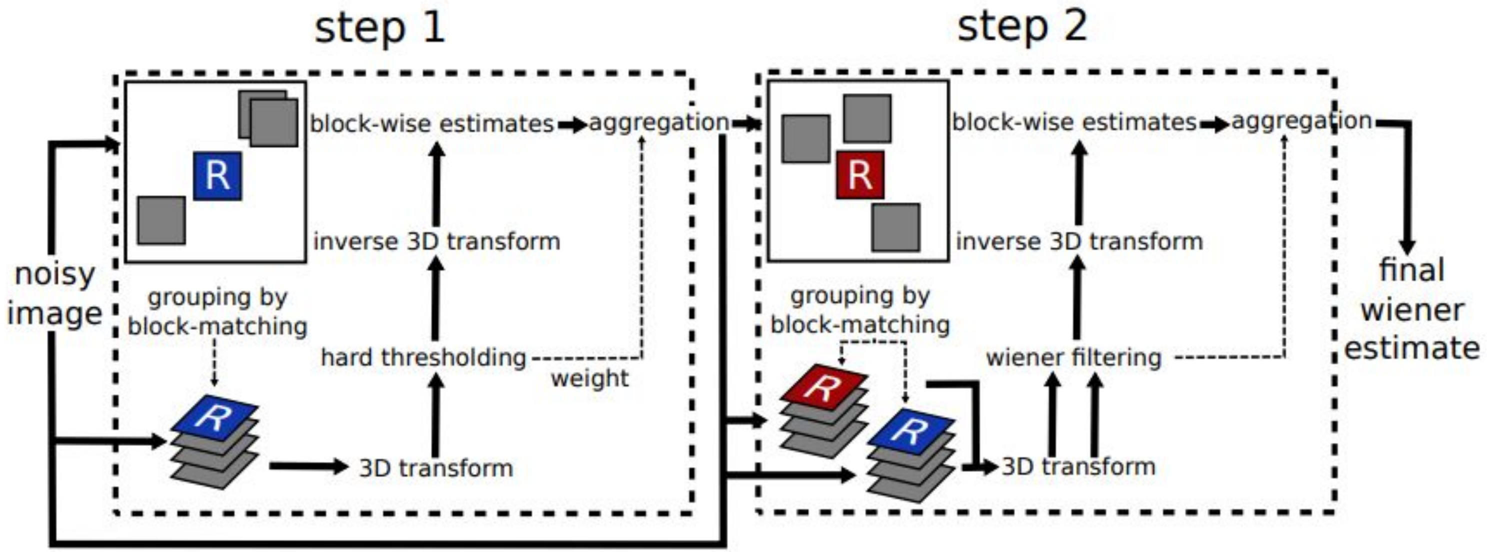
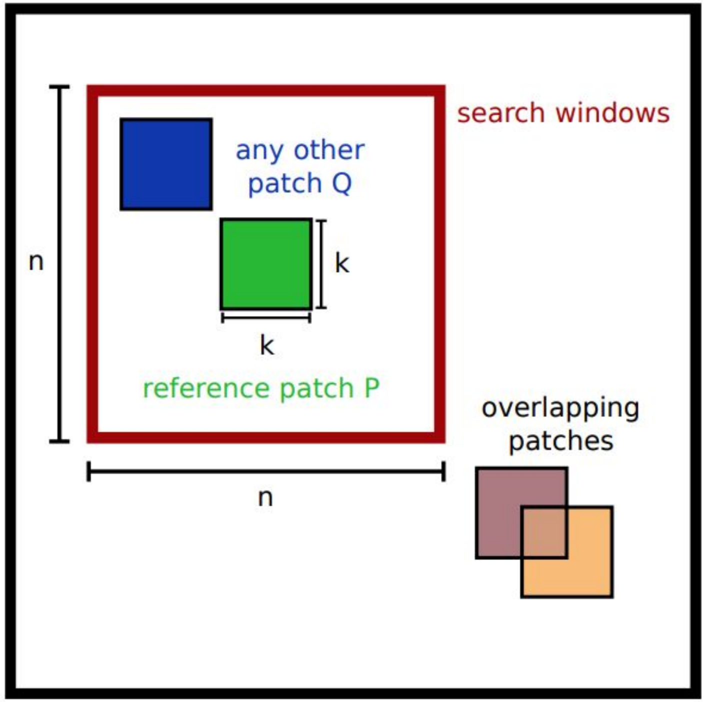


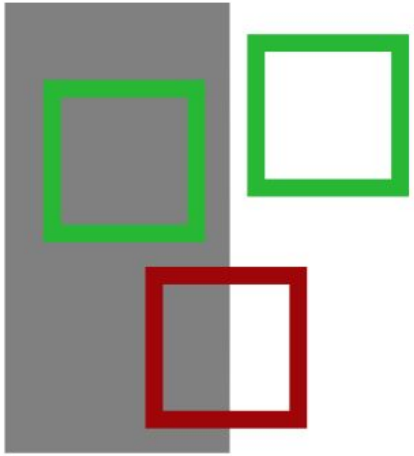
Figure 1: Scheme of the BM3D algorithm.

BM3D (Grouping)



The first sub-step is grouping. The original noisy image u is searched in a P-centered n(hard) × (n hard) neighborhood for patches Q similar to the reference patch P

BM3D



Green patches will have a weight superior to the red patch, because they are more sparse (have less nonzero transform coefficients)

- Collaborative Filtering: Applies 3D isometric linear transform followed by shrinkage of transform spectrum and inverse linear transform to estimate each patch.

$$\mathbb{P}(P)^{\mathbf{hard}} = \tau_{3D}^{\mathbf{hard}^{-1}}(\gamma(\tau_{3D}^{\mathbf{hard}}(\mathbb{P}(P))))$$

where γ is a hard thresholding operator with threshold $\lambda_{3D}^{\mathbf{hard}}\sigma$:

$$\gamma(x) = \begin{cases} 0 & \text{if} \quad |x| \leq \lambda_{3D}^{\mathbf{hard}}\sigma \\ x & \text{otherwise} \end{cases}$$

- Aggregation: Combines estimates for each pixel, giving priority to homogeneous patches. Uses weighting to prioritize homogeneous patches over those containing edges, reducing artifacts. Kaiser Window is applied to further attenuate patch borders for reduced border effects.

BM3D

- Patch-Matching: Processed on the basic estimate. Similar patches obtained, forming two 3D groups.

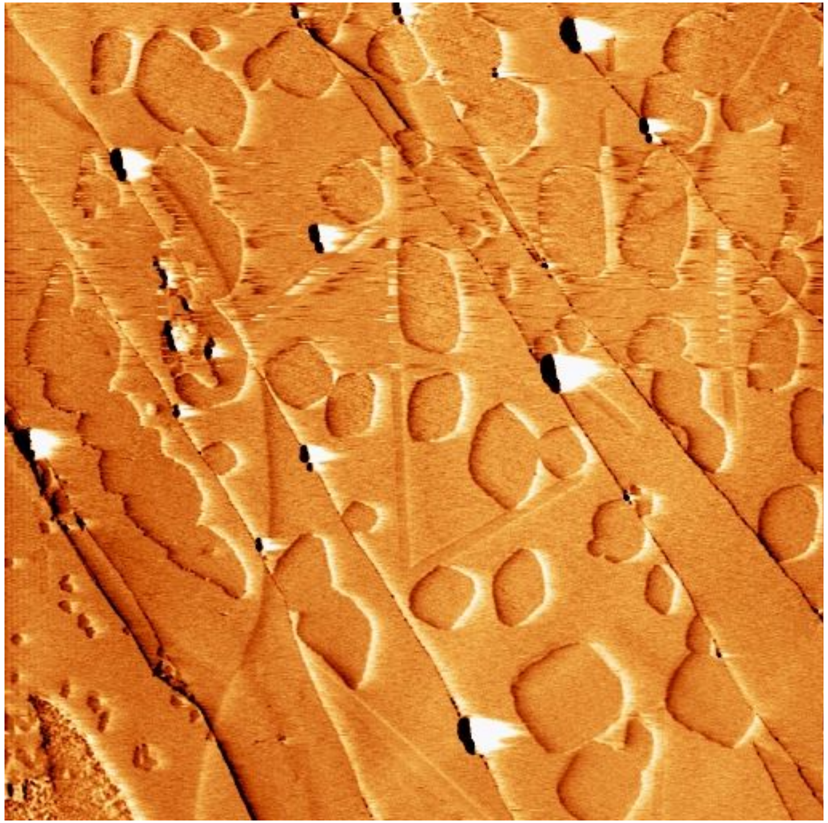
- Collaborative Filtering: Employs empirical Wiener coefficients and performs Wiener collaborative filtering on obtained 3D groups.

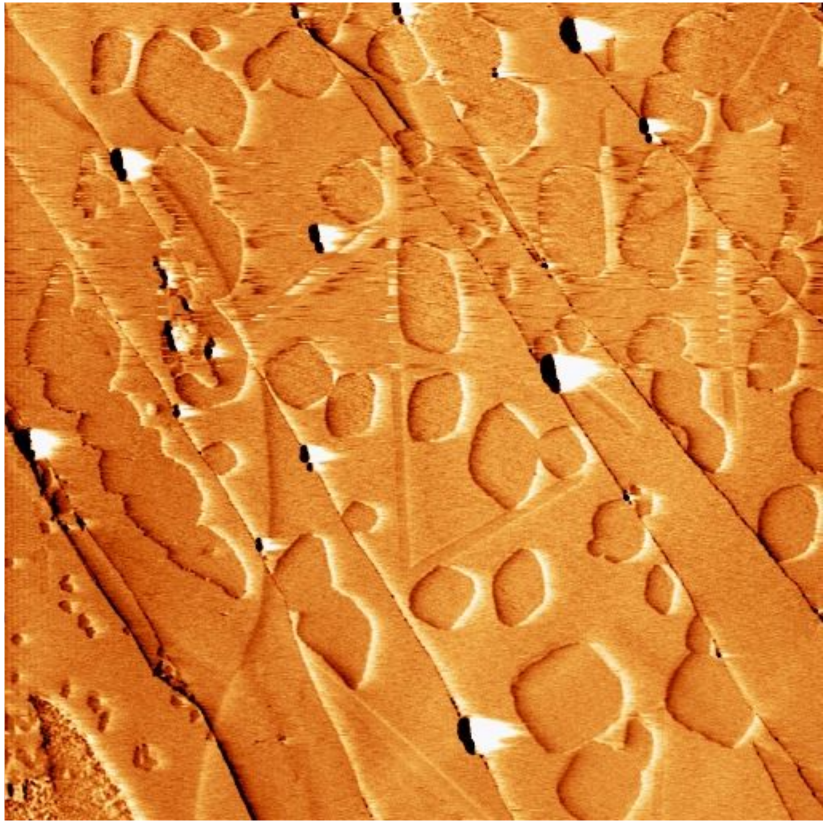
- Aggregation: Similar to the first step, estimates stored for every pixel. Kaiser window applied to reduce border effects.

- Final Estimate: Equation provided for the final estimate obtained after the second step.

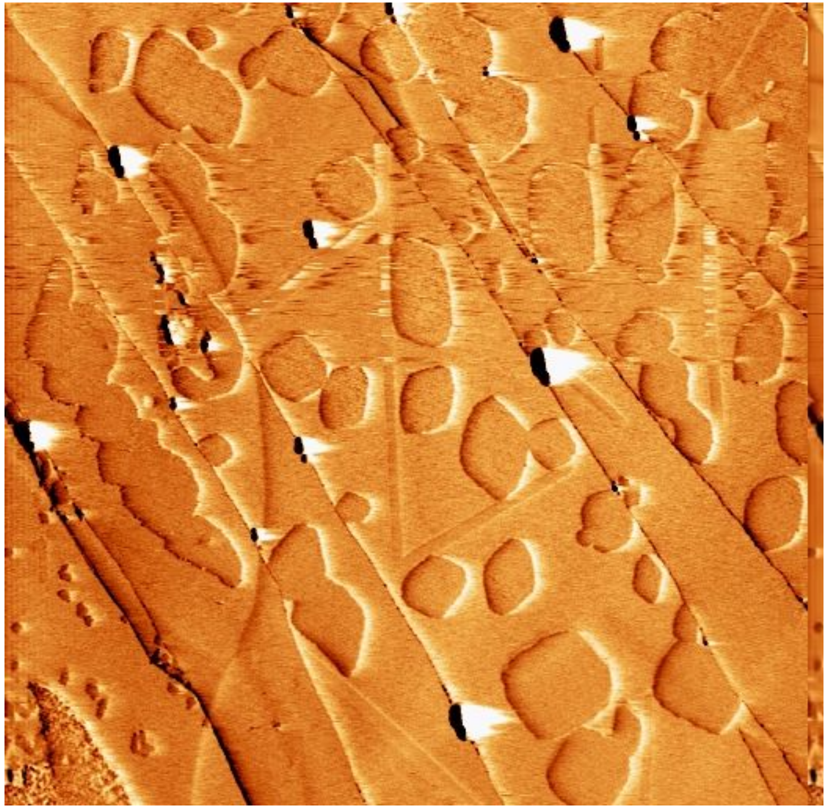
$$u^{\textbf{final}}(x) = \frac{\sum_{P} w_P^{\textbf{wien}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x) u_{Q,P}^{\textbf{wien}}(x)}{\sum_{P} w_P^{\textbf{wien}} \sum_{Q \in \mathcal{P}(P)} \chi_Q(x)}$$

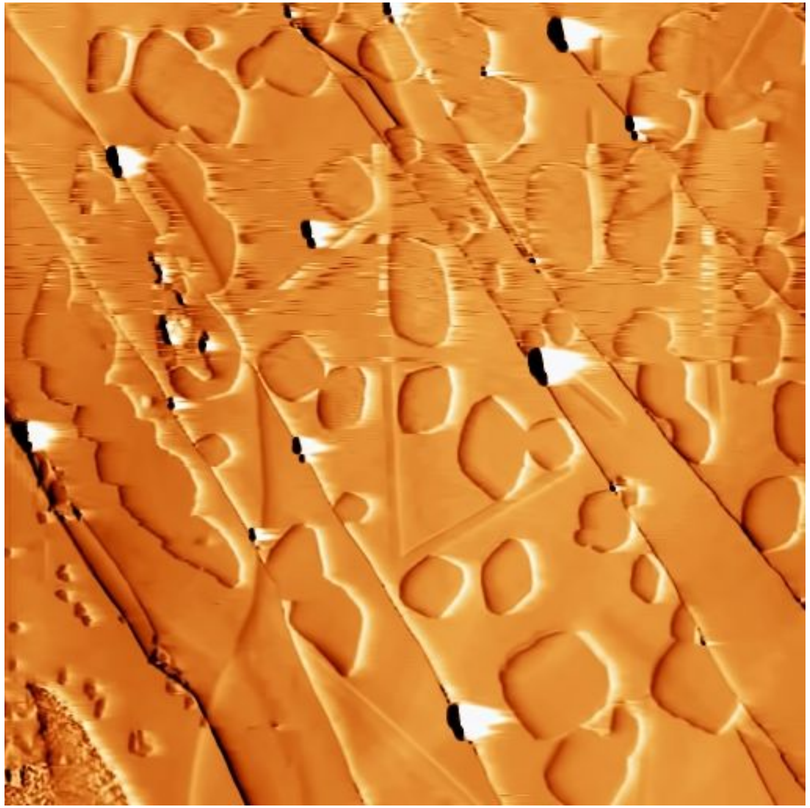
Bm3D (sigma=0): Amplitude Image



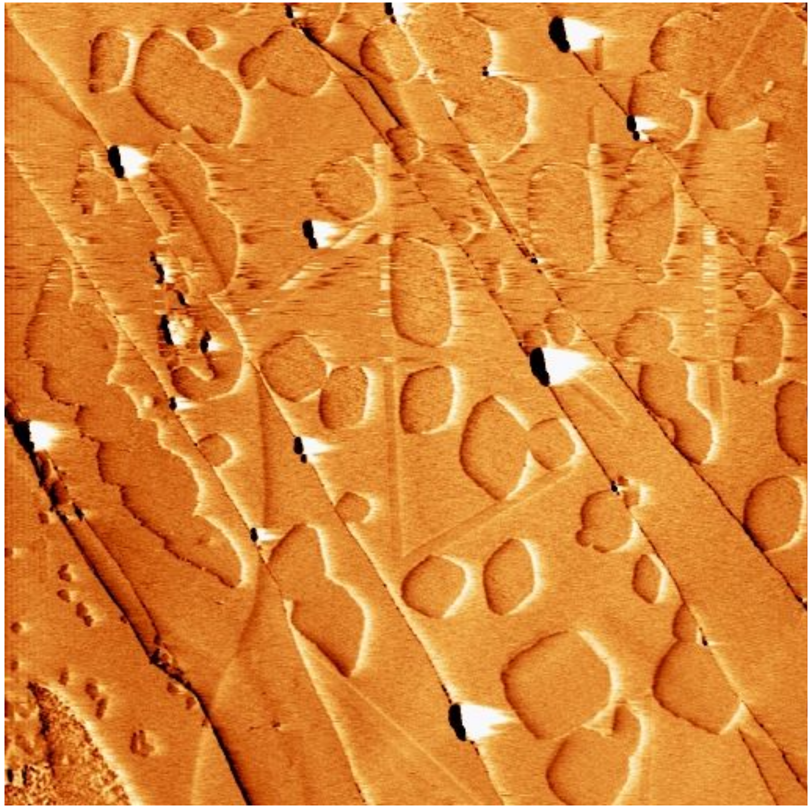


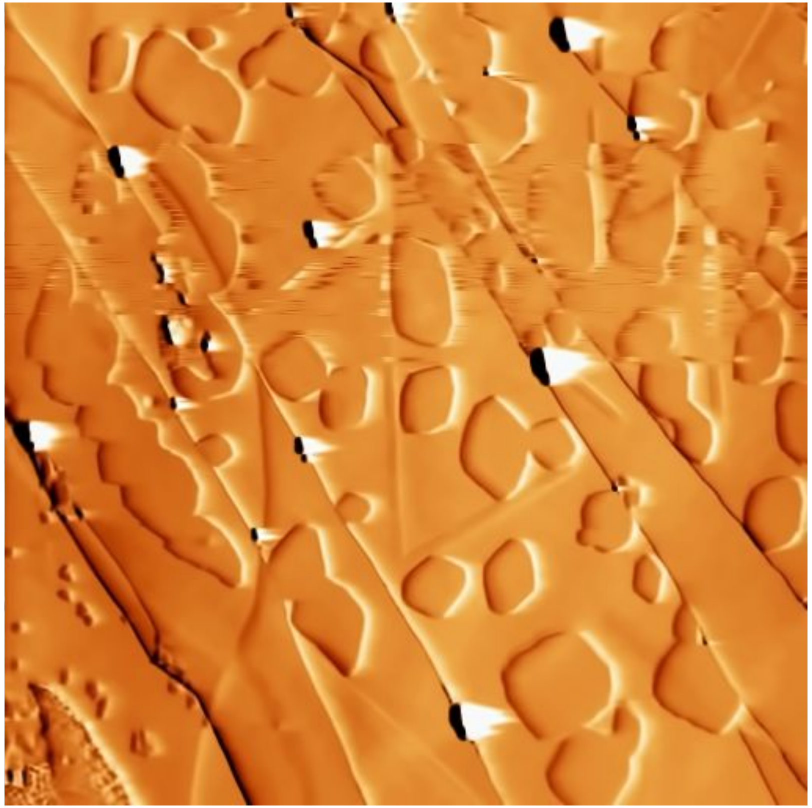
Bm3D (sigma=25): Amplitude Image



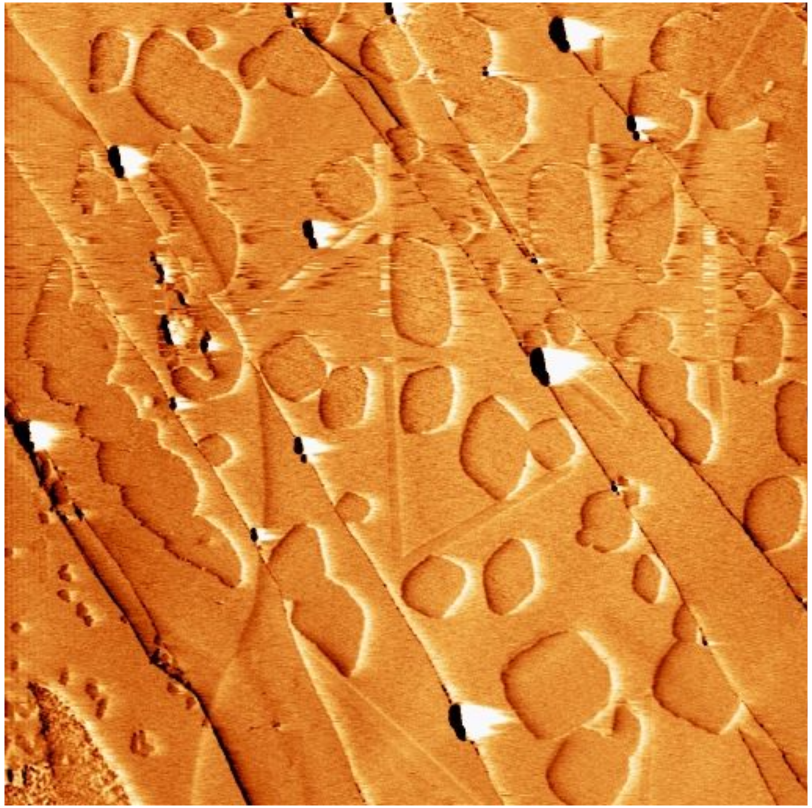


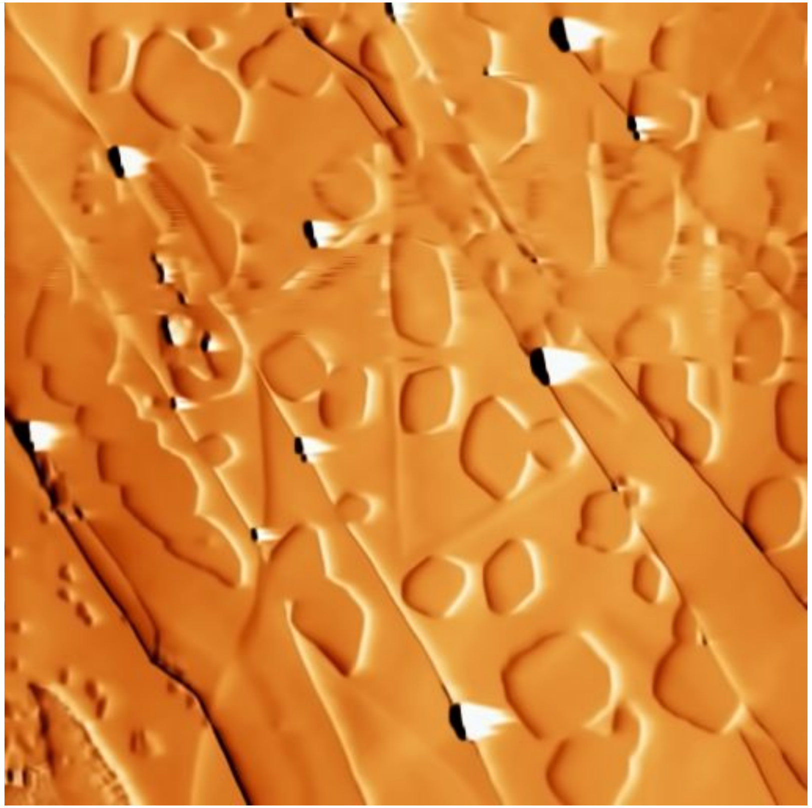
Bm3D (sigma=50): Amplitude Image



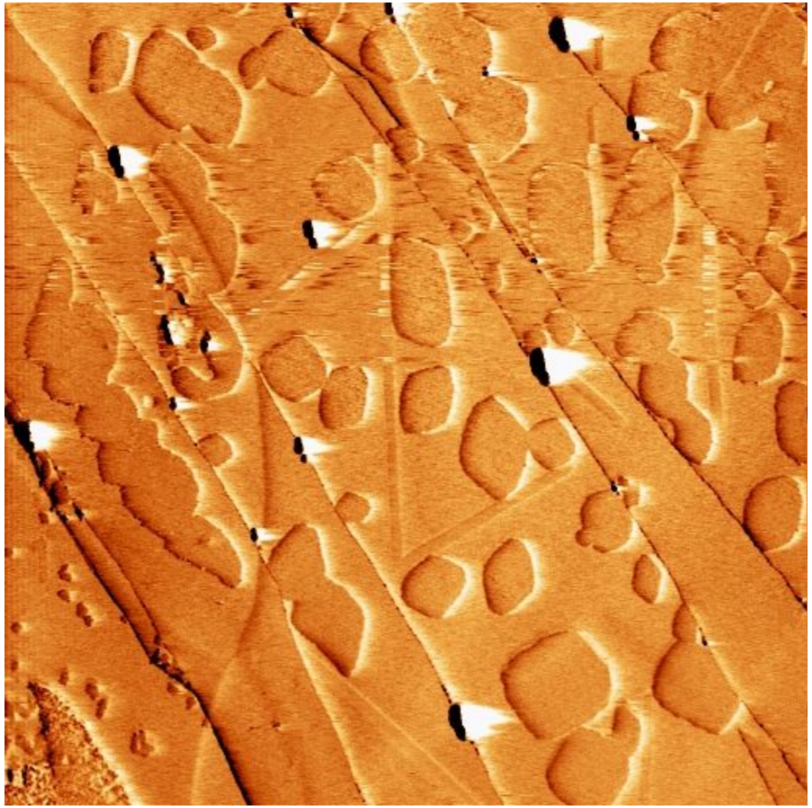


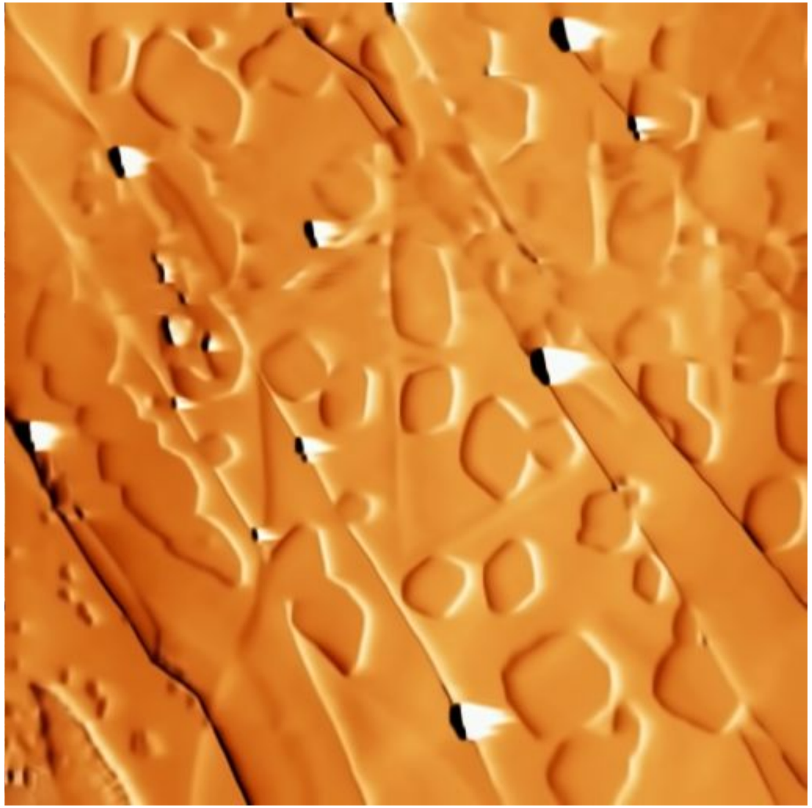
Bm3D (sigma=75): Amplitude Image





Bm3D (sigma=100): Amplitude Image





Further Proceedings...

We thought of trying deep learning models to denoise, but due to constraint of not having ground truth images we were restricted to try this domain.

But we have found a research paper using **Stein's unbiased risk estimator** to tackle this issue and train deep neural network denoisers only based on the use of noisy images.

So, we are currently implementing this paper and will try to get results by the upcoming week.

THANK YOU!