Manual for the Geometric calibration of 4D-STEM datasets

Shoucong Ning
Department of Physics
University of Science and Technology of China
Hefei, Anhui

Core reference paper:

[1]. Ning, Shoucong, et al. "Accurate and Robust Calibration of the Uniform Affine Transformation Between Scan-Camera Coordinates for Atom-Resolved In-Focus 4D-STEM Datasets." Microscopy and Microanalysis 28.3 (2022): 622-632.
[2]. Ning, Shoucong, et al. "An integrated constrained gradient descent (iCGD) protocol to correct scan-positional errors for electron ptychography with high accuracy and precision." Ultramicroscopy 248 (2023): 113716.

Acknowledgment

This software and the attached manual were developed by Dr. Shoucong Ning with the support of Singapore NRF. We also thank Dr. Qian He, Dr. Wenhui Xu and Prof. Fucai Zhang for their contributions and discussions in the development of the code used in this research. We would like to express our gratitude to Prof. Stephen Pennycook and reviews of M&M for their insightful comments and suggestions that have greatly improved the quality of this work.

1. Installation.

Our hybrid calibration package is written in python3 language, and executed using Jupyter Notebook. Consequently, this package can be deployed in any platform supporting Python language.

1.1. Dependencies.

Only ***python3*** supported is included in our package, and the required Python libraries are *Numpy*, *Scipy*, *Matplotlib*, *Skimage*, and *Jupyter Notebook*, and there is a specific requirement for the version of these libraries. As the most efficient way to access these packages, the ***anaconda/miniconda*** is recommended. Anaconda provides a convenient way to install and manage packages for scientific computing, data analysis, and machine learning, without having to deal with dependency conflicts or compatibility issues. It includes popular packages such as NumPy, Pandas, Scikit-learn, TensorFlow, and PyTorch, among others. Anaconda also includes tools such as Jupyter Notebook, Spyder, and RStudio, which provide an integrated development environment (IDE) for data science workflows.

To install Anaconda, please follow the following steps:

1. Go to the Anaconda download page: https://www.anaconda.com/products/distribution. Select the appropriate version of Anaconda for your operating system (Windows, macOS, or Linux) and download the installer.
2. Run the installer and follow the on-screen instructions to complete the installation process. You may need to provide administrative privileges to install Anaconda on your computer. During the installation process, you will be prompted to select a destination folder for Anaconda and to choose whether to add Anaconda to your system PATH environment variable. In the Windows operating system, please do not add Anaconda to your system PATH environment variable, while this is adding Anaconda to your system PATH environment variable is suggested for beginners.
3. Once the installation is complete, you can launch Anaconda Navigator, which is a graphical user interface for managing and launching Anaconda applications. Alternatively, you can use the Anaconda command-line interface (CLI) by opening a terminal or command prompt and typing "anaconda" or "conda" followed by the desired command.

1.2. Execution of Jupyter Notebooks

Download all of the Jupyter Notebook files (.ipynb) from this repo and copy your notebooks to your preferred directory. On the Windows platform, search the Anaconda prompt after clicking the search button, input Jupyter Notebook, and press enter to call the Jupyter Notebook UI in your web browser. On the linux platform, please activate the default enviroment using the terminal and type the jupyter notebook command to run the script.

1.3. Introduction to packages.

2. Workflow for thin (2D materials) and thick (bulk materials) specimens.

2.1 Simple processings of 4D-STEM datasets.

First, pre-processing of the 4D-STEM dataset is required to account for the **variation of gain** among the pixels for accurate geometric calibration and phase reconstruction.

a. Collect and process gain reference in ***Gain Calibration advanced notebook***

- The gain reference of the 4D-STEM detector is acquired at the correct accelerating voltage (high tension) by illuminating the detector with a uniform beam (without specimen).
- Load the dataset for gain reference in ***Gain Calibration advanced notebook***. The user may choose to collect gain reference separately for odd/even acquisition (depending on the counter of 4D-STEM detector ?) by using numpy slicing in line 2 of this cell.

```
1  dat4d = np.reshape(dat4d, (256*256, 256, 256))
2  pacbed = np.average(dat4d[1::2], axis=(0,))
3  #generate the background signal inside PACBED and reduce it.
4  hot = np.zeros_like(pacbed)
5  dead = np.zeros_like(pacbed)
6  avg_pacbed = np.average(pacbed)
7  dead[np.where(pacbed<1e-1)] = 1
8  hot[np.where(pacbed>1.7*avg_pacbed)] = 1
9  pacbed[np.where(hot==1)] = avg_pacbed
10 pacbed[np.where(dead==1)] = avg_pacbed
11 fig =plt.figure(1, figsize=(75, 25))
```

- Dead and hot pixels are detected on the Position-average CBED (PACBED)

- The effect of non-uniform electron dose is accounted for by fitting a 2D-ROI of PACBED with a 2D linear function. The fitted 2D plane is the estimated electron dose distribution on the detector during gain acquisition. The gain file is then calculated on line 4:

```
:  1 gain = np.ones_like(pacbed)
   2 gain[np.where(dead>0)] = 0
   3 intense = PlaneGeneration(params, pacbed.shape[0], pacbed.shape[1])
   4 gain = (pacbed/intense)
   5 gain[np.where(dead>0)] = 0 # dead pixel has 0 gain
```

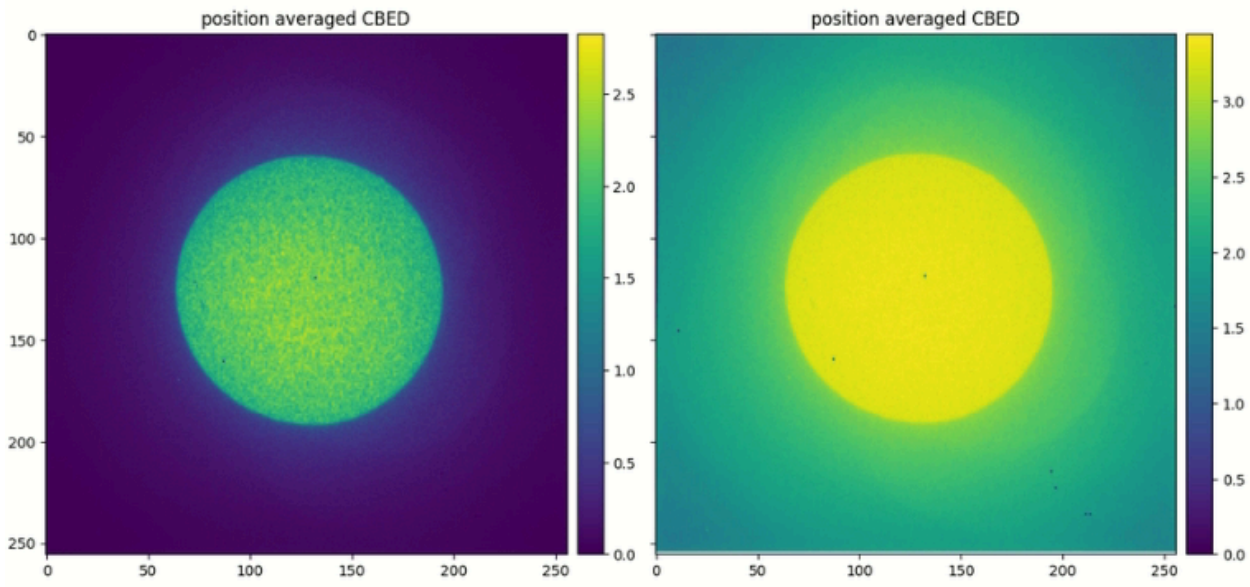
- Gain file(s) are saved in .npy format

b. Preprocessing 4D-STEM dataset in ***Preprocessing notebook***

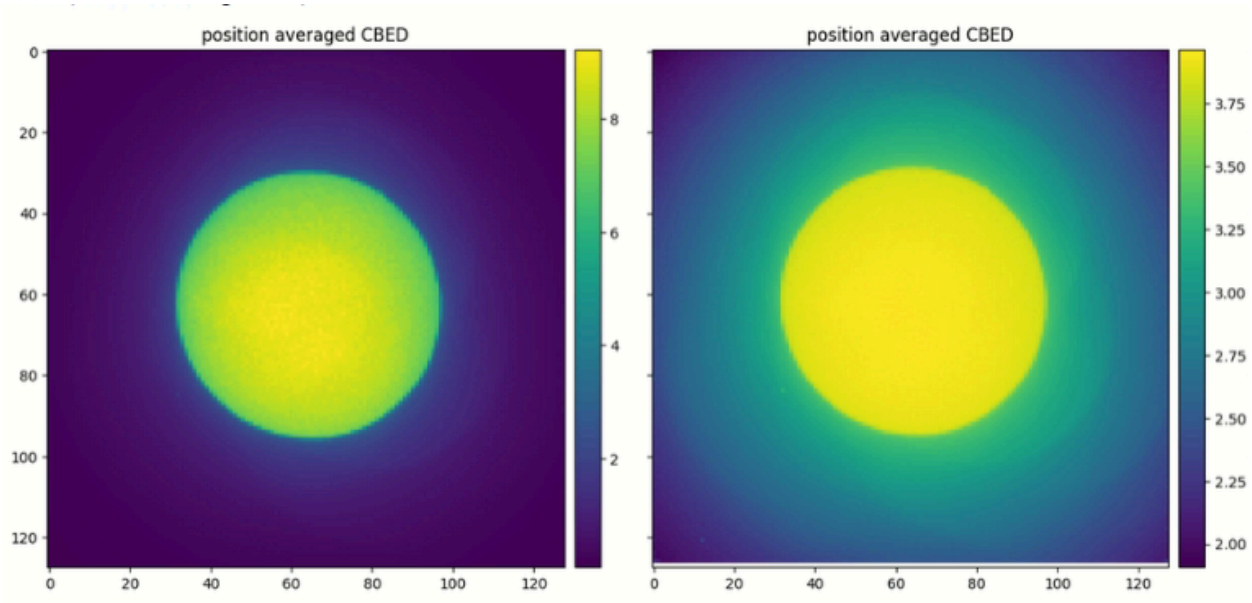
- Read 4D-STEM dataset according to the type of detector and save in .npy format. The numpy array should have the shape (H,V,X,Y) where H, V are the size of fast and slow scan directions, X and Y are the shape of the detector.
- Detect hot and dead pixels, set their values to 0.
- Bin the 4D-STEM dataset along X and Y direction to average out the dead and hot pixels (alternatively we can interpolate the value of these pixels using the neighboring pixels in case the detector used has a limited number of pixels ?)
- Gain normalization is done by dividing the 4D-STEM dataset with the gain array. Users can also specify gain normalization of odd and even acquisition with corresponding gain array for accuracy

```
dat4d[0::2,:,:,:] =dat4d[0::2,:,:,:]/gain1 # assuming scan size is even
dat4d[1::2,:,:,:] =dat4d[1::2,:,:,:]/gain2
```

Before pre-processing:



After pre-processing and binning 2x2:



- Save dataset after preprocessing as "ndata.npy" for Geometric Calibration

2.2 Geometric Calibration

Geometric Calibration is performed in ***Thin Calibration notebook*** or ***Bulk Calibration notebook*** as described in detail in Section 3. The aim of geometric calibration is to accurately obtain the geometric relation between the scanning coordinates (H,V) and CBED coordinates (X,Y) using J-matrix method (maybe?) and Fourier method as discussed in detail in Ning-2022. The geometric relation includes: angle θx (between H and X) (but this is theta_y in the script), angle θy (between V and Y) (theta_x in the script), the scan interval ratio between fast and slow scanning directions γ.

There are 4 sets of solutions for (theta_y, theta_x) in the range [0, 2π]. The set of solutions produced by this script depends on the initialization values for (theta_y, theta_x). Systematically, the initialization can be done with the J-matrix method to identify the uniform rotation angle θ, together with the identification of the flipping between the scanning coordinates and the camera coordinates. Flipping here indicates a change in chirality between the two coordinate systems (H cross V and X cross Y are opposite vectors). Flipping can also be detected with only the Fourier method through the sign of the scan interval ratio γ. The 4 sets of solutions are 4 permutations of θx±π and θy±π: {(θx,θy),(θx+π,θy),(θx,θy+π),(θx+π,θy+π)} corresponding to 4 sets of initial guesses {(θ,θ),(θ+π,θ),(θ,θ+π),(θ+π,θ+π)}. Two of 4 cases can be eliminated when γ<0. The

remaining two valid cases should have the same chirality but with a 180° rotation, resulting in an opposite contrast of the atom columns after phase retrieval. We detect whether a flip occurs if $\pi/2<(\theta_x - \theta_y) <3\pi/2$ for $\theta_x>\theta_y$.

*I introduced flip = np.sign(np.cross([*fast_shift,0],[*slow_shift,0])[-1]) to detect whether there is a flip. Then flip is added to the phase after SSB here:

```
objectR =IFFT_2D(RTrotterSum*TrotterPixelNum)
fig =plt.figure(1, figsize=(60, 30))
grid=AxesGrid(fig, 236, nrows_ncols=(1,2),
              axes_pad=0.5,
              share_all=False,
              cbar_location="right",
              cbar_mode="each",
              cbar_size="5%",
              cbar_pad="2%")
im=grid[0].imshow(np.angle(objectR)*flip)
grid.cbar_axes[0].colorbar(im)
grid[0].set_title('Right Trotter phase')
im=grid[1].imshow(np.abs(objectR))
grid.cbar_axes[1].colorbar(im)
grid[1].set_title('Right Trotter amp')
```

Let me know if you're ok with it

We then try one of the two remaining cases with simple Single Side Band (SSB) reconstruction. If the atom columns appear bright in the retrieved phase (positive), no rotation is needed. If the atom columns are dark (negative), we rotate both (θ_y, θ_x) with a 180°-rotation.

Column phase	No rotation	180 Rotation°
No flip	Positive	Negative
With flip	Negative	Positive

2.3 Phase retrieval.

After obtaining the accurate geometric relations, phase retrieval can be performed with SSB method [Pennycook-2015]. In this script, aberrations of the electron probe can also be calculated

through the method described in [xxx]. The calculated geometric relations and probe information will be used as inputs for further iterative reconstruction.

3. Examples.

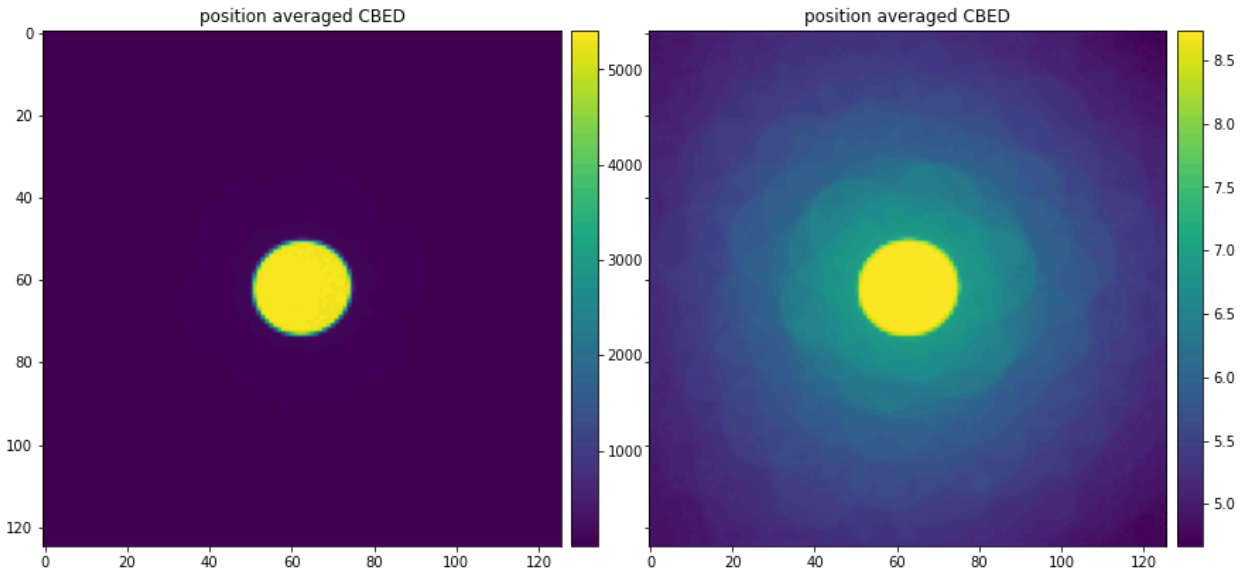
3.1. 2D materials. (Thin Calibration.ipynb)

Taking the 4D-STEM dataset of a WS_2-bilayer captured using EMPAD under 60kV accelerating voltage as an example, the 4D-STEM dataset consists of 256*256*128*128 pixels, and the EMPAD detector consists of 128*128 pixels. The field of view of the 4D-STEM dataset is set to 10.8 nm, indicating a 0.423Å scanning step size.

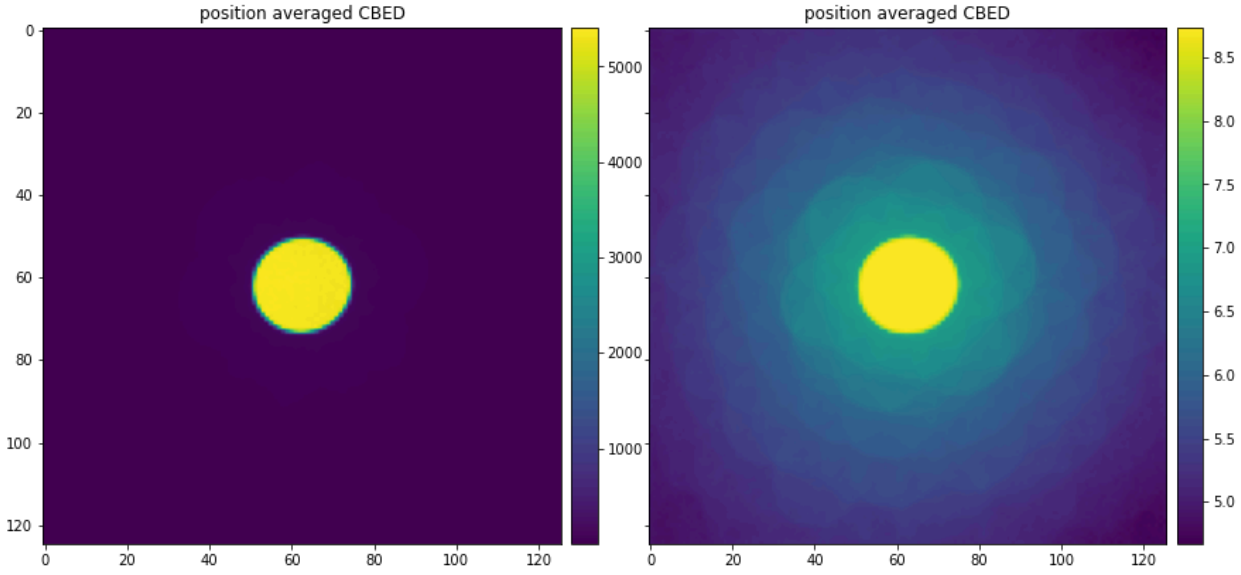
Firstly, the 4D-STEM dataset is converted from .raw format to the .npy format using the ***Read EMPAD data notebook***. To read the 4D-STEM in .raw format, the directory and file name should be set first.

```
path = "D:/4D-STEM Data/EMPAD/WS2/F20230315/06/"
file = "data.raw"
#frame size of the EMPAD detector
frame_size = (128,128)
#numbers of scanning positions along fast and slow direction.
scan_size = (256,256)
```

To set the pixel number of 4D-STEM datasets, the frame size and scan size should be set. In this notebook, the border signal of the EMPAD detector is abandoned considering the enhanced signal around the boundary. In addition, a threshold is set to eliminate dark current and thermal noise. The positional averaged CBED (PACBED) in linear and logarithm space are shown in the following figure:

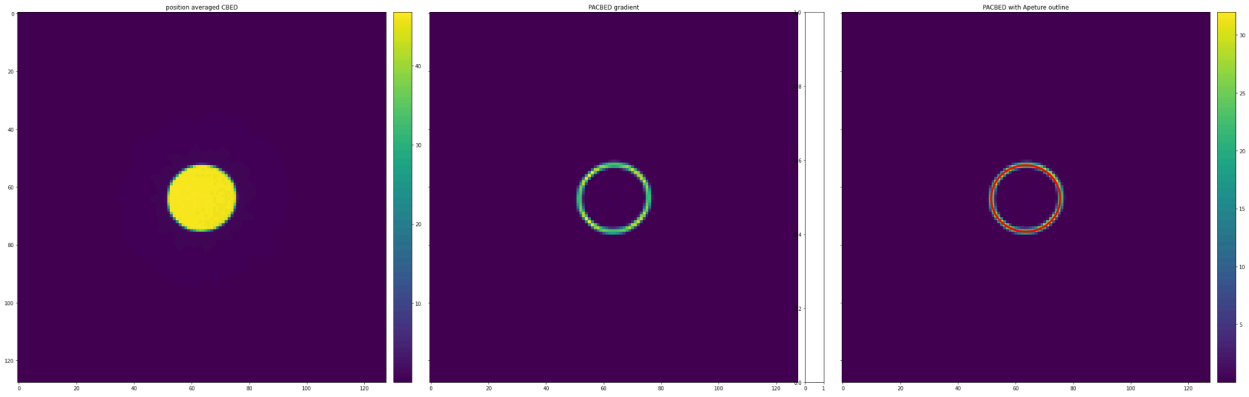


As that, the 4D-STEM dataset will be saved in the .npy format within the same directory after running the last line. Then the ***Thin Calibration notebook*** is called to calibrate the geometric relationship, measure the residual aberration, and reconstruct the phase of imaged sample. The save .npy file is loaded in ***Thin Calibration notebook*** and the PACBED is first shown to validate that the file is successfully transferred:

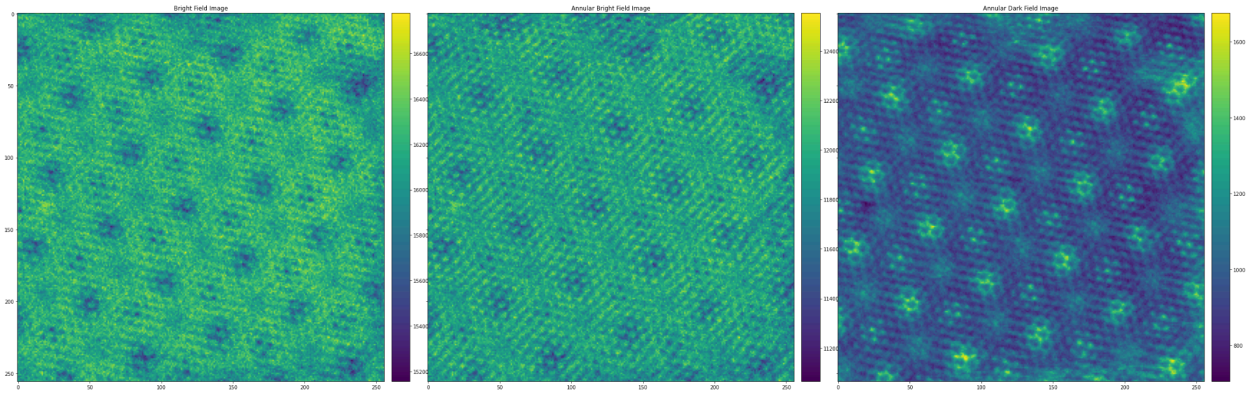


In the ptychography reconstruction, the physical length of each detector pixel should be known. Experimentally, the aperture size is recorded, and it's possible to determine the angular value of each pixel from the PACBED. By treating the aperture as an oval shape, the center and the diameter of the aperture are determined. Then the angular value of each detector pixel is acquired. Taking the PACBED recorded by the EMPAD detector as an example, the aperture is

17.9 mrad, and the angular value of each pixel is 1.52 mrad for the current settings of the projector lens.



The benefit of determining the aperture position and radius is not limited to further iterative ptychography reconstruction. The STEM frames, including ABF-STEM, BF-STEM, and ADF-STEM can be generated on 4D-STEM datasets. To ensure the diffraction patterns match their scanning positions, and remove the bad scanning positions, the generation of STEM frames is also compulsory. As shown in the following Figure, different contrasts of the WS2 Moire lattice are delivered, and there are no bad scanning lines to be processed, then the Fourier transformation will be applied to the 4D-STEM dataset along scanning positions.



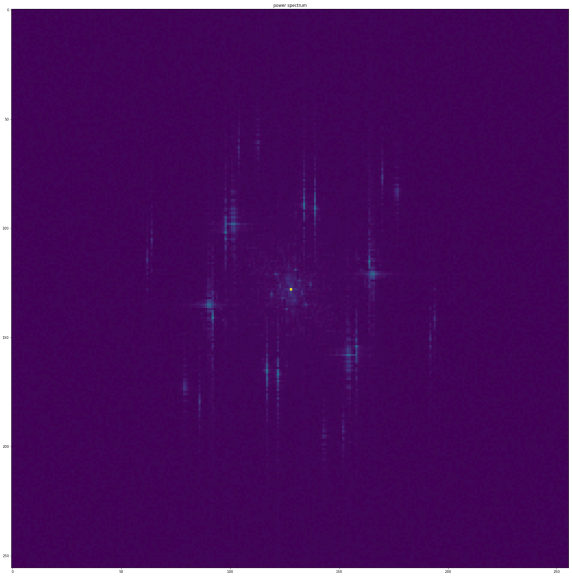
Before Fourier transformation, diffraction patterns should be trimmed in order to reduce memory consumption. There is a parameter named expansion controlling the frame size of the trimmed 4D-STEM dataset, it's the ratio between the frame width and the aperture diameter. So a large expansion will lead to a larger trimmed frame and slower computation.

```
#trim the 4D-STEM dataset to reduce the computation time.
expansion = 1.4
x_start = int(x_center-expansion*radius)
y_start = int(y_center-expansion*radius)
x_end = int(x_center+expansion*radius)
y_end = int(y_center+expansion*radius)
```

In addition, to reduce the influence of the beam current variation, we suggest users to normalize each diffraction pattern by assuming that the total electron number in each diffraction pattern should be the same. This is a reasonable assumption especially when the collection angle of the electron detector is high, say 80 mrad.

```
#normalize the intensity of each line.
ratio = 1.0/np.average(dat4d, axis= (2, 3))
ratio = np.reshape(ratio, (dat4d.shape[0], dat4d.shape[1],1,1))
```

The Fourier transformation is applied to the trimmed and normalized (if necessary) 4D-STEM dataset, and the power spectrum of the transformed 4D-STEM dataset is shown as follows:



This power spectrum is quite similar to the power spectrum of the 2D STEM frames (ABF-STEM, ADF-STEM, BF-STEM) since the Fourier transformation is applied to scanning positions. Due to the scanning distortion, the Bragg peaks in the power spectrum of the 4D-STEM dataset are obviously elongated along the vertical direction. Each pixel in the above power spectrum corresponds to a 2D G-set slice of the transformed 4D-STEM dataset for a specific spatial frequency (sample plane). The intensity value of each pixel in the above power

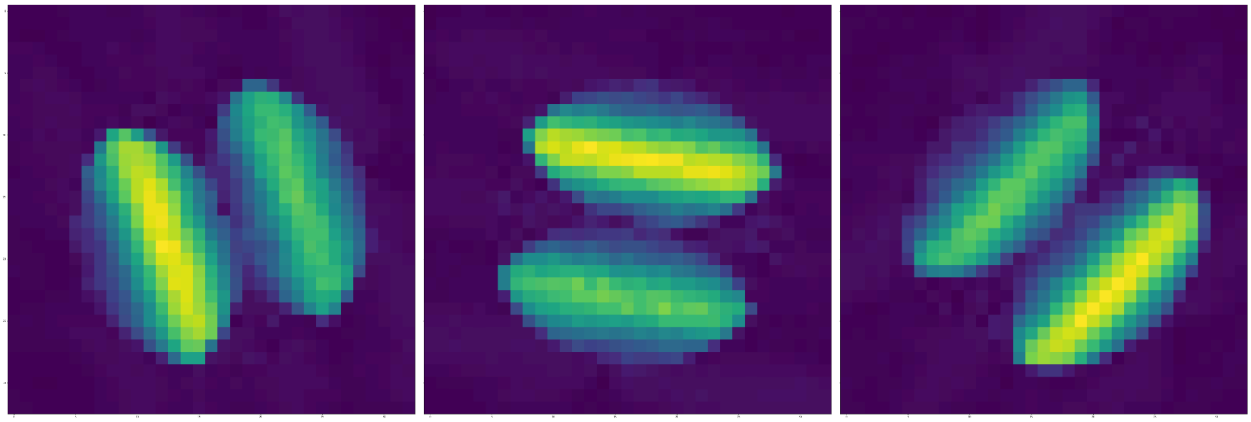
spectrum is computed by integrating the intensity of each G-set slice at different frequencies (detector plane). Consequently, the peaks in the power spectrum usually have the highest signal-to-noise ratio for the further determination of geometric relationships. Sorting the Brag peaks by their intensity, the index of optional Bragg peaks is shown in the following cell (with (0,0) is at k=0 in reciprocal space):

```
print(descend_idx[0][1:27]-pcenter_y)
print(descend_idx[1][1:27]-pcenter_x)
```

```
[-13  13 -26  26  39 -39  37 -37   7  -7 -30  30  38 -38   7  -7 -30  30
 -38  38  12 -12  37 -37  -8   8]
[ 36 -36 -30  30  -6   6 -11  11 -38  38 -26  26 -11  11 -37  37 -27  27
   6  -6 -36  36  -6   6  38 -38]
```

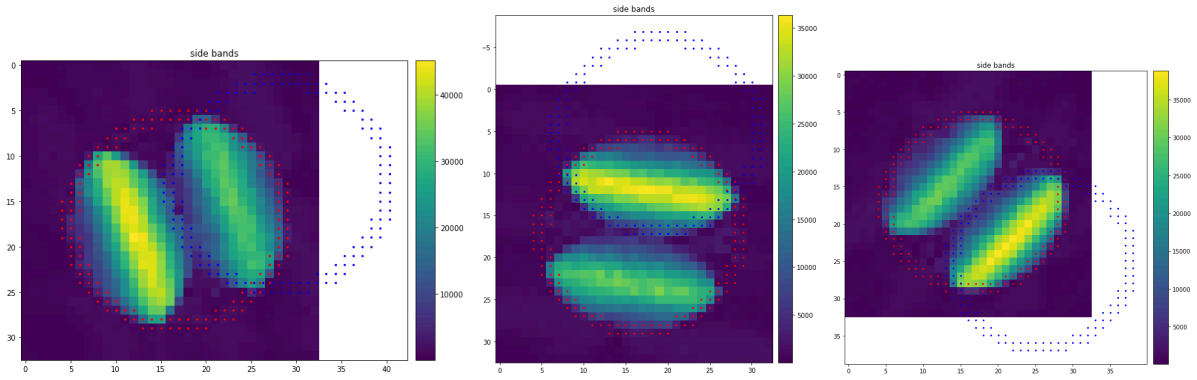
Pick up three Bragg peaks with large intersection angles and high intensity; the intensity distribution of these G-set slices will be plotted in the following cell:

```
idx_i = [ 13,39, 26]
idx_j = [-36,-6, 30]
```



In these three G-set slices, the position of the BF disk, and two diffracted disks will be obviously seen. These two diffracted disks are shifted at the same distance but in the opposite direction. The key to geometric calibration is locating the diffracted disks precisely. For 2D materials, the outlines of these disks are obvious, while these outlines will be interrupted by multiple scattering in bulk materials. Moreover, the incoherence of the electron beam, in addition to residual aberrations influences the intensity distribution inside the overlapped regions of these disks. To determine the position of diffracted disks, a coarse rotation angle and shift radius should be initialized by users as shown by the following cell:

```
angle1 =-0.20
dist1 = 35.8
```



The value of each angle and distance should be coarsely tuned to ensure the outline of the aperture matches the intensity distribution of G-set slices. After this is done, the angles and distances will be optimized iteratively in the following cell:

```
pacbed_grad[np.where(pacbed_grad < np.max(pacbed_grad)*thresh)] = 0
#automatic optimization of the above parameters by assuming the aperture is a rounded shape.
trotter_params1 = np.asarray([dist1, angle1])
slices_grad = np.sqrt(np.sum(np.square(np.gradient(slice1)), axis = 0))
slices_grad[np.where(slices_grad < np.max(slices_grad)*thresh)] = 0
trotter_params1 = optimize.minimize(GradMaximizer, trotter_params1, args=(slices_grad, pacbed_grad), method='Nelder-Mead').x
print(trotter_params1)
#the second one.
trotter_params2 = np.asarray([dist2, angle2])
slices_grad = np.sqrt(np.sum(np.square(np.gradient(slice2)), axis = 0))
slices_grad[np.where(slices_grad < np.max(slices_grad)*thresh)] = 0
trotter_params2 = optimize.minimize(GradMaximizer, trotter_params2, args=(slices_grad, pacbed_grad), method='Nelder-Mead').x
print(trotter_params2)
#the third one.
trotter_params3 = np.asarray([dist3, angle3])
slices_grad = np.sqrt(np.sum(np.square(np.gradient(slice3)), axis = 0))
slices_grad[np.where(slices_grad < np.max(slices_grad)*thresh)] = 0
trotter_params3 = optimize.minimize(GradMaximizer, trotter_params3, args=(slices_grad, pacbed_grad), method='Nelder-Mead').x
print(trotter_params3)
```

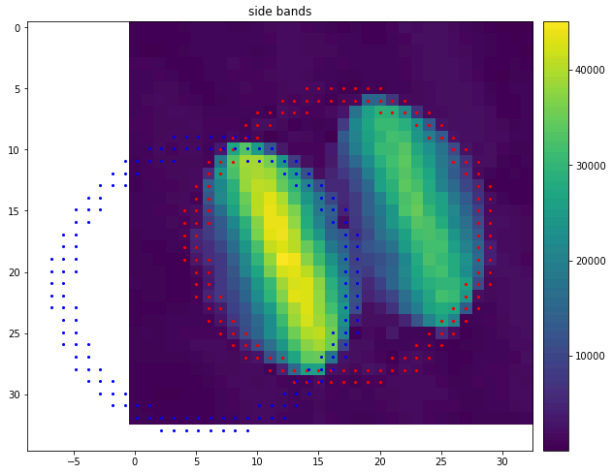
```
[11.775     -0.3474375]
[12.52441406 -1.41353516]
[12.16875  0.73125]
```

Based on these angles, the geometric parameters including the rotation angle of the fast, and slow scanning direction, and the scanning step size will be computed by solving the equation groups:

```
Rotation angle of the fast scanning direction is:  -0.0767549575295729
Rotation angle of the fast scanning direction is:  -0.082014891433705
Scanning Step ration between fast/slow direction is:  1.2941696798543538
The scanning vector along the fast scanning direction is:  [ 0.42063291 -0.03234921]
The scanning vector along the slow scanning direction is:  [0.02670535 0.32488548]
The dimension of the 4D-STEM dataset is:  (256, 256, 126, 126)
```

These parameters will be used in both SSB/WDD and iterative ptychographic reconstruction and dramatically influences the reconstruction result, and the resolved aberration distribution inside the aperture. To verify the accuracy of the determined geometric relationship, one arbitrary G-set slice is chosen, and the value dl should be input to determine the shift distance of diffracted disks:

```
#compute the rotation angle and length of reciprocal scanning vector
#For any diffracton frame indexed as: [center_y+ i, center_x + j]
#the shift vector is: j*kx+i*ky = j*dl*(cos(theta_x), sin(theta_x)) + i*dl*gama*(cos(theta_y), sin(theta_y))=(dx, dy)
dl = 0.30
slice_idx = 0
```



The value of dl (scan interval in reciprocal space) can be manually tuned or computed using the following equation:

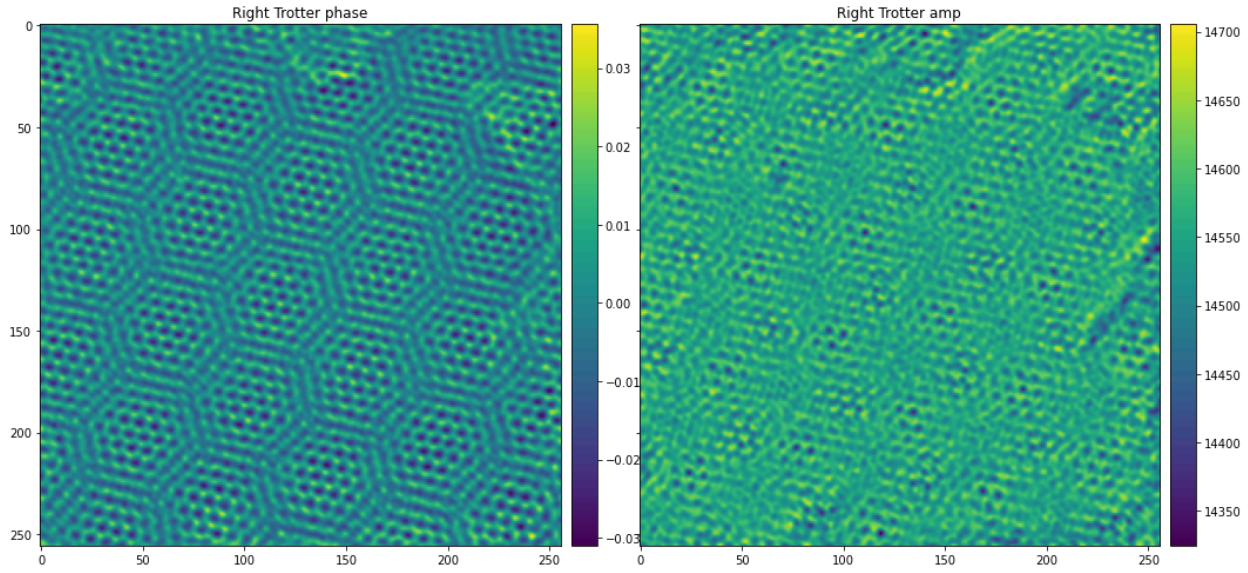
$$dl = \lambda/FoV/dg$$

Where λ is the wavelength of the electron, FoV is the width of the scanned field of view, and dg is the angular value of each detector pixel. We use Å and mrad as length and angle unit throughout this script.

```
#firstly set the pixel length in the reciprocal space.
def Wavelength(Voltage):
    """Compute the wave length of the electron
       the unit of voltage should be kV and returned unit of wavelength is angstrom
       parameters Voltage: Microscope accelerating voltage.
     """
    emass = 510.99906
    hc = 12.3984244
    wavelength = hc/np.sqrt(Voltage * (2*emass + Voltage))
    return wavelength
dl = 1000*Wavelength(60.0)/108/pixel_angle
print(dl)
```

```
0.2966283678558262
```

After determining the dl value, the accurate position of diffracted disks for each G-set slice can be solved. Then the SSB reconstruction without aberration correction can be conducted:



When the Geometric relationship is correct, the atom columns show bright phase contrast. When the flip appears and the geometric relationship is correct, the atom columns show bright phase contrast in the inversed phase distribution. After that, the aberration correction should be conducted to give a better phase reconstruction using SSB.

From the phase distribution in the double overlapped regions of the BF disk and two diffracted disks, it's possible to solve the aberrations inside the BF disk from the G-set slices. Instead of solving the 2D distribution function, the aberration coefficients (Zernike polynomial coefficient) are fitted to ensure robustness. For 4D-STEM datasets with a relatively low signal-to-noise ratio, the aberration order can be set to 2 or 3, and the order can be improved to 5 even 6 order when the signal-to-noise ratio is high enough.
The user can specify the maximum aberration order and number of G-set slices with *order* and *valid_slices* parameters in line 6 and 7 below. The G-set slices are sorted in descending order of intensity. *Valid_slices=N* should be chosen such that the double overlapped regions of the n-th G-set slices are well-defined for n<N.

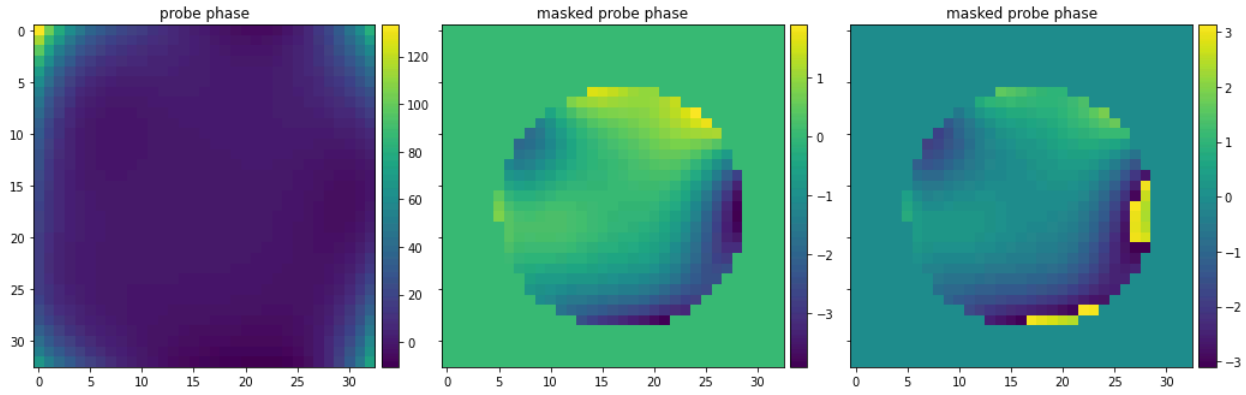
```
# How to solve the aberration coefficient?
# AX = b;
# A is the construction matrix,
# X is the aberration coefficient [C1, C12a, C12b, C23a, C23b, C21a, C21b, C3, C34a, C34b, C32a, C32b, Qp1, Qp2.. Qpn]';
# b is the phase{G(Kf,Qp)} inside the double overlapped areas.
order = 6 # Choose the maximum aberration order to solve
valid_slices = np.arange(17) # choose the top N brightest G-glices based on the distribution of the double overlapping region above
#valid_slices = np.asarray([0, 1, 2, 4, 5, 6, 7, 9, 10, 11, 13, 14, 17, 18])
for ipt in range(valid_slices.shape[0]):
    slice_idx = valid_slices[ipt]
    row_idx=descend_idx[0][slice_idx+1]
    col_idx=descend_idx[1][slice_idx+1]
    shift_x = col_idx-pcenter_x
    shift_y = row_idx-pcenter_y
    delta_x = dl*(shift_x*np.cos(theta_x)+shift_y*np.cos(theta_y)*gama)
    delta_y = dl*(shift_x*np.sin(theta_x)+shift_y*np.sin(theta_y)*gama)
    AMatrixT = GenAMatrix(x_freq, y_freq, delta_x*freqx_inter, delta_y*freqy_inter, valid_regions[slice_idx],order)
    BMatrixT = unwraps[slice_idx][valid_regions[slice_idx]]
    if ipt ==0:
        BMatrix = BMatrixT
        AMatrix = np.zeros((AMatrixT.shape[0]+1,AMatrixT.shape[1]), AMatrixT.dtype)
        AMatrix[0:AMatrixT.shape[0],0:AMatrixT.shape[1]] = AMatrixT
        AMatrix[-1,0:AMatrixT.shape[1]] = 1
    else:
        BMatrix = np.concatenate((BMatrix, BMatrixT))
        AMatrixF = np.zeros((AMatrix.shape[0]+1,AMatrix.shape[1]+AMatrixT.shape[1]), AMatrix.dtype)
        AMatrixF[0:AMatrix.shape[0],0:AMatrix.shape[1]] = AMatrix
        AMatrixF[0:AMatrixT.shape[0], AMatrix.shape[1]:AMatrixF.shape[1]]=AMatrixT
        AMatrixF[-1,AMatrix.shape[1]:AMatrixF.shape[1]] =1
        AMatrix = AMatrixF
```

The fitted aberration coefficient and the averaged phase value of each double-overlapped region of G-set slices are:

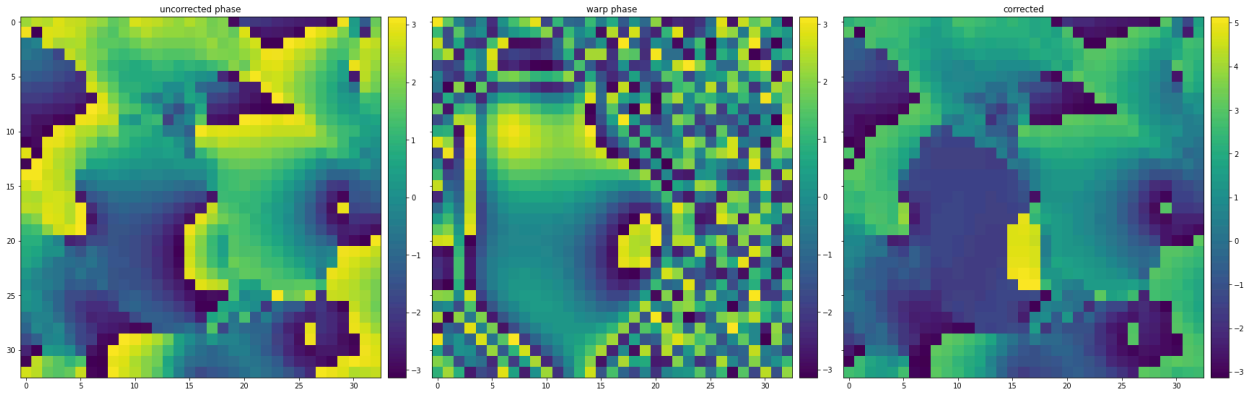
```
BMatrixP = AMatrix.dot(BMatrix)
AMatrixP = np.dot(AMatrix, AMatrix.T)
Aberrations = np.linalg.solve(AMatrixP, BMatrixP)
print(Aberrations)
```

```
[ 6.55534427e+00  9.84825486e+00 -2.47527939e+01  4.77405265e+01
 -1.25379446e+02 -4.96843240e+01  7.99549336e+01 -8.63529281e+02
 -9.56681170e+02 -1.71227268e+01  7.04618098e+00  3.59942862e+02
 -1.45743174e+03 -3.02310865e+01  1.74735321e+02 -6.75002458e+02
 -7.46517275e+02  7.13342045e+00  7.14113354e+03  8.40449179e+03
  1.97529019e+03 -6.37523357e+02 -2.44317179e+03  1.62179046e+02
 -7.18467777e+02  4.98791989e+00 -1.47929420e+00  3.30205011e+00
  4.37511272e-01  3.06281519e-01  3.34393778e+00  2.57869510e+00
  1.04036937e+00 -7.12274462e-01  4.33887671e+00 -7.68472829e+00
 -8.88963348e-01 -7.08505625e-01  4.39570769e+00 -3.78500556e+00
  7.44747417e+00  1.72751367e+00  1.94804012e+00  5.71881251e+00
 -2.08549455e+00  1.91147586e+00  1.59602147e+00  4.02964965e+00
 -5.14546000e-01  8.90069902e-01  2.67101511e+00]
```

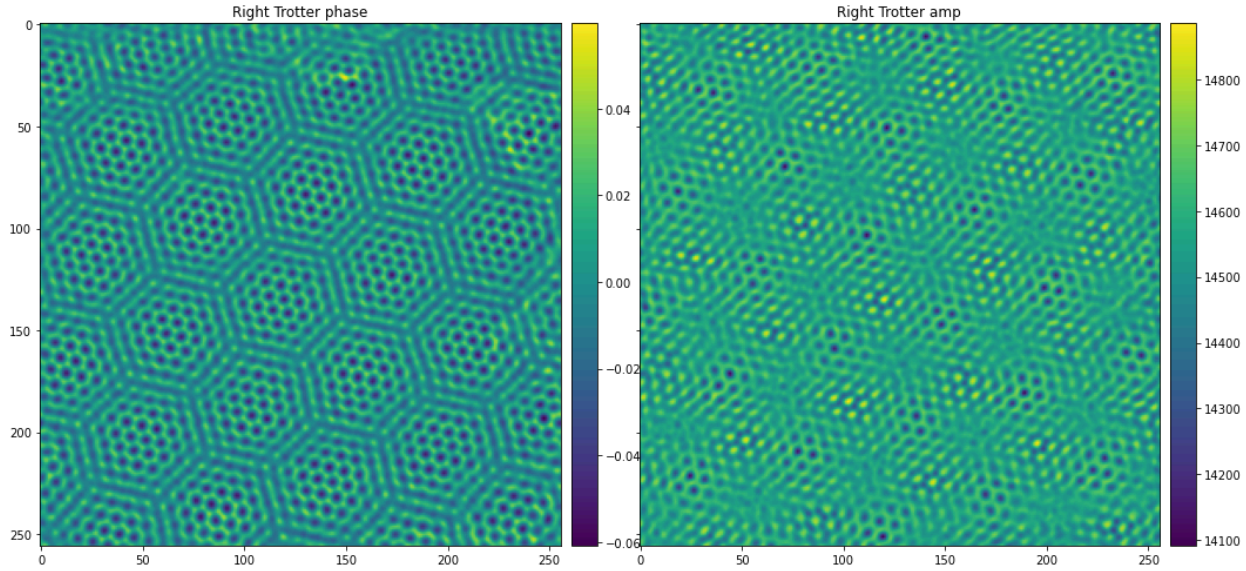
The corresponding 2D phase distribution inside the aperture are:



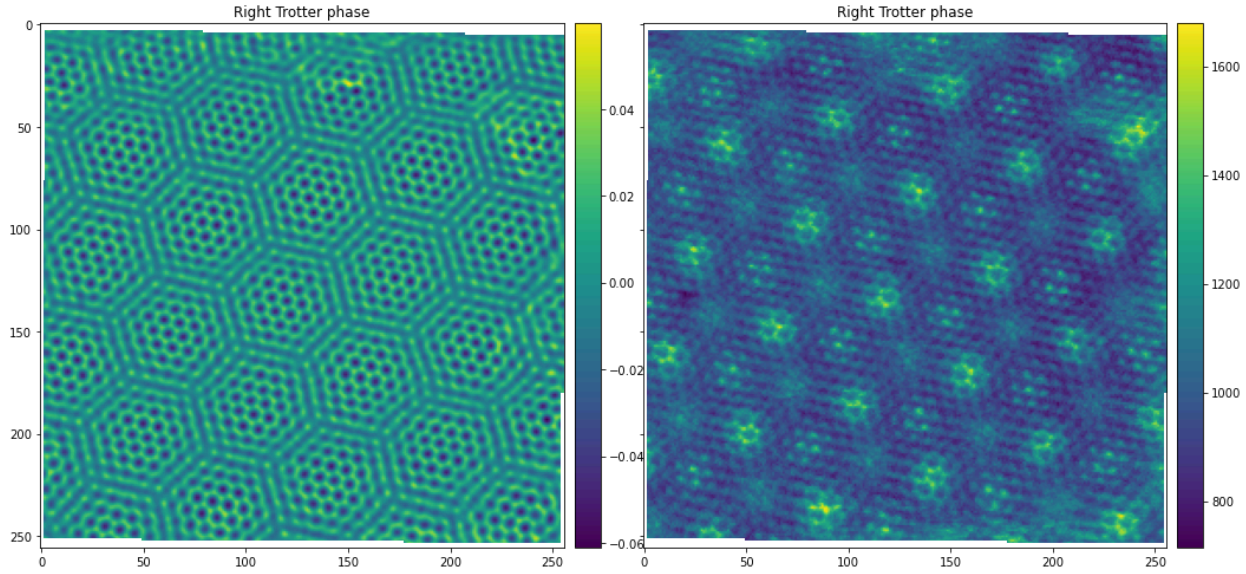
Taking the G-set slice with index [9, -36] as an example, the extra phase shift in one of the double-overlapped region of BF disk and diffracted disks can be eliminated, and a uniform phase distribution is unveiled to show the effectiveness of aberration correction.



The determined aberration coefficients can be further used to eliminate the residual aberrations in the double-overlapped regions of each G-set slice, and the influence of residual aberrations on the reconstructed phase can be fully eliminated as shown in the following phase distribution.



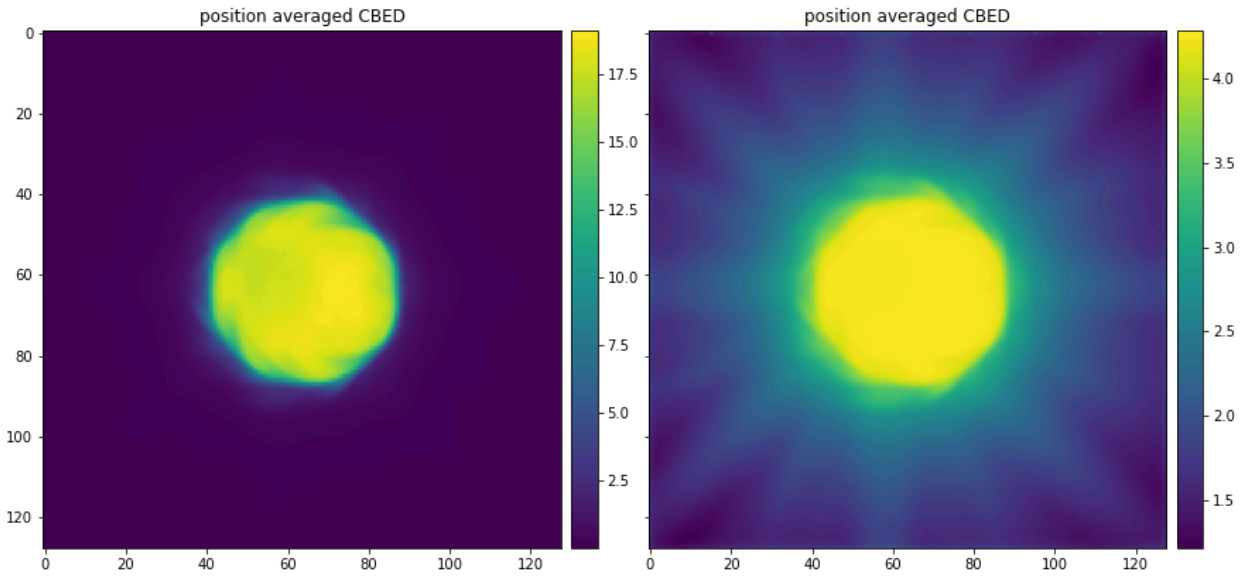
Using the determined geometric relationship, the ADF-STEM image and the phase image are transformed to the coordinate system of the electron camera:



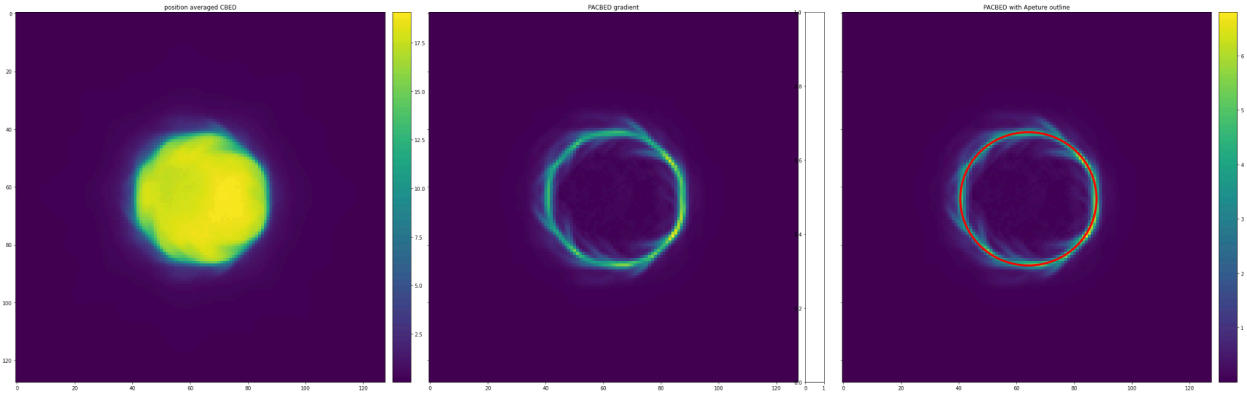
3.2 Thin bulk materials. (Bulk Calibration.ipynb)

This python script is called when the sample is around several tens of nanometers (usually less than 50), and the outline of the aperture can still be partly seen in the PACBED pattern and G-set slices. For thicker samples, please check the following two examples. Similar to the case of 2D materials, the 4D-STEM dataset in .npy format is first loaded and the PACBED pattern is generated and plotted in the first step. Taking the 4D-STEM dataset captured using the EMPAD detector at 300kV accelerating voltage as an example, the defocus of the probe-forming lens is set to -20 nm, indicating the electron beam focuses inside the sample. The scanning step size is set to 0.26 A, and the aperture size of the probe-forming lens is 25mrad. After running the following cell, the PACBED is generated and will be visualized in the following cell:

```
folder = "D:/4D-STEM Data/EMPAD/STO/Tsinghua/experiment/1/"
dat4d = np.load(folder+'data.npy')
#dat4d = dat4d[:,:, 68:188, 68:188]
#dat4d = dat4d[10:246,10:246]
pacbed = np.average(dat4d, axis=(0,1))
#np.save(folder + "data.npy", dat4d)
print(dat4d.shape)
```



The frame size of the diffraction patterns and scanning position numbers can be modified in this cell to prevent dead pixels in the electron detector and invalid scanning positions (synchronization problem or strong distortion). Due to the multiple scattering of the electron inside thick samples, the outline of the aperture is interrupted as shown in the above PACBED pattern. It's challenging to determine the angular value of each pixel even if the size of the aperture in mrad is provided. Fortunately, the oval-fitting function is powerful enough to overcome this problem and reliably determine the angular value of each detector pixel as shown below:



For this 4D-STEM dataset, the determined angular value of each detector pixel is 1.072 mrad, and the aperture intensity 2D distribution (***probe mask.npy***) is saved for further iterative ptychography reconstruction.

```
1.0235543048275826
radius of the aperture in pixels:  23.599995394366985  pixel
angular value of each pixel:  1.0716528209881866  mrad
```

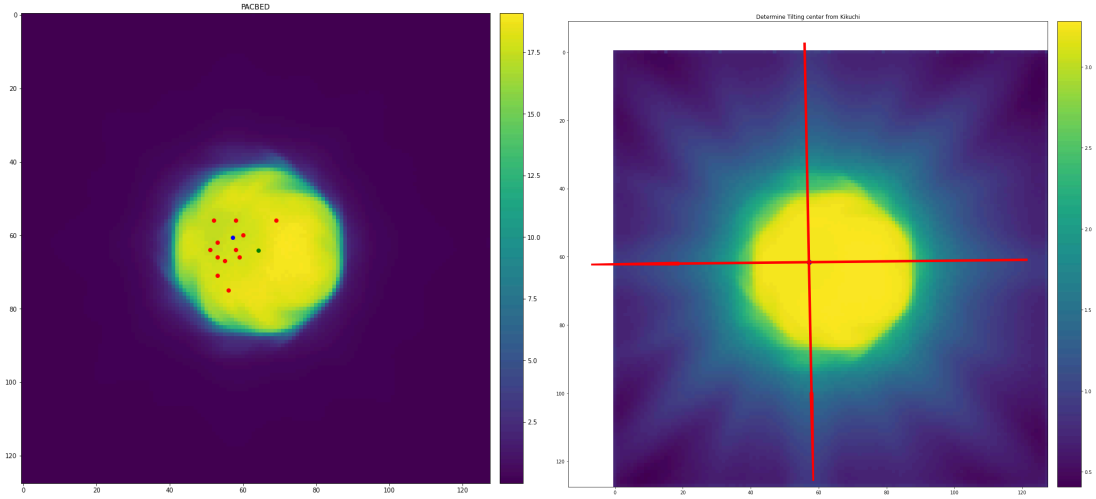
Different from the calibration of 2D materials, the tilting angle of the sample should be determined for further iterative ptychography reconstruction and the determination of geometric

relationships. There are two different options to determine the tilting angle of the sample: 1). determine the peaks/troughs in the BF part of the PACBED pattern. 2). Determine the tilting center of the sample from the Kikuchi line. The following cell is for the first option:

```
from skimage.morphology import square
from skimage.morphology import dilation
from skimage.morphology import erosion
from scipy.spatial import distance_matrix
#pacbedb = dilation(pacbed, square(6))
pacbedb = erosion(pacbed, square(4))
peak_pos = np.asarray(np.where((pacbed <= pacbedb) & (pacbed > np.max(pacbed)*0.6)))
center_pos = np.reshape(np.asarray([y_center, x_center]), (1, 2))
#get the nearest peak to the center of the CBED.
tilt_center = peak_pos[:,(np.argmin(distance_matrix(peak_pos.T, center_pos)))]
xt_center = tilt_center[1]
yt_center = tilt_center[0]
#xt_center = x_center - 7
#yt_center = y_center - 3.5
#fit the Gaussian blob to the subpixel precision.
```

If the tilt center of the sample shows higher intensity than its surrounding pixels, the dilation function should be called to identify the intensity peak. If the tilt center of the sample shows lower intensity than its surrounding pixels, the erosion function should be called to identify the trough. If the feature of the tilting center is not obvious, users are suggested to manually set the tilting center. In the following plot, the tilting center of the sample is marked by the blue dot, and the center of the aperture is marked by the green dot. Users can iteratively adjust the position of the tilting center to make the position of the tilting center match the intensity distribution inside the BF aperture.

```
xt_center = x_center - 7
yt_center = y_center - 3.5
```

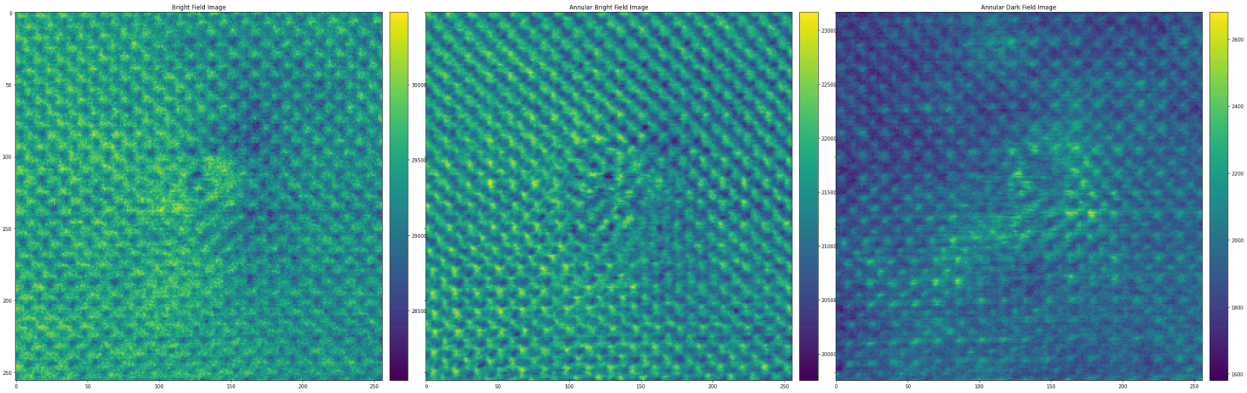


Besides the BF center, the Kikuchile also offers rich features for the identification of the tilting center. For crystalline samples that are able to scatter electrons strongly, the Kikuchi lines become obvious features in the PACBED, and the crossing of these Kikuchi lines is right at the tilting center. In the current software version, users need to manually adjust the tilting angle and the center of these Kikuchi lines. After the tilting center is well set, the tilting angle of the sample is printed and used in further iterative ptychography reconstruction.

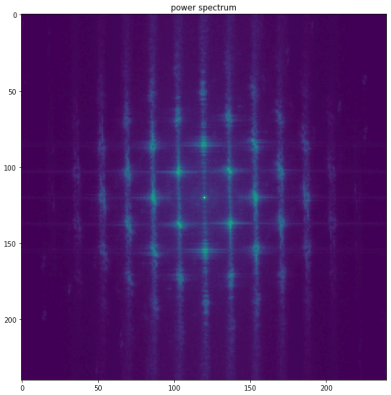
```
print("Tilting angle along x direction: ",(xt_center-x_center)*pixel_angle)
print("Tilting angle along y direction: ",(yt_center-y_center)*pixel_angle)
```

```
Tilting angle along x direction:  -7.501569746917307
Tilting angle along y direction:  -3.7507848734586533
```

After the determination of the tilting center and aperture center, the STEM images including the BF-STEM, ABF-STEM, and ADF-STEM images are generated to check if the scanning position are registered with diffraction patterns, and if diffraction patterns are well captured. If scanning pixels have zero ADF-STEM intensity, then the whole scanning lines where these pixels belong to have to be trimmed and abandoned.



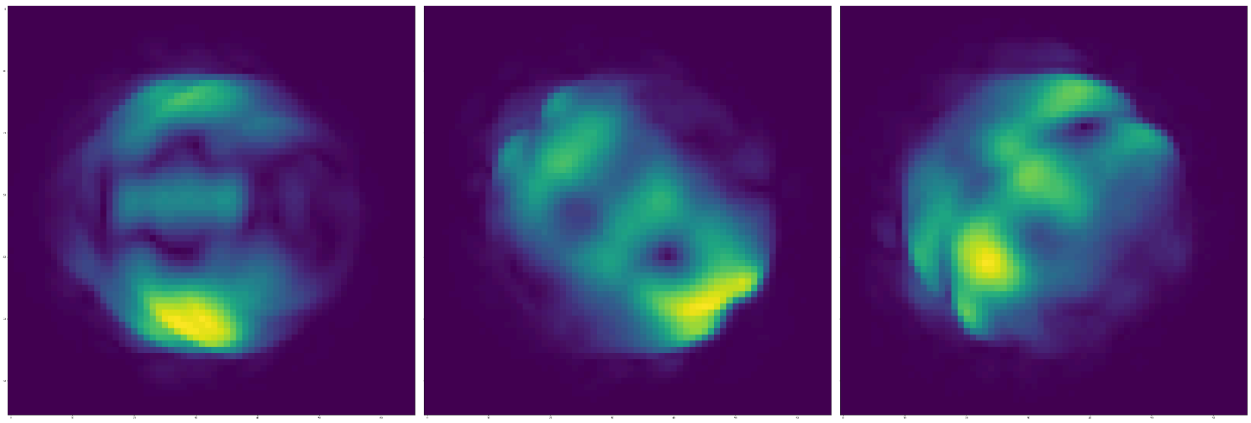
When the number of scanning positions is well set, and the dead pixel lines in diffraction patterns are well processed, the Fourier transformation should be applied to the 4D-STEM dataset along scanning positions. This procedure is the same as the case of 2D materials, and the corresponding diffractogram is printed as follows:




```
idx=np.unravel_index(np.argsort(power_spectra,
descend_idx=(idx[0][::-1],idx[1][::-1])
print(descend_idx[0][1:27]-pcenter_y)
print(descend_idx[1][1:27]-pcenter_x)
```

```
[ 34 -34  35 -35  33 -33  34 -34 -35  35  33 -
  34 -34   2  -2 -34  34  68 -68]
[-33  33  34 -34 -34  34  35 -35 -33  33 -35
 -32  32  68 -68 -34  34  -1   1]
```

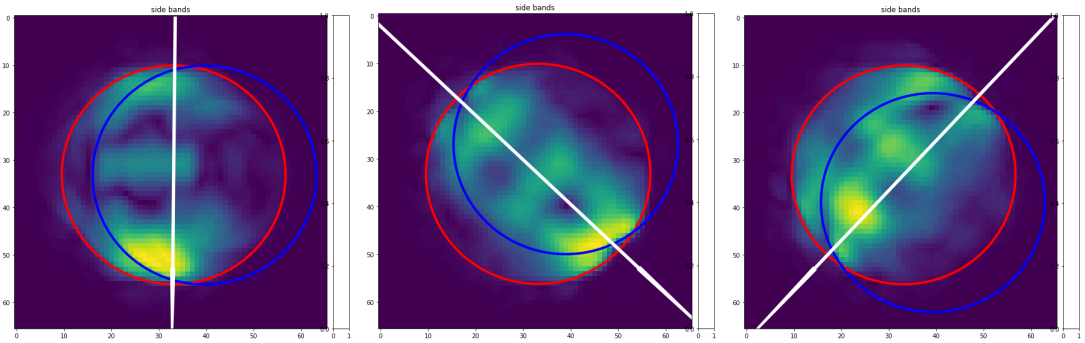
The G-set slices are sorted by their total intensities and appear in pairs since the transformed 4D-STEM dataset are conjugated in the reciprocal space corresponding to scanning positions. After selecting three G-set slices with the highest intensity and large intersection angle, their corresponding intensity distribution is plotted and the outline of the aperture is smeared by multiple scattering.



The index of these G-set slices are :

```
idx_i = [ 18, 0,34]
idx_j = [-16,33, 1]
```

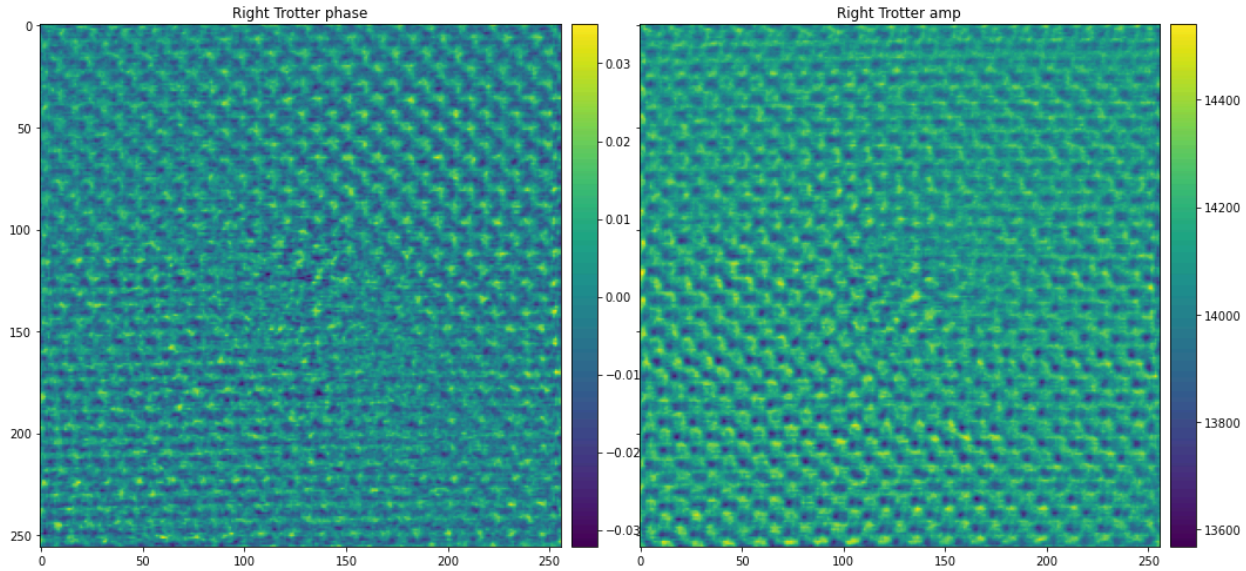
To identify the position of diffracted disks, both the symmetry axis of G-set slices and the outline of the diffracted disks are used.



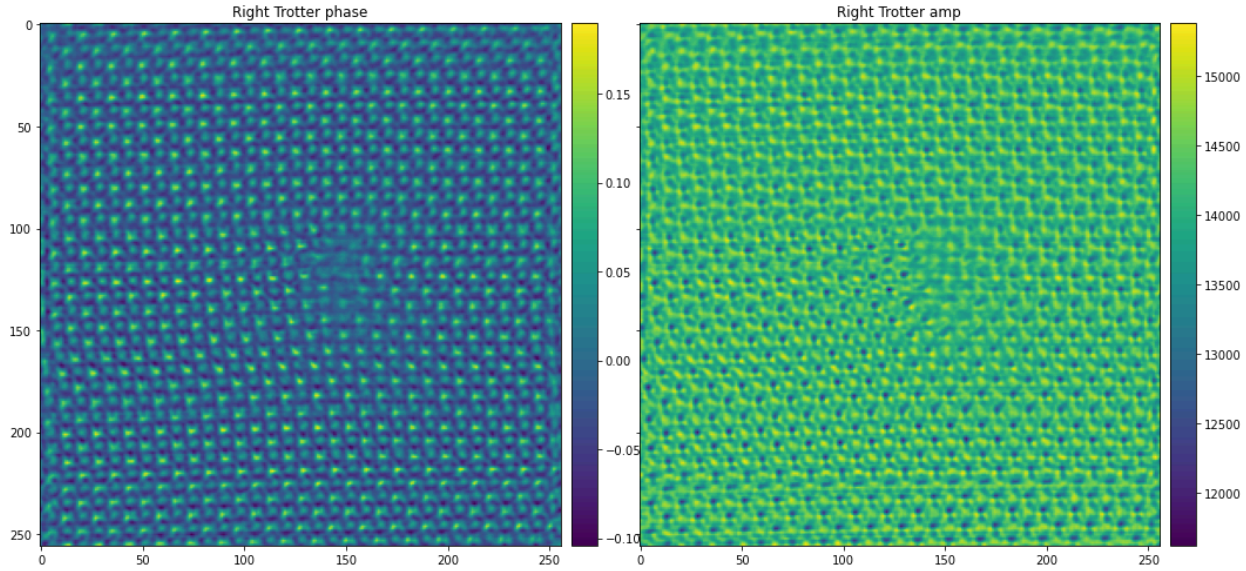
Then the geometric relationship is determined as follows:

```
Rotation angle of the fast scanning direction is:  2.30121295657963
Rotation angle of the fast scanning direction is:  -0.8199999999999982
Scanning Step ration between fast/slow direction is:  0.9943021749856402
The scanning vector along the fast scanning direction is:  [-0.17357106  0.19378948]
The scanning vector along the slow scanning direction is:  [0.19130216 0.17850118]
The dimension of the 4D-STEM dataset is:  (256, 256, 128, 128)
```

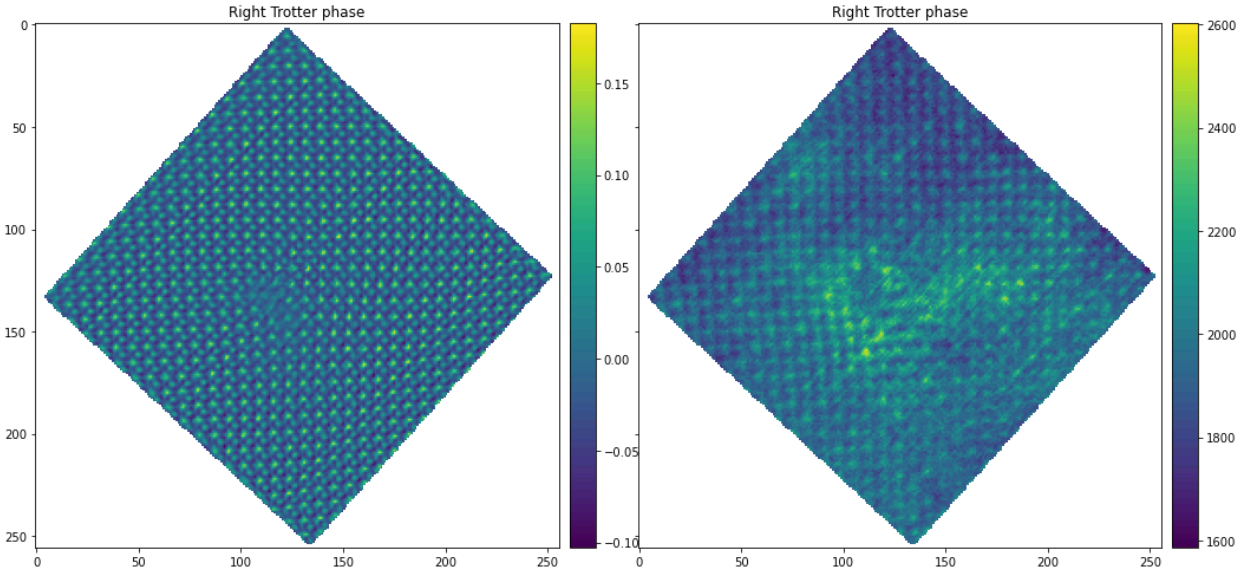
The SSB reconstruction is conducted based on the current geometric relationship:



For thick specimens, it's challenging to determine if the extra 180-degree rotation exits without aberration correction. Consequently, the aberration correction should be conducted, and the reconstructed phase image with aberration correction should be compared with the ADF-STEM image to ensure the uniqueness of the geometric relationship. The aberration correction of the thick specimen is similar to the case of 2D materials, and the reconstructed phase after aberration correction is shown as follows.

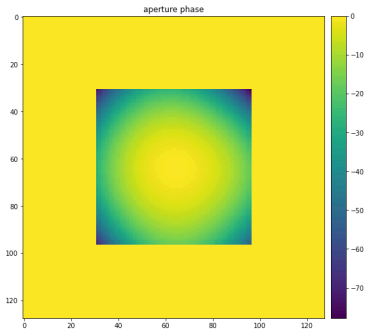


The reconstructed lattice well matches the lattice of STO, and it seems the 180-degree rotation does not exist. To double-check the effectiveness of the geometric relationship, both the phase image and the ADF-STEM image are affinely transformed and registered in the following plot:



Consequently, the ADF-STEM intensity distribution matches the reconstructed phase image, and showing the effectiveness of aberration correction and geometric relationship. In the last step, the phase distribution inside the aperture of the probe-forming lens is saved for further iterative ptychography reconstruction.

```
aper_phase = np.zeros_like(pacbed)
aper_phase[y_start:y_end,x_start:x_end] = probe_phase
fig =plt.figure(1, figsize=(30, 30))
grid=AxesGrid(fig, 236, nrows_ncols=(1,1),
              axes_pad=0.5,
              share_all=False,
              cbar_location="right",
              cbar_mode="each",
              cbar_size="5%",
              cbar_pad="2%")
im=grid[0].imshow(aper_phase)
grid.cbar_axes[0].colorbar(im)
grid[0].set_title('aperture phase')
np.save(folder + "probe phase.npy", np.float32(aper_phase))
```



3.3 Thick materials with strong scattering. (Kikuchi Calibration.ipynb)

Similar to other script, the basic library and functions will be defined in the first two cells of ***Kikuchi Calibration.ipynb***. Then the 4D-STEM dataset will be loaded in the following cell, and the dimension of the 4D-STEM dataset should be trimmed for the following two cases:

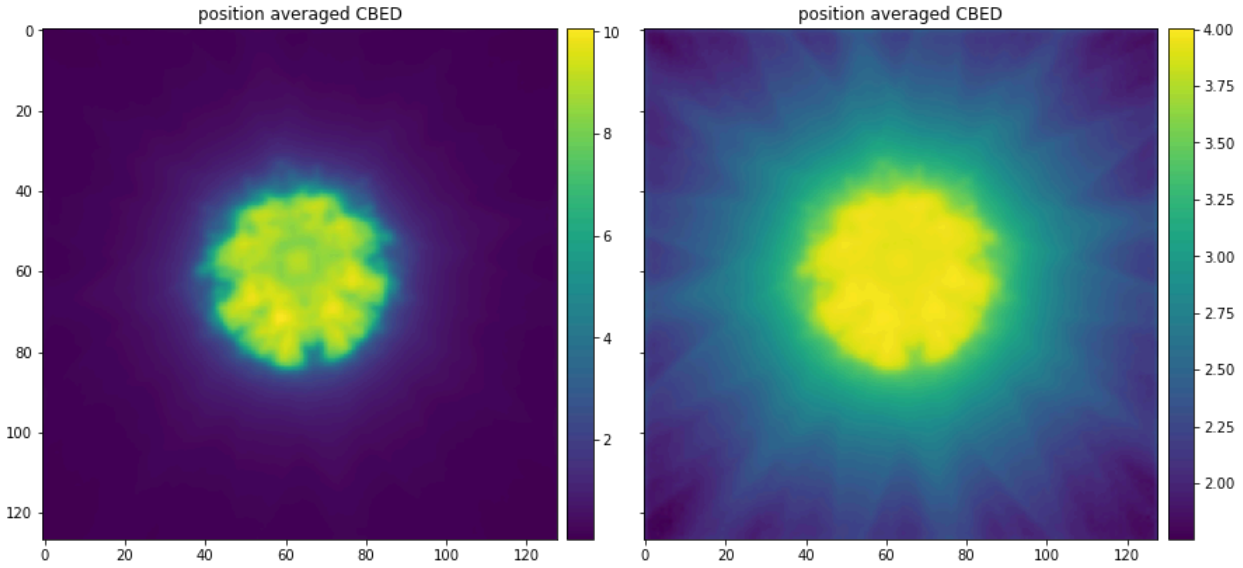
1. The border of the detector is dead or enhanced.
2. The first few scanning lines have no signal or have unsynchronized signals, and this is frequently observed in the Merlin Medipix3 and EMPAD systems.

The correction of the CBED size is usually done after checking the PACBED, and the correction of scanning size is done after checking the corresponding STEM frames of the loaded 4D-STEM

dataset. (Please go back to the third cell if you found the above two cases, and run this script again.)

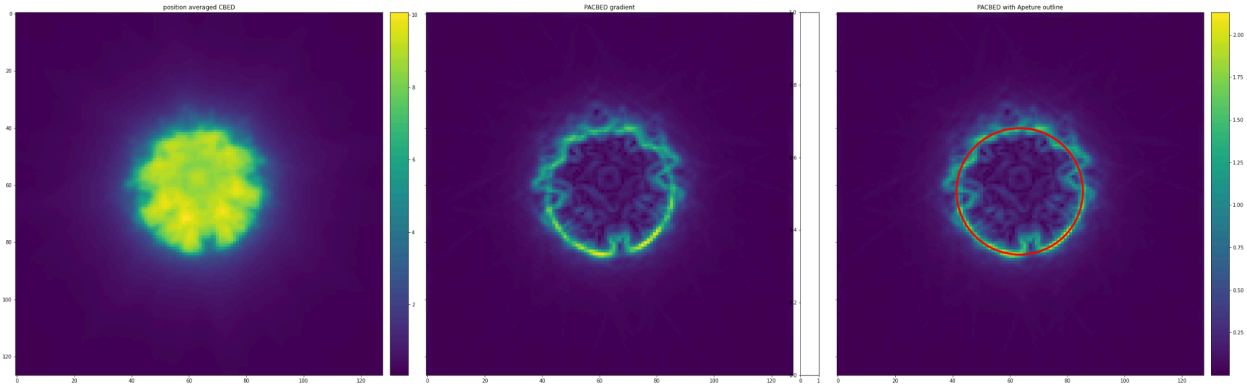
```
folder = "D:/4D-STEM Data/EMPAD/STO/6/"
dat4d = np.load(folder+'data.npy')
print(dat4d.shape)
dat4d = dat4d[:,:, 0:126, 1:127]
#dat4d = dat4d[10:246,10:246]
pacbed = np.average(dat4d, axis=(0,1))
#np.save(folder + "data.npy", dat4d[:,:,1:])
```

In above cell, the PACBED is generated and shown in linear and logrithm space as follows:



For thick specimen with strong scattering, the outline of the aperture is interrupted, and this is quite different from the 2D materials. The modified shape of the BF disk is related to the sample symmetry, sample thickness and orientation (misalignment).

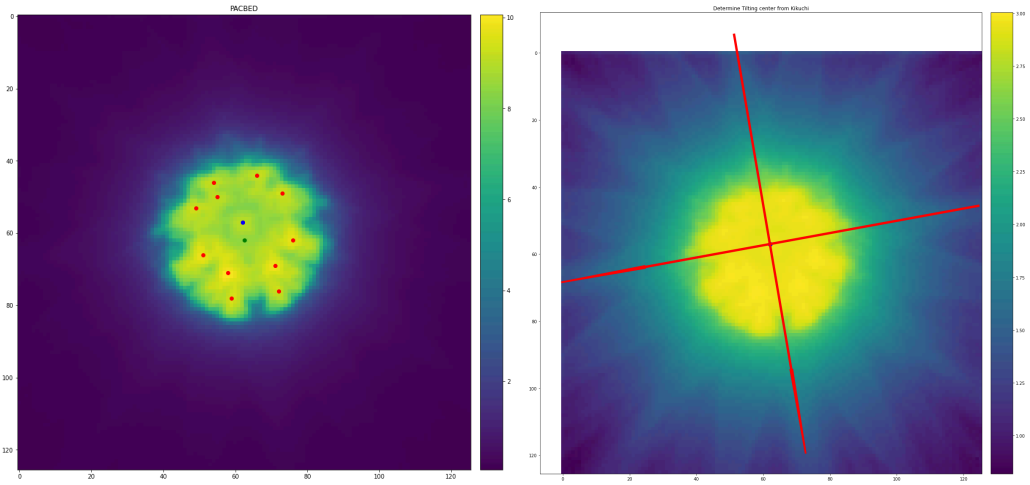
To determine the position of the aperture center, an oval shape of the aperture is assumed. The interrupted outline of the aperture can be well recovered as shown in the following figure:



In addition, the angular size of each pixel in the detector can be determined when the semi-angle of the aperture is provided. For this sample, the aperture size is ***17.9 mrad***, and the angular size of each pixel is ***0.8108 mrad***. This value will be used in the following iterative ptychography reconstruction.

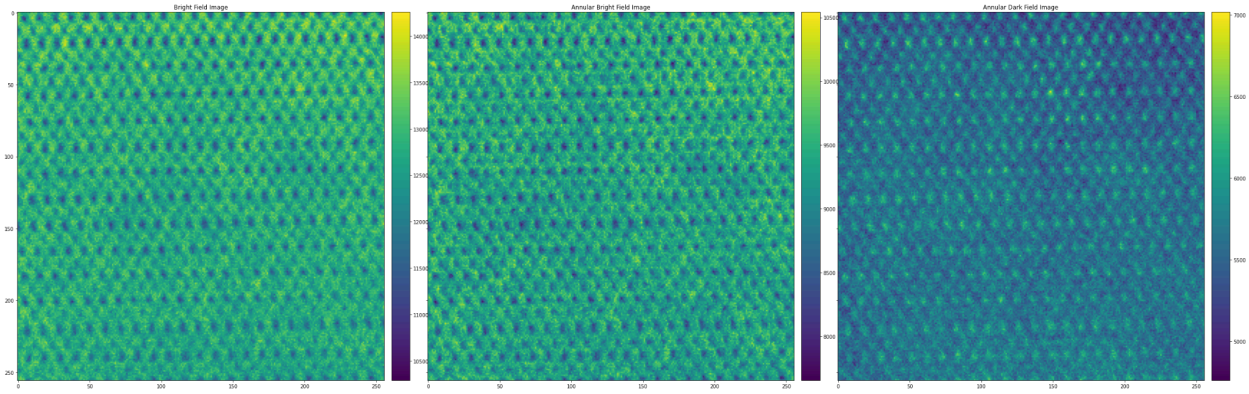
```
#generate the probe mask used for probe initialization.
aperture = 17.9#mrad
pixel_angle = 2* aperture/ (radius + radius / new_params[3])
print(new_params[3])
print("radius of the aperture in pixels: ", radius, " pixel")
print("angular value of each pixel: ", pixel_angle, " mrad")
```

The positional averaged CBED will be further processed in order to get the tilting angle of the sample. In this script, the tilt center can be automatically determined, or determined manually according to the intensity distribution inside the BF aperture or according to Kikuchi line:



The determined tilting angle of the sample will be used in the iterative ptychography reconstruction, in addition to the determination of geometric relationships between scanning positions and the detector in the far field. After processing the PACBED, the STEM frames

including the ABF-STEM, BF-STEM and ADF-STEM image will be generated for this 4D-STEM dataset, and please check if there is obvious appearance of unsynchronized scanning lines and



3.4 Thick materials with weak scattering. (Line Calibration.ipynb)

For thick specimens with weak scattering, the Kikuchi line might not be obviously seen in the PACBED and the Fourier-transformed 4D-STEM datasets. In addition, the outline of the diffracted disk disappears due to the multiple scattering. Then the Kikuchi-based calibration script and the general calibration script for thick specimens are no longer valid. Fortunately, the achromatic line due to the incoherence of the electron source and the inelastic scattering becomes a more obvious feature in the G-set slices of bulk materials. Theoretical analysis and simulations are provided in the **Basic Theory** part. According to our theory, the achromatic line in the G-set slices is usually perpendicular to the shift direction of diffracted disks, and the calibration can be accurate when the orientation of achromatic lines can be well determined.

To determine the orientation of the achromatic line, two different algorithms are designed. The first one is based on the symmetry fitting, the other one is based on the line fitting. The source codes for these two functions are shown as follows:

```
def LineTarget(params, center_x, center_y, Gslice, threshold):
    #measure the distance of each pixel to the line
    distance = 0.0
    angle_val = params[0]
    bvalues = np.cos(angle_val)*center_y - np.sin(angle_val)*center_x
    #select the pixels suitable for the fitting of
    pixels= np.where(Gslice>np.average(Gslice)*threshold)
    weights = Gslice[pixels]
    x_coord = np.asarray(pixels[1])
    y_coord = np.asarray(pixels[0])
    for i in range(weights.shape[0]):
        distance += weights[i]*np.square(np.cos(angle_val)*y_coord[i] - np.sin(angle_val)*x_coord[i] - bvalues)
    return distance
def LineFitting(Gslice, center_x, center_y, threshold, radius_limit):
    Gslice_mask = Gslice.copy()
    Y_idx, X_idx = np.indices(Gslice_mask.shape)
    Gslice_mask[np.where(np.sqrt(np.square(Y_idx - center_y) + np.square(X_idx - center_x))<radius_limit)]=0
    #set the center pixels to zero.
    pixels= np.where(Gslice_mask>np.average(Gslice)*threshold)
    x_coord = np.asarray(pixels[1])
    y_coord = np.asarray(pixels[0])
    weights = Gslice[pixels]
    return np.arctan(np.polyfit(x_coord, y_coord, 1, w = weights)[0])
```

For all of the four scripts, the only difference is the determination of the shift angle of diffracted disks, in addition to the calibration of the rotation angle.

4. Basic theories.

4.1 Coordinate system.

4.2 Computation of Geometric relationship

4.3 Detection of aperture outline.

4.4 Detection of achromatic line.

4.5 Detection of achromatic line.

4.6 Simulations.

4.7 Computation of iCOM(iDPC) from 4D-STEM datasets.