PySABER

Release 1.0.0

K. Aditya Mohan

May 24, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

PySABER is a python package for characterizing the X-ray source and detector blurs in cone-beam X-ray imaging systems. SABER is an abbreviation for systems approach to blur estimation and reduction. Note that even parallel beam X-rays in synchrotrons are in fact cone beams albeit with a large source to object distance (SOD). X-ray images, also called radiographs, are simultaneously blurred by both the X-ray source spot blur and detector blur. This python package uses a numerical optimization algorithm to disentangle and estimate both forms of blur simultaneously. The point spread function (PSF) of X-ray source blur is modeled using a density function with two parameters. The first parameter is the full width half maximum (FWHM) of the PSF along the x-axis (row-wise) and second is the FWHM along the y-axis (column-axis). The PSF of detector blur is modeled as the sum of two density functions, each with its own FWHM parameter, that are mixed together by a mixture (or weight) parameter. All these parameters are then estimated using numerical optimization from normalized radiographs of a sharp edge such as a thick Tungsten plate rollbar. To simultaneously estimate the PSFs of both source and detector blurs, radiographs must be acquired at two different values for the ratio of the source to object distance (SOD) and object to detector distance (ODD). If each radiograph has a single straight edge, then the measurements must be repeated for two different, preferably perpendicular, orientations of the edge. If the radiograph consists of two intersecting perpendicular edges, then a single radiograph at each specified SOD/ODD is sufficient.

Once the parameters of both source and detector blurs are estimated, this package is also useful to reduce blur in radiographs using deblurring algorithms. Currently, Wiener filtering and regularized least squares deconvolution are two deblurring algorithms that are supported for deblurring. Both these techniques use the estimated blur parameters to deblur radiographs. For more detailed explanation on the experimental methodology and theory of PySABER, please read the paper in *References*.

CHAPTER

TWO

REFERENCES

If you use *pysaber Package*, we request you to cite the following article -

- K. Aditya Mohan, Robert M. Panas, and Jefferson A. Cuadra. "SABER: A Systems Approach to Blur Estimation and Reduction in X-ray Imaging." arXiv preprint arXiv:1905.03935 (2019) [pdf]

CHAPTER

THREE

LICENSE

LLNL-CODE-766837. This project is licensed under the MIT License.

MIT License

CHAPTER

FOUR

INSTALLATION

pysaber is installed using the python package manager pip. To install pysaber, run the following command in a terminal:

```
pip install pysaber
```

Alternatively, to install using the source code in the github repository pysaber, first download the repository using the download link in the top right corner of the webpage. Or, you can also git clone the repository directly from github. In a terminal, change the current directory to the outermost folder of this downloaded repository, which contains the README file, and run the following command:

```
pip install .
```

It is recommended to install the *pysaber Package* within a python virtual environment.

CHAPTER

FIVE

TUTORIAL

The steps involved in estimating blur PSFs and deblurring radiographs are outlined in the links below.

5.1 Input Sanity Check

- The first step to estimating the blur model involves computation of the transmission function, which is the ideal radiograph image that is formed in the absence of X-ray source and detector blurs. This computation of transmission function is performed internally in the function *pysaber.estimate_blur()*, which is used to estimate the blur model by computing the parameters of X-ray source and detector blurs. However, this computation of transmission function is not fail-proof and may result in inaccurate edge localization if certain assumptions made when computing the transmission function are not satisfied.
- Before using *pysaber.estimate_blur()* to estimate blur model, it is recommended to check for accurate edge localization in the transmission function. The transmission function can be computed using the function *pysaber.get_trans_masks()*.
- The function *pysaber.get_trans_masks()* also returns the mask arrays for the transmission function and radiograph, which are used to include or exclude certain pixels from blur estimation. By default, the radiograph mask only excludes a small number of pixels along the boundary of the radiograph from blur estimation. Additional pixels can be excluded from blur estimation by appropriately setting the input arguments of the functions *pysaber.get_trans_masks()* and *pysaber.estimate_blur()*. The mask for transmission function should also exclude the padded pixels in addition to those pixels excluded by the radiograph mask. Hence, *pysaber.get_trans_masks()* is also useful to check if user expectations for the mask arrays are satisfied.
- Example python scripts that demonstrate the above procedure are shown below. To obtain the data that is required to run this script, download and unzip the zip file at the link `data`. To run the script as is within the current working directory, the files in the zip file must be placed within a folder called `data`.

Listing 1: Verify transmission function and masks for horizontal edge radiograph.

```
import numpy as np
from PIL import Image #To read images in TIFF format
from pysaber import get_trans_masks #To compute transmission function and masks
import matplotlib.pyplot as plt #To display images

pix_wid = 0.675 #Width of each pixel in micrometers

#Read a horizontal edge radiograph for which the transmission function must be computed
rad = Image.open('data/horz_edge_25mm.tif') #Read radiograph
rad = np.asarray(rad) #Convert to numpy array
```

(continues on next page)

(continued from previous page)

```
bright = Image.open('data/horz_bright.tif') #Read bright field
bright = np.asarray(bright) #Convert to numpy array
dark = Image.open('data/horz_dark.tif') #Read dark field
dark = np.asarray(dark) #Convert to numpy array
nrad = (rad-dark)/(bright-dark) #Normalize radiograph

#Get the transmission function and masks
trans,trans_mask,rad_mask = get_trans_masks(nrad,edge='straight',pad=[3,3])
#Use pad = [1,1] if you do not want padding

#Display the array trans
#Visually check for inaccurate localization of the sharp-edge
plt.imshow(trans,cmap='gray')
plt.colorbar()
plt.show()

#Display and inspect the mask for transmission function
plt.imshow(trans_mask,cmap='gray')
plt.show()

#Display and inspect the mask for radiograph
plt.imshow(rad_mask,cmap='gray')
plt.show()

#Show a line plot comparing the measured radiograph and transmission function
sz = nrad.shape
coords = np.arange(-(sz[0]//2),sz[0]//2,1)*pix_wid
mid = (trans.shape[0]//2,trans.shape[1]//2)

plt.plot(coords,nrad[:,sz[1]//2])
#Due to padding, trans is three times the size of nrad in each dimension
#For proper alignment in the presence of padding, both nrad and trans are center aligned
#Center alignment is used since an equal amount of padding is applied at both ends of␣
↪each axis
plt.plot(coords,trans[mid[0]-(sz[0]//2):mid[0]+(sz[0]//2),mid[1]])
plt.xlabel('micrometers')
plt.legend(['Measured','Transmission'])
plt.show()
```

Listing 2: Verify transmission function and masks for vertical edge radiograph.

```
import numpy as np
from PIL import Image #To read images in TIFF format
from pysaber import get_trans_masks #To compute transmission function and masks
import matplotlib.pyplot as plt #To display images

pix_wid = 0.675 #Width of each pixel in micrometers

#Read a vertical edge radiograph for which the transmission function must be computed
rad = Image.open('data/vert_edge_25mm.tif') #Read radiograph
rad = np.asarray(rad) #Convert to numpy array
```

(continues on next page)

(continued from previous page)

```
bright = Image.open('data/vert_bright.tif') #Read bright field
bright = np.asarray(bright) #Convert to numpy array
dark = Image.open('data/vert_dark.tif') #Read dark field
dark = np.asarray(dark) #Convert to numpy array
nrad = (rad-dark)/(bright-dark) #Normalize radiograph

#Get the transmission function and masks
trans,trans_mask,rad_mask = get_trans_masks(nrad,edge='straight',pad=[3,3])
#Use pad = [1,1] if you do not want padding

#Display the array trans
#Visually check for inaccurate localization of the sharp-edge
plt.imshow(trans,cmap='gray')
plt.colorbar()
plt.show()

#Display and inspect the mask for transmission function
plt.imshow(trans_mask,cmap='gray')
plt.show()

#Display and inspect the mask for radiograph
plt.imshow(rad_mask,cmap='gray')
plt.show()

#Show a line plot comparing the measured radiograph and transmission function
sz = nrad.shape
coords = np.arange(-(sz[1]//2),sz[1]//2,1)*pix_wid
mid = (trans.shape[0]//2,trans.shape[1]//2)

plt.plot(coords,nrad[sz[0]//2,:])
#Due to padding, trans is three times the size of nrad in each dimension
#For proper alignment in the presence of padding, both nrad and trans are center aligned
#Center alignment is used since an equal amount of padding is applied at both ends of␣
↪each axis
plt.plot(coords,trans[mid[0],mid[1]-(sz[1]//2):mid[1]+(sz[1]//2)])
plt.xlabel('micrometers')
plt.legend(['Measured','Transmission'])
plt.show()
```

5.2 Estimate Blur Model

- To estimate the blur model, we must acquire radiographs of a sharp edge such as a Tungsten rollbar. Each sharp edge radiograph can either contain a single straight edge or two mutually perpendicular intersecting edges. If imaging a single straight edge, then radiographs must be acquired at two different perpendicular orientations of the straight edge. Also, radiographs must be acquired at two different values of SOD/ODD, where SOD is the source to object distance and ODD is the object to detector distance.
- Next, the radiographs must be appropriately normalized. For each radiograph, acquire a bright field image (measurements with X-rays but no sample) and a dark field image (measurements without X-rays). Then, compute the normalized radiograph by dividing the difference between the radiograph and the dark field image with the difference between the bright field and the dark field image.

- Using the normalized radiographs, estimate parameters of X-ray source blur and detector blur using the function *pysaber.estimate_blur()*.
- An example python script that demonstrates blur estimation using radiographs of a single straight edge at various orientations and SOD/ODD values is shown below. To obtain the data that is required to run this script, download and unzip the zip file at the link `data`. To run the script as is within the current working directory, the files in the zip file must be placed within a folder called `data`.

```
import numpy as np #For mathematics on vectors
from PIL import Image #To read images in TIFF format
from pysaber import estimate_blur #To estimate blur PSF parameters

pix_wid = 0.675 #Width of each pixel in micrometers

#Horizontal and vertical edge radiographs
edge_files = ['data/horz_edge_25mm.tif','data/horz_edge_50mm.tif',
                'data/vert_edge_25mm.tif','data/vert_edge_50mm.tif']
#Filenames of bright field images for normalization
bright_files = ['data/horz_bright.tif','data/horz_bright.tif',
                    'data/vert_bright.tif','data/vert_bright.tif']
#Filenames of dark field images for normalization
dark_files = ['data/horz_dark.tif','data/horz_dark.tif',
                'data/vert_dark.tif','data/vert_dark.tif']
#Source to object (SOD) distances in micrometers for each radiograph in edge_files
sod = [24751.89,50251.79,24753.05,50253.35]
#Source to detector (SDD) distances in micrometers for each radiograph in edge_files
sdd = [71003.08,71003.08,71010.86,71010.86]

rads = [] #List that will contain normalized radiographs
odd = [] #Object to detector distance (ODD) for each radiograph in rads
for i in range(len(edge_files)): #Loop through all the radiograph files
    rad = Image.open(edge_files[i]) #Read radiograph
    rad = np.asarray(rad) #Convert to numpy array
    bright = Image.open(bright_files[i]) #Read bright field
    bright = np.asarray(bright) #Convert to numpy array
    dark = Image.open(dark_files[i]) #Read dark field
    dark = np.asarray(dark) #Convert to numpy array
    nrad = (rad-dark)/(bright-dark) #Normalize radiograph
    rads.append(nrad) #Add normalized radiograph to the list rads
    odd.append(sdd[i]-sod[i]) #Add corresponding ODD to the list odd

#Estimate X-ray source blur, detector blur, and every radiograph's transmission function
#To reduce run time and quickly produce a result, the value for argument thresh can be␣
↪reduced.
#However, reducing thresh may produce an inaccurate blur model that does not fit the␣
↪measured data
src_params,det_params,trans_params = estimate_blur(rads,sod,odd,pix_wid,
                edge='straight',thresh=1e-6,pad=[3,3],power=1.0,save_dir='./')
#src_params is a python dictionary of parameters that quantify X-ray source blur
#det_params is a python dictionary of parameters that quantify blur from the detector␣
↪panel
#Both src_params and det_params characterize the effective blur and are used during␣
↪deblurring
#trans_params is a list of lists, each of which contains the low/high values of␣
```

(continues on next page)

(continued from previous page)

```
↪transmission function
#trans_params is useful to check accuracy of fit and not useful for deblurring.

#help(estimate_blur)
#Uncomment above line to get help on using the function estimate_blur

print("------------------------------------------------------------")
print("Source blur model parameters are {}".format(src_params))
#Print parameters of source blur
print("Detector blur model parameters are {}".format(det_params))
#Print parameters of detector blur
print("Transmission function parameters are {}".format(trans_params))
#Print parameters of transmission functions
```

5.3 Validate Blur Model

- We must ensure that the estimated parameters are indeed a good fit for the measured data. This is done by comparing line profiles across the sharp edge between the measured radiograph and the predicted radiograph from the blur model. The output of the blur model given parameters of source blur, detector blur, and transmission function is computed using the function *pysaber.get_trans_fit()*. The fit must be evaluated for every sharp-edge radiograph that is input to *pysaber.estimate_blur()*.
- Verify the agreement between the measured radiograph and the blur model prediction. Carefully zoom into the region containing the sharp edge and verify if the predicted blur matches with the blur in the measured radiograph. Also, verify the agreement between the measured radiograph values and blur model prediction in regions further away from the sharp edge. The predicted radiograph that is output by the blur model contains additional padding. Hence, it is necessary to account for this padding when comparing with the measured radiograph.
- If the fit is not tight, consider reducing the value of the input argument `thresh` of the function *pysaber.estimate_blur()* to obtain a better fit. Reducing the convergence threshold, `thresh`, can improve the agreement between the measured radiograph and the blur model prediction, but will inevitably result in longer run times. A good fit indicates that the blur model is able to accurately model the X-ray source and detector blurs.
- Example python scripts for line profile comparisons between the blur model prediction and measured radiograph are shown below. To obtain the data that is required to run this script, download and unzip the zip file at the link `data`. To run the script as is within the current working directory, the files in the zip file must be placed within a folder called `data`.

Listing 3: Line profile comparisons across a horizontal edge radiograph.

```
import numpy as np
from PIL import Image #To read images in TIFF format
from pysaber import get_trans_fit #To get blurred radiograph as predicted by the blur␣
↪model
import matplotlib.pyplot as plt #To display images

pix_wid = 0.675 #Width of each pixel in micrometers

#Read a horizontal edge radiograph for which accuracy of fit must be analyzed
rad = Image.open('data/horz_edge_25mm.tif') #Read radiograph
rad = np.asarray(rad) #Convert to numpy array
bright = Image.open('data/horz_bright.tif') #Read bright field
```

(continues on next page)

(continued from previous page)

```
bright = np.asarray(bright) #Convert to numpy array
dark = Image.open('data/horz_dark.tif') #Read dark field
dark = np.asarray(dark) #Convert to numpy array
nrad = (rad-dark)/(bright-dark) #Normalize radiograph
sod = 24751.89 #Source to object distance (SOD) of radiograph
sdd = 71003.08 #Source to detector distance (SDD) of radiograph

#Parameters of X-ray source blur
src_params = {'source_FWHM_x_axis':2.69,
                'source_FWHM_y_axis':3.01,
                'norm_power':1.0,
                'cutoff_FWHM_multiplier':10}
#Parameters of detector blur
det_params = {'detector_FWHM_1':1.85,
                'detector_FWHM_2':126.5,
                'detector_weight_1':0.916,
                'norm_power':1.0,
                'cutoff_FWHM_1_multiplier':10,
                'cutoff_FWHM_2_multiplier':10}
#Transmission function parameters
trans_params = [0.015,0.98]

#Get the blurred radiograph as predicted by the blur model
pred_nrad,_ = get_trans_fit(nrad,sod,sdd-sod,pix_wid,src_params,det_params,trans_params,
↪edge='straight',pad=[3,3])

#Show a line plot comparing the measured radiograph and the predicted blurred radiograph
sz = nrad.shape
coords = np.arange(-(sz[0]//2),sz[0]//2,1)*pix_wid
mid = (pred_nrad.shape[0]//2,pred_nrad.shape[1]//2)

plt.plot(coords,nrad[:,sz[1]//2])
#Due to padding, pred_nrad is three times the size of nrad in each dimension
#For proper alignment in the presence of padding, both nrad and pred_nrad are center␣
↪aligned
#Center alignment is used since an equal amount of padding is applied at both ends of␣
↪each axis
plt.plot(coords,pred_nrad[mid[0]-(sz[0]//2):mid[0]+(sz[0]//2),mid[1]])
plt.xlabel('micrometers')
plt.legend(['Measured','Prediction'])
plt.show()
```

Listing 4: Line profile comparisons across a vertical edge radiograph.

```
import numpy as np
from PIL import Image #To read images in TIFF format
from pysaber import get_trans_fit #To get blurred radiograph as predicted by the blur␣
↪model
import matplotlib.pyplot as plt #To display images

pix_wid = 0.675 #Width of each pixel in micrometers
```

(continues on next page)

(continued from previous page)

```
#Read a vertical edge radiograph for which accuracy of fit must be analyzed
rad = Image.open('data/vert_edge_25mm.tif') #Read radiograph
rad = np.asarray(rad) #Convert to numpy array
bright = Image.open('data/vert_bright.tif') #Read bright field
bright = np.asarray(bright) #Convert to numpy array
dark = Image.open('data/vert_dark.tif') #Read dark field
dark = np.asarray(dark) #Convert to numpy array
nrad = (rad-dark)/(bright-dark) #Normalize radiograph
sod = 24753.05 #Source to object distance (SOD) of radiograph
sdd = 71010.86 #Source to detector distance (SDD) of radiograph

#Parameters of X-ray source blur
src_params = {'source_FWHM_x_axis':2.69,
                'source_FWHM_y_axis':3.01,
                'norm_power':1.0,
                'cutoff_FWHM_multiplier':10}
#Parameters of detector blur
det_params = {'detector_FWHM_1':1.85,
                'detector_FWHM_2':126.5,
                'detector_weight_1':0.916,
                'norm_power':1.0,
                'cutoff_FWHM_1_multiplier':10,
                'cutoff_FWHM_2_multiplier':10}
#Transmission function parameters
trans_params = [0.015,0.98]

#Get the blurred radiograph as predicted by the blur model
pred_nrad,_ = get_trans_fit(nrad,sod,sdd-sod,pix_wid,src_params,det_params,trans_params,
↪edge='straight',pad=[3,3])

#Show a line plot comparing the measured radiograph and the predicted blurred radiograph
sz = nrad.shape
coords = np.arange(-(sz[1]//2),sz[1]//2,1)*pix_wid
mid = (pred_nrad.shape[0]//2,pred_nrad.shape[1]//2)

plt.plot(coords,nrad[sz[0]//2,:])
#Due to padding, pred_nrad is three times the size of nrad in each dimension
#For proper alignment in the presence of padding, both nrad and pred_nrad are center␣
↪aligned
#Center alignment is used since an equal amount of padding is applied at both ends of␣
↪each axis
plt.plot(coords,pred_nrad[mid[0],mid[1]-(sz[1]//2):mid[1]+(sz[1]//2)])
plt.xlabel('micrometers')
plt.legend(['Measured','Prediction'])
plt.show()
```

5.4 Visualize Blur PSF

- For further analysis and visualization, we can also compute the point spread functions (PSF) of source blur and detector blur. The PSF of source blur is computed using the function *pysaber.get_source_psf()* and PSF of detector blur is computed using *pysaber.get_detector_psf()*.
- The function *pysaber.get_source_psf()* is useful to compute PSF either in the plane of the X-ray source or the plane of the detector. Since source blur PSF on the detector plane is a function of the object's source to object distance (SOD) and object to detector distance (ODD), SOD and ODD must be specified when computing source PSF in the plane of the detector. To compute source blur PSF in the source plane, it is sufficient to use the default values for SOD and ODD in *pysaber.get_source_psf()*.
- The detector blur PSF obtained using *pysaber.get_detector_psf()* models blur due to the scintillator and detector panel. Hence, it is independent of SOD and ODD.
- Example python scripts that demonstrate visualization of source and detector PSFs are shown below.

Listing 5: Plot X-ray source blur PSF

```
import numpy as np #For mathematics on vectors
import matplotlib.pyplot as plt #For plotting and showing images
from pysaber import get_source_psf #To compute PSF of source blur

pix_wid = 0.675 #Width of each pixel in micrometers
#Parameters of X-ray source blur
src_params = {'source_FWHM_x_axis':2.69,
              'source_FWHM_y_axis':3.01,
              'norm_power':1.0,
              'cutoff_FWHM_multiplier':10}

#Get point spread function (PSF) of source blur in the plane of the X-ray source.
#Do not supply SOD and SDD if you need PSF in the source plane.
source_psf = get_source_psf(pix_wid,src_params)

#Display the source blur PSF on the source plane as an image
sz = source_psf.shape
x = np.arange(-(sz[1]//2),(sz[1]//2)+1,1)*pix_wid
y = np.arange(-(sz[0]//2),(sz[0]//2)+1,1)*pix_wid
plt.pcolormesh(x,y,source_psf,cmap='gray')
plt.xlabel('micrometers')
plt.ylabel('micrometers')
plt.title('X-ray source PSF at source plane')
plt.colorbar()
plt.show()

#To get source blur PSF on the detector plane, supply SOD and ODD = SDD - SOD to the␣
↪function get_source_psf
sod = 25000 #in micrometers
sdd = 71000 #in micrometers
source_psf = get_source_psf(pix_wid,src_params,sod,sdd-sod)

#help(get_source_psf)
#Uncomment the above line to get help on using the function get_source_psf
```

(continues on next page)

(continued from previous page)

```
#Display the source blur PSF on the detector plane as an image
sz = source_psf.shape
x = np.arange(-(sz[1]//2),(sz[1]//2)+1,1)*pix_wid
y = np.arange(-(sz[0]//2),(sz[0]//2)+1,1)*pix_wid
plt.pcolormesh(x,y,source_psf,cmap='gray')
plt.xlabel('micrometers')
plt.ylabel('micrometers')
plt.title('X-ray source PSF at detector plane')
plt.colorbar()
plt.show()
```

Listing 6: Plot X-ray detector blur PSF

```
import numpy as np #For mathematics on vectors
import matplotlib.pyplot as plt #For displaying images
from matplotlib.colors import LogNorm #To display image values in logarithm scale
from pysaber import get_detector_psf #To compute PSF of detector blur

pix_wid = 0.675 #Width of each pixel in micrometers
#Parameters of detector blur
det_params = {'detector_FWHM_1':1.85,
              'detector_FWHM_2':126.5,
              'detector_weight_1':0.916,
              'norm_power':1.0,
              'cutoff_FWHM_1_multiplier':10,
              'cutoff_FWHM_2_multiplier':10}

#Get point spread function (PSF) of detector blur as a 2D numpy array.
detector_psf = get_detector_psf(pix_wid,det_params)

#help(get_detector_psf)
#Uncomment the above line to get help in using the function get_detector_psf

#Display the PSF of detector blur
sz = detector_psf.shape
x = np.arange(-(sz[1]//2),(sz[1]//2)+1,1)*pix_wid
y = np.arange(-(sz[0]//2),(sz[0]//2)+1,1)*pix_wid
plt.pcolormesh(x,y,detector_psf,cmap='gray',norm=LogNorm())
plt.xlabel('micrometers')
plt.ylabel('micrometers')
plt.title('Detector PSF')
plt.colorbar()
plt.show()
```

5.5 Deblur Radiographs

- Once the parameters of source and detector PSFs are estimated, radiographs of any arbitrary sample acquired at any source to object distance (SOD) and source to detector distance (SDD) can be deblurred using various techniques.
- To deblur a radiograph using Wiener filter, the function *pysaber.wiener_deblur()* is used. To deblur using regularized least squares deconvolution (RLSD), use the function *pysaber.least_squares_deblur()*.
- Deblurring increases sharpness and resolution. However, it also introduces ringing artifacts and increases noise. To reduce noise and ringing artifacts, the regularization parameter can be increased. Ringing artifacts also increase with increasing inaccuracy of the blur model. Thus, it is essential to obtain a good fit between the measured radiograph and the blur model prediction as explained in the section *Validate Blur Model*.
- The python scripts shown below demonstrate deblurring of radiographs using Wiener filter and RLSD. To obtain the data that is required to run this script, download and unzip the zip file at the link `data`. To run the script as is within the current working directory, the files in the zip file must be placed within a folder called `data`.

Listing 7: Deblurring using Wiener filter

```
import numpy as np
from pysaber import wiener_deblur #To deblur using Wiener filtering
import matplotlib.pyplot as plt #To display images
from PIL import Image #To read images in TIFF format

rad_file = 'data/horz_edge_25mm.tif' #Filename of radiograph
bright_file = 'data/horz_bright.tif' #Bright field
dark_file = 'data/horz_dark.tif' #Dark field

sdd = 71003.08 #Source to detector distance (SDD) in micrometers
sod = 24751.89 #Source to object distance (SOD) in micrometers
pix_wid = 0.675 #Pixel width in micrometers
reg_param = 0.1 #Regularization parameter

rad = np.asarray(Image.open(rad_file)) #Read radiograph and convert to numpy array
bright = np.asarray(Image.open(bright_file)) #Read bright field image and convert to␣
↪numpy array
dark = np.asarray(Image.open(dark_file)) #Read dark field image and convert to numpy␣
↪array
norm_rad = (rad-dark)/(bright-dark) #Normalize radiograph

#X-ray source blur parameters
src_params = {'source_FWHM_x_axis':2.69,
              'source_FWHM_y_axis':3.01,
              'norm_power':1.0,
              'cutoff_FWHM_multiplier':10}

#Detector blur parameters
det_params = {'detector_FWHM_1':1.85,
              'detector_FWHM_2':126.5,
              'detector_weight_1':0.916,
              'norm_power':1.0,
              'cutoff_FWHM_1_multiplier':10,
              'cutoff_FWHM_2_multiplier':10}
```

(continues on next page)

(continued from previous page)

```
#Deblur the radiograph using Wiener filter
wiener_rad = wiener_deblur(norm_rad,sod,sdd-sod,pix_wid,src_params,det_params,reg_param)

#Display deblurred radiograph
sz = wiener_rad.shape
x = np.arange(-(sz[1]//2),(sz[1]//2)+1,1)*pix_wid
y = np.arange(-(sz[0]//2),(sz[0]//2)+1,1)*pix_wid
plt.pcolormesh(x,y,wiener_rad,cmap='gray')
plt.xlabel('micrometers')
plt.ylabel('micrometers')
plt.title('Wiener deblur')
plt.colorbar()
plt.show()
```

Listing 8: Deblurring using Regularized Least Squares Deconvolution

```
import numpy as np
from pysaber import least_squares_deblur #To deblur using regularized least squares␣
↪deconvolution (RLSD)
import matplotlib.pyplot as plt #To display images
from PIL import Image #To read images in TIFF format

rad_file = 'data/horz_edge_25mm.tif' #Filename of radiograph
bright_file = 'data/horz_bright.tif' #Bright field image
dark_file = 'data/horz_dark.tif' #Dark field image

sdd = 71003.08 #Source to detector distance (SDD) in micrometers
sod = 24751.89 #Source to object distance (SOD) in micrometers
pix_wid = 0.675 #Pixel width in micrometers
reg_param = 0.001 #Regularization parameter

rad = np.asarray(Image.open(rad_file)) #Read radiograph and convert to numpy array
bright = np.asarray(Image.open(bright_file)) #Read bright field image and convert to␣
↪numpy array
dark = np.asarray(Image.open(dark_file)) #Read dark field image and convert to numpy␣
↪array
norm_rad = (rad-dark)/(bright-dark) #Normalize radiograph

#X-ray source blur parameters
src_params = {'source_FWHM_x_axis':2.69,
              'source_FWHM_y_axis':3.01,
              'norm_power':1.0,
              'cutoff_FWHM_multiplier':10}

#Detector blur parameters
det_params = {'detector_FWHM_1':1.85,
              'detector_FWHM_2':126.5,
              'detector_weight_1':0.916,
              'norm_power':1.0,
              'cutoff_FWHM_1_multiplier':10,
              'cutoff_FWHM_2_multiplier':10}
```

(continues on next page)

(continued from previous page)

```
#Deblur the radiograph using regularized least squares deconvolution (RLSD)
rlsd_rad = least_squares_deblur(norm_rad,sod,sdd-sod,pix_wid,src_params,det_params,reg_
↪param,thresh=2e-4)

#Display deblurred radiograph of artifact
sz = rlsd_rad.shape
x = np.arange(-(sz[1]//2),(sz[1]//2)+1,1)*pix_wid
y = np.arange(-(sz[0]//2),(sz[0]//2)+1,1)*pix_wid
plt.pcolormesh(x,y,rlsd_rad,cmap='gray')
plt.xlabel('micrometers')
plt.ylabel('micrometers')
plt.title('RLSD deblur')
plt.colorbar()
plt.show()
```

CHAPTER

SIX

PYSABER PACKAGE

6.1 Functions

apply_blur_psfs(rad, sod, odd, pix_wid, ...)	Function to blur the input radiograph with point spread functions (PSF) of X-ray source and detector blurs.
estimate_blur(rads, sod, odd, pix_wid, edge)	Estimate parameters of point spread functions (PSF) that model X-ray source blur and/or detector blur from normalized radiographs of a straight sharp edge or mutually perpendicular intersecting pair of sharp edges.
get_detector_psf(pix_wid, det_pars)	Function to compute point spread function (PSF) of detector blur.
get_effective_psf(pix_wid, src_pars, det_pars)	Function to compute the effective point spread function (PSF), which is the convolution of X-ray source and detector PSFs.
get_source_psf(pix_wid, src_pars[, sod, odd])	Function to compute the point spread function (PSF) of X-ray source blur in the plane of the X-ray source or the detector.
get_trans_fit(rad, sod, odd, pix_wid, ...[, pad])	Function to compute the blur model prediction and ideal transmission function for a radiograph with a single straight edge or two mutually perpendicular edges.
get_trans_masks(rad, edge[, tran_pars, pad, ...])	Function to compute transmission function and masks for a radiograph with a single straight edge or two mutually perpendicular edges.
least_squares_deblur(rad, sod, odd, pix_wid, ...)	Function to reduce blur (deblur) in radiographs using a regularized least squares iterative algorithm.
wiener_deblur(rad, sod, odd, pix_wid, ...)	Function to reduce blur (deblur) in a radiograph using Wiener filtering.

6.1.1 apply_blur_psfs

pysaber.**apply_blur_psfs**(*rad*, *sod*, *odd*, *pix_wid*, *src_pars*, *det_pars*, *padded_widths=[0, 0]*, *pad_type='constant'*, *pad_constant=0*)

Function to blur the input radiograph with point spread functions (PSF) of X-ray source and detector blurs.

This function blurs the input radiograph with X-ray source blur and detector blur with the specified point spread function (PSF) parameters. This function is useful to observe the effect of source and detector blurs on a simulated radiograph.

Parameters

- **rad** (*numpy.ndarray*) – Radiograph of type numpy.ndarray that is normalized using the bright-field (also called flat-field) and dark-field images.
- **sod** (*float*) – Source to object distance (SOD) of radiograph.
- **odd** (*float*) – Object to detector distance (ODD) of radiograph.
- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **src_pars** (*dict*) – Dictionary containing the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.
- **det_pars** (*dict*) – Dictionary containing the estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`.
- **padded_widths** (*list*) – List of two integers that specifies the amount of padding already applied to input radiograph. The first integer specifies the padding applied along the first dimension of radiograph. The second integer specifies the padding applied along the second dimension of radiograph. Assumes `padded_widths[k]/2` amount of padding is applied at both the left and right extremities of dimension `k`, where `k` is `0` or `1`.
- **pad_type** (*str*) – Type of additional padding that must be used if amount of padding specified in `padded_widths` is insufficient. Supported values are `edge` and `constant`.
- **pad_constant** (*float*) – If `pad_type` is *constant*, specify the constant value that must be padded.

Returns

Radiograph that is blurred using X-ray source and detector blurs.

Return type

numpy.ndarray

6.1.2 estimate_blur

pysaber.**estimate_blur**(*rads*, *sod*, *odd*, *pix_wid*, *edge*, *thresh=1e-06*, *pad=[3, 3]*, *masks=None*, *bdary_mask=5.0*, *perp_mask=5.0*, *power=1.0*, *save_dir='./'*, *only_src=False*, *only_det=False*, *mix_det=True*)

Estimate parameters of point spread functions (PSF) that model X-ray source blur and/or detector blur from normalized radiographs of a straight sharp edge or mutually perpendicular intersecting pair of sharp edges.

This function is used to estimate parameters of the PSFs that model X-ray source blur and/or detector blur. It takes as input the normalized radiographs at multiple source to object distances (SOD) and object to detector distances (ODD). If each radiograph has a single straight edge, then the measurement must be repeated for two different, preferably perpendicular, orientations of the edge. If the radiograph consists of two intersecting perpendicular edges, then a single radiograph at each specified SOD/ODD is sufficient. Simultaneous estimation of source and detector blur will require radiographs at a minimum of two different value pairs for SOD/ODD. During PSF parameter estimation, the influence of certain regions within each radiograph can be removed by masking. For more details, please read ahead and also refer to the documents listed in *References*.

Parameters

- **rads** (*list*) – List of radiographs, each of type numpy.ndarray, at various SODs and ODDs. Each radiograph must be normalized using the bright-field (also called flat-field) and dark-field images.
- **sod** (*list*) – List of source to object distances (SOD), each of type *float*, at which each corresponding radiograph in the list `rads` was acquired.
- **odd** (*list*) – List of object to detector distances (ODD), each of type *float*, at which each corresponding radiograph in the list `rads` was acquired.
- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **edge** (*str*) – Used to indicate whether there is a single straight edge or two mutually perpendicular edges in each radiograph. If `edge` is `perpendicular`, then each radiograph is assumed to have two mutually perpendicular edges. If it is `straight`, then each radiograph is assumed to have a single straight edge. Only `perpendicular` and `straight` are legal choices for `edge`.
- **thresh** (*float*) – Convergence threshold for the minimizer during parameter estimation. The iterations stop when the ratio of the reduction in the error function (cost value) and the magnitude of the error function is lower than `thresh`. This is the parameter `ftol` that is specified in the `options` parameter of `scipy.optimize.minimize`. The optimizer used is L-BFGS-B. During joint estimation of source and detector blur, the convergence threshold for the minimizer during the first two initialization steps is ten times this value.
- **pad** (*list*) – List of two integers that determine the amount of padding that must be applied to the radiographs to reduce aliasing during convolution. The number of rows/columns after padding is equal to `pad_factor[0]`/`pad_factor[1]` times the number of rows/columns in each radiograph before padding. For example, if the first element in `pad_factor` is `2`, then the radiograph is padded to twice its size along the first dimension.
- **masks** (*list*) – List of boolean masks, each of type numpy.ndarray and same shape as the radiograph, that is used to exclude pixels from blur estimation. This is in addition to the masking specified by `bdary_mask` and `perp_mask`. An example use case is if some pixels in the radiograph `rads[i]` are bad, then those pixels can be excluded from blur estimation by setting the corresponding entries in `masks[i]` to `False` and `True` otherwise. If None, no user specified mask is used.

- **bdary_mask** (*float*) – Percentage of image region in the radiographs as measured from the outer edge going inwards that must be excluded from blur estimation. Pixels are excluded (or masked) beginning from the outermost periphery of the image and working inwards until the specified percentage of pixels is reached.
- **perp_mask** (*float*) – Percentage of circular region to ignore during blur estimation around the intersecting corner of two perpendicular edges. Ignored if `edge` is `straight`.
- **power** (*float*) – Shape parameter of the density function used to model each PSF. For example, choosing a value of one for `power` creates an exponential (Laplacian) density function. Choosing a value of two for `power` creates a Gaussian density function.
- **save_dir** (*str*) – Directory where estimated parameters are saved in *yaml* file format. Source blur parameters are saved in the file `source_params.yml` within the folder `save_dir`. Similary, detector blur and transmission function parameters are saved as `detector_params.yml` and `transmission_params.yml`.
- **only_src** (*bool*) – If `True`, only estimate source blur parameters.
- **only_det** (*bool*) – If `True`, only estimate detector blur parameters.
- **mix_det** (*bool*) – If `True`, do not use mixture model for detector blur.

Returns

Tuple of objects containing the estimated parameters. If estimating both source and detector blur parameters, returns the three element tuple (`src_pars`, `det_pars`, `tran_pars`). If estimating only source blur parameters, returns the two element tuple (`src_pars`, `tran_pars`). If estimating only detector blur parameters, returns the two element tuple (`det_pars`, `tran_pars`). `src_pars` and `det_pars` are python dictionaries. `tran_pars` is a list of lists.

`src_pars` contains the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.

`det_pars` contains estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`. If `mix_det` is `False`, then value for key `detector_weight_1` is fixed at `1` and value for key `detector_FWHM_2` is fixed at `0`.

`tran_pars` contains estimated parameters of the transmission function for each input radiograph. This return value is a list of lists, where each inner nested list consists of two parameters of type

float. These *float* values give the low and high values respectively of the transmission function. The number of nested lists in the returned list equals the number of input radiographs. Note that the transmission function is the normalized radiograph image that would have resulted in the absence of blur and noise.

Return type
tuple

6.1.3 get_detector_psf

pysaber.**get_detector_psf**(*pix_wid*, *det_pars*)

Function to compute point spread function (PSF) of detector blur.

Parameters

- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **det_pars** (*dict*) – Dictionary containing the estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`.

Returns
PSF of detector

Return type
numpy.ndarray

6.1.4 get_effective_psf

pysaber.**get_effective_psf**(*pix_wid*, *src_pars*, *det_pars*, *sod=1*, *odd=1*)

Function to compute the effective point spread function (PSF), which is the convolution of X-ray source and detector PSFs.

Parameters

- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **src_pars** (*dict*) – Dictionary containing the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of

source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.

- **det_pars** (*dict*) – Dictionary containing the estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`.
- **sod** (*float*) – Source to object distance (SOD).
- **odd** (*float*) – Object to detector distance (ODD).

Returns
PSF of effective blur in the plane of detector.

Return type
numpy.ndarray

6.1.5 get_source_psf

pysaber.**get_source_psf**(*pix_wid*, *src_pars*, *sod=1.0*, *odd=1.0*)

Function to compute the point spread function (PSF) of X-ray source blur in the plane of the X-ray source or the detector.

If source to object distance (SOD) is equal to object to detector distance (ODD), then the PSF on the detector plane is same as that on the plane of the X-ray source. If PSF on detector plane is desired, it is required to specify the SOD and ODD. If PSF on source plane is desired, use the default values for SOD and ODD.

Parameters

- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **src_pars** (*dict*) – Dictionary containing the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the

PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.

- **sod** (*float*) – Source to object distance (SOD).
- **odd** (*float*) – Object to detector distance (ODD).

Returns
PSF of X-ray source blur.

Return type
numpy.ndarray

6.1.6 get_trans_fit

pysaber.**get_trans_fit**(*rad*, *sod*, *odd*, *pix_wid*, *src_pars*, *det_pars*, *tran_pars*, *edge*, *pad=[3, 3]*)

Function to compute the blur model prediction and ideal transmission function for a radiograph with a single straight edge or two mutually perpendicular edges.

For a measured radiograph consisting of a straight sharp edge or two mutually perpendicular edges, get the ideal transmission function and the predicted radiograph from the blur model. Here, the blur model is used to model the impact of blur due to X-ray source and detector.

Parameters

- **rad** (*numpy.ndarray*) – Normalized radiograph of a straight sharp edge or two mutually perpendicular edges.
- **sod** (*float*) – Source to object distance (SOD) for the radiograph `rad`.
- **odd** (*float*) – Object to detector distance (SDD) for the radiograph `rad`.
- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **src_pars** (*dict*) – Dictionary containing the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.
- **det_pars** (*dict*) – Dictionary containing the estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF.

The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`.

- **tran_pars** (*list*) – List containing the estimated parameters of the transmission function for the input radiograph. It consists of two parameters of type *float*. These *float* values give the low and high values respectively of the transmission function. Note that the transmission function is the normalized radiograph image that would have resulted in the absence of blur and noise. If not specified (or specified as None), then the best fitting transmission function parameters are estimated using RANSAC regression.
- **edge** (*str*) – Used to indicate whether there is a single straight edge or two mutually perpendicular edges in each radiograph. If `edge` is `perpendicular`, then each radiograph is assumed to have two mutually perpendicular intersecting edges. If it is `straight`, then each radiograph is assumed to have a single straight edge. Only `perpendicular` and `straight` are legal choices for `edge`.
- **pad** (*list*) – List of two integers that determine the amount of padding that must be applied to the radiographs to reduce aliasing during convolution. The number of rows/columns after padding is equal to `pad_factor[0]/pad_factor[1]` times the number of rows/columns in each radiograph before padding. For example, if the first element in `pad_factor` is `2`, then the radiograph is padded to twice its size along the first dimension.

Returns

Tuple of two arrays of type `numpy.ndarray`. The first array is blurred radiograph as predicted by the blur model. The second array is transmission function, which is the ideal readiograph in the absence of source and detector blur.

Return type

tuple

6.1.7 get_trans_masks

pysaber.**get_trans_masks**(*rad*, *edge*, *tran_pars=None*, *pad=[1, 1]*, *mask=None*, *bdary_mask=5.0*, *perp_mask=5.0*)

Function to compute transmission function and masks for a radiograph with a single straight edge or two mutually perpendicular edges.

For a measured radiograph consisting of a straight sharp edge or two mutually perpendicular edges, get the transmission function, mask for transmission function, and mask for radiograph.

Parameters

- **rad** (*numpy.ndarray*) – Normalized radiograph of a straight sharp edge or two mutually perpendicular edges.
- **edge** (*str*) – Used to indicate whether there is a single straight edge or two mutually perpendicular edges in each radiograph. If `edge` is `perpendicular`, then each radiograph is assumed to have two mutually perpendicular edges. If it is `straight`, then each radiograph is assumed to have a single straight edge. Only `perpendicular` and `straight` are legal choices for `edge`.
- **tran_pars** (*list*) – List containing the estimated parameters of the transmission function for the input radiograph. It consists of two parameters of type *float*. These *float* values give the low and high values respectively of the transmission function. Note that the transmission function is the normalized radiograph image that would have resulted in the absence of blur and noise. If not specified (or specified as None), then the best fitting transmission function

parameters are estimated using RANSAC regression. If specified as `[0,1]`, this function returns the ideal transmission function.

- **pad** (*list*) – List of two integers that determine the amount of padding that must be applied to the radiographs to reduce aliasing during convolution. The number of rows/columns after padding is equal to `pad_factor[0]`/`pad_factor[1]` times the number of rows/columns in each radiograph before padding. For example, if the first element in `pad_factor` is `2`, then the radiograph is padded to twice its size along the first dimension.
- **mask** (*numpy.ndarray*) – Boolean mask of the same shape as the radiograph that is used to exclude pixels from blur estimation. This is in addition to the masking specified by `bdary_mask` and `perp_mask`. An example use case is if some pixels in the radiograph `rad` are bad, then those pixels can be excluded from blur estimation by setting the corresponding entries in `mask` to `False` and `True` otherwise. If None, no user specified mask is used.
- **bdary_mask** (*float*) – Percentage of image region in the radiographs as measured from the outer edge going inwards that must be excluded from blur estimation. Pixels are excluded (or masked) beginning from the outermost periphery of the image and working inwards until the specified percentage of pixels is reached.
- **perp_mask** (*float*) – Percentage of circular region to ignore during blur estimation around the intersecting corner of two perpendicular edges. Ignored if `edge` is `straight`.

Returns

Tuple of three arrays each of type `numpy.ndarray`. The first array is the transmission function, which is the ideal readiograph in the absence of source and detector blur. The second and third arrays are the masks for the transmission function and radiograph respectively. The mask array indicates what pixels must be included (pixel value of `True`) or excluded (pixel value of `False`) during blur estimation.

Return type

tuple

6.1.8 least_squares_deblur

pysaber.**least_squares_deblur**(*rad*, *sod*, *odd*, *pix_wid*, *src_pars*, *det_pars*, *reg_par*, *init_rad=None*, *weights=None*, *thresh=0.0001*)

Function to reduce blur (deblur) in radiographs using a regularized least squares iterative algorithm.

Parameters

- **rad** (*numpy.ndarray*) – Normalized radiograph to deblur
- **sod** (*float*) – Source to object distance (SOD) of the radiograph
- **odd** (*float*) – Object to detector distance (ODD) of the radiograph
- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **src_pars** (*dict*) – Dictionary containing the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the

PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.

- **det_pars** (*dict*) – Dictionary containing the estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`.
- **reg_par** (*float*) – Regularization parameter for the least squares deblurring algorithm. Noise and ringing artifacts in the deblurred radiograph decreases with increasing values for `reg_par` and vice versa. But, note that increasing `reg_par` can also result in excessive blurring due to over-regulization. It is recommended to empirically choose this parameter by increasing or decreasing it by a factor, greater than one (such as `2` or `10`), until the desired image quality is achieved.
- **init_rad** (*numpy.ndarray*) – Initial estimate for the deblurred radiograph. If set to `None`, then the blurred radiograph is used as an initial estimate. If not `None`, then `init_rad` must be a numpy.ndarray of the same shape as the radiograph `rad`.
- **weights** (*numpy.ndarray*) – Array of weights of the same shape as `rad` that is useful to model measurement noise. It can be used to increase or decrease the influence of a particular pixel of `rad` in the forward model cost function. If set to `None`, every pixel is assigned the same weight of `1` in the cost function.
- **thresh** (*float*) – Convergence threshold for the minimizer used to deblur the input radiograph. The iterations stop when the ratio of the reduction in the error function (cost value) and the magnitude of the error function is lower than `thresh`. This is the parameter `ftol` that is specified in the `options` parameter of `scipy.optimize.minimize`. The optimizer used is L-BFGS-B.

Returns

Deblurred radiograph using regularized least squares algorithm.

Return type

numpy.ndarray

6.1.9 wiener_deblur

pysaber.**wiener_deblur**(*rad*, *sod*, *odd*, *pix_wid*, *src_pars*, *det_pars*, *reg_par*)

Function to reduce blur (deblur) in a radiograph using Wiener filtering.

Parameters

- **rad** (*numpy.ndarray*) – Normalized radiograph to deblur
- **sod** (*float*) – Source to object distance (SOD) for the radiograph `rad`.
- **odd** (*float*) – Object to detector distance (SDD) for the radiograph `rad`.
- **pix_wid** (*float*) – Effective width of each detector pixel. Note that this is the effective pixel size given by dividing the physical width of each detector pixel by the zoom factor of the optical lens.
- **src_pars** (*dict*) – Dictionary containing the estimated parameters of X-ray source PSF. It consists of several key-value pairs. The value for key `source_FWHM_x_axis` is the full width half maximum (FWHM) of the source PSF along the x-axis (i.e., second numpy.ndarray dimension). The value for key `source_FWHM_y_axis` is the FWHM of source PSF along the y-axis (i.e., first numpy.ndarray dimension). All FWHMs are for the source PSF in the plane of the X-ray source (and not the plane of the detector). The value for key `cutoff_FWHM_multiplier` decides the non-zero spatial extent of the source PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_x_axis']/2` and `src_pars['cutoff_FWHM_multiplier']` times `src_pars['source_FWHM_y_axis']/2`.
- **det_pars** (*dict*) – Dictionary containing the estimated parameters of detector PSF. It consists of several key-value pairs. The value for key `detector_FWHM_1` is the FWHM of the first density function in the mixture density model for detector blur. The first density function is the most dominant part of detector blur. The value for key `detector_FWHM_2` is the FWHM of the second density function in the mixture density model. This density function has the largest FWHM and models the long running tails of the detector blur's PSF. The value for key `detector_weight_1` is between `0` and `1` and is a measure of the amount of contribution of the first density function to the detector blur. The values for keys `cutoff_FWHM_1_multiplier` and `cutoff_FWHM_2_multiplier` decide the non-zero spatial extent of the detector PSF. The PSF is clipped to zero beginning at a distance, as measured from the PSF's origin, equal to the maximum of `det_pars['cutoff_FWHM_1_multiplier']` times `det_pars['detector_FWHM_1']/2` and `det_pars['cutoff_FWHM_2_multiplier']` times `det_pars['detector_FWHM_2']/2`.
- **reg_par** (*float*) – Regularization parameter for Wiener filter. Noise and ringing artifacts in the deblurred radiograph decreases with increasing values for `reg_par` and vice versa. But, note that increasing `reg_par` can also result in excessive blurring due to over-regulization. It is recommended to empirically choose this parameter by increasing or decreasing it by a factor, greater than one (such as `2` or `10`), until the desired image quality is achieved.

Returns

Deblurred radiograph using a Wiener filter.

Return type

numpy.ndarray

CHAPTER

SEVEN

FEEDBACK

This software is under development and may have bugs. If you run into any problems, please raise a issue on github at the link issues. There is a lot of scope to improve the performance and functionality of this python package. Furthermore, since this package solves a non-convex optimization problem, there is a remote possibility that the final solution may be a local optima that does not properly fit the data. If there is sufficient interest, we will invest time to significantly reduce the run time, improve convergence and usability, and add additional features and functionalities.

CHAPTER

EIGHT

SPONSOR

8.1 Acknowledgements

LLNL-CODE-766837. LLNL-SM-809826. This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

8.2 Disclaimer

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX