preprocess_demo

May 25, 2022

[ ]:
```python
import sys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import collections as mc
import scipy.signal as sig
import sys
import os
import time
from tqdm.notebook import tqdm, trange
import importlib as imp
import scipy.optimize as opt

import preprocess_4dstem as prep
imp.reload(prep)

# Set matplotlib format
SMALL_SIZE = 12
MEDIUM_SIZE = 15
BIGGER_SIZE = 18

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=BIGGER_SIZE)  # fontsize of the figure title
```

1 Process Overview

This notebook is a companion to “preprocess_4dstem.py” and provides a sample workflow for that module.

Preprocessing 4D-STEM data can use a lot of RAM. This program is organized in a way that reduces RAM use, so the total RAM use should only be 2-3x the size of the original file. To do this, instead of saving a new set of images to memory after each step (centering, artifact subtraction, etc), the modifications are tabulated and only applied when necessary.

For example, Box 1 calculates the centers of each image, but doesn't save the centered images to memory - it simply records the image centers to that they can be used as needed. The same is done for artifact subtraction.

2 Input Parameters

[ ]:
```
# These can be defined manually, or captured from your DM4 file or filename␣
↪using automated
# scripts.  Be aware that metadata in DM4 files is organized inconsistently.

# File System
filename = "sample_4dstem_data_80x80_ss=10_cl=480.dm4"
directory = "./"
dm_path = directory + filename
output_directory = directory

# Diffraction Data
n_grid = 80
n_pixels = 512
step_size = 10
camera_length = 480

print("Reading from File %s\nWriting to Directory %s" % (dm_path,␣
↪output_directory))

# Calculate values from input data
n_images = n_grid**2
q_per_pixel = 1152 * np.pi / (camera_length * n_pixels) # At 256 pixels, 1/d =␣
↪6nm^-1 for CL=480 (GMS 3 shows 5.996).
                                                        # This is equivalent to␣
↪1.2*pi nm^-1 from center to edge.
```

3 Prepare Image Centers

[ ]:
```
x_offset, y_offset = prep.calculate_centroids(dm_path, n_grid, n_pixels,␣
↪verbose=False)

max_error = 2  # Pixels.  Images farther than this from the fit are omitted.
n_cycles = 10  # Number of times to omit high-error images and re-fit
x_centers, y_centers, blocklist = prep.iterative_fit(x_offset, y_offset,␣
↪n_pixels, max_error, n_cycles)
```

4 Prepare Artifacts

[ ]:
```
imp.reload(prep)
average_image = prep.get_average_centered_image(dm_path, n_pixels, x_centers,␣
 ↪y_centers, blocklist, verbose=False)
streak_image = prep.get_streak(average_image, verbose=True)
plt.imshow(average_image, vmax=50)
plt.show()
```

5 Conservative Background Subtraction

[ ]:
```
current_image = average_image - streak_image
background_image = prep.get_background_image(current_image, reduction_factor =␣
 ↪0.9, fit_start=40, fit_end=300, verbose=False)
```

6 View Corrected Diffraction Patterns

[ ]:
```
total_artifact = streak_image + background_image
prep.stitch_diffraction_images(dm_path, n_pixels, x_centers, y_centers,␣
 ↪blocklist, artifact=total_artifact, low_percentile=80, high_percentile=95,␣
 ↪alpha=0.2,
                              save=False, output_directory="/home2/luke/
 ↪simulations_paper/figures/", max_images=1, start_image=0, figsize=(12, 12),␣
 ↪filter_size=None,
                              patterns_per_block=4, colorbar=False,␣
 ↪cmap='viridis')
```

7 Integrate and Convert to Q vs Chi

[ ]:
```
# Set range and resolution
q_min, q_max, q_step = (1.0, 3.0, 0.05)
angle_step = 5

plt.rc('axes', titlesize=BIGGER_SIZE)
q_theta_matrix = prep.integrate_diffraction(dm_path, n_pixels, x_centers,␣
 ↪y_centers,
                                  q_min, q_max, q_step, angle_step, q_per_pixel,
                                  artifact=streak_image,␣
 ↪background=background_image,
```

```
                                        blocklist=blocklist, check_images=5,␣
↪medfilt_size=3, verbose=False,
                                        crosshairs=False, check_vmin=0,␣
↪check_vmax=20, colorbar=False, big=False,
                                        save_check_images=False, save_path='../
↪figures/check_diffraction/')
```

[ ]:
```
path = output_directory + filename.split('.')[0] + '__q %.2f %.2f %.2f a %.2f.
↪npy' % (q_min, q_max, q_step, angle_step)
print(path)
np.save(path, q_theta_matrix)
```

[ ]:
```
```