Project Introduction

Programming Language： Python

Software Requirements ： Anaconda + cuda

Project Address： https://github.com/ryougishiki02

```
DeepEM
├── image
│   ├── element_sort
│   │   └── • element.py
│   ├── strain
│   │   ├── • pkg.py
│   │   └── • strain.py
│   ├── test
│   ├── • remove_background.py
│   ├── • remove_denoise.py
│   └── • pkg.py
├── video
│   ├── • crop.py
│   └── • refine.py
├── model
│   └── unet
│       ├── unet3
│       │   ├── • denoise
│       │   └── • segment
│       ├── unet5
│       │   ├── • denoise
│       │   └── • segment
│       └── • pkg.py
└── • requirements.txt
```

Image background removal, denoising, element analysis, strain analysis

Video cropping, frame extraction (denoising and drift correction will be added later)

Neural networks and parameters

Table of Contents

STEM Image Background Removal

remove_background.py

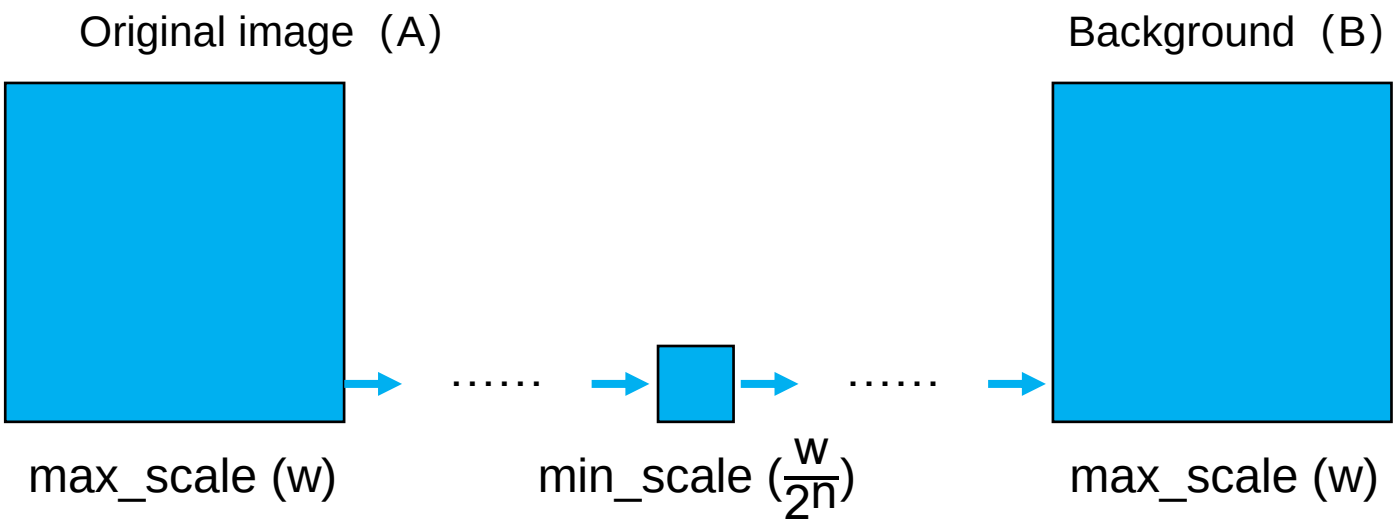
```python
from image.pkg import Backgroundremover, load_img

image = load_img()

# remove background
#Define the scaling parameters and remove the background from the image
remover = Backgroundremover(image, max_scale=2048, min_scale=32)
image = remover.remove()
remover.visualize()  # 可视化去背景后的图像
remover.save()       # 保存去背景后的图像
```

Scope of Application: STEM, TEM images (any size)

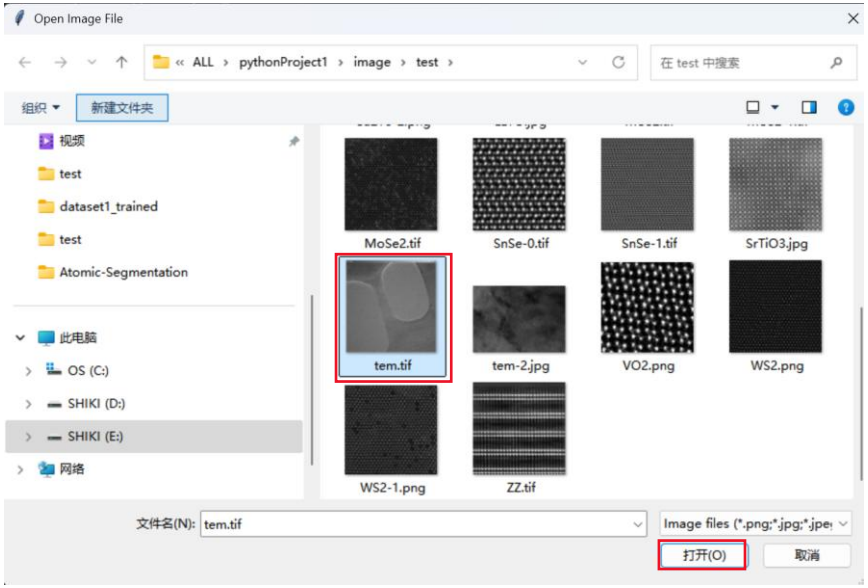
Adjust Parameters: Maximum and minimum image sizes

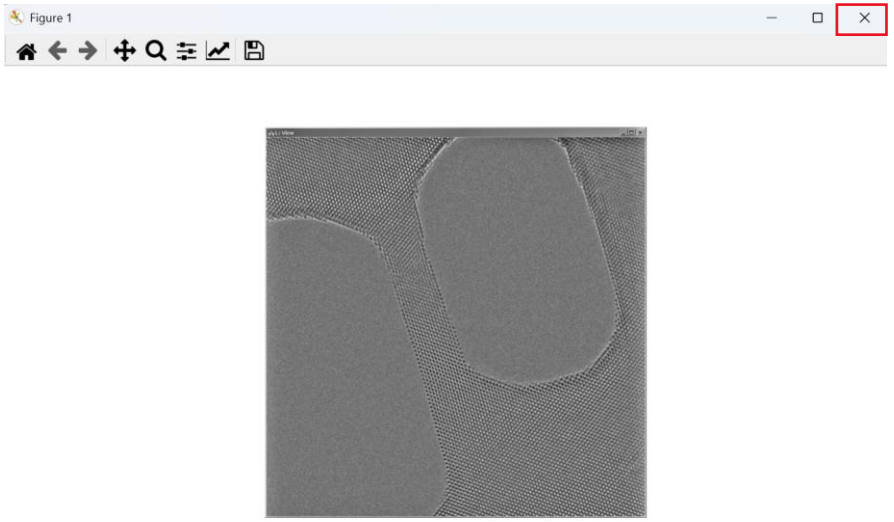
运行 'remove_background (1...'(U) Ctrl+Shift+F10

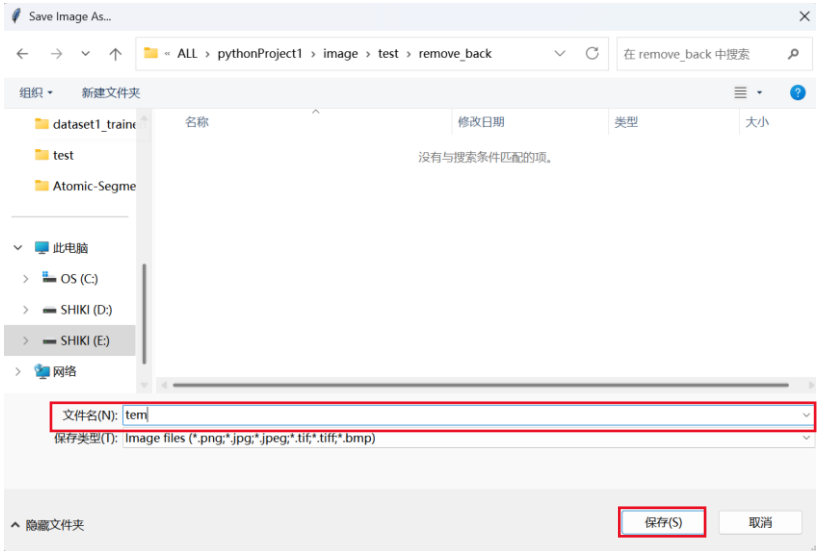
Principle: Background-removed image （C） = A -B



STEM Image Background Removal







Process:

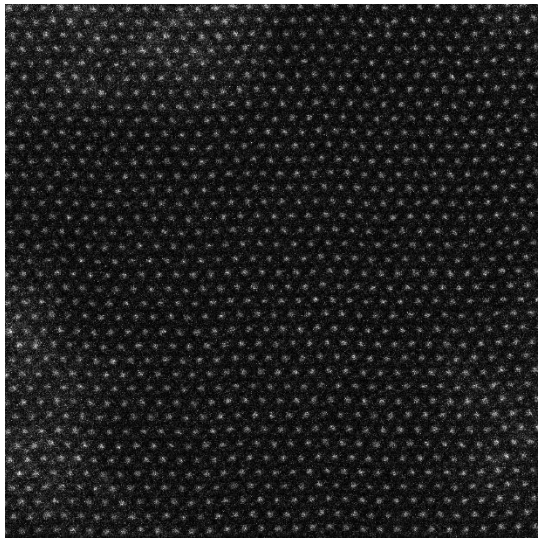
Select image - Preview output image -

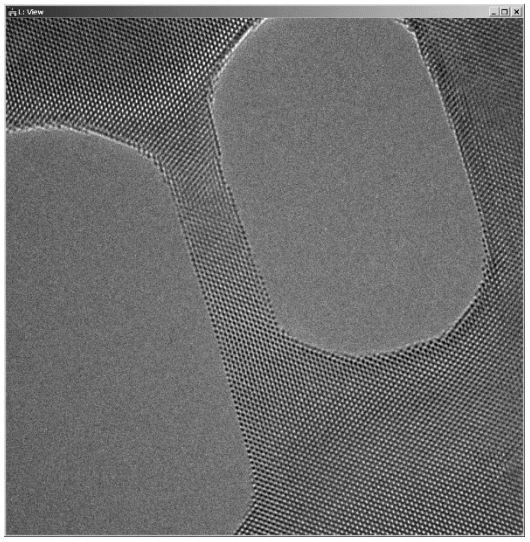
Select storage path

STEM Image Background Removal

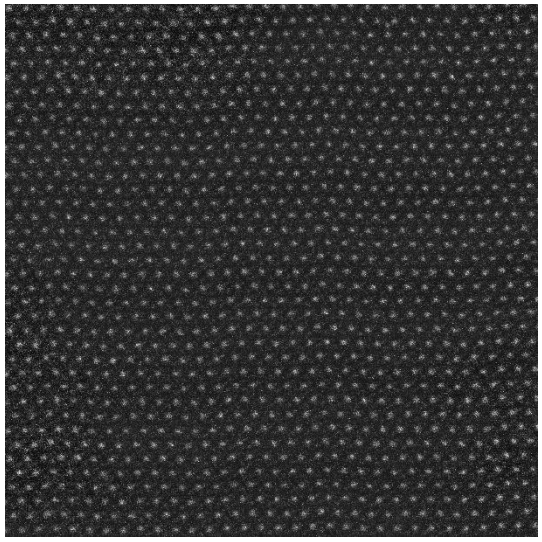
Display:

Original image





Processed image



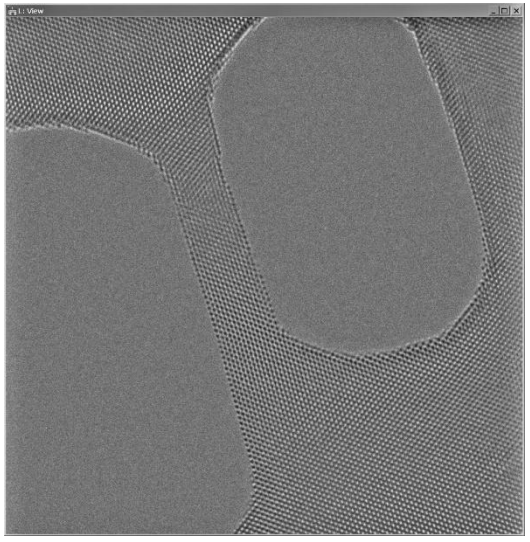


Table of Contents

STEM Image Denoising

remove_denoise.py

```python
from image.pkg import Backgroundremover, load_img, Prediction, Average_img
from model.unet.unet import Unet
from model.unet.unet3 import Unet as Unet3

# load Image
image = load_img()

# remove background:
# Define the scaling parameters and remove the background from the image
remover = Backgroundremover(image, max_scale=2048, min_scale=16)
image = remover.remove()
remover.visualize()  # 可视化去背景后的图像

periodicity = 2
stride = 6/8
batch_size = 8
# periodicity: number of boxes in one row or column; stride: stride of the sliding window;
# model: Unet(n_classes=1); batch_size: batch size for prediction
denoiser = Prediction(image, periodicity, stride, Unet(n_classes=1), batch_size)
ori_image, denoised_image = denoiser.denoise()
denoiser.visualize(denoised_image)  # 可视化去噪后的图像
# denoiser.save(denoised_image)

Average_img(denoised_image, ori_image, windows_size = 768)
```

Background removal (omitted))

Denoising

Image superposition

Scope of Application: STEM images (any size)

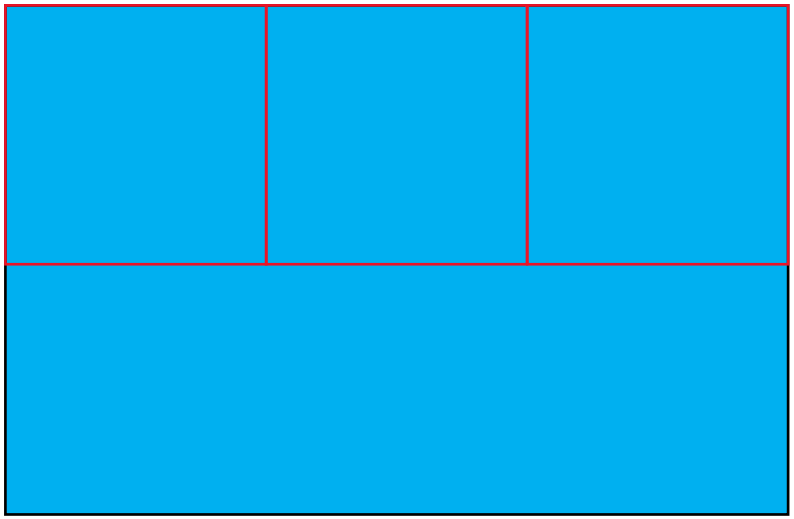
Usage: Right-click - Run

STEM Image Denoising

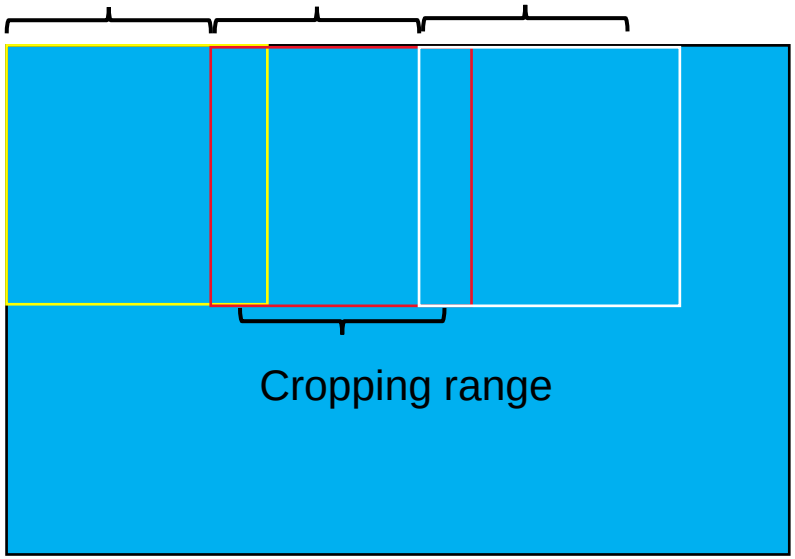
Main Parameters

```
14    periodicity = 2
15    stride = 6/8
16    batch_size = 8
```

periodicity (eg.3 as below)

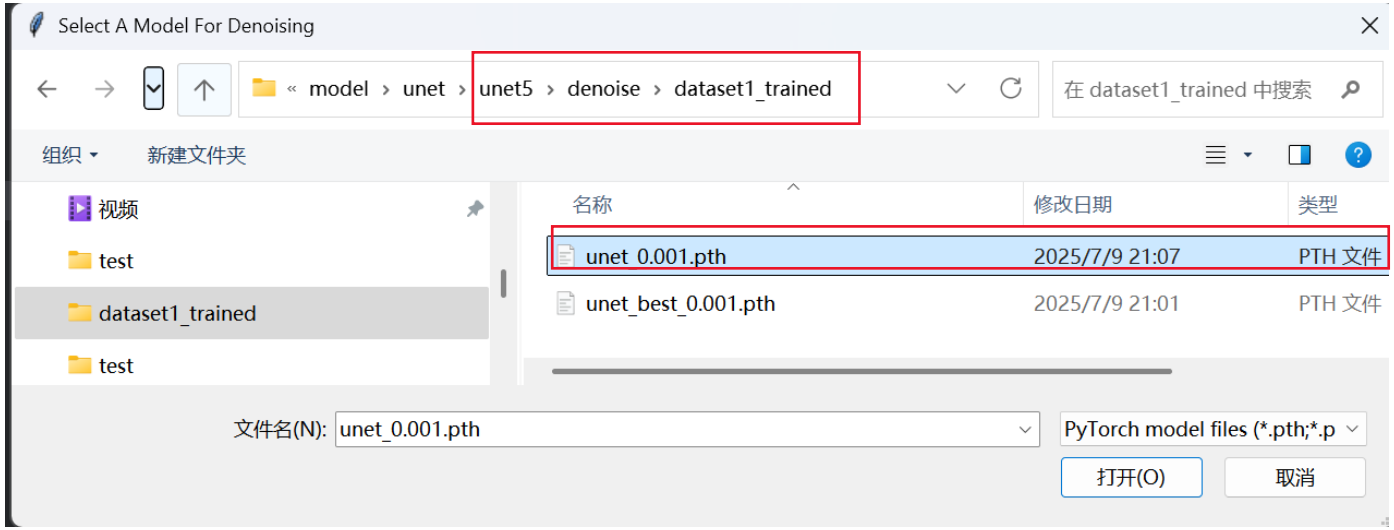


stride



STEM Image Denoising

Process:
1.Select image - 2. Preview background-removed image - 3. Select model weights (weights correspond to the selected model) - 4. Preview denoised image

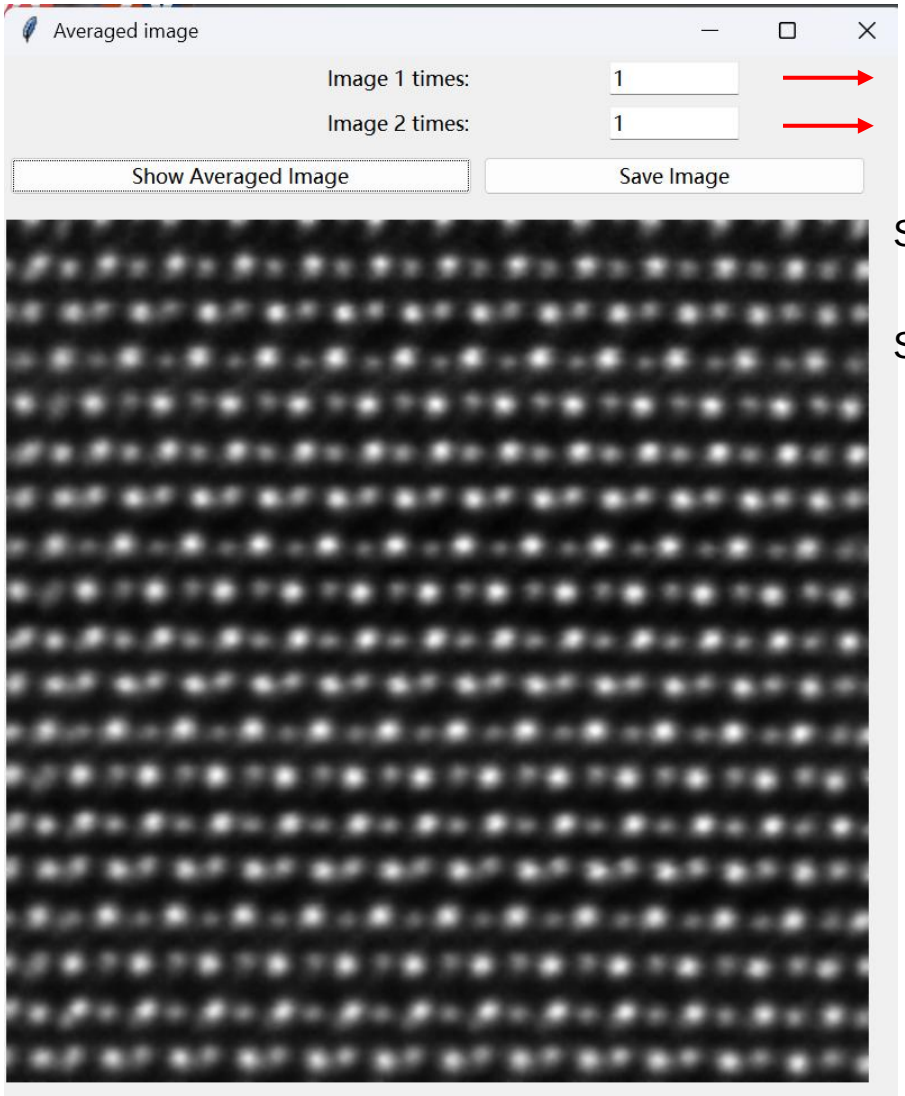



```
19  denoiser = Prediction(image, periodicity, stride, Unet5(n_classes=1), batch_size)
```

Unet3 or Unet5

STEM Image Denoising

5、Image superposition



Number of denoised image superpositions

Number of original image superpositions

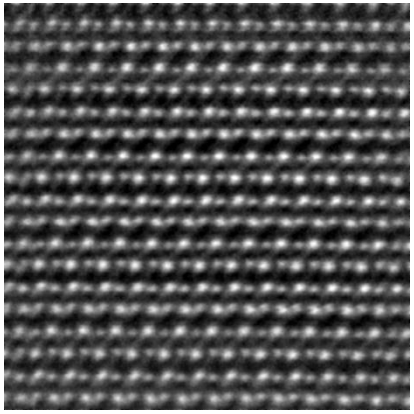
Show Averaged Image

Save Image

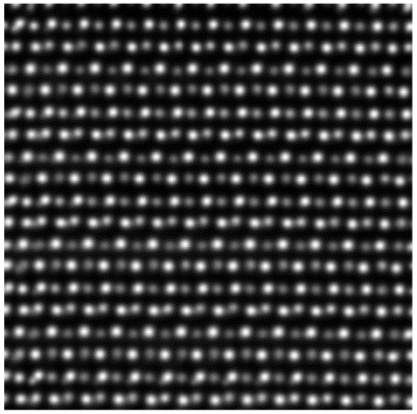
STEM Image Denoising

Display:

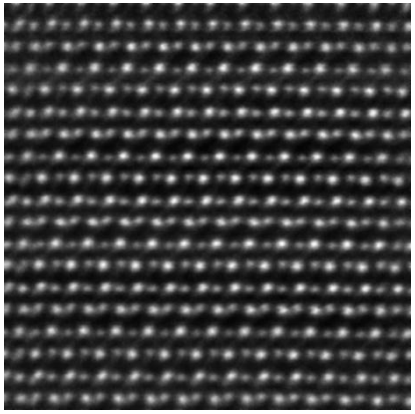
Original image (SnSe)



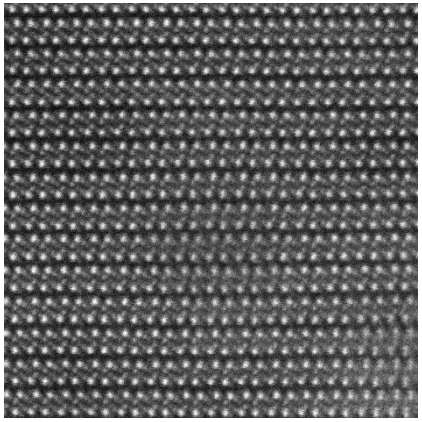
Denoised Image



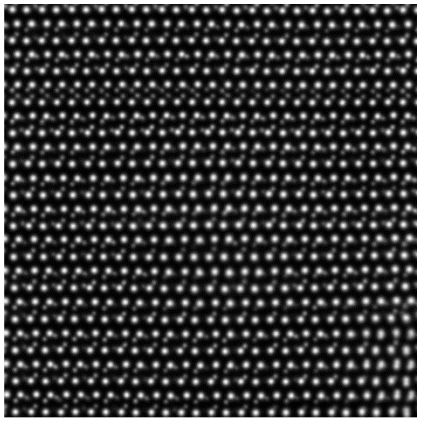
1:1 superposition



Original image (Cu2Te)



Denoised Image



1:1 superposition

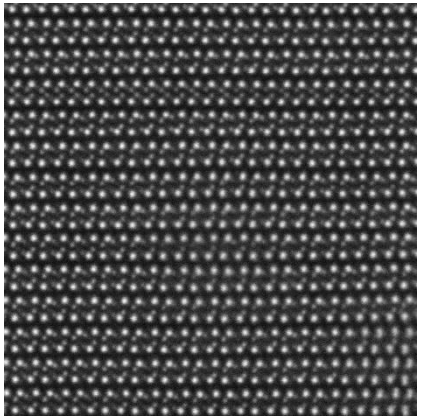
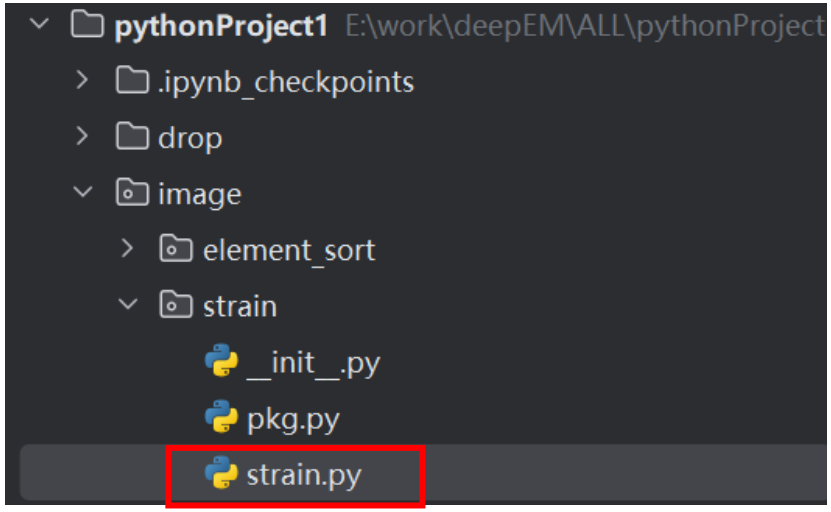


Table of Contents

Strain Analysis



1、Select image - Remove background (omitted)

```
# Load the original STEM image
image = load_img()

# Step 1: Remove the background from the image
# max_scale and min_scale define the scale range for background removal
remover = Backgroundremover(image, max_scale=2048, min_scale=32)
image = remover.remove()
remover.visualize()              # Visualize the image after background removal
```

Strain Analysis

2、Denoising and position detection

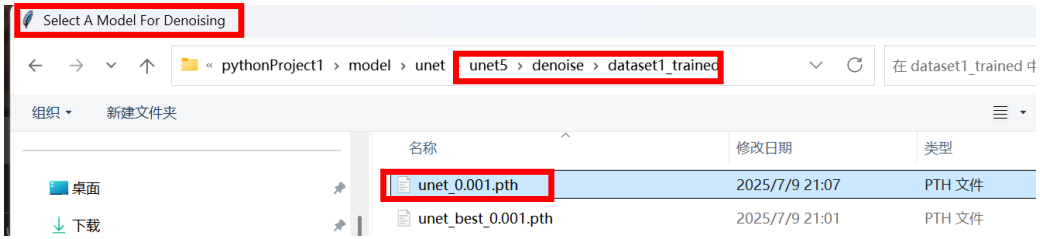
```
# Step 2: Denoise the image and detect atomic positions
# periodicity: number of boxes per row/column (the bigger one) for denoising and segmentation
# stride: step size of sliding window
# batch_size: batch size for the denoising model
# model: model type for denoising ('unet' or 'unet3')
# split_method: method to split denoised regions ('watershed' or 'floodfill')
# min_distance: minimum distance between detected atoms
# filter_ratio: ratio threshold to filter weak detections (usually 0.5-1)
# I: intensity metric to consider ('i_peak' or 'i_mean')
denoised_image, filtered_image, cropped_image, filtered_cropped_image, atoms, atoms_crop = Denosie_Position(
            image,
            periodicity=3,
            stride=6/8,
            batch_size=8,
            model='unet',
            split_method='floodfill',
            min_distance=3,
            filter_ratio=0.8,
            I='i_peak',
            save=False
        )
```

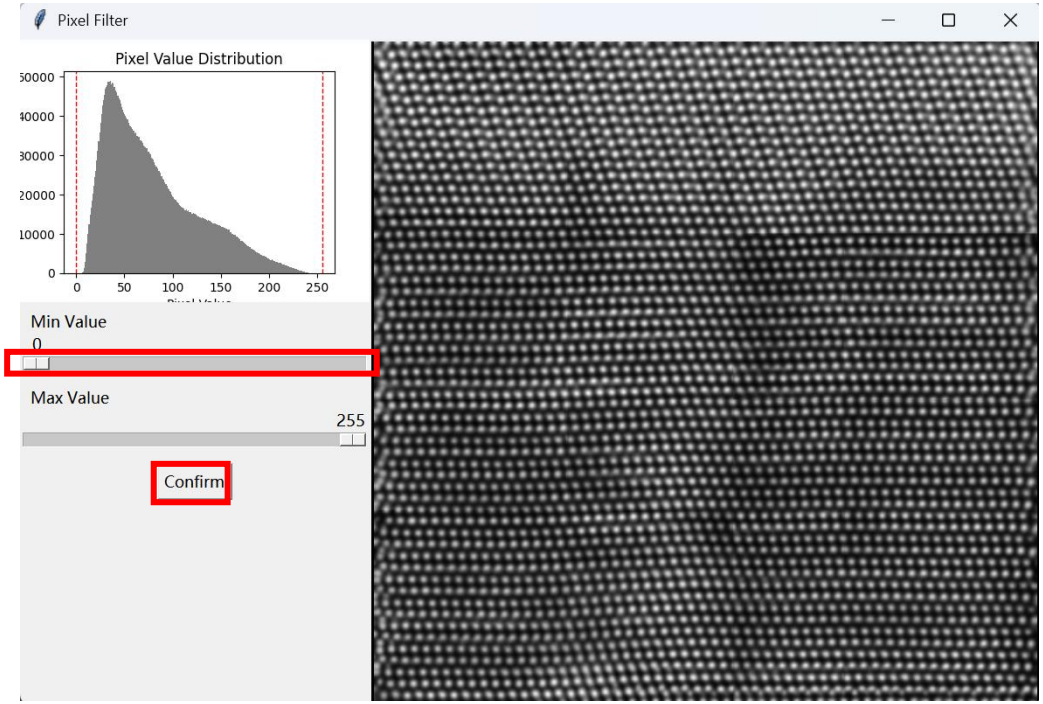
图像去噪和分割参数设置

原子识别参数设置

Strain Analysis

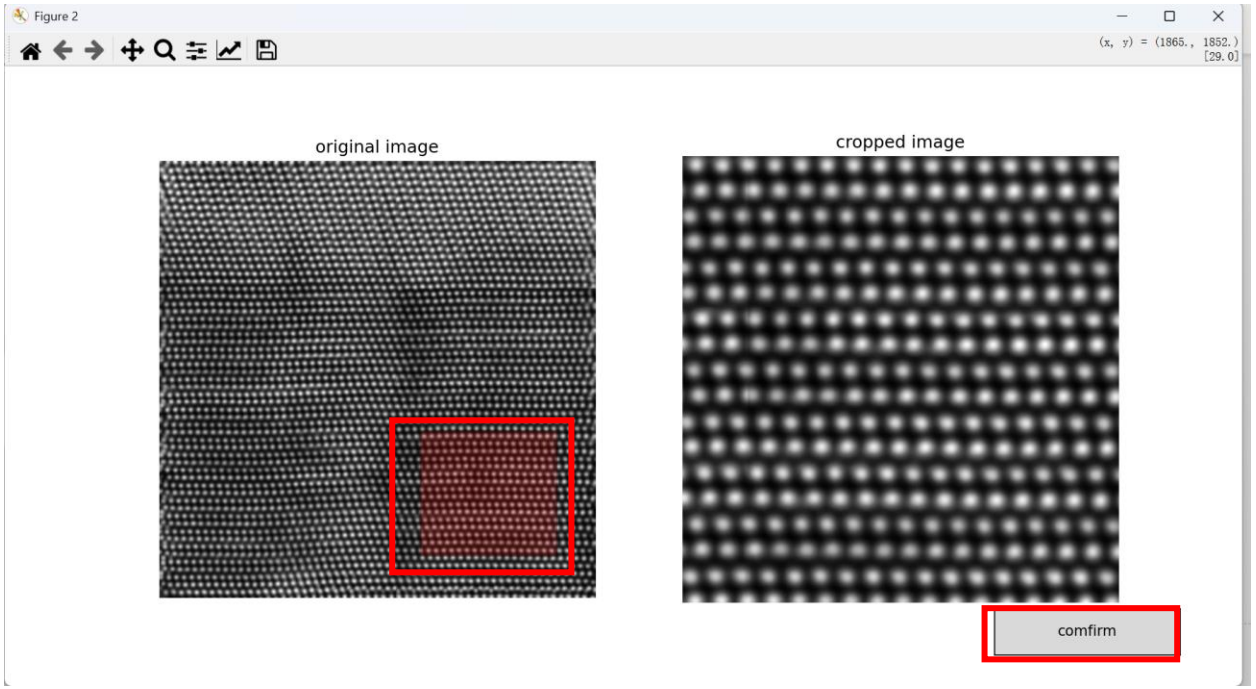
2.1、Denoising - Pixel filtering (filter out non-frame atoms as much as possible)





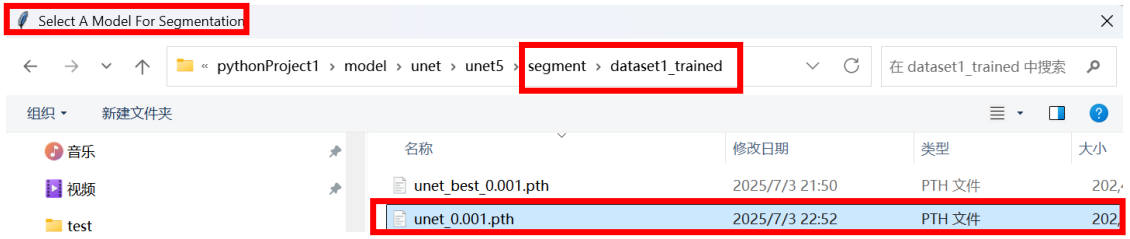
Strain Analysis

2.2、Select standard region (region with minimal distortion)

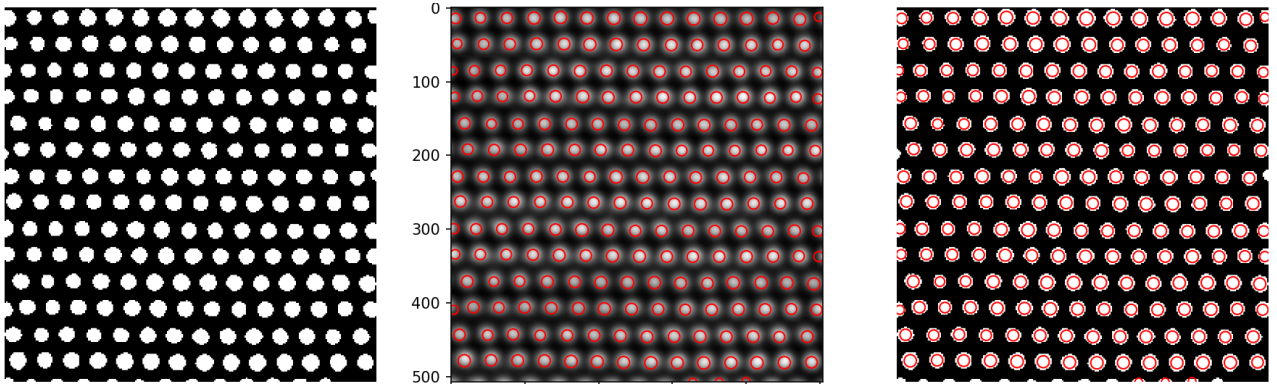


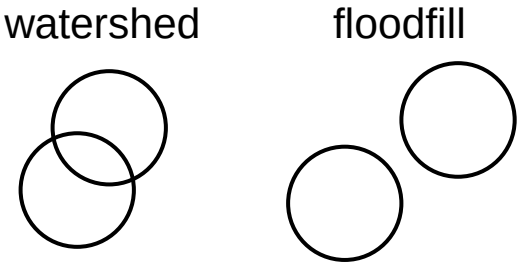
Strain Analysis

2.3、Atomic recognition model weights



Preview of atomic recognition results in standard region





Parameters：

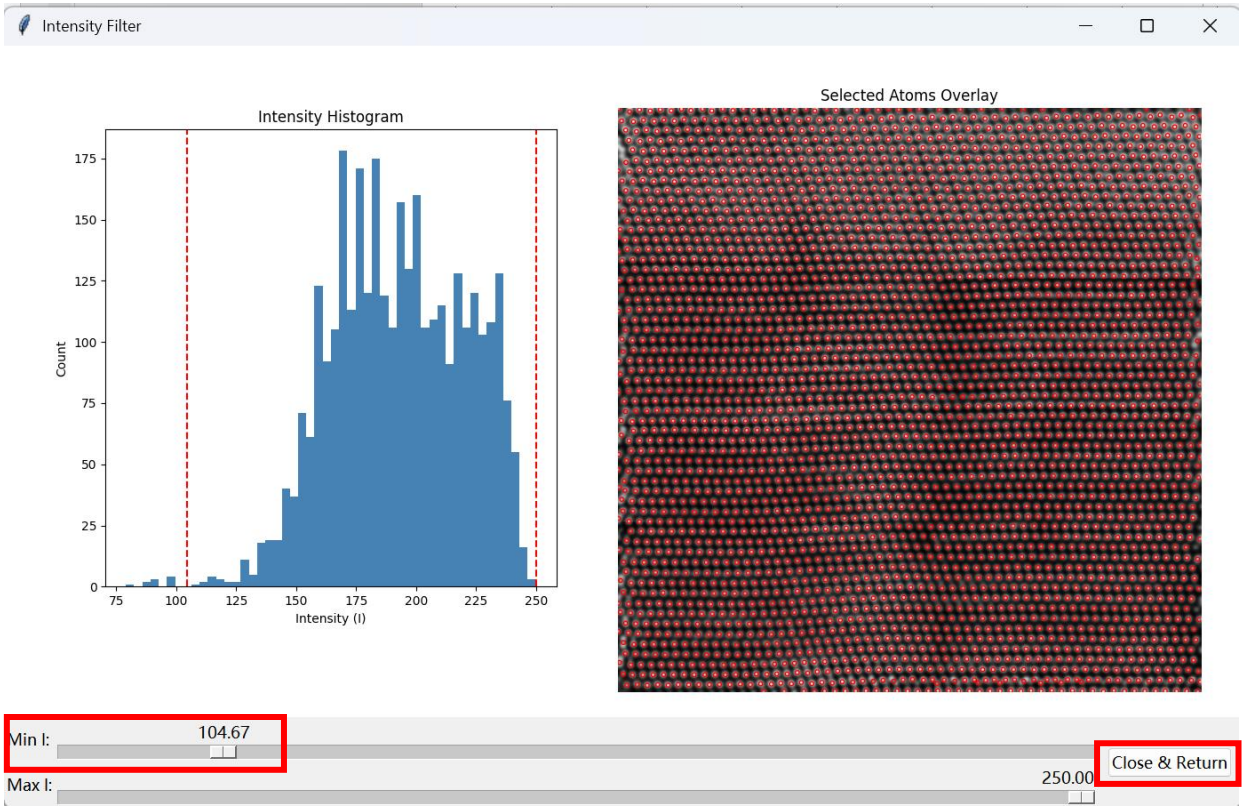
split_method: Detection algorithm (‘watershed’ or ‘floodfill’)

min_distance: Minimum distance between atoms

filter_ratio: Filter out atoms with radii exceeding the limit $\bar{r}\cdot\left(1-\frac{k}{2}\right) < r < \bar{r}\cdot(1+k)$

Strain Analysis

2.4、Select atoms with pixels in a certain range (filter out non-frame atoms as much as possible)



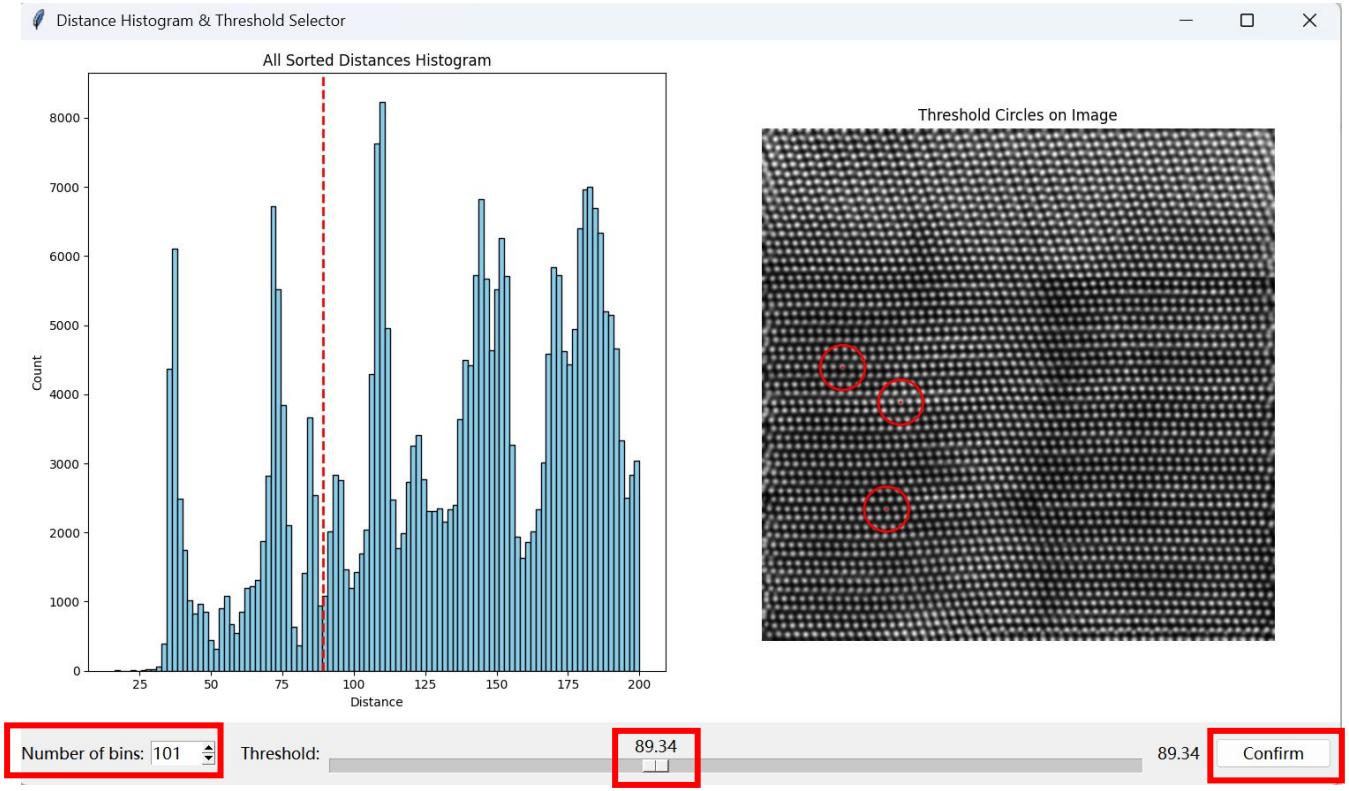
Adjustment of relevant input parameters:

I: Filter based on peak pixels or average pixels (i_peak or i_mean)

Strain Analysis

3、Select the range of neighboring atoms (must be larger than one period in the direction of a set of basis vectors)

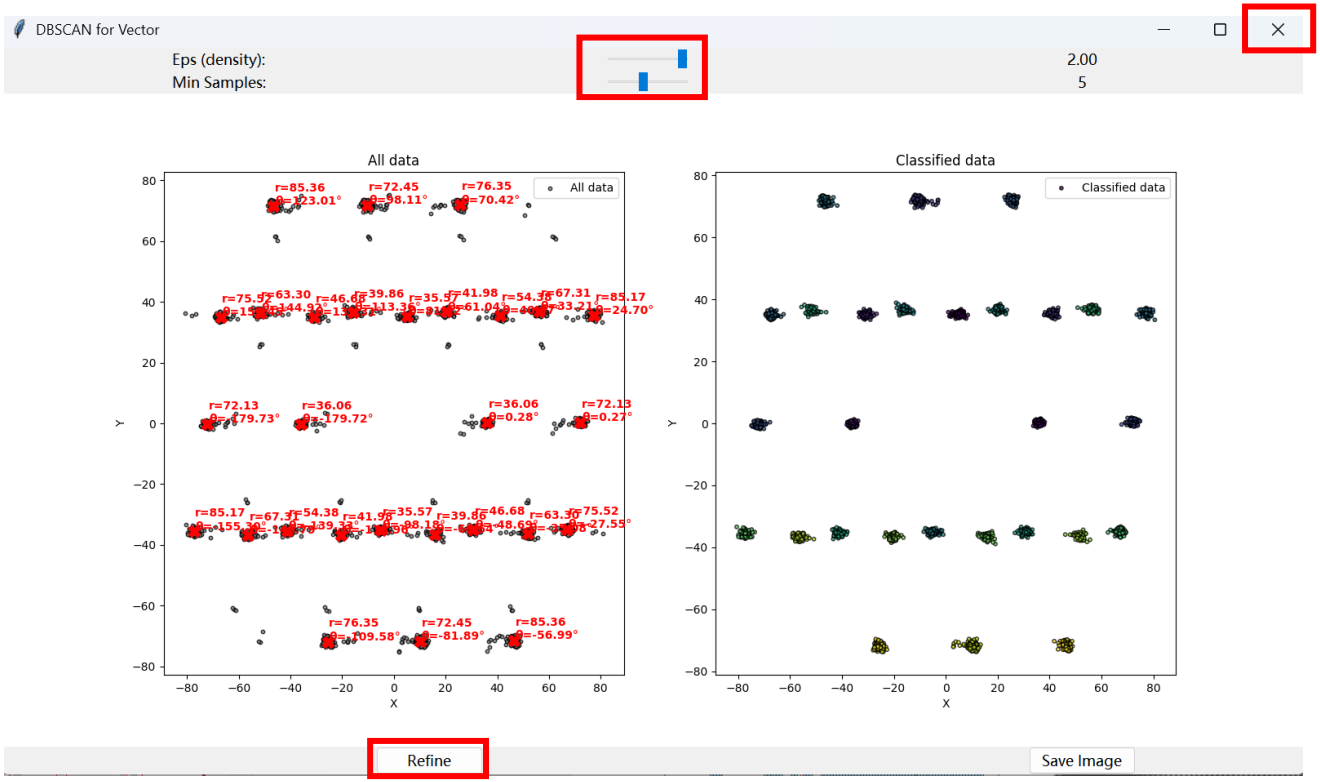
```
# Step 3: Calculate neighboring atoms for each detected atom within radius r
# neighbor_dis: distances to neighbors
# neighbor_pos: neighbor positions
# cutoff: distance cutoff threshold
r = 200            # Neighbor search radius in pixels (or unit matching image scale)
neighbor_dis, neighbor_pos, cutoff = Calculate_neighbor(atoms.to_numpy(), r, denoised_image)
neighbor_dis_crop, neighbor_pos_crop, _ = Calculate_neighbor(atoms_crop.to_numpy(), r, cropped_image, if_crop=True)
```



Strain Analysis

4、Cluster the basis vectors (standard region)

```
# Step 4: Classify standard lattice vectors and label atoms in the cropped region
# vector_centers: centers of standard lattice vectors
# atoms_labels_cropped: labels assigned to atoms in cropped region based on vector classification
vector_centers, atoms_labels_cropped = Vector_centers(
        atoms_crop.to_numpy(), neighbor_dis_crop, neighbor_pos_crop, cutoff
        )
```

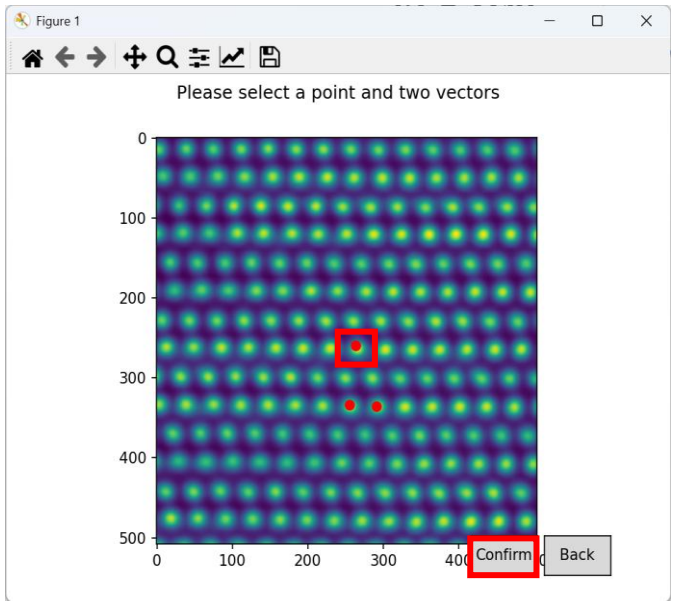


Strain Analysis

5-6. Classify atoms according to basis vectors and select one class for strain analysis (different spatial positions of the same element)

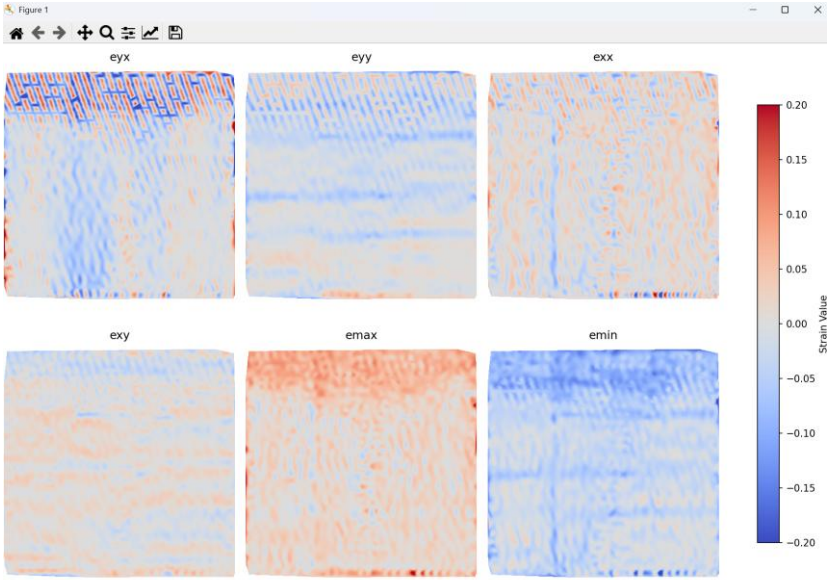
7. Select a set of basis vectors (the first point is O, the second and third are A, B, forming OA, OB basis vectors)

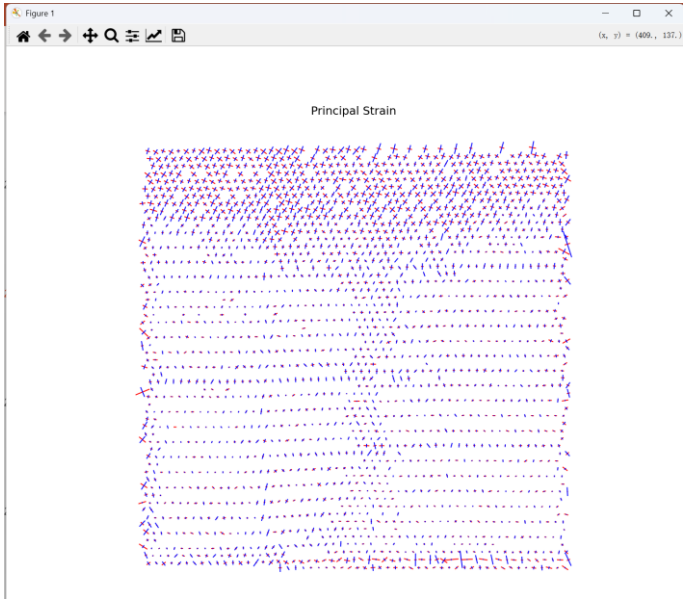
```
# Step 5: Extract lattice vectors for all atoms in the original image and classify them
# vectors_list: list of lattice vectors for each atom
# vector_labels_list: labels for each lattice vector
# atoms_labels: atom labels based on lattice vector classification
vectors_list, vector_labels_list, atoms_labels = classify_vectors(
    vector_centers,  atoms.to_numpy(),  neighbor_dis,
    neighbor_pos, cutoff, dist_multi=1)

# Step 6: Interactive atom selector GUI for the user to select specific lattice vector classes
# selected_vectors_list: lattice vectors selected by user
# selected_vector_labels_list: labels of selected vectors
# selected_atoms: atom positions corresponding to the selected vectors
atomselector = AtomSelector(vectors_list, vector_labels_list, atoms.to_numpy(), atoms_labels, denoised_image)
selected_vectors_list, selected_vector_labels_list, selected_atoms = atomselector.get_result()

# Step 7: Define a set of standard lattice vectors for strain calculation
standard_a, standard_b, label_a, label_b = standard_vector(cropped_image, vector_centers)
```



Strain Analysis

8、Strain analysis results





9、 Redraw by changing the maximum value of the color bar

```
********** Calculate Strain **********

Enter new color bar max value (or press Enter to exit): 0.15

Enter new color bar max value (or press Enter to exit): 0.2
```

Strain Analysis

Supplementary: Output log

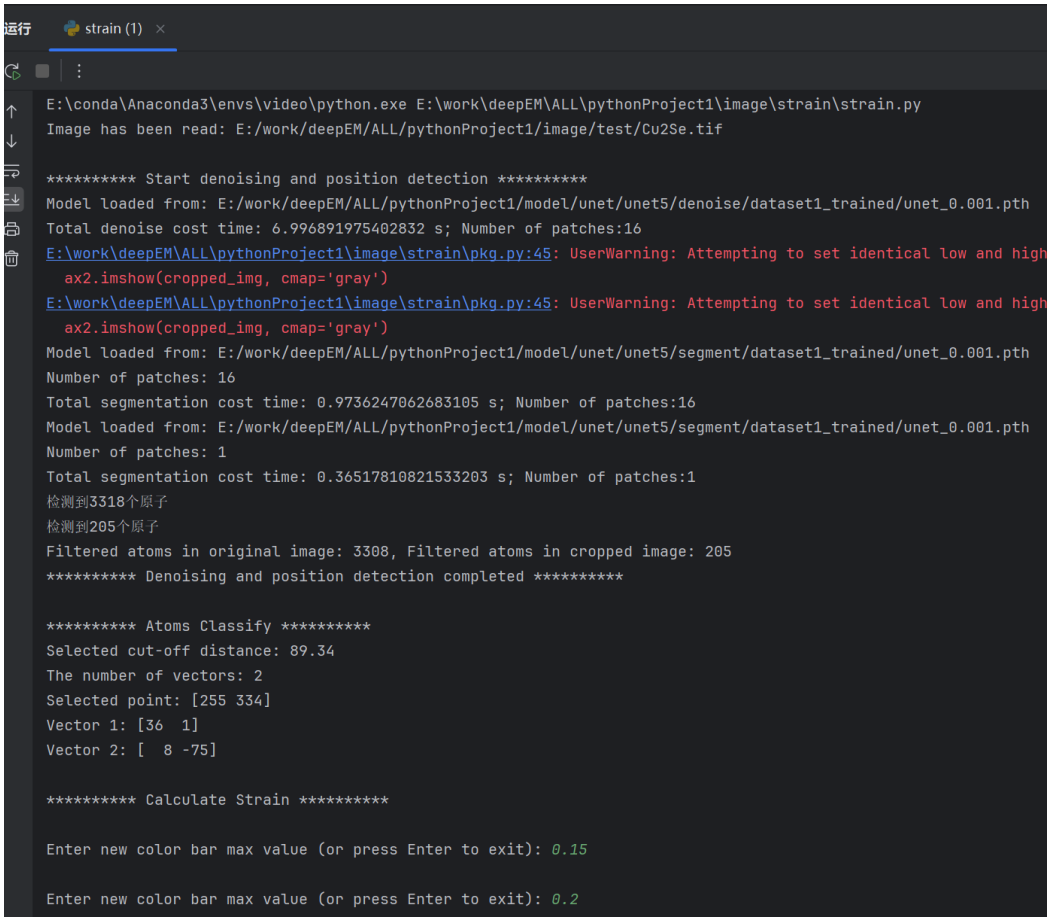


Table of Contents

Element Analysis

Select image - Remove background - Denoise - Atomic recognition (omitted)

element.py

```python
from image.pkg import Backgroundremover, Prediction, Position, Element, load_img
from model.unet.unet import Unet as Unet
from model.unet.unet3 import Unet as Unet3
import numpy as np

image = load_img()

#remove background
remover = Backgroundremover(image, max_scale=2048, min_scale=32)
image = remover.remove()
remover.visualize()  # 可视化去背景后的图像

periodicity = 2
stride = 6/8
batch_size = 8
# periodicity: number of boxes in one row or column; stride: stride of the sliding window;
# batch_size: batch size for prediction
denoiser = Prediction(image, periodicity, stride, Unet3(n_classes=1), batch_size)
ori_image, image = denoiser.denoise()
denoiser.visualize(image)  # 可视化去噪后的图像

segmenter = Prediction(image, periodicity, stride, Unet3(), batch_size)
image, output = segmenter.segment()
segmenter.visualize(output)  # 可视化分割后的图像

#二值化,检测原子位置
output_np = np.array(output.convert("L")).astype(np.uint8)
ori_image_np = np.array(ori_image.convert("L")).astype(np.float32)
#atoms为一个DataFrame：x, y, r, fit_area, ori_area, i_mean, i_peak
#split_method = 'watershed'/'floodfill'
position_detecter = Position(ori_image_np, output_np, erosion_iter = 0,
                             split_method = 'floodfill', min_distance = 5,filter_ratio = 0.8)
atoms = position_detecter.position_detect()
position_detecter.visualize(atoms, save_path = None)
```

Element Analysis

Element Analysis

```
36    # 检测元素
37    element_detecter = Element(ori_image_np, atoms, I_ = 'i_peak', start_bins=20, bin_step=20,
38                               min_prominence_ratio=0.25, smooth_sigma_ = 2)
39    atoms = element_detecter.element_detect()
40    element_detecter.visualize(save_path_ = None,
41                               colors_ = ['red', 'blue', 'orange', 'green', 'purple', 'brown', 'pink'])
42
```

Adjustment of relevant input parameters：

I: Classify based on peak pixels or average pixels （i_peak or i_mean）

start_bin: Initial number of bins in the histogram

bin_step: Bin increment step

min_prominence_ratio: Filter out smaller peaks

smooth_sigma: Gaussian fitting parameter

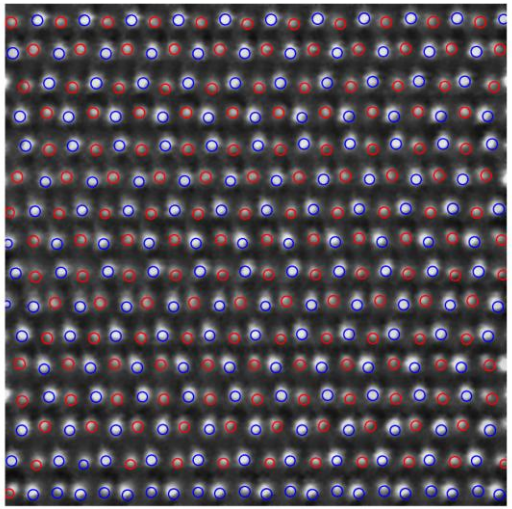
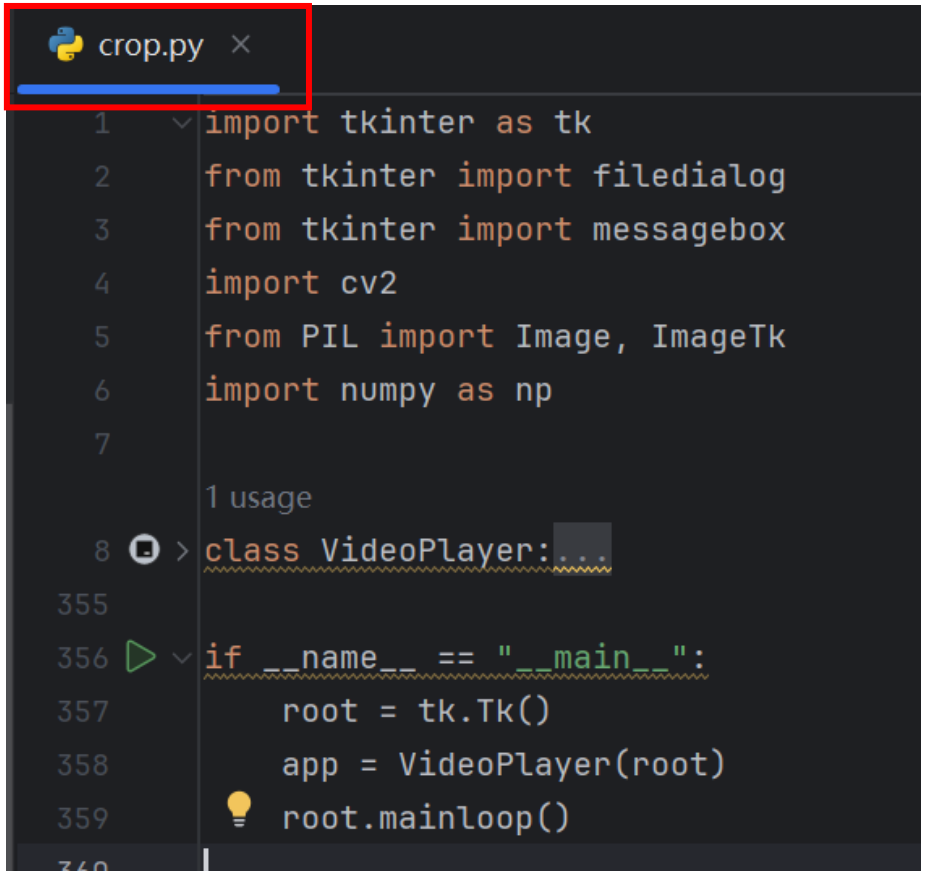


Table of Contents

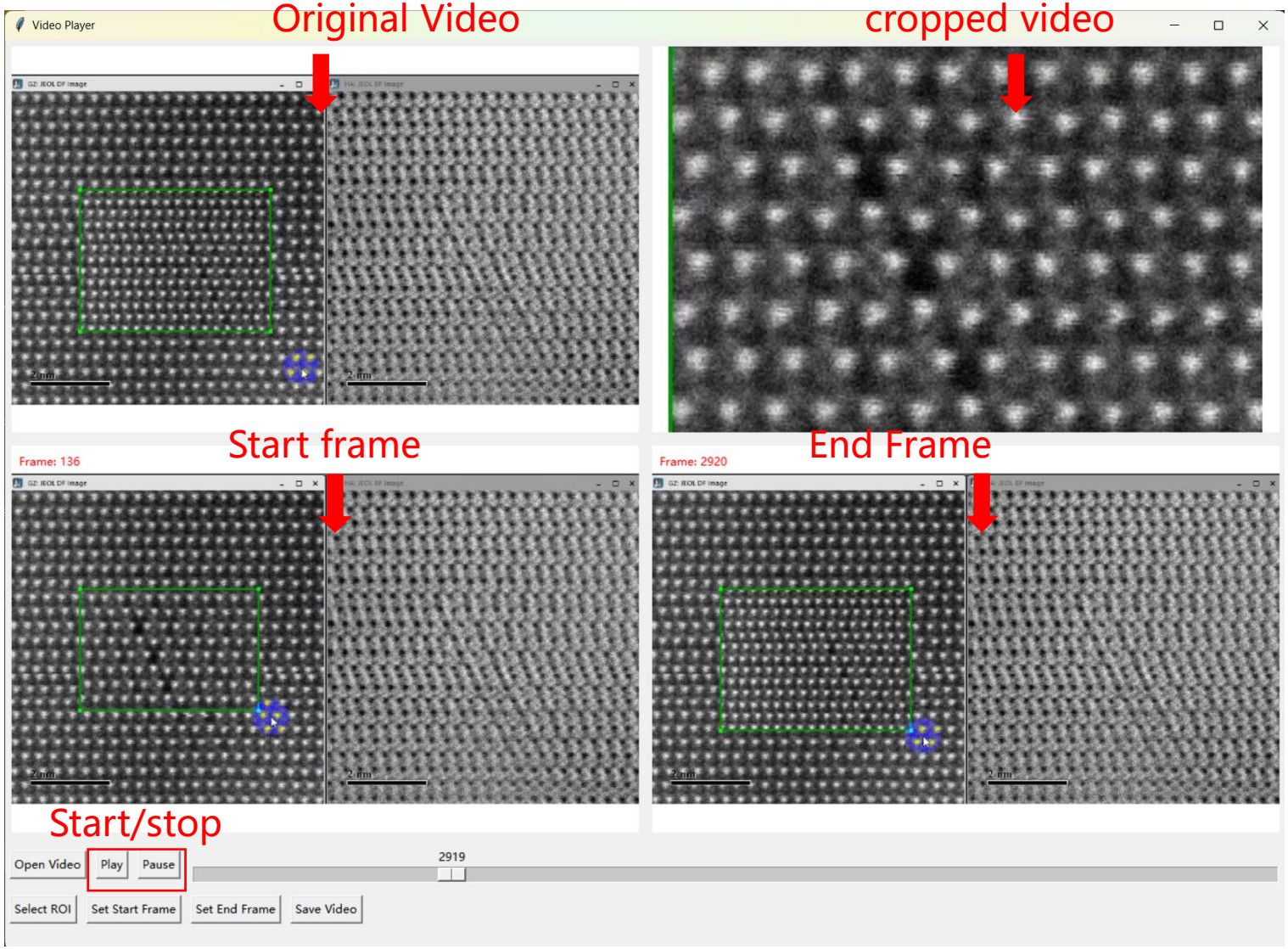
STEM Video Cropping



Scope of Application: STEM videos (any size)

Usage: Right-click - Run

STEM Video Cropping



Process

1、Open Video (Load video)

2、Select ROI (Select cropping area)

3、Set Start Frame

4、Set End Frame

5、Save Video

Table of Contents

STEM Video Frame Extraction

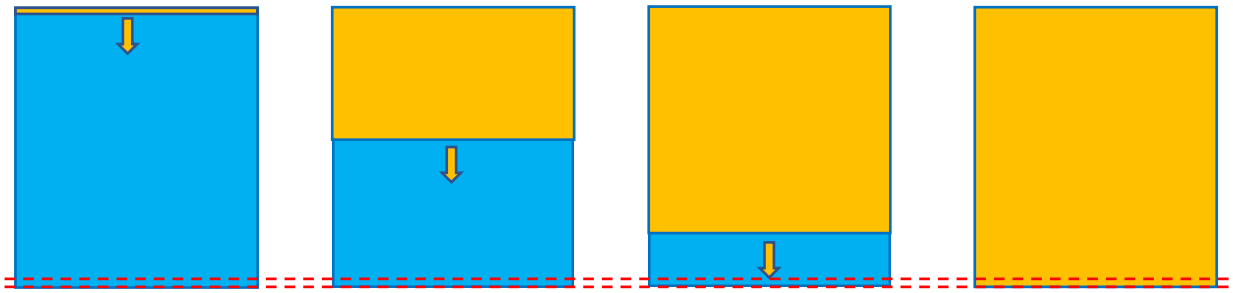
refine.py

```
import ...

1 usage
def calculate_bottom_difference(frame1, frame2, num_rows=3):...

1 usage
def refine_video(input_path, threshold=1, num_rows=3):...

1 usage
class VideoApp:...

if __name__ == '__main__':...
```

Scope of Application: STEM videos (any size)

Usage: Right-click - Run

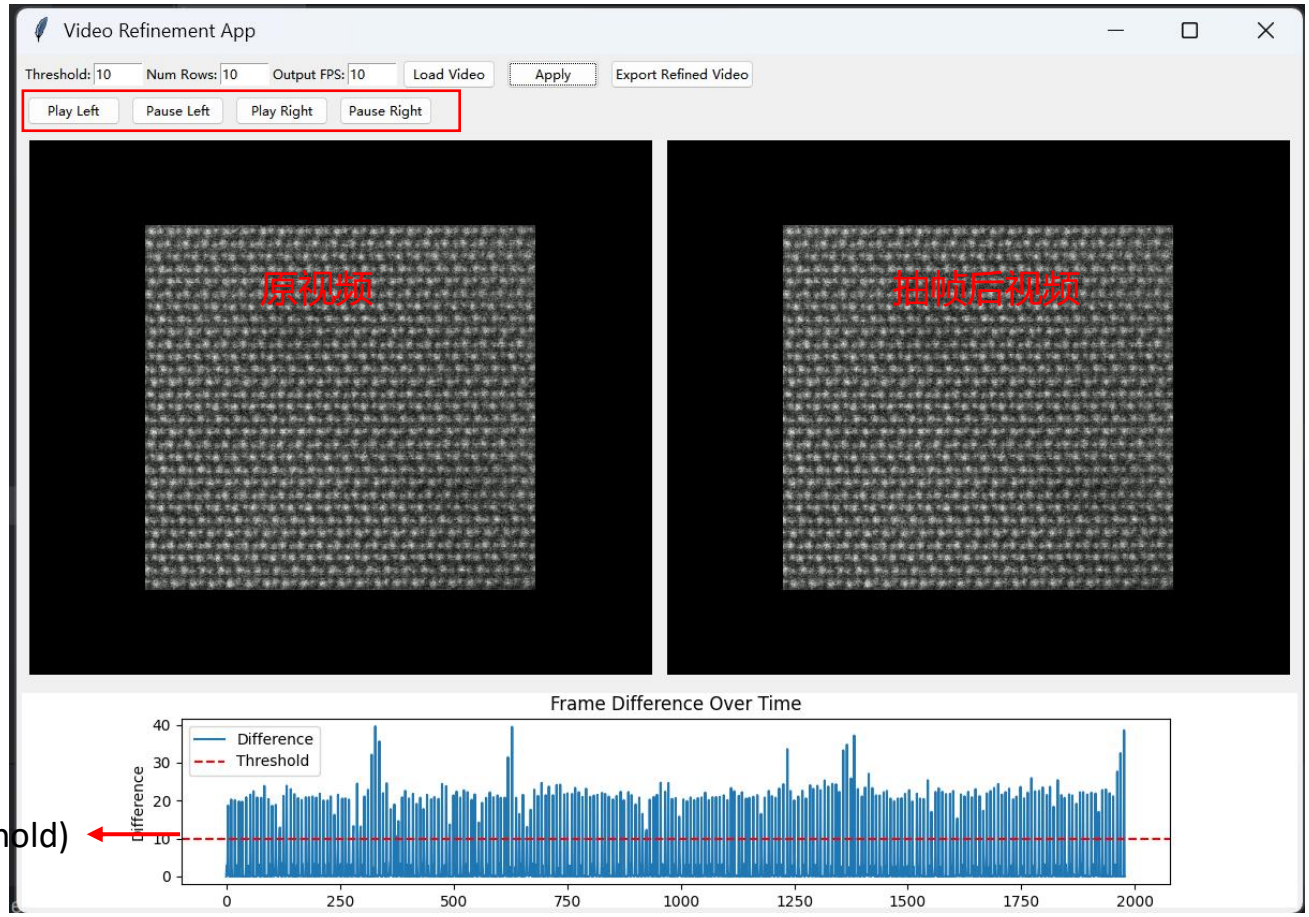
Principle:



$$A_t - A_{t\text{-}1} > \Delta$$

STEM Video Frame Extraction

Video playback



Δ (Threshold)

$A_t - A_{t\text{-}1}$(for the last Num Rows rows)

Process

1、Load Video

2、Adjust Threshold and Num Rows

3、Apply

4、 Adjust Output FPS

5、Export