IT Project - ENVI Editor

Introduction

This project aims to create a graphical user interface for the ENVI spectral image editor that has the features listed in Figure 1.

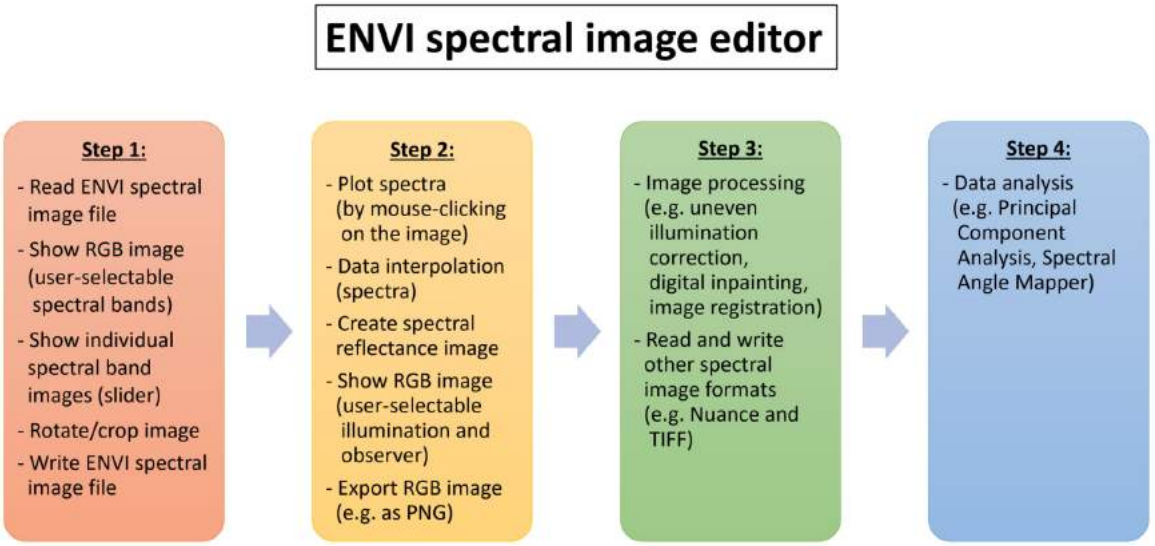


Figure 1. Features for the ENVI Editor.

For this project, the GUI was implemented using python's built-in Tkinter library. Also, other libraries such as Numpy, Matplotlib, Sklearn, SPy, Open-CV, and Pillow were used for data processing and image processing.

The project file structure is listed as follows:

```
.
├── envi_code.py
├── requirements.txt
├── tk.py
└── widget
    ├── gray.png
    ├── customScale.py
    ├── page3.py
    ├── page4.py
    ├── page5.py
    ├── page6.py
    ├── page7.py
    ├── page8.py
    └── page9.py
```

The *envi_code.py* contains all the core codes for reading and writing the ENVI data and the data processing. The *tk.py* is the main driver code for the GUI, which controls the GUI's behavior and layout. The */widget* directory contains all the custom widgets for the Tkinter

app. It is also easily expandable by just simply modifying the *tk.py* and calling the added custom frame widget.

In this project, there are 9 pages and each page contains the user interface of 1 feature, which are listed as follows:

01. **Band reader**: Preview one band of the ENVI file as a grayscale image by dragging a slider. The image can be saved as a PNG file.
02. **RGB reader**: Preview 3 bands of the ENVI file as an RGB image by dragging three sliders. The RGB image can be saved as a PNG file.
03. **Image cropper**: Cropping and rotating the ENVI spectral file and save file either as PNG or ENVI format.
04. **Reflectance reader**: Show the reflectance plots (white-corrected) by selecting the points.
05. **Illumination correction**: Correct the uneven illumination for the ENVI file and save it in either PNG or ENVI format.
06. **Digital inpainting**: This is a process that can be used to remove certain objects in a picture and fill that area based on the neighboring area.
07. **Image registration**: The purpose is to compare or fuse images acquired under different conditions for the same object, such as images from different acquisition devices, taken at different times, from different shooting perspectives, etc.
08. **Principal component analysis**: Principal component analysis aims to use the idea of dimensionality reduction to transform multiple indicators into a few comprehensive indicators. This transformation transforms the data into a new coordinate system so that the first large variance of any data projection is on the first coordinate (called the first principal component), and the second-largest variance is on the second coordinate (the second principal component). Ingredients), and so on. The principal component analysis is often used to reduce the dimensionality of the data set while maintaining the feature that contributes the most to the variance of the data set.
09. **Spectral angle mapper**: The SAM algorithm considers the spectrum of each image element as a high-dimensional vector, and measures the similarity between the spectra by calculating the angle between the two vectors; the smaller the angle, the more similar the two spectra are, and the more likely they belong to the same kind of feature, thus the category of unknown data can be identified based on the spectral angle. In classification, the spectral angle between the unknown data and the known data is calculated, and the category of the unknown data is classified into the category corresponding to the smallest spectral angle.

Methods

This project is a multi-page app, and each page is contained inside a Tkinter *frame* widget. And by clicking the button in the start menu, it will go to the corresponding frame.

For the first function, the grayscale image reader of the ENVI file, first reads the header and stores the information into a dictionary, then the spectral data was read into the NumPy memory map and normalized the data according to the data type that is stated in the header file, then reshape the data according to the interleave. After the data is opened and

processed, the specific band can be read and presented inside the canvas widget in the current frame. But some transformations are needed to fit into the canvas. The custom slider widget slider can be used to control the band number.

For the second function, the RGB mode is similar to the first function, the difference is that there are three sliders and each controls the R, G, B value, after three bands are selected, it will be merged into an RGB image and presented in the canvas.

For the image cropper, this is simply just slicing the spectral cube based on the user's mouse coordinates and rotation of the spectral cube using the NumPy rot90() function.

For the reflectance reader, the user will have to choose points (max of 10 points), and if the white reference data is within the same folder, it will automatically do the white correction and present the reflectance data as a line plot in a separate window according to the sequence of clicking.

The illumination correction is done by using OpenCV's CLAHE (Contrast Limited Adaptive Histogram Equalization) algorithm. CLAHE operates on small regions in the image, called tiles, rather than the entire image. The neighboring tiles are then combined using bilinear interpolation to remove the artificial boundaries. This algorithm can be applied to improve the contrast of images.

The digital inpainting is done by using the cv2.inpaint() function, in this GUI, the Navier-Stokes method was used. The algorithm is based on fluid dynamics and uses partial differential equations. The basic principle is heuristic. It first travels along the edges of the known area to the unknown area (because the edges are continuous). It continues with isolines (lines connecting points with the same intensity, just as contours connect points with the same height) while matching the asymptotic vectors at the boundaries of the repaired region. For this purpose, some methods from fluid dynamics are used. After obtaining the color, fill the color to reduce the minimum difference in the area. The user has to draw the area that needs to be inpainted and then click preview to see the results.

For the image registration, firstly we create an ORB detector with defined features (default set as 5000). Then compute the key points and descriptors. Then use Hamming distance to match features between reference and target image. Then select the top 90 % matches and then find a perspective transformation between two images using the RANSAC-based robust method.

For the principal component analysis, the sklearn is used for PCA and the data is preprocessed in a way that all the pixels within one band is flattened into one column so the data is now transformed into a 2d matrix with the row (lines*sample) and column (bands) then perform the PCA analysis to get the results.

For the spectral angle mapper, the spectral angle can be calculated using the following formula:

$$\alpha = cos^{-1}\left(\frac{\sum_{i=1}^{nb} t_i r_i}{(\sum_{i=1}^{nb} t_i^2)^{\frac{1}{2}}(\sum_{i=1}^{nb} r_i^2)^{\frac{1}{2}}}\right)$$

where as:

- α = spectral angle between the reference and the target pixel
- nb = number of spectral bands
- t = vector of the target pixel
- r = vector of the reference pixel

Below is the function for the SAM using numpy matrix multiplication. Firstly, we need the dot product of the whole spectral image and the selected pixel (tr), then the dot product of the image itself (rr), however, it is time consuming for the large file and it can be saved as a global variable for later use. Then calculate the dot product of itself and calculate the spectral angle relative to selected pixels.

```
def sam(array, target, rr=None, threshold = 0.1):
    tr = np.dot(array,target.T).reshape(array[:,:,0].shape)
    if rr is None:
        rr = (array*array).sum(-1)
    tt = np.dot(target,target.T)
    cos_alpha = tr/(np.sqrt(tt)*np.sqrt(rr))
    cos_alpha[cos_alpha > 1.0] = 1.0
    cos_alpha[cos_alpha < -1.0] = -1.0
    alpha = np.arccos(cos_alpha)
    # The value smaller than threshold will be 1, else 0
    output = (alpha <= threshold).astype(np.int_)
    return output
```

IT Project - User Guide

Usage

This project uses the pipenv to manage the virtual environment and packages.

First install pipenv:

```
pip install pipenv
```

Then using pipenv to create the virtual environment and install all the necessary packages:

```
pipenv install
```

Finally, run the GUI program:

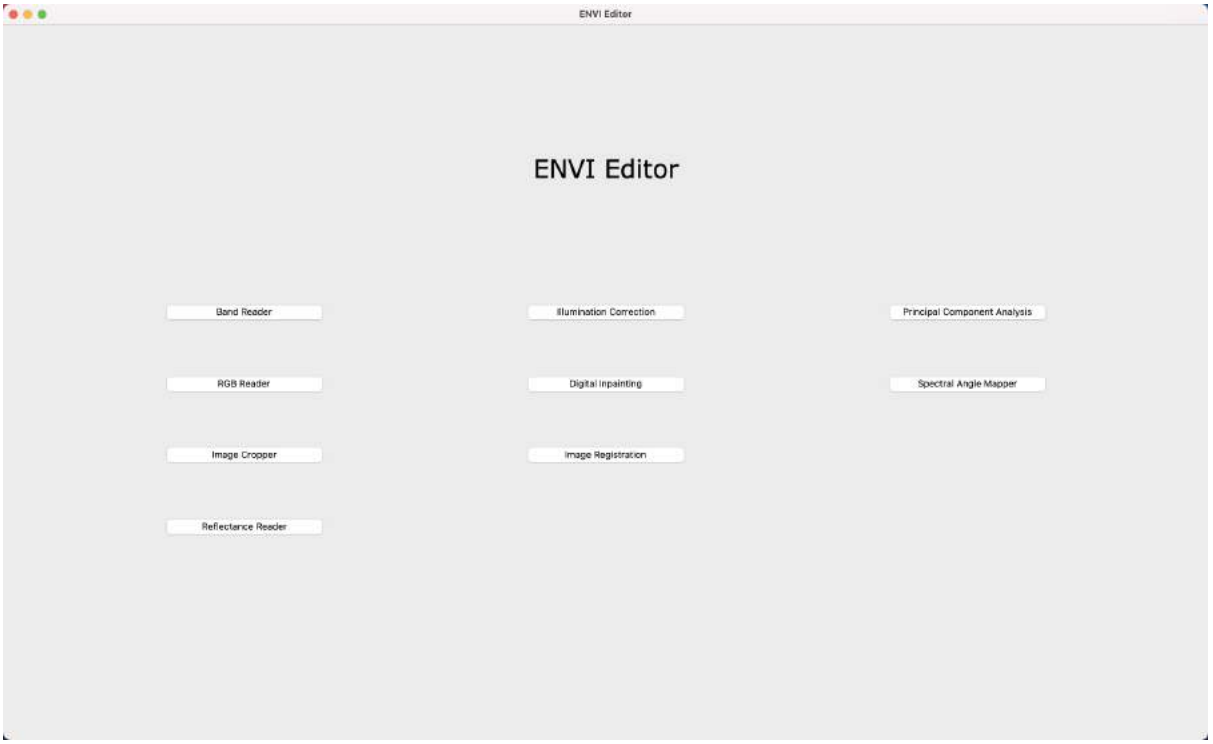
```
pipenv run python3 tk.py
```

User Interface

The video version of the user guide is also uploaded to YouTube via this link.

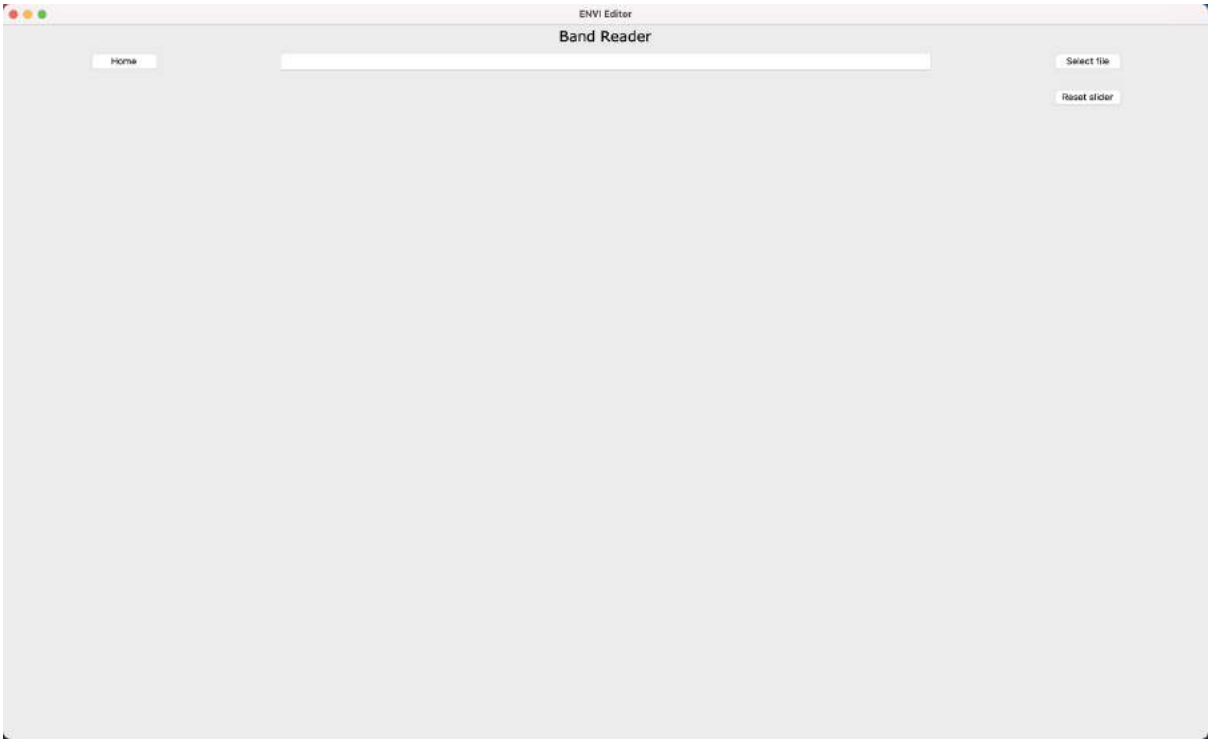
Menu page

The first page is the manu page which has just 9 buttons that can go to the corresponding pages.

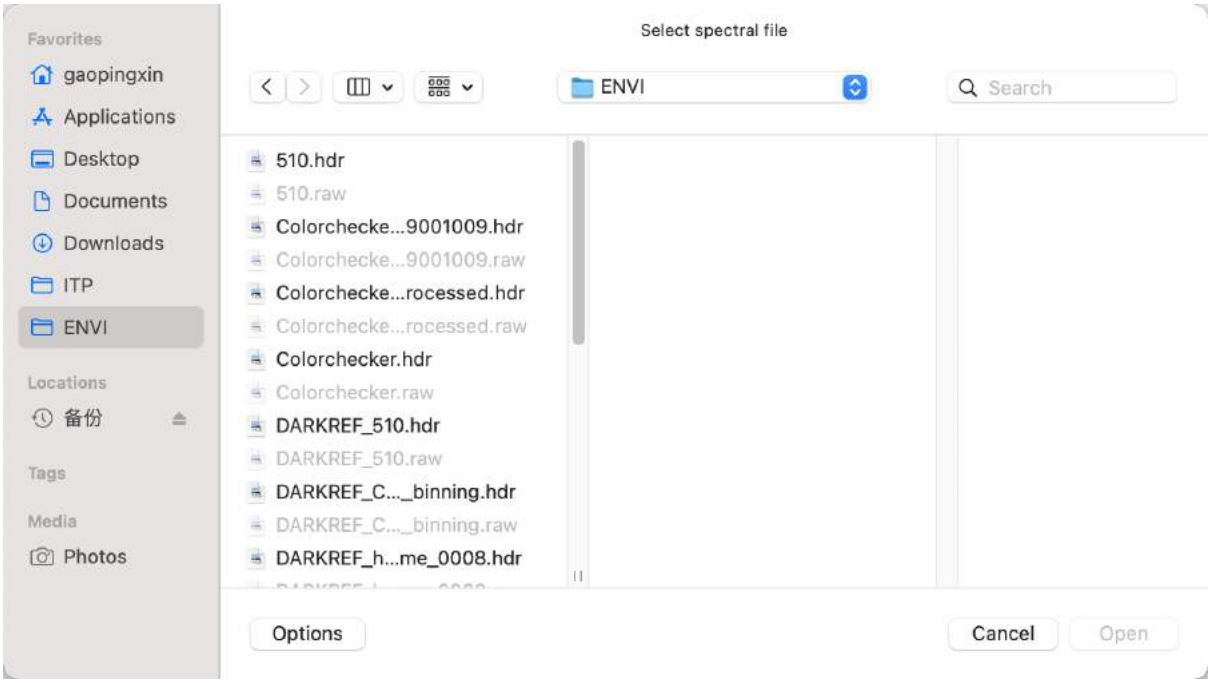


Band Reader

In this page, there is a “Home” button, which will back the menu page after clicked. It is accessible on every page.

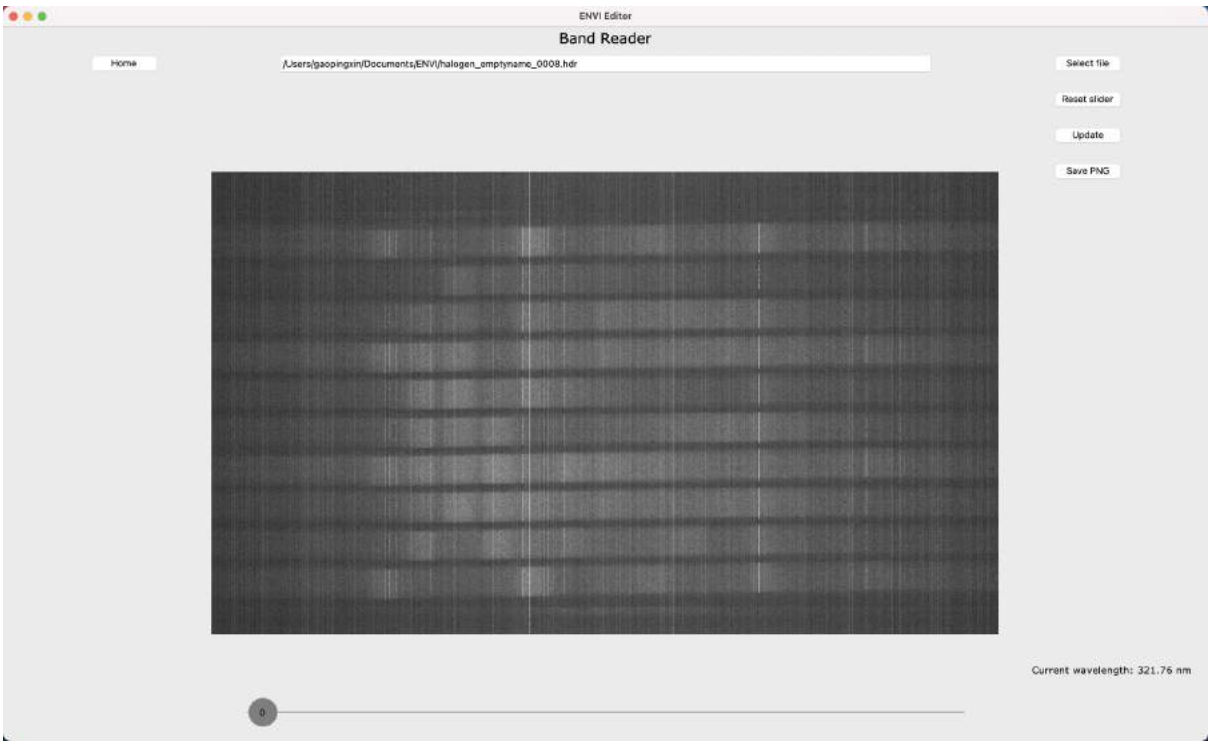


Then in the center there is a text input area, the file path will show in this area. Users can select files by clicking the “Select file” button. Then a window will pop-up as shown below.



The user can just select the ".hdr" file and it will automatically read the corresponding data file ('img', 'dat', 'sli', 'hyspex', 'raw'). The **file name has to be exactly the same** in order to read the data.

After the file has been selected, some other widgets will appear. The default band is 0, which means it will preview the first band as a grayscale image. The slider can be used to change the band and then click the "Update" button, the image will appear in the center area. The "Reset" button will reset the band number to 0. In the right bottom corner, it will show the current band's wavelength. The "Save PNG" will save the current image as a .png file in the same directory as the ENVI file with the same file name as the ENVI file and a "_grayscale_bandNo" suffix will be added to the file name to avoid file name conflict. In this case the saved file name is "halogen_emptyname_0008_grayscale_0".


ENVI Editor
Band Reader
Home
/Users/gaopingxin/Documents/ENVI/halogen_emptyname_0008.hdr
Select file
Reset slider
Update
Save PNG
Current wavelength: 321.76 nm
0

RGB Reader

The RGB reader is similar to the Band reader, the only difference is that there are three sliders and the default band will be the “default bands” in the header file. The image will appear as an RGB file. The new file name will add the suffix “_rgb_(R band,G band,B band)” to the original file name.
In this case is “halogen_emptyname_0008_rgb_(563,327,161)”.

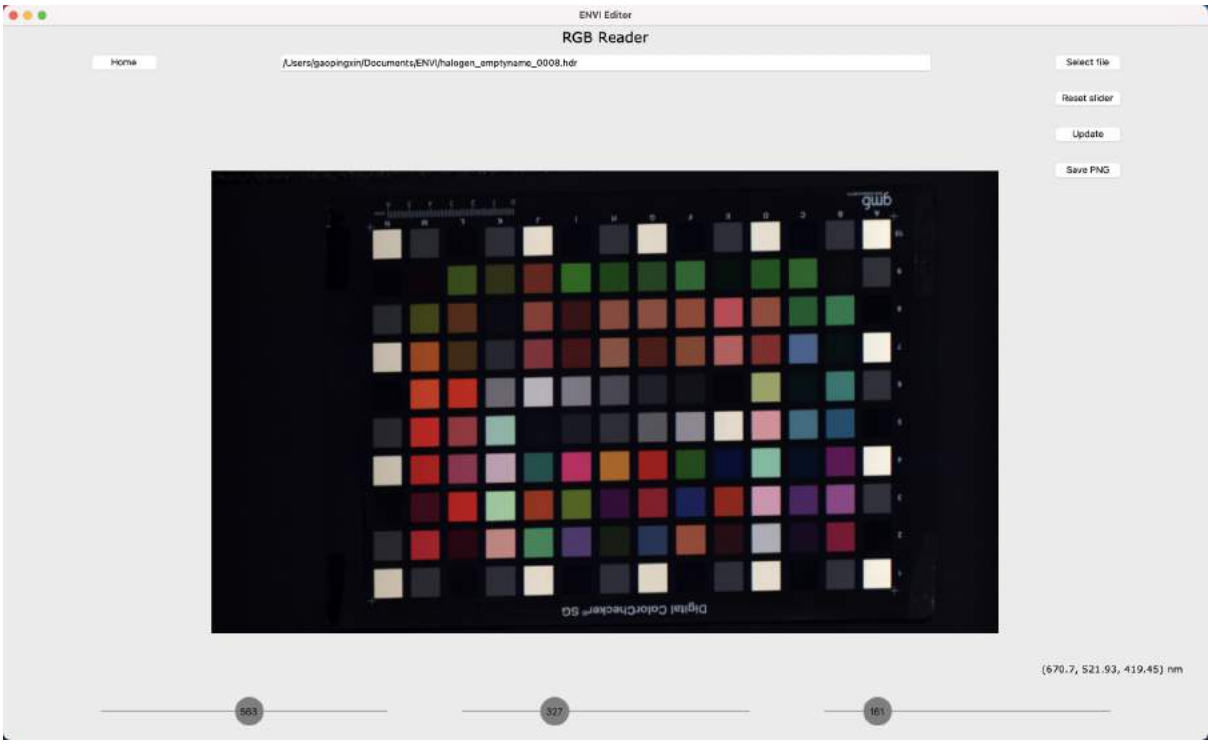
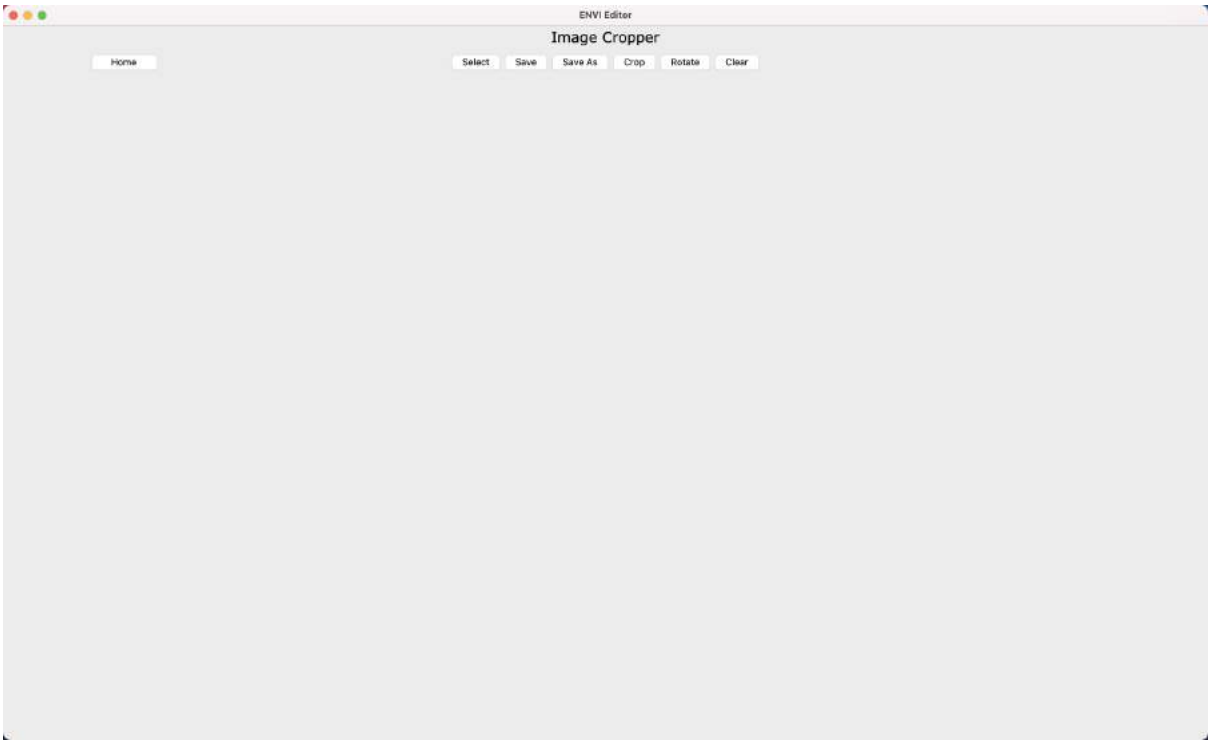


Image Cropper

In the image cropper, the interface is slightly different from the first two pages.



After the ENVI file was selected, a few more widgets will be added to the control bar.
First the “Save” button will save the png file of the current band and the suffix of the current timestamp will be added to the filename. The “”Save As” button will open a new window so that the user can choose where and the file name of the image will be saved.

The “Crop” button will enable the cropping mode, which means the user can just press the mouse on the image and drag it to the desired position to get the cropped image.

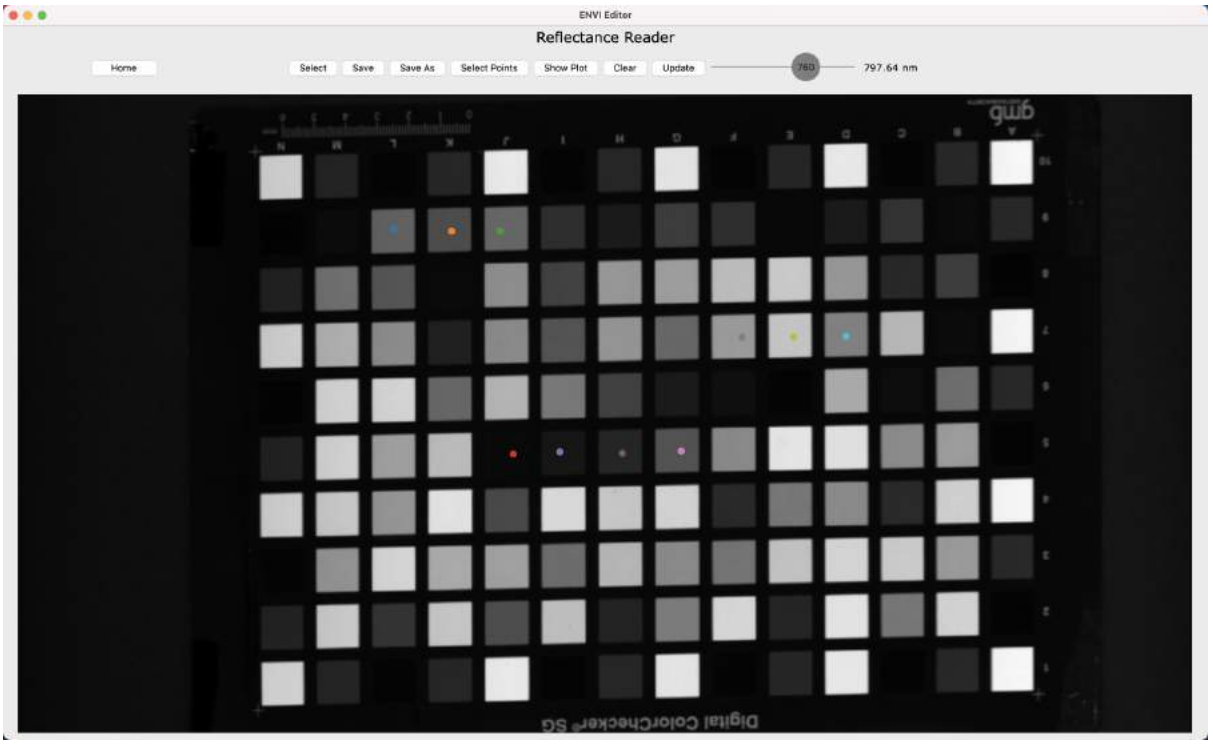
The “Rotate” button will rotate the image clockwise by 90 degrees.

The “Clear” button will remove all the operations and return to the band 0.

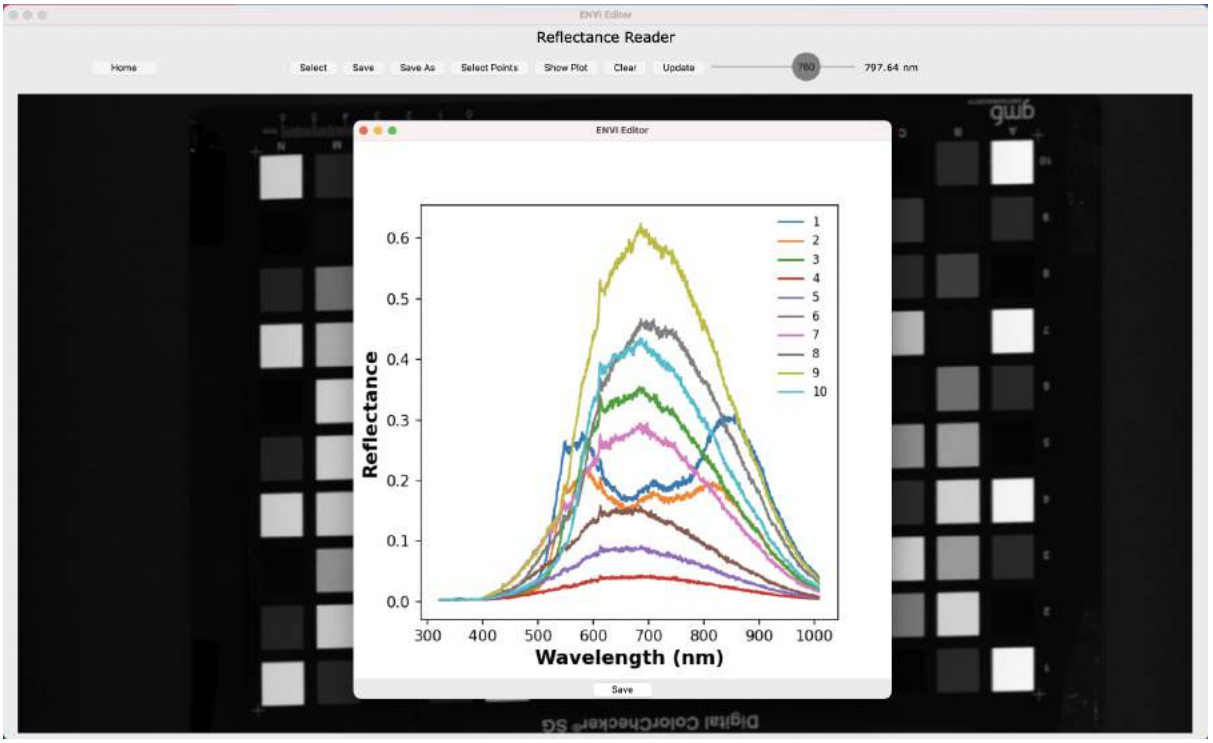
The “Apply to all and save” button will apply all the operations to the whole image and save it as an ENVI file. The file will be saved in the same folder and the timestamp suffix will be added to the file name.

Reflectance Reader

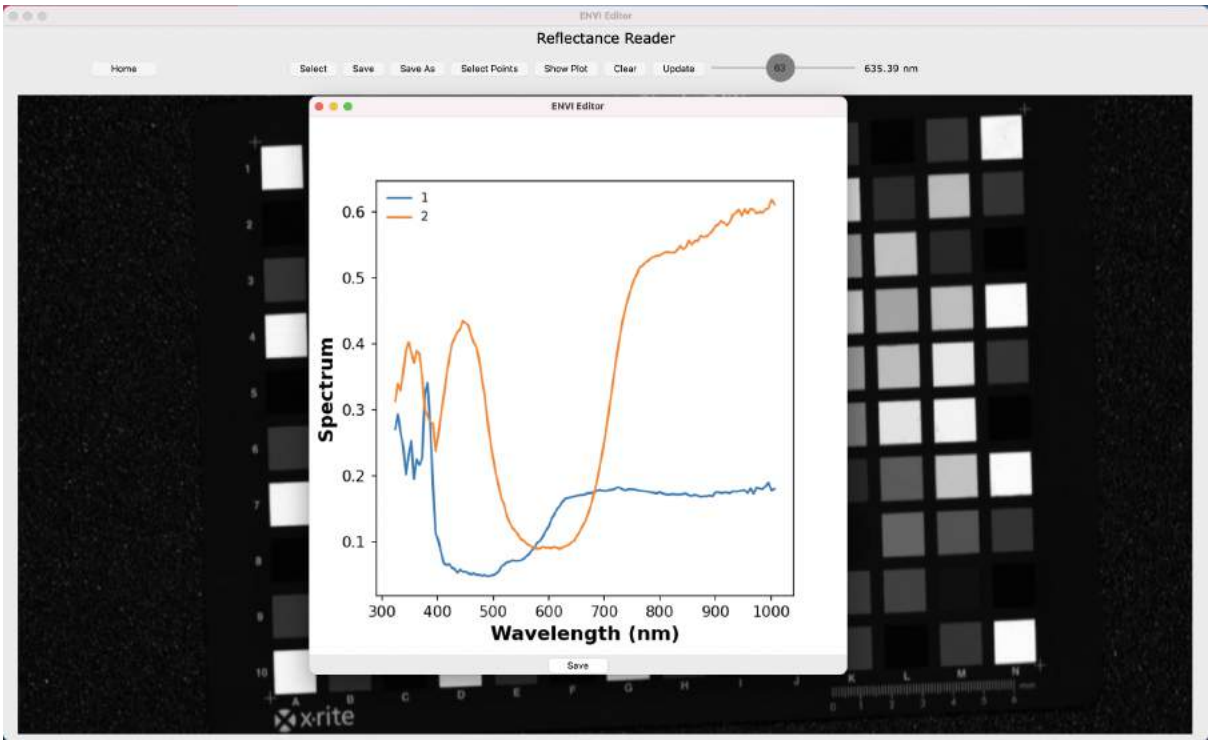
In this page, users can select a maximum of 10 points by clicking “Select Points” and the points will also be marked in the image.



And then click the “Show Plot” it will pop-up a matplotlib image with the reflectance plot. And it can be saved as a png file.

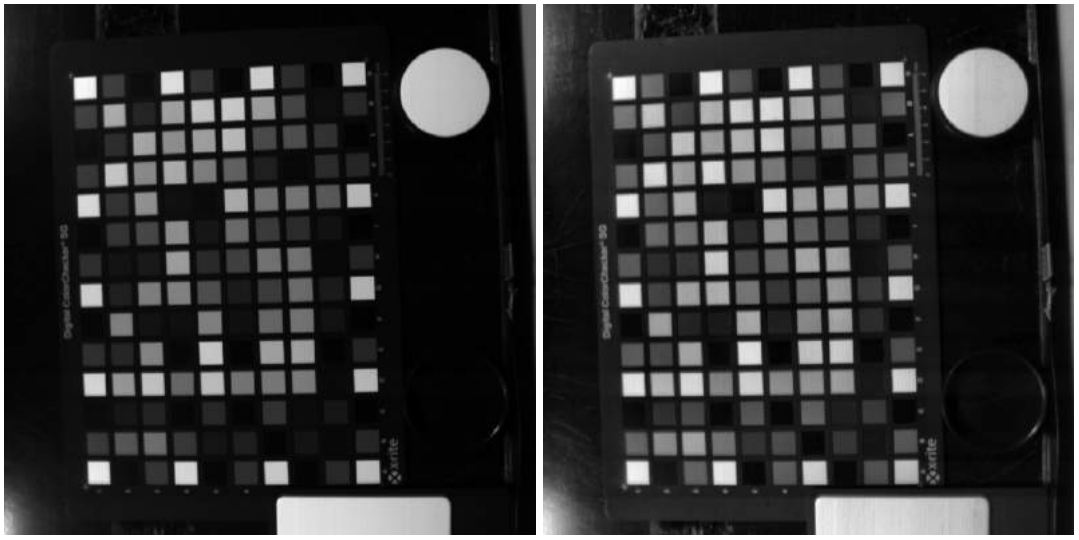


This function can automatically do the white correction only if the name of the file has a “WHITEREF_” prefix and the same file name. Otherwise, it will simply show the spectrum.



Illumination Correction

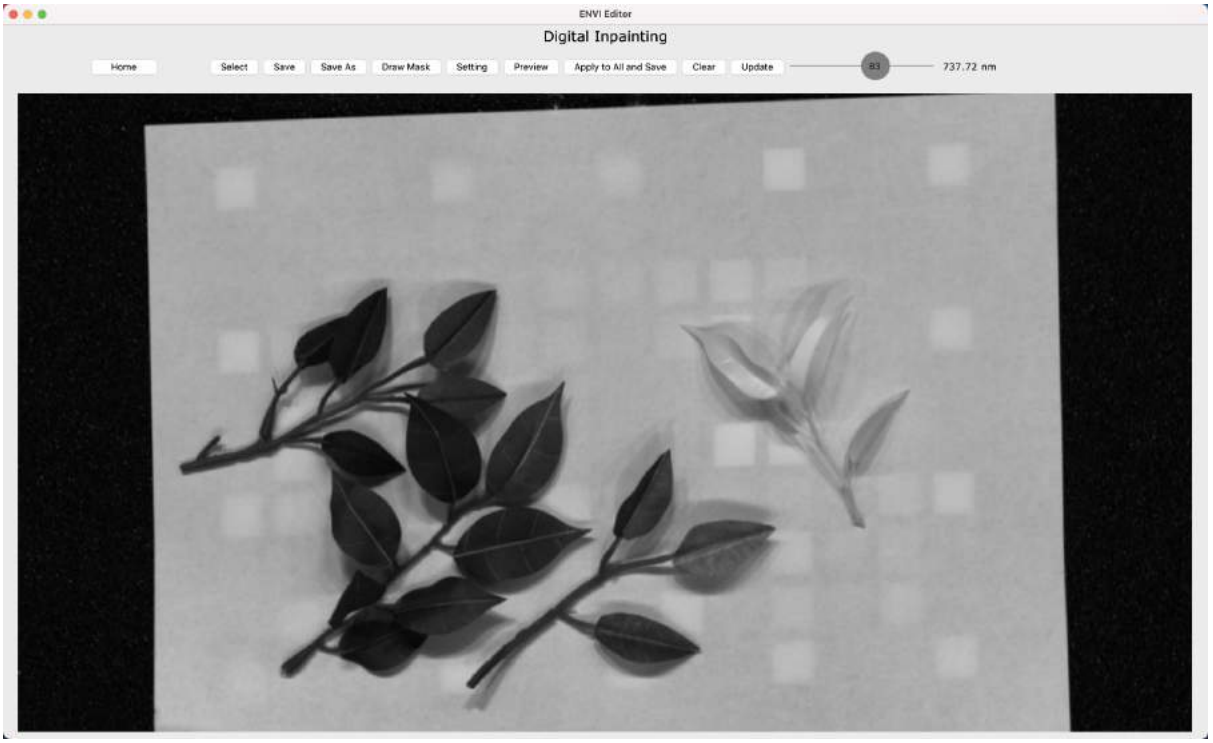
This CLAHE (Contrast Limited Adaptive Histogram Equalization) method gives better results. CLAHE divides the image into small blocks 8x8 which are equalised as usual using equalize Hist.

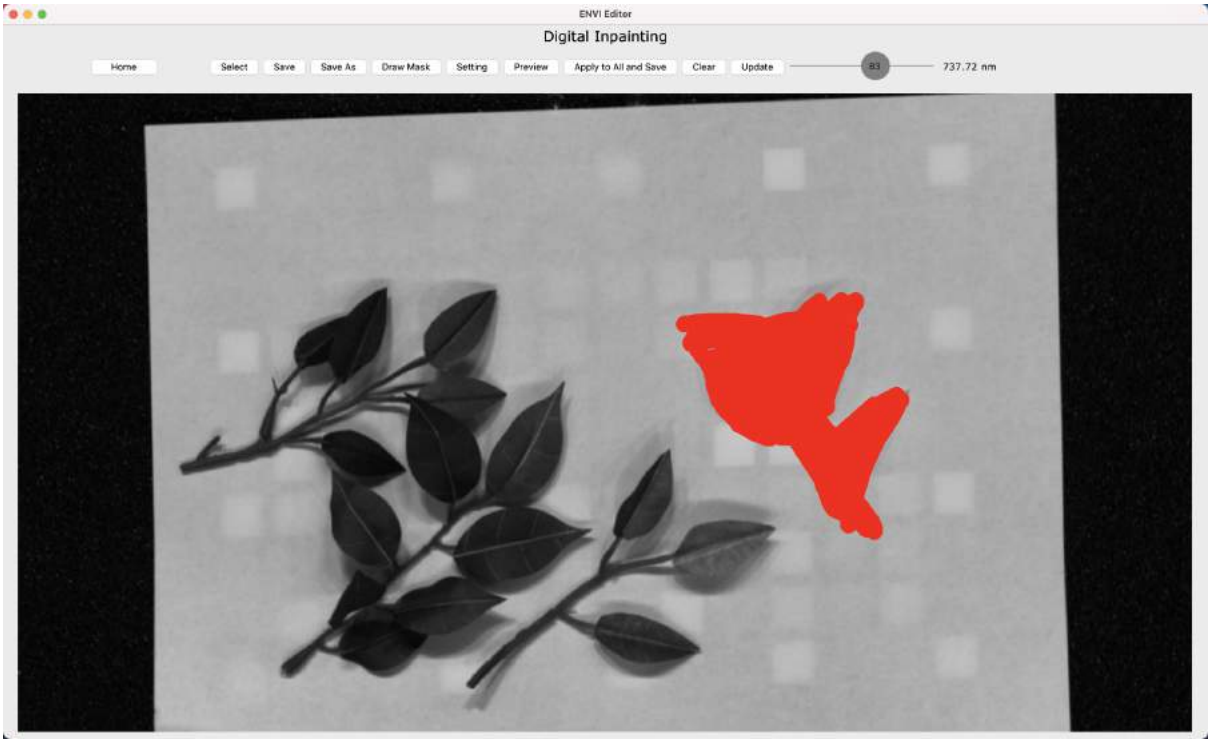


Before After

Digital Inpainting

In this page, users need to draw a mask manually for the algorithms to start processing. After clicking the “Draw Mask” button, it will enable the drawing method and the user can draw the mask to remove the area that needs to be removed. The brush size can be changed by clicking the “Setting” button.





After clicking the “Preview” Button, the result appears in the canvas. However, this method is not perfect and it might have some flaws. This operation can be also applied to the whole ENVI file by clicking the “Apply to All and Save” button.

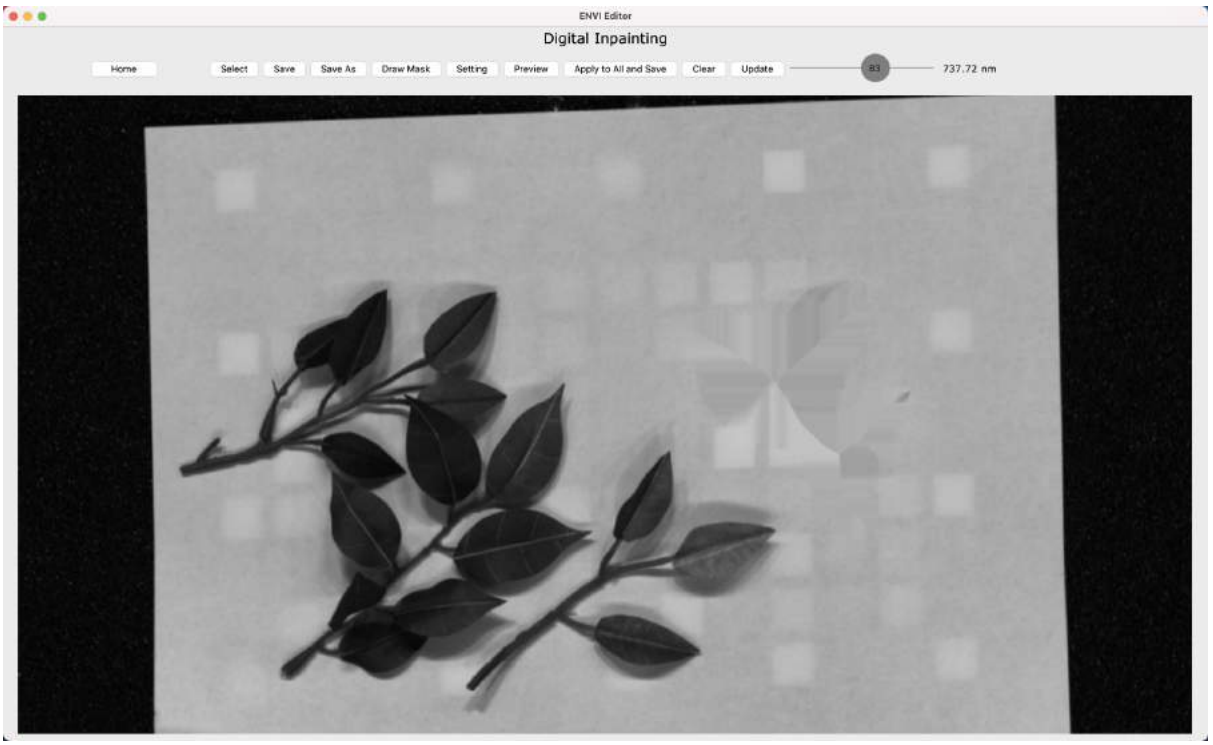
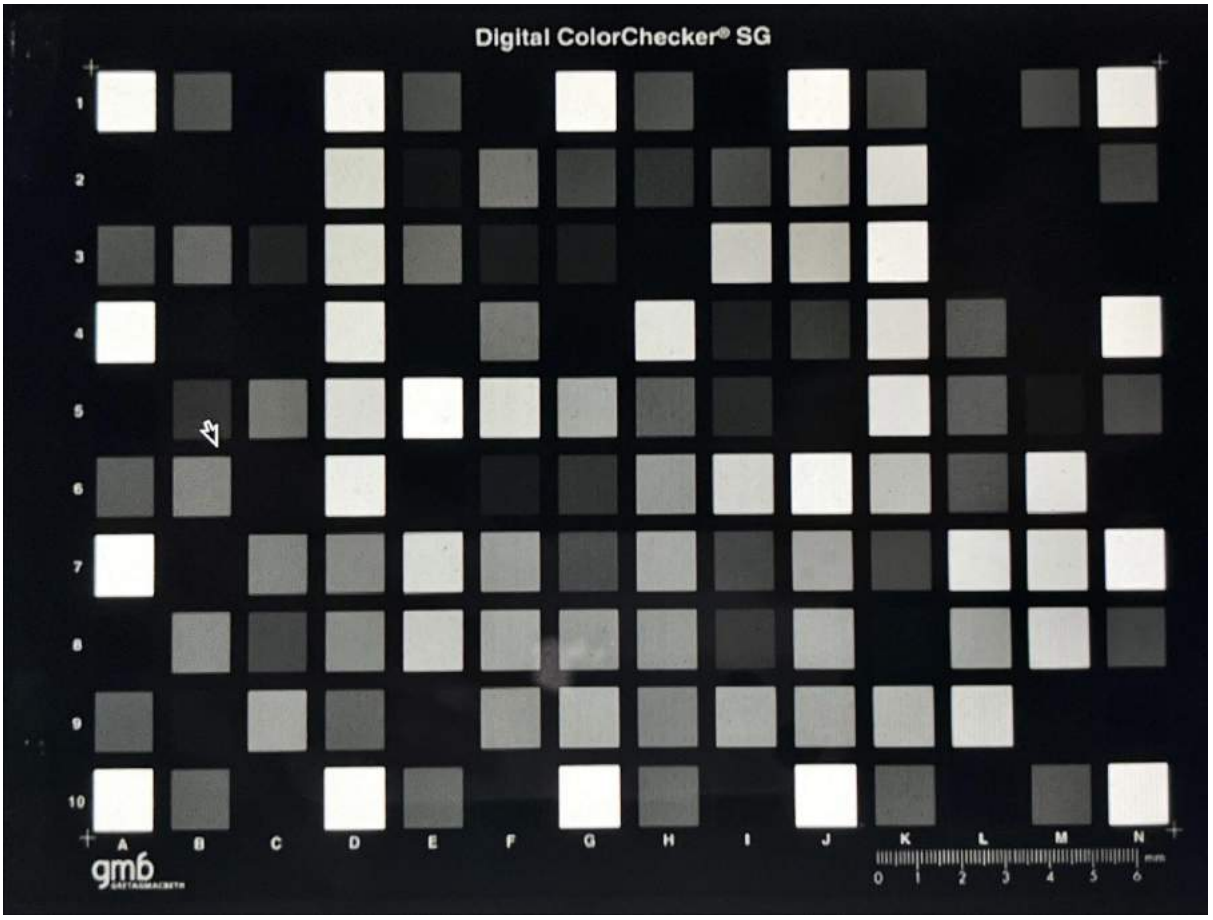


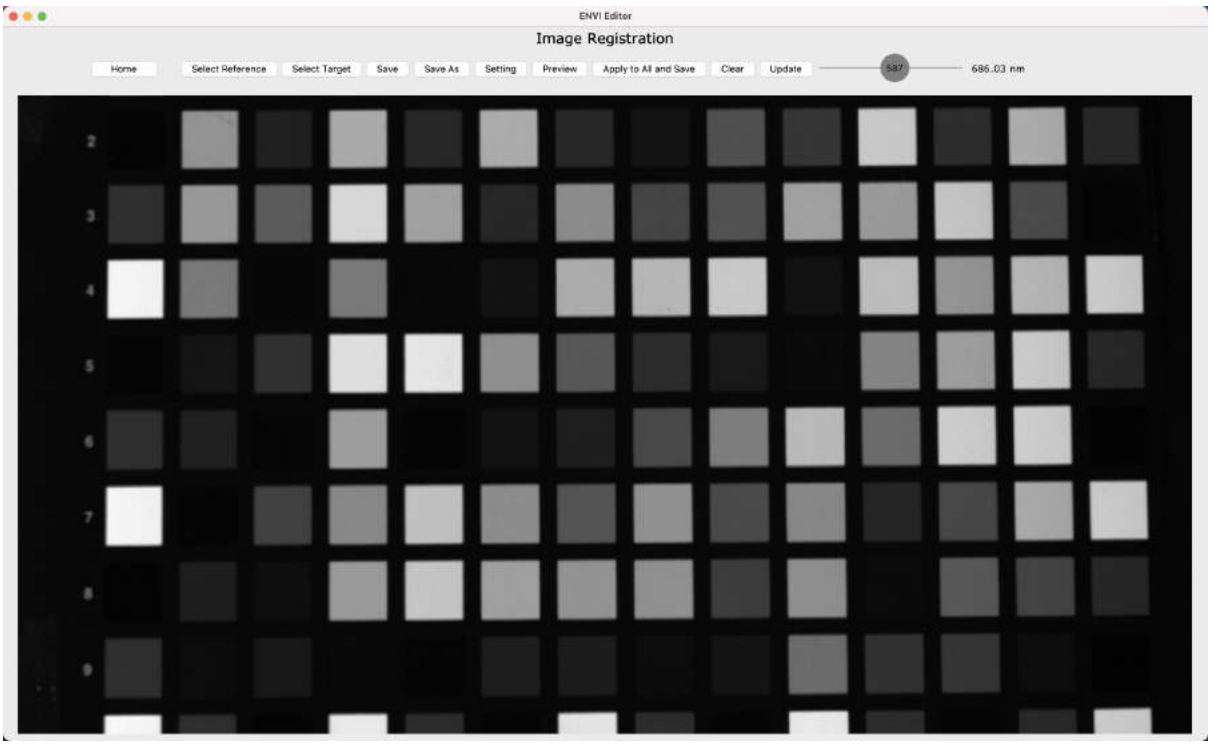
Image Registration

In this page, the user has to create a reference image using the target ENVI file that is needed for the registration. In the test case, I use the Microsoft Lens to process the “halogen_emptyname_0008.raw”. And the processed image is shown below.



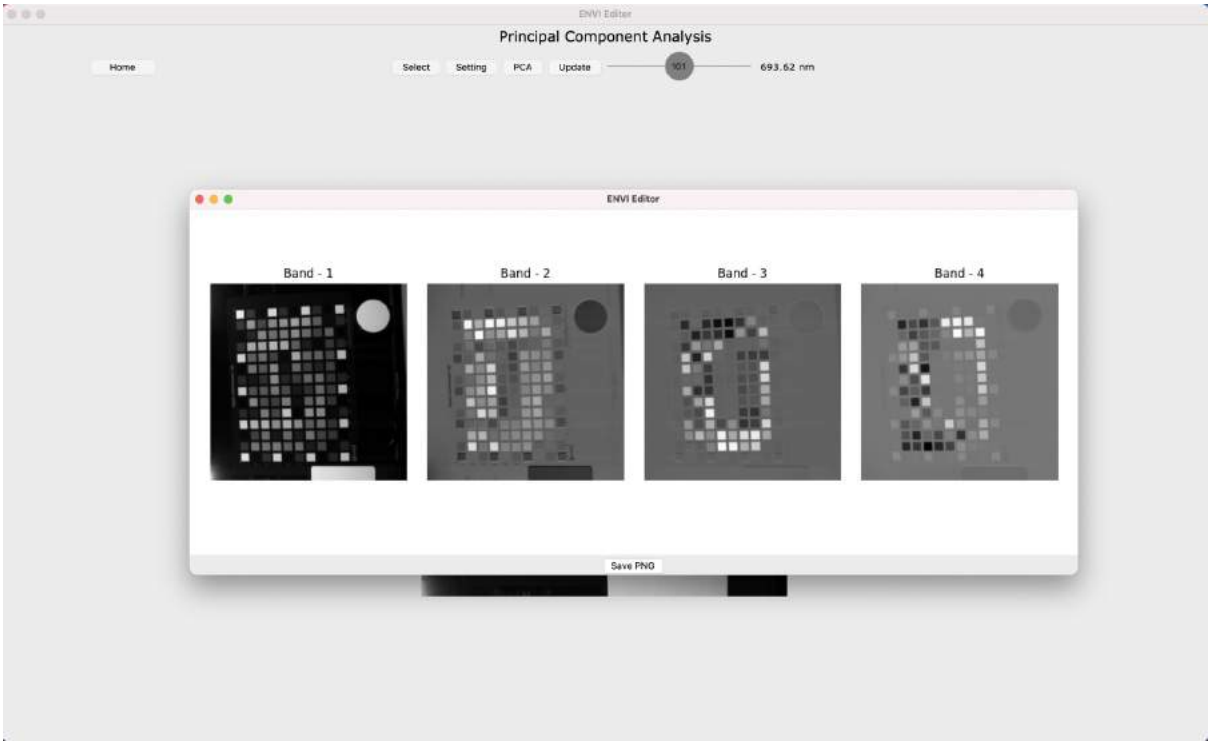
Reference image

After the reference image and target ENVI file was selected, the user can just press the “Preview” button and in the process the image will appear and it will look similar to the reference image. The user can also change the setting by changing the number of detection features (default is set as 5000). Then it can be applied to all the bands and saved as ENVI file.

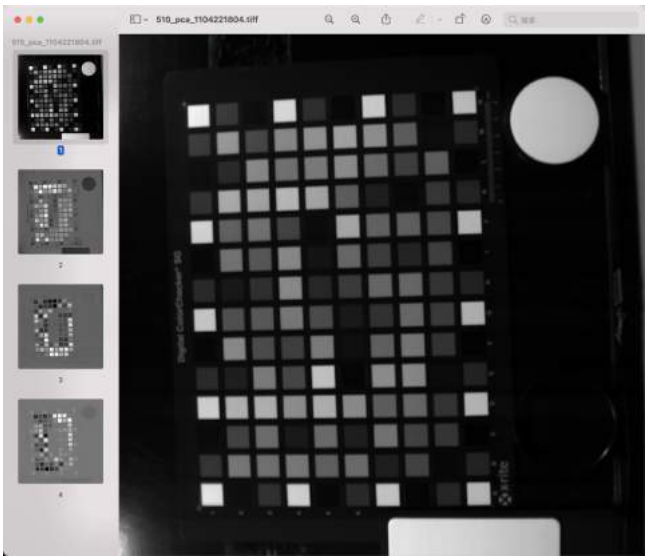

ENVI Editor
Image Registration
Home
Select Reference
Select Target
Save
Save As
Setting
Preview
Apply to All and Save
Clear
Update
587
686.03 nm

Principal Component Analysis

In this page, the user can change the number of components or the variance, if the input is an integer larger than 1 or the float number smaller than 1, respectively. The file can be saved as a png or tiff file.

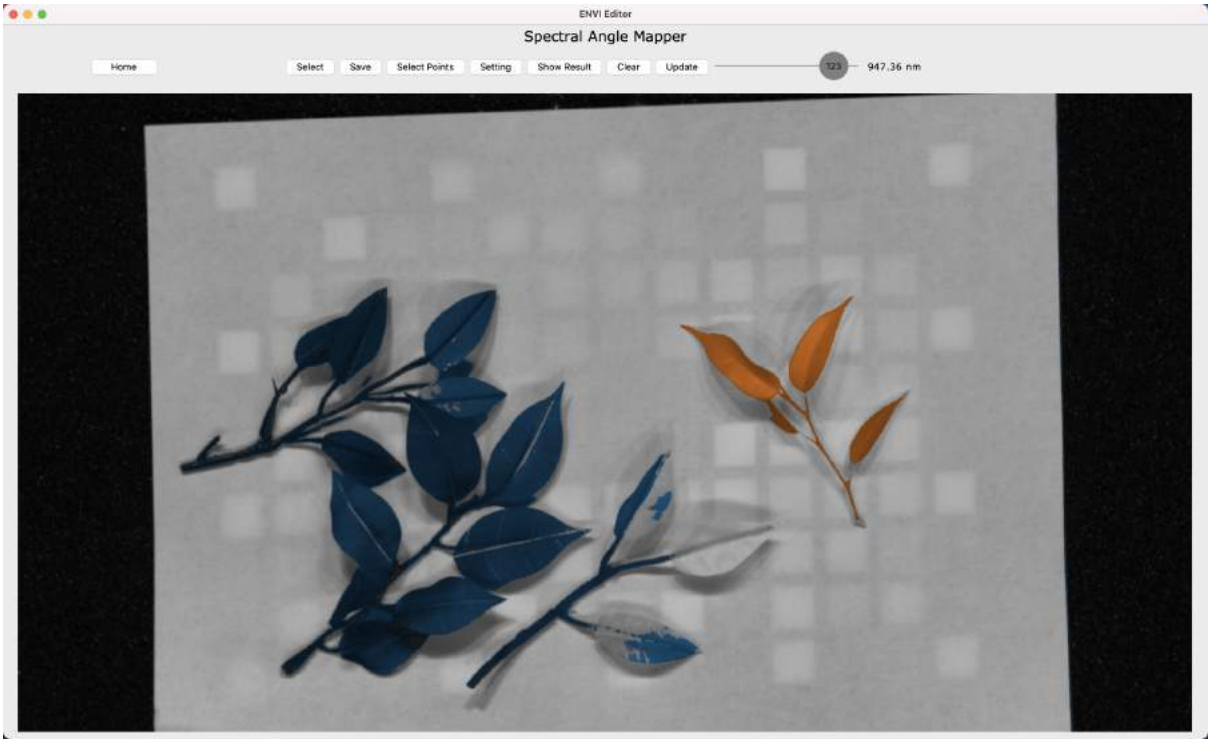


Preview of the saved .tiff file.



Spectral Angle Mapper

In the spectral angle mapper, after the file is loaded, the user has to choose at least 1 point of interest as the reference point. The maximum of selected points is 10. And then click “Show Result”, the program will calculate the spectral angle between the reference point and the rest pixels, and the areas which have the spectral angle smaller than the threshold (default as 0.15, changeable by clicking “Setting”) will be marked with the corresponding color.



References

Lundh, F. (1999). An introduction to tkinter. *URL: Www. Pythonware. Com/Library/Tkinter/Introduction/Index. Htm*.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*, 357–362. https://doi.org/10.1038/s41586-020-2649-2

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95.

Clark, A. (2015). *Pillow (PIL Fork) Documentation*. readthedocs. Retrieved from https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf

Pedregosa, F., Varoquaux, Ga"el, Gramfort, A., Michel, V., Thirion, B., Grisel, O., … others. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*(Oct), 2825–2830.

malibursali/simpleImageEdtor: Simple Image Editor With Python

Python Tkinter slider customization

Tkinter, overlay foreground image on top of a background image with transparency

victorgzv/Lighting-correction-with-OpenCV: Cleaning an image to obtain a more usable document using Python

Python | Image Registration using OpenCV

Image Inpainting using OpenCV

Image inpainting with OpenCV and Python

Python | Image Registration using OpenCV

Dimensionality Reduction in Hyperspectral Images using Python | by Syam Kakarla

In Depth: Principal Component Analysis | Python Data Science Handbook

PCA on HyperSpectral Data. A Beginner friendly tutorial on… | by Richa Dutt

cjaca/Hyperspectral-Image-Analysis: Spectral Angle Mapper used to classify Hyperspectral Image