# ARTEFACT REMOVAL, NOISE REDUCTION, AND CONTRAST ENHANCEMENT OF ELECTRON MICROSCOPY IMAGES WITH NAPARI

## User Guide

Version 1

# Contents

# Abbreviations

CLI – command-line interface

FIB/SEM – Focused ion beam / scanning electron microscope

GUI – graphical user interface

Cryo-EM

SPA

Cryo-ET

vEM

# Introduction

## Cryo-FIB/SEM

In the past decade, great advancements have occurred for electron microscopy (EM). The Nobel Prize in Chemistry 2017 recognized a ground-breaking development in cryogenic electron microscopy (cryo-EM), which improved the high-resolution imaging of biomolecular structures. For cryo-EM, samples are rapidly vitrified in water to yield crystal-free ice in a glass-like state which allows to preserve the biomacromolecules as close to their native state as possible. Then, transmission electron microscopy (TEM) is employed to visualize these structures. The structural biology field continuous to benefit from these techniques, such as cryo-EM single particle analysis (SPA) and cryogenic electron tomography (cryo-ET), the former studying purified macromolecules in solution and the latter investigating molecules within the context of cell or tissue sections (Saibil 2022 Cryo-EM).

More recently, another revolution in the EM field is taking place, specifically in volumetric image acquisition. Volume electron microscopy (vEM) approaches are based on microscopy techniques such as TEM and scanning electron microscopy (SEM). Classically, these microscopy methods are limited to sample thicknesses of around 200 nm (Collinson 2023). However, vEM addresses these challenges through various methods of serial sectioning and/or imaging (Collinson 2023). With the TEM, serial sections of a sample block are laid on grids or on tape and then images are acquired simultaneously or section by section. Meanwhile with the SEM, serial sections on a substrate can be sequentially imaged similar to TEM methods, but more advanced methods offer automation of the serial sectioning. A section of a sample block is removed, and the newly exposed surface is imaged in succession. These methods breaks the depth limitations, revealing 3D ultrastructure of biological samples of up to 1 micrometer in continuous depth. As such, vEM was declared as one of the seven technologies to watch in 2023 and has been utilized in several biological fields (Peddie 2023).

The serial cryogenic focused ion beam / scanning electron microscope (cryo-FIB/SEM) is a technique that sits at the exciting junction of cryo-EM and vEM. The combination of sample preservation at near-native fully-hydrated states and serial acquisition of nanometer-thin sections results in visualization of molecular structures in large volumes with high resolution. With this technique, research questions such as the mineral distribution in cellular organelles (Vidavsky 2016) and subcellular architectural disruptions in mutated primary cells (Zhu 2021) can be explored in a structural context. Moreover, concurrent detection of secondary and back-scattered electrons reveal information on the surface topography and the atomic composition of the sample, respectively.

The FIB/SEM is the only microscopy technique that produces isotropic voxels as small as 5x5x5 nm3 (Kizilyaprak 2018), giving finer resolution in the z-direction than with manual sectioning or automated sectioning with a diamond knife, i.e., serial block face / scanning electron microscopy (SBF/SEM). However, as larger volumes 50x50x50 μm3 would be extremely time consuming to image (Kizilyaprak 2018), this technique is more suited to smaller objects such as cells, bacteria, parasites and such. A sample is imaged by sequentially milling a layer off of the sample surface with the FIB and imaging the exposed surface by the SEM. After which the individual images representing the planes of the sample can be reconstructed into the 3D volume. Traditionally, samples for FIB/SEM are prepared with classic preservation methods, such as with chemical fixation with aldehydes or with high-pressure freezing, freeze-substitution with organic solvents. After fixation, samples are stained with heavy metals like osmium, lead and uranyl to enhance the contrast of molecular structures and then embedded in resin to stabilize the sample against the electron beam of the SEM (Kizilyaprak 2018).

Although traditional sample preparation methods have resulted good insights into cellular structures, artefacts from fixation, staining and drying can hinder accurate structural determination. Biological specimens are inherently hydrated, thus cryogenic fixation offers a better approach in preserving structures at near-to-native state (Vidavsky 2016). However, cryo-fixed samples face its own set of challenges (Dumoux 2023).

Firstly, native biological matter are mostly composed of elements such as carbon, hydrogen and oxygen, which have low atomic numbers. Because of the lack of heavy metal staining, fewer backscattered and secondary electrons are returned from the interaction of the beam with the low atomic number elements, resulting in lower sample contrast. Secondly, biological materials are heterogenous. The heterogeneity in hardness causes differential milling of the FIB through the sample, thus features parallel to the milling direction or “curtaining” artefacts ensue. Thirdly, certain biological substances can be non-conductive. In consequence, electrons can accumulate on the surface and produce bright regions in the image or electrons can be emitted in greater rates than the beam provides and lead to dark areas with low signal. Other issues of images from cryo-fixed 3D volumes are varying brightness levels across the slices of the stack and high noise levels due to the high-resolution image (source is a website though). All these issues hinder the precise structural information in FIB/SEM data.

## Python and the napari library

Python is one of the popular programming language among biologists, applied in fields such as in image processing, bioinformatics, and natural language processing. Its multiple advantages are its simplicity and readability, the shallow learning curve, and its active community. The large Python community continually creates and maintains a collection of packages for a wide range of functionalities, including biological applications.

A recent great addition to the Python toolbox is napari, “a fast, interactive viewer for multi-dimensional images.” This library allows “n-dimensional image visualization, annotation and analysis.” What is special about napari is that the idea arose from electron microscopists who identified a lack of this tool in Python, so this viewer was particularly intended for large volumetric microscopy data. At its base, napari offers a great graphical user interface (GUI) for viewing and interacting with multidimensional image data. Moreover, users can further extend its features with plugins. Thus, this processing application for electron microscopy data is built as a plugin for the napari viewer.

## The boommarter EM image processing napari plugin

Along with the sample preparation and imaging techniques, the post-processing workflows are as important. Often the raw images from microscopes are riddled with artefacts and quality issues. Cryo-FIB/SEM technology is advancing at a fast rate, but the image processing part of the whole workflow is lagging behind. Currently, there are only a few tools that are tailored for processing FIB/SEM images: Okapi-EM and bmiptools. Okapi-EM is a convenient plugin for napari and offers functions for charging removal and stack alignment. However, it does not tackle other issues such as curtaining or denoising. On the other hand, bmiptools provides a whole toolbox of functions and possibilities to link these functions into a pipeline. However, it is a simple GUI that does not allow viewing of the images as the users runs the functions.

Currently, the boommarter plugin (named as such since the consonants of the syllables spell out BMT, which is also the abbreviation of the group’s name Biochemistry of Mineralized Tissues) combines both the convenience of the image viewer such as napari and several functions required for processing FIB/SEM stacks. The processing functions are decurtaining (i.e., vertical stripe removal), decharging, denoising, enhancing contrast, horizontal stripe removal, brightness correction through the stack. It also offers other functions such as identifying the same images in a stack and linking the processing functions into a pipeline.

The processing functions are intended for FIB/SEM data, so it only handles gray scale images. However, the functions are modality-agnostic, enabling its use on other types of EM data such as SEM images of samples prepared with chemical fixation or freeze substitution or even TEM images. Especially useful are the denoising and enhancing contrast functions, since these are issues that present in most, if not all, types of imaging data.

Outline

* Electron microscopy developed quickly in the past years
* Cryo-EM (Saibil 2022 Cryo-EM)
  + Cryo-EM for single particle analysis – purified macromolecule in solution
  + Cryo-ET – in situ, molecules in the context of cell or tissue sections
* vEM (Collinson 2023)
  + Depth limitation
  + TEM – serial sectioning
  + SEM – serial sectioning and automated sectioning.
  + 7 technologies to watch in 2023 <https://www.nature.com/articles/d41586-023-00178-y>
* Clarifies differences of cryo-EM and vEM <https://analyticalscience.wiley.com/content/article-do/volume-electron-microscopy-rise%C2%A0>
  + SE vs BSE
  + <https://www.thermofisher.com/de/de/home/materials-science/learning-center/applications/sem-electrons.html#:~:text=BSE%20come%20from%20deeper%20regions,material%20appears%20in%20the%20image>.
* Cryo-FIB/SEM at the merging point of cryo-EM and vEM
  + Cryo-preservation of fully hydrated frozen cells at near native state and serial imaging of 10 nm and below sections = nanometer resolution of molecular structures in 3D volumes
  + Successful biological applications of cryo-FIB/SEM (list in Collinson 2023)
    - Distribution of mineral deposition sites in cellular organelles of sea urchin embryos and zebrafish larvae tail fins (Vidavsky 2016 <https://doi.org/10.1016/j.jsb.2016.09.016>)
    - Subcellular architectural disruptions in primary fibroblasts in Leigh syndrome patient with USMG5 mutation (Zhu 2021 <https://doi.org/10.1016/j.str.2020.10.003>)
      * Linking genetic and cytologic information
  + How does cryo-FIB/SEM work
    - Iso-voxels (Luckner 2018 <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6096567/>)
    - Sample preparation
      * Traditional methods: chemical fixation or high pressure freezing and freezing substitution, then heavy metal staining, then embedding (Kizilyaprak 2018)
      * Cryogenic fixation
* Cryo samples issues (Dumoux 2023)
  + Contrast: native biological material mostly have low atomic number elements 🡪 less backscattered and secondary electrons 🡪 low sample contrast
  + Curtaining: heterogenous biological material 🡪 differential milling by the FIB 🡪 features parallel to milling direction = “curtains”
    - <https://doi.org/10.2138/am-2001-8-917>
  + Charging
    - <https://www.nanoscience.com/blogs/how-to-combat-electric-charge-buildup-in-scanning-electron-microscopy/>
    - Positive charging: dark areas with lack of signal, electron beam against highly insulating substances in sample such as with regions of high lipid content
    - Negative charging: electrons trapped within sample and accumulate on surface of non-conductive specimen, bright regions
    - Basically large regions of brightness or darkness in image, obscuring finer details
  + Brightness variation across slices of the stack
    - Source?
  + Noise levels
    - High resolution image, low S/N ratio, grainy appearance
      * <https://myscope.training/SEM_Scan_rate_and_signal_to_noise>

Outline

* Boommarter = BMT
* Identified a gap in image quality processing softwares
  + Currently available: Okapi-EM (only offers for charging but nice that it has stack registration) or bmiptools (not user friendly)
  + Lack of automated image analysis methods!!
    - Can do stack processing for each function
      * STILL WORKING ON LINKING ALL 4 STEPS OR CHOOSING WHICH FUNCTIONS TO LINK TOGETHER and then saving the metadata
* Addresses: decurtaining, decharging, contrast enhancement, denoising, horizontal destriping, histrogram matching
* These functions are intended for EM data, so it only handles gray scale images. The functions were developed for the processing of FIB/SEM data.
* However, the functions are modality-agnostic, so can be used for TEM
  + Contrast and noise is also problems with AFS and Chemically fixed samples
    - AFS
      * How it works
      * Does curtaining and charging happen with AFS?
      * Problems: contrast? Noise?
    - Heavy metal staining
      * How it works
      * Contrast is not a problem. Only noise?

# Initialization and Utilization

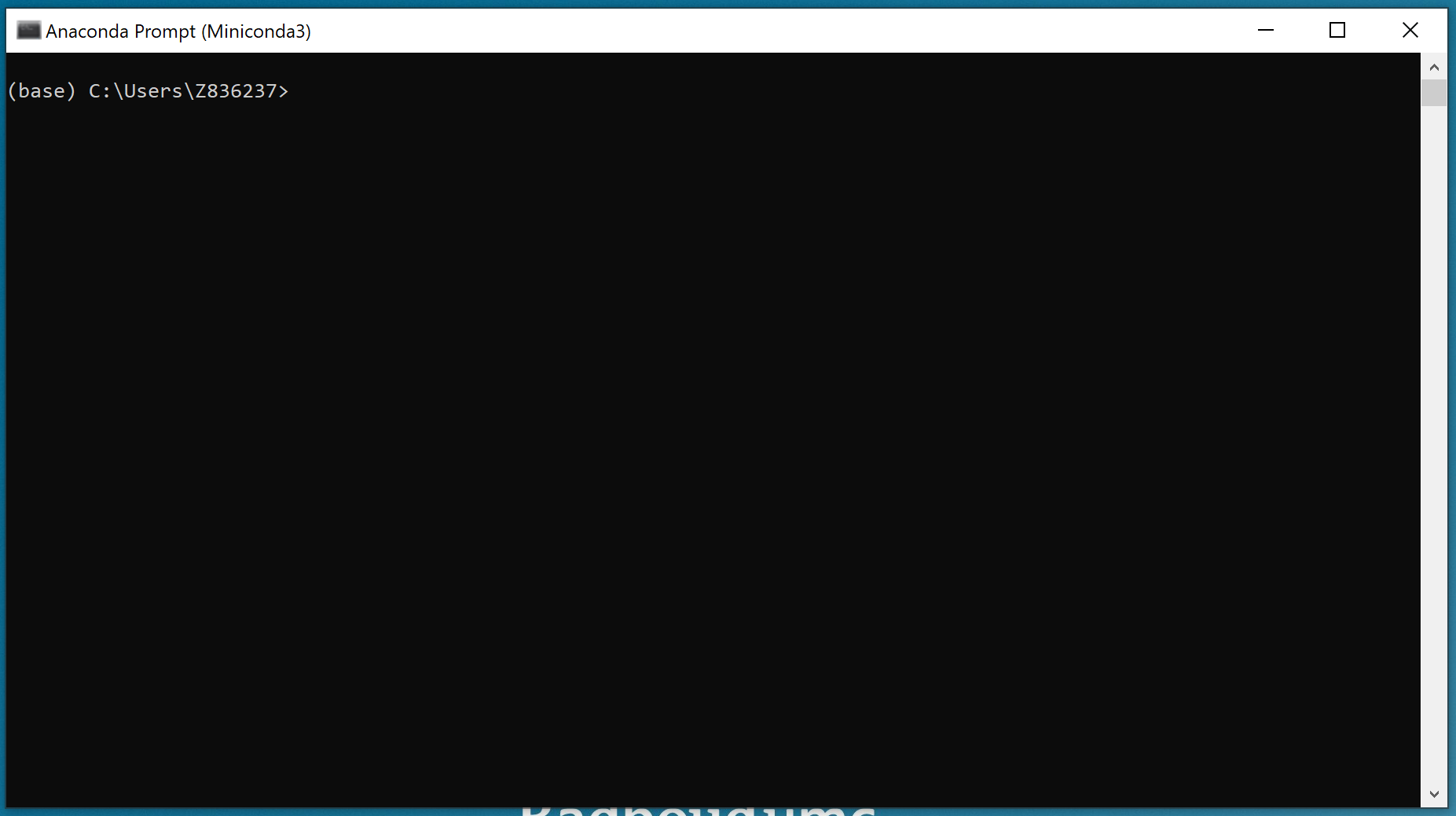
## Miniconda

The processing application is best run with Miniconda or Anaconda, which are tools for managing Python environments and packages. Miniconda is a free minimal installer for conda, the package and environment management system.

Install Miniconda by following the instructions in this website.

<https://docs.anaconda.com/free/miniconda/miniconda-install/>

Once Miniconda is installed, launch the terminal by opening the Anaconda Prompt (Miniconda3) program. In case the program cannot be found, simply type the program name into the search bar of the operating system.

  
Figure 1. The terminal or command-line interface (CLI) of Miniconda. This Anaconda Prompt provides a way to interact with the Conda package manager. Instead of a GUI in which the user interacts with the operating system via buttons, the terminal or the CLI takes lines of text or command-lines as input from the user.

A virtual environment is a useful tool within Miniconda to ensure that dependencies (such as Python and package versions) of a project can be kept isolated from other projects. Create a virtual environment for the processing application by typing the following command-line in the terminal and afterwards clicking Enter:

conda create -n napari-environment python=3.11 napari pyqt -c conda-forge

Continue the creation of the virtual environment by entering “y” or simply clicking Enter after this prompt.



Activate the virtual environment with this command-line:

conda activate napari-environment

The virtual environment is activated once the prompt changes

from 

to 

Test if the virtual environment is successfully created and napari is properly installed with this command-line, which launches the napari window:

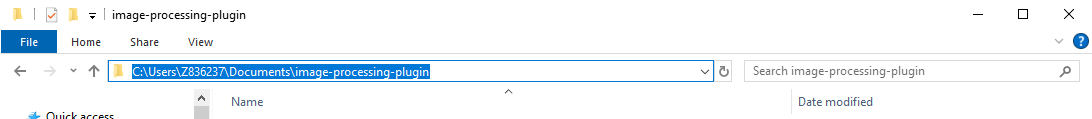
napari

The napari interface should appear in a several seconds to a minute. Close the window before installing the plugin.

Once napari is ready, the image processing plugin can be installed into the virtual environment.

Move to the directory containing the plugin. For Windows OS, it is important that the folder path is enclosed in parentheses and each folder name is separated by a double backslash. The folder path will differ per user depending on where they placed the plugin folder.

Determine the folder path name by going to the location of the plugin folder. Copy the folder path.



Then, execute this command-line in the Miniconda terminal:

cd “folder\_path\_name\_to\_plugin”

Example: cd “C:\\Users\\Z836237\\Documents\\image-processing-plugin”

The current directory is shown in the prompt.



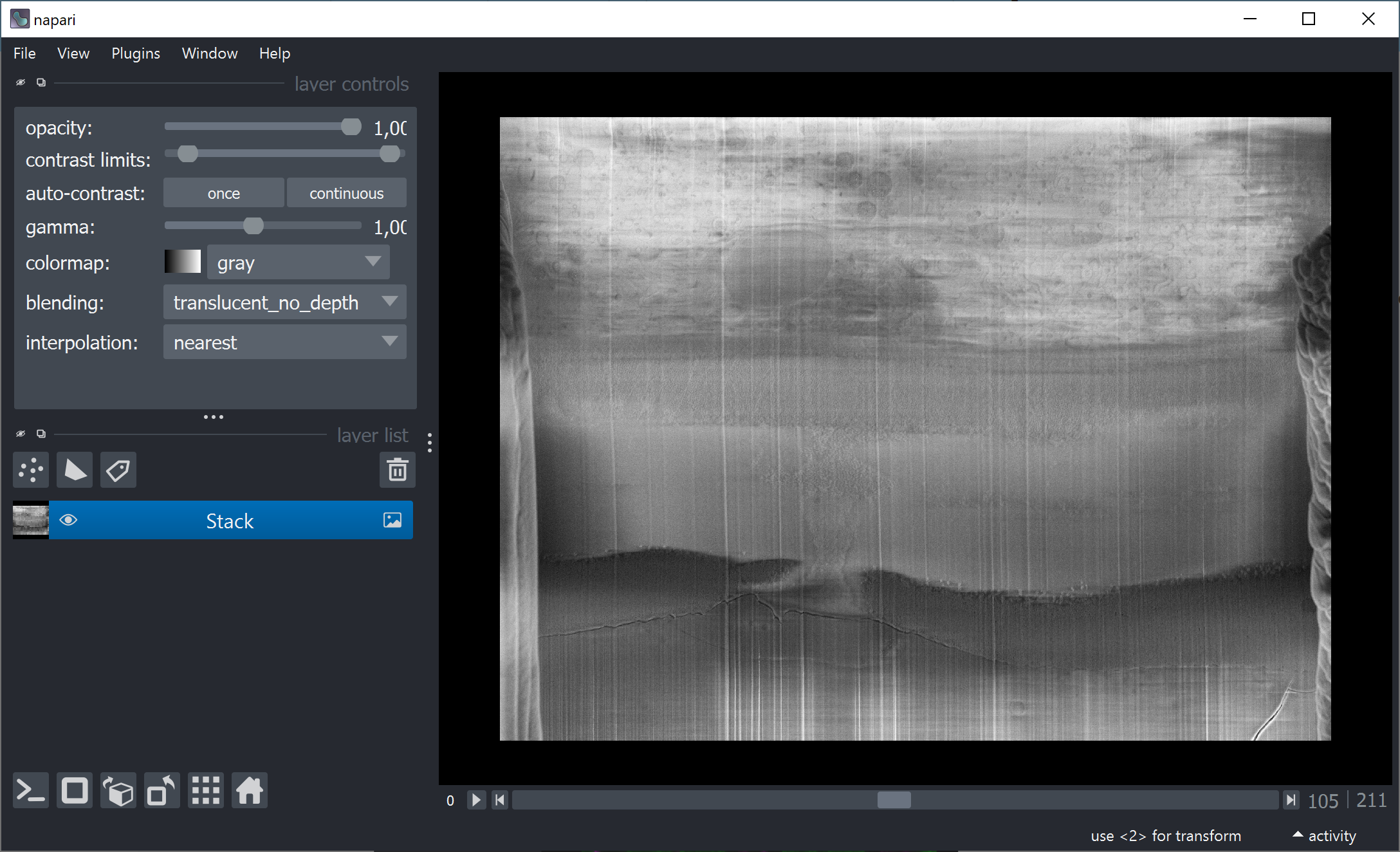
Install the plugin with this command-line:

pip install .

Launch napari again with the “napari” command-line.

## Napari

Here are some key areas in the napari viewer. The Main Menu contains the important content and commands of the window. The Canvas displays the data in napari. The Layer List shows the all layers in the viewer. The Layer Controls control the aspects of the layers. In this case a volume stack is imported, so the viewer also has Dimension Sliders.



**Layer Controls**

**Layer List**

**Main Menu**

**Dimension Sliders**

**Canvas**

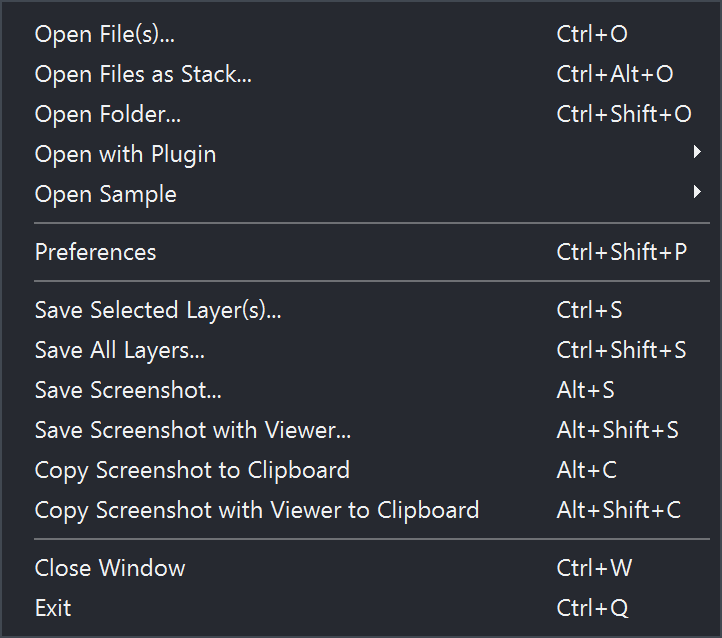
A more in-depth look at the key areas of the viewer is provided in this website:

<https://napari.org/stable/tutorials/fundamentals/viewer.html#layout-of-the-viewer>

Layers are one of the basic napari objects. napari supports different layer types, e.g., Image, Labels, Points, Shapes, Surface, Tracks, and Vectors. In-depth explanations of layer types and layer controls are available in this website:

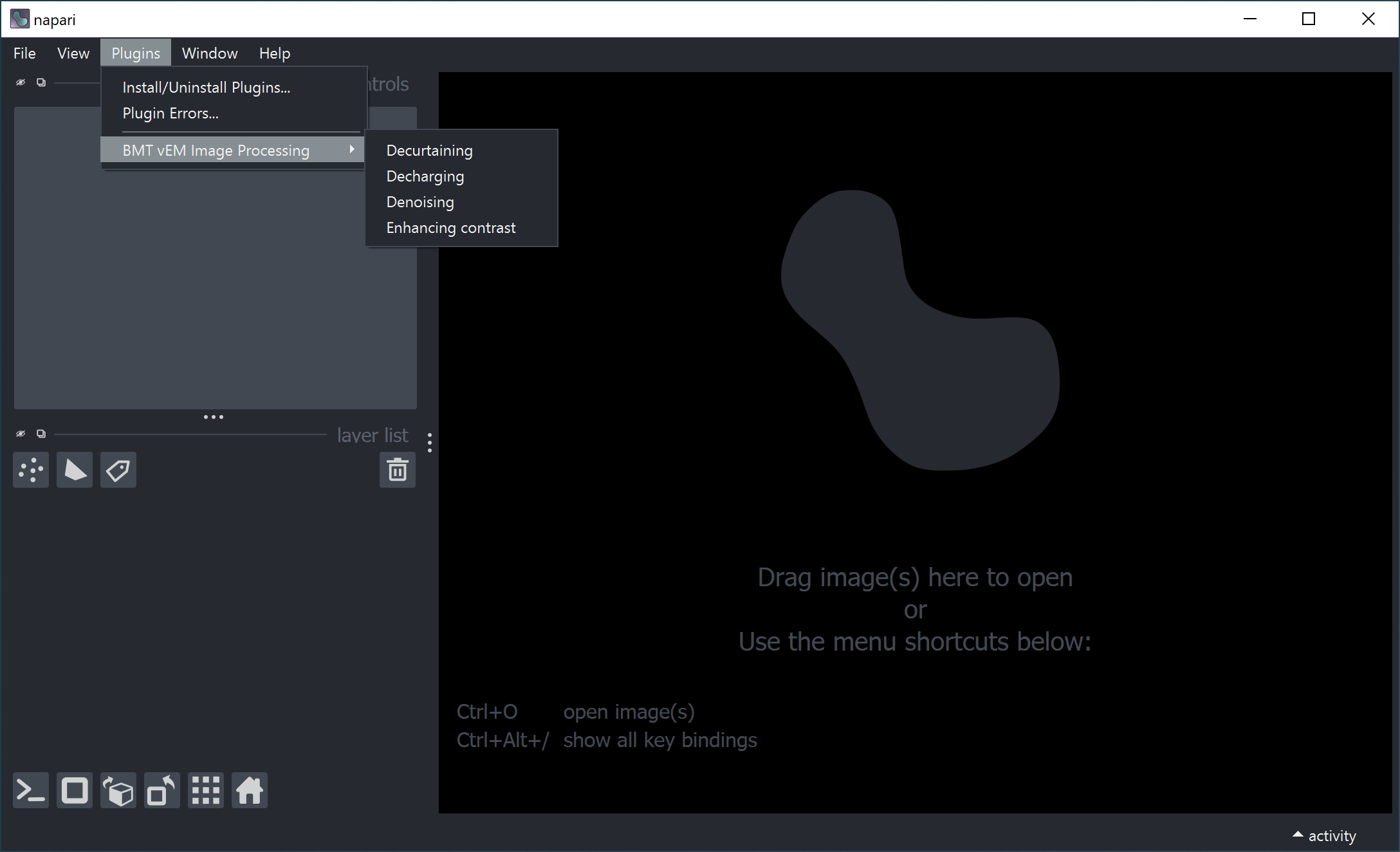
<https://napari.org/stable/guides/layers.html>

In the File menu of the Main Menu, use the **Open File(s)** option to open single files or **Open Files as a Stack…** or **Open Folder** to open volume stacks. Once processing is finished, select the desired layer in the Layer List and use the **Save Selected Layer(s)…** or **Save All Layers…** options to save the processed images.

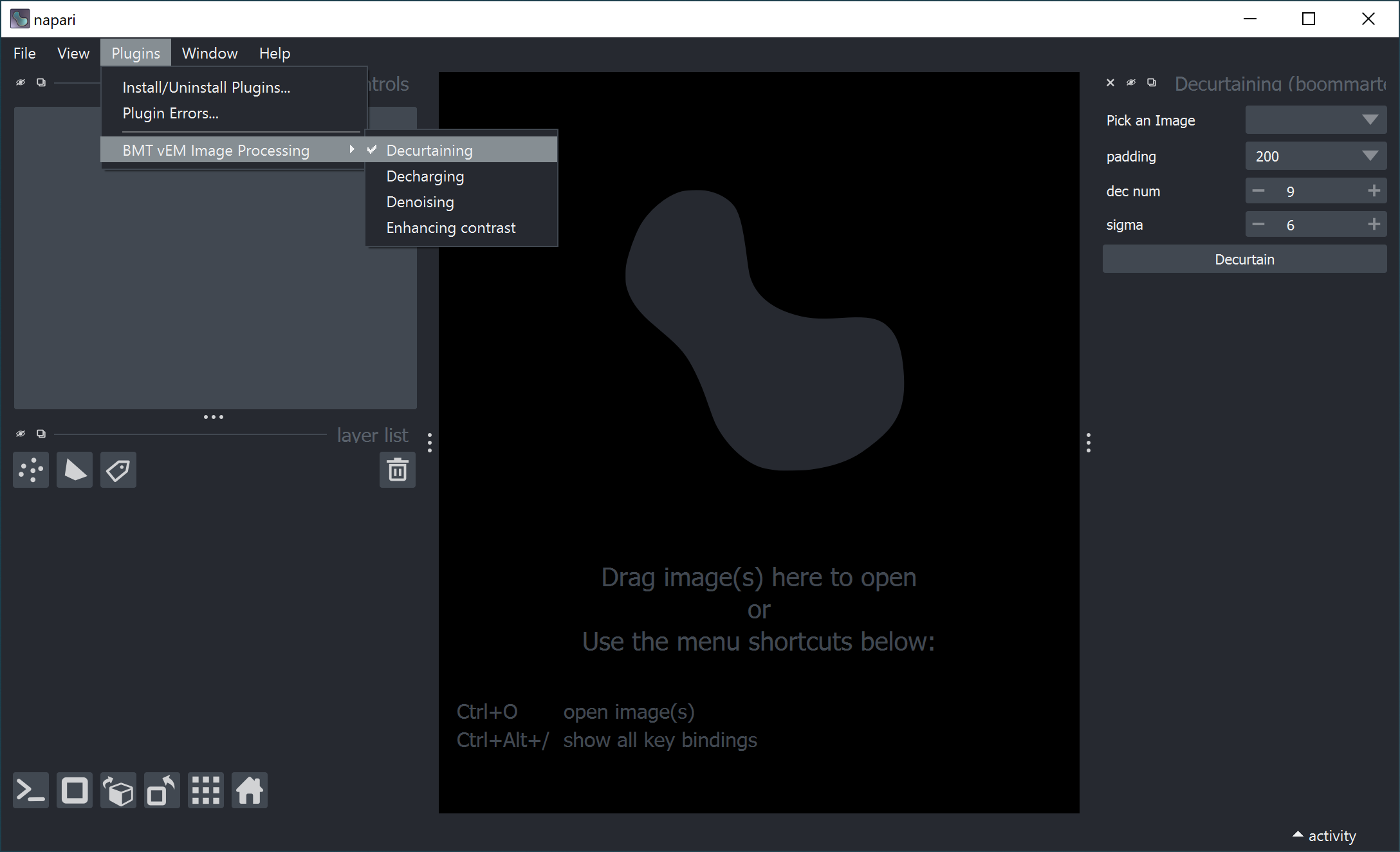


## Plugin

If the plugin was successfully installed, then it should be available via the Plugin menu.



The plugin is very straightforward. Users can click on which function they want to use. For the decurtaining function as an example, click on the Decurtaining option. A widget will appear on the right side of the viewer.



The following steps are applicable to all the widgets of the processing steps:

Choose the desired image or stack to be processed.

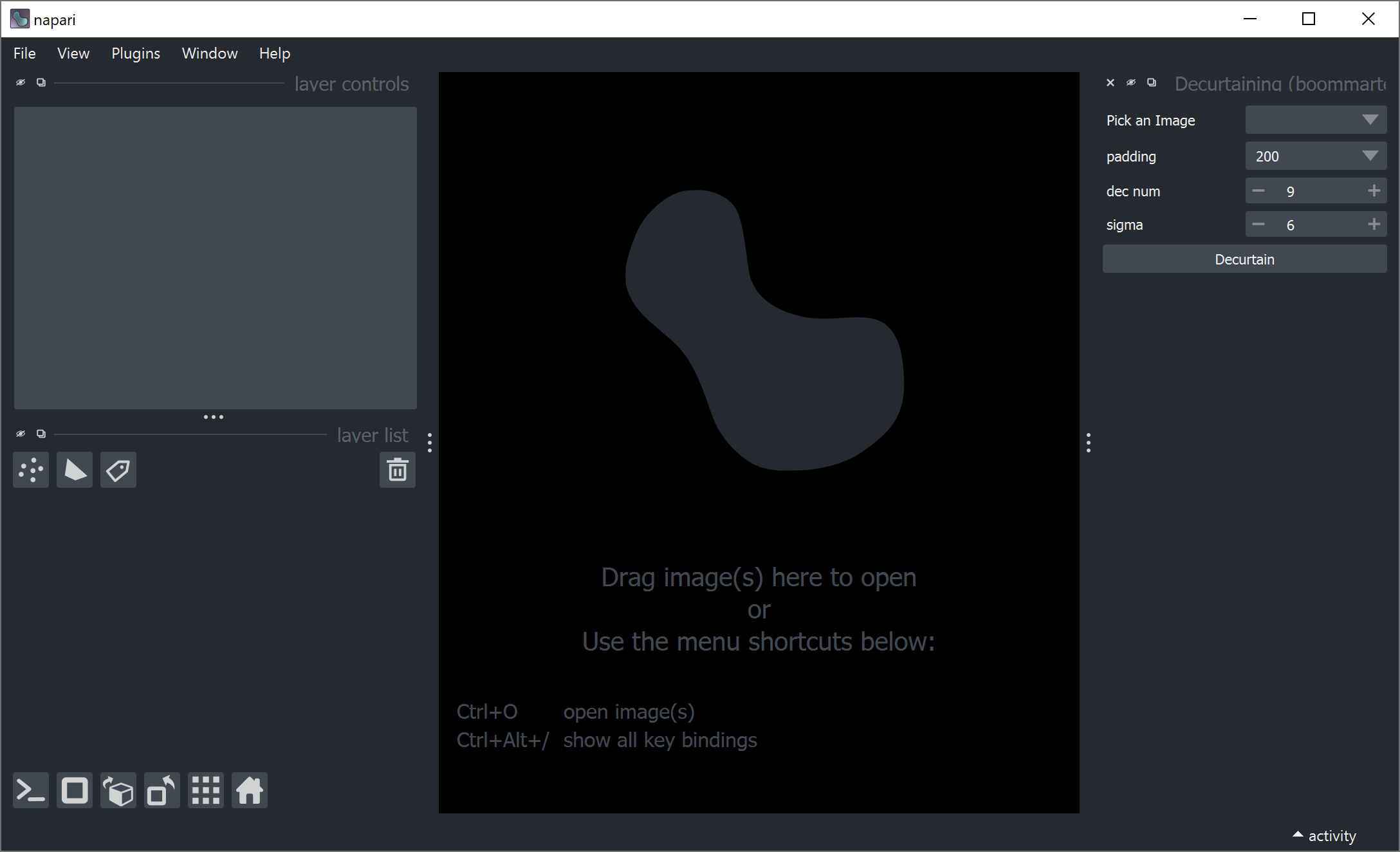
Set the function parameters.

Click the function call button.

Wait until the processed image appears as a new layer in the left side.

The images will be named with the processing step and the parameter values. Single images should appear in about a few seconds, but stacks may take several minutes.

In the case users want a side-by-side view of the images, the grid mode can be toggled on and off at the bottom left button menu.



# Decurtaining

Layers of the sample is milled off by the FIB to allow imaging through the sample.

When the FIB hits a harder material and mills unevenly, creating ridges in the surface.

The SEM is very sensitive to surface changes.

Thus, curtains or waterfall effect manifests on the image

* Show image

Algorithms that are available for decurtaining??

* Why choose this wavelet decomposition and FFT filtering

## Algorithm

The algorithm for the removal of the curtaining the wavelet-Fourier filtering (Münch 2009)

Decomposes image to details 🡪 transforms vertical details to Fourier space 🡪 filters the vertical details 🡪 inverse transform 🡪 reconstructs the image from details

Figure from Münch paper

### Decomposition number

* Number of times the image is broken down
* Implication: Higher number, more fine separation of details

### Sigma

* Strength of filter in the Fourier space
* Implication: Higher number, stronger filtering

Padding

* Keep this hidden from users?
* Padding 400

Change in Plugin – change names of new layers: with underscore

Good default: decomposition 6 sigma 4 or decomposition 7



## Guidelines for parameter values

Increase decomposition number first, (what is the upper limit? Look through the images to find that?)

Only then increase sigma, since sigma removes details so ideal not to increase sigma too much

## Deeper dive into the Python code

Graphical workflow of the python function

* Have to understand the function deeply first lol
* Basically
  + Decomposes images into details
  + Takes vertical decomposition image
  + Transform to Fourier space
  + Filters strong vertical details manifesting in the horizontal orientation
  + Transforms back to spatial dimension
  + Reconstructs image from details

# Decharging

Biological materials are non-conductive and can accumulate charge at the surface as the SEM scans

Charge can be emitted in one go

Manifests are bright regions in the image, hiding the details from the image

## Algorithm

Basically background subtraction

Gaussian blur image to capture only the background and subtract that from image

### Sigma X

* Strength of filtering in the X direction
* Higher sigma, wider filter 🡪 can capture large bright regions in the background

### Sigma Y

* Strength of filtering in the Y direction
* Keep small so no details are lost

Padding

* Set to 400 and hide from users

## Guidelines for parameter values

It is recommended to keep the default sigma Y value. Adjust sigma X lower if more background subtraction is needed or higher if too much detail is being lost

## Deeper dive into the Python code

Basically

* Pad
* Gaussian blur
* Subtract gaussian blur image to original image
* Normalize resulting image back to the bit scale (is that the right way of saying it?)
* Remove padding

# Denoising

Cryo-fixed sample require careful dosing of the electron beam

Can result in low signal to noise ratios manifesting in a grainy noise in the image

What have others done to solve noise problem

* Most popular since almost all image have this problem

## Algorithm

Anisotropic diffusion

Basically blurring with respect to edges

What is gradient threshold?

### Steps

* Steps? Number of iterations of the blurring effect

### Gradient threshold (b)

* Cut off point of gradient of edges that are respected and not blurred
* Low gradient threshold 🡪 any gradients higher than this threshold will be respected 🡪 preserves more edges

Do I need to pad?? Check the edges of the images

## Guidelines for parameter values

Recommended to keep gradient threshold below 0.05 or 0.1?

* How much freedom should I give to the users? i.e., how many values to make available?

## Deeper dive into the Python code

Basically

* Anisodiff function from python

# Enhancing Contrast

Biological materials do not have good contrast since low atomic numbers

Heavy metal staining was the traditional solution for this, but cryo cannot have heavy metal staining

* No time for staining
* Heavy metals cannot penetrate cells without fixation (?)
* Heavy metals are toxic to cells, which defeats the purpose of cryo which attempts to preserve cells at near native conditions

## Algorithm

CLAHE

* Cutting off histogram and redistributing pixels

### Clip limit

* The limit of the pixel intensity to cut off??

### Tile size

* Row and column wise size of tile regions
* Number of rows or columns that the image will be divided to

## Guidelines for parameter values

Higher clip limit, larger tile sizes??

Some clip limits can suddenly produce artifacts. Ways to fix that

* Increase clip limit
* Make tile sizes larger

## Deeper dive into the Python code

Basically

* Pad
* Gaussian blur
* Subtract gaussian blur image to original image
* Normalize resulting image back to the bit scale (is that the right way of saying it?)
* Remove padding

# References

# Outline

Introduction

* On what software is the app based on? Python and napari
* What type of data is this processing app targeted for?
* What is FIB/SEM data?
* What are the types of preparation?
  + Cryo
    - What are the common artefacts that arise?
      * Curtains
      * Charging
    - What are common problems?
      * Low contrast for cryo.
      * Is denoising a general problem?
  + HM and AFS
    - Contrast and denoising

Content of sections for each function

* What is the problem that it tackles? Explain more deeply
* How does the algorithm fix the problem?
* What do the parameters mean?
* What is the optimal range of parameters?
  + Even better if I can explain it to certain cases.