Supporting Information

**Removing Stripes, Scratches, and Curtaining with Non-Recoverable Compressed Sensing**

Jonathan Schwartz1, Yi Jiang2, Yongjie Wang3, Anthony Aiello3, Pallab Bhattacharya3,

Hui Yuan4, Zetian Mi3, Nabil Bassim5, Robert Hovden1

*1 Department of Materials Science and Engineering, University of Michigan, Ann Arbor, MI, USA*

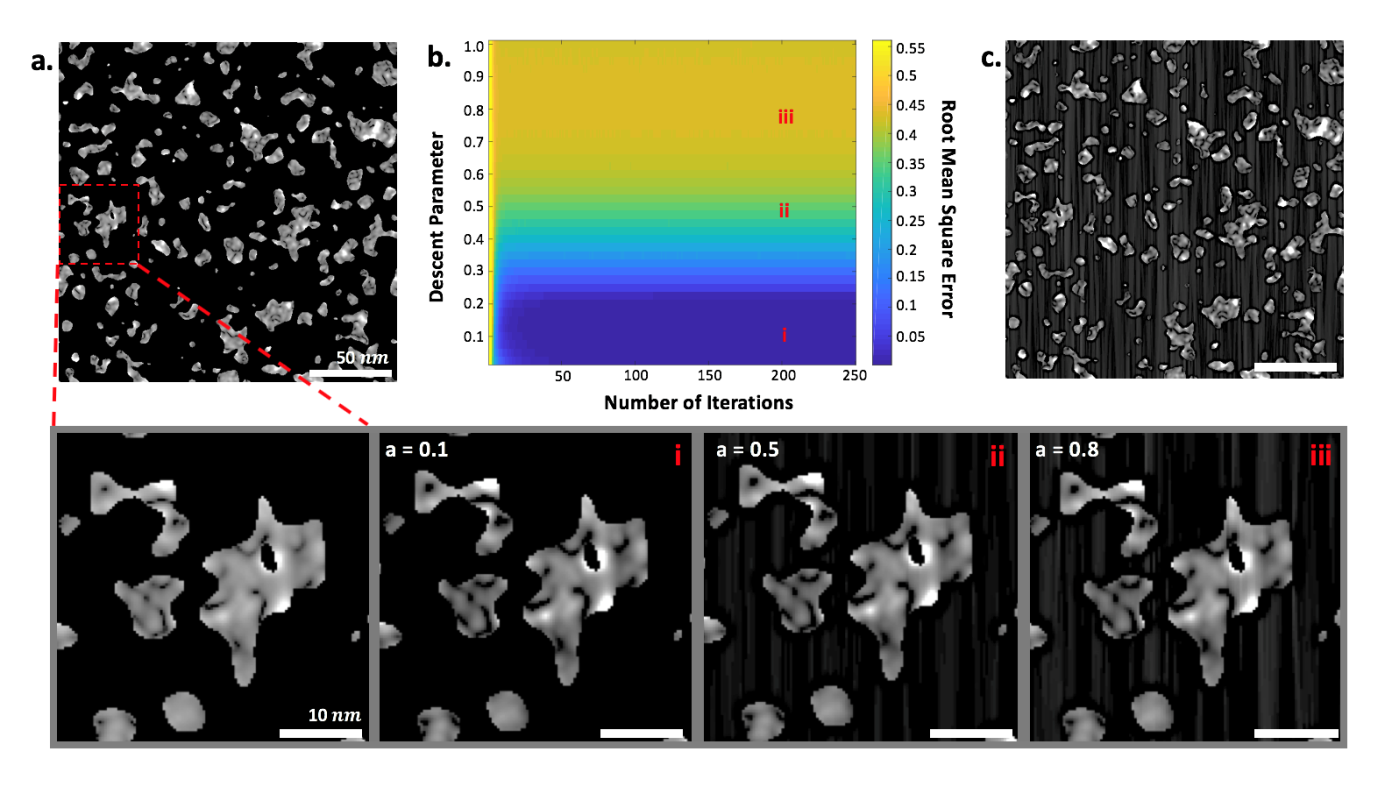
*2 X-ray Science Division, Advanced Photon Source, Argonne National Laboratory, Argonne, IL, USA*

*3 Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, USA*

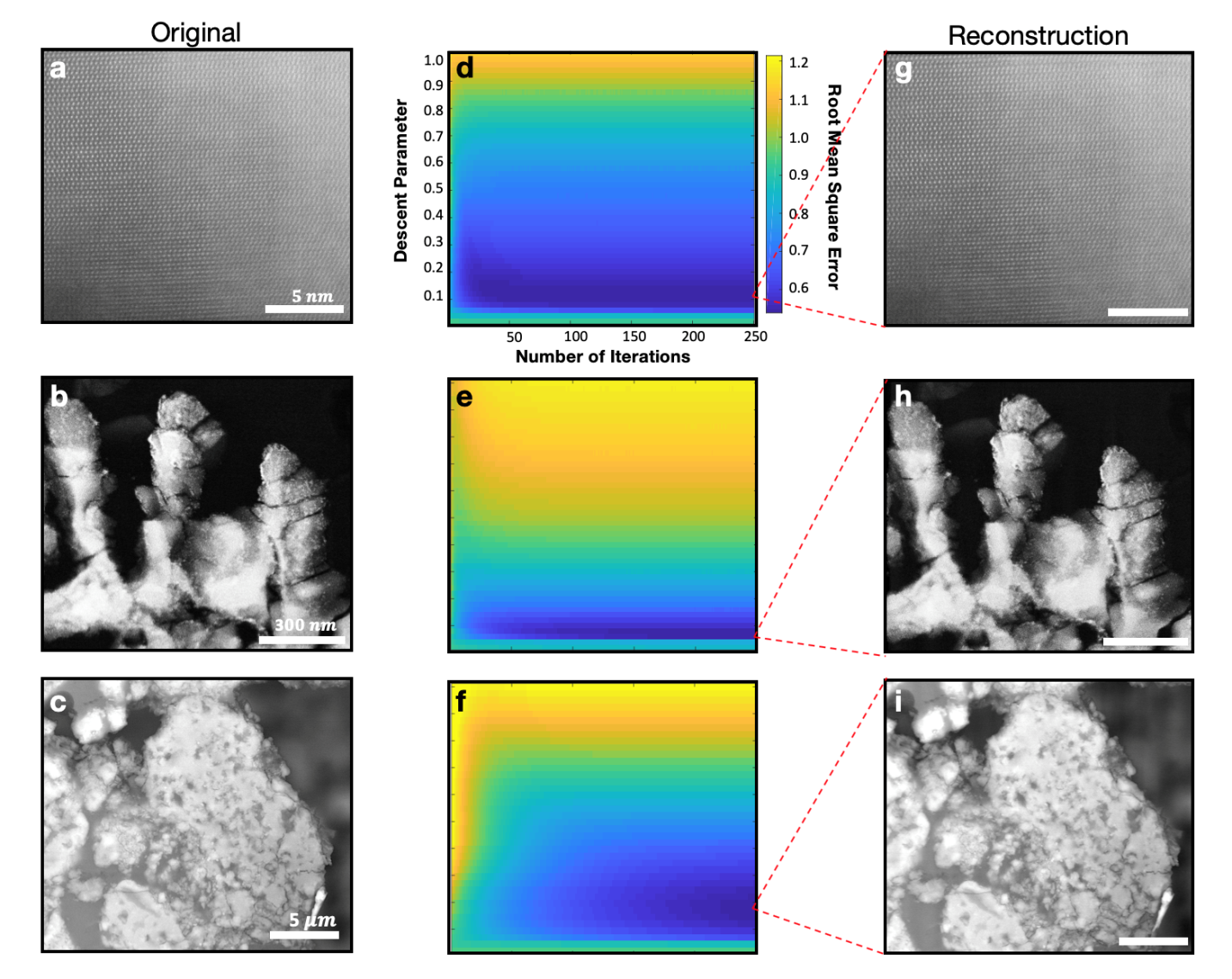
*4 Canadian Centre for Electron Microscopy, McMaster University, Hamilton, ON, Canada*

*5 Department of Materials Science and Engineering, McMaster University, Hamilton, ON, Canada*

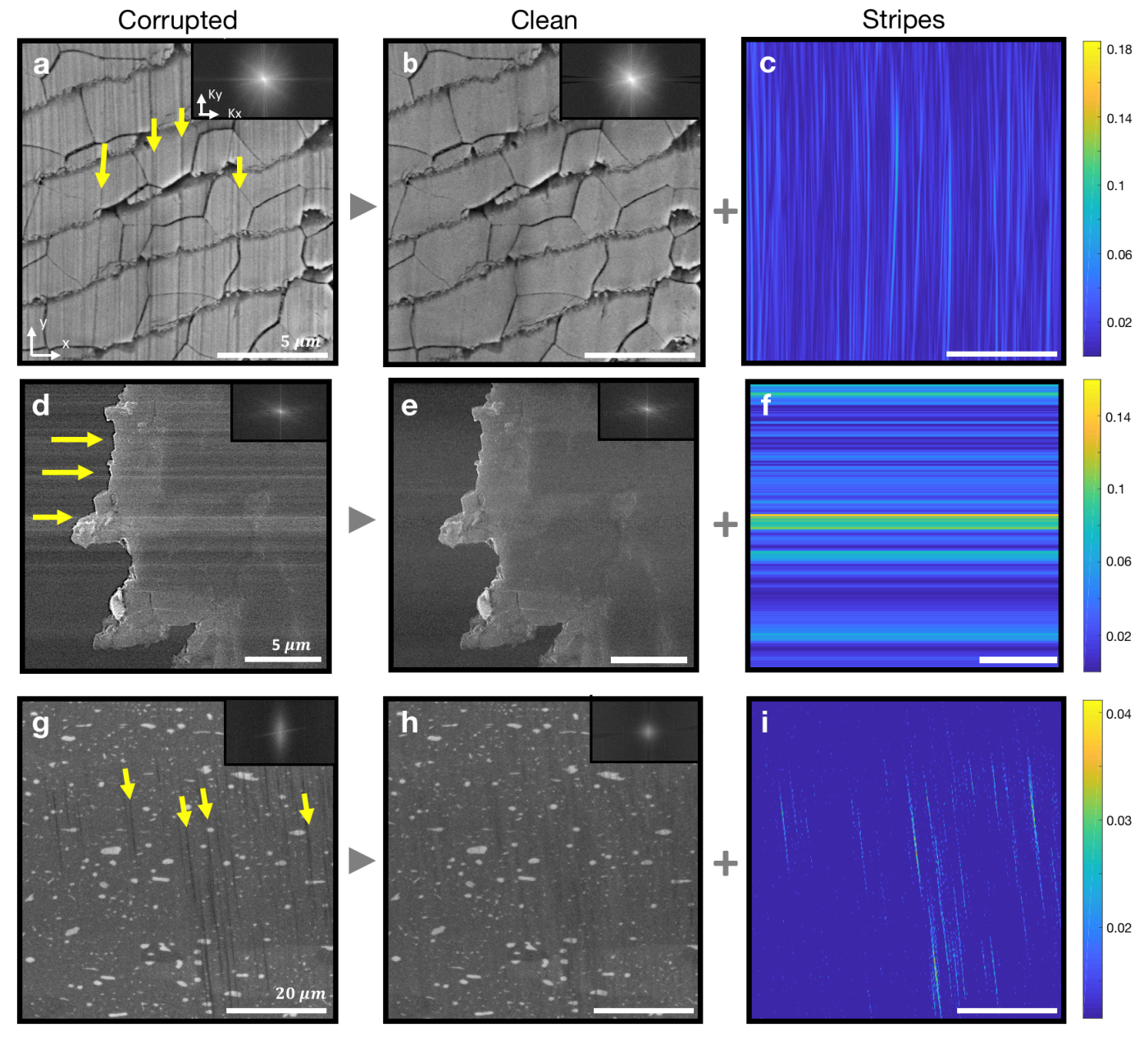
Corresponding author: jtschw@umich.edu – (818) 625-0933



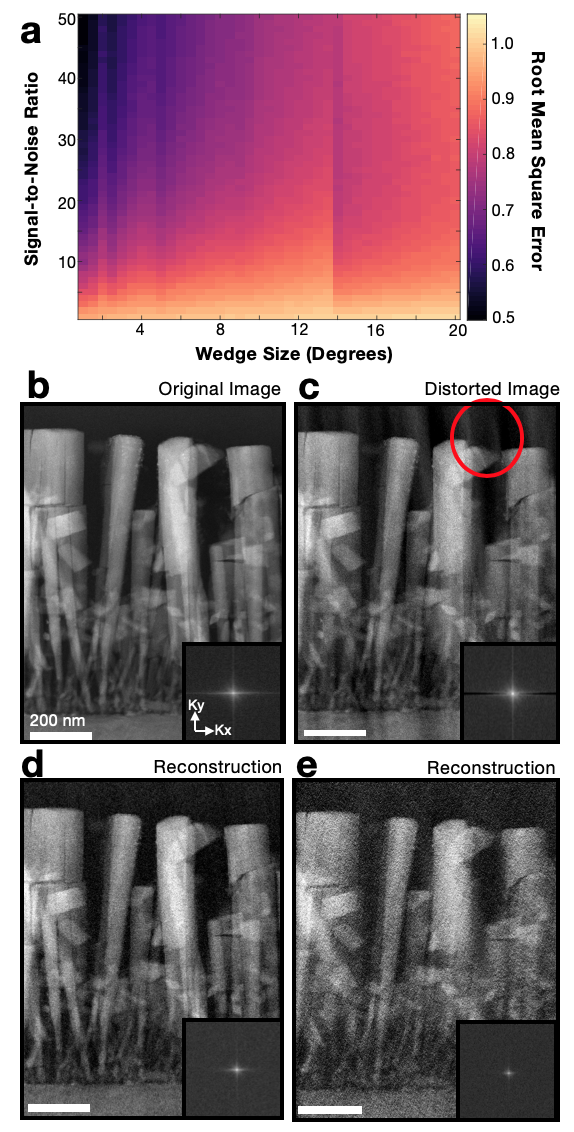
**Fig. S1: Convergence Rate and Parameter Determination with Simulated Nanoparticles** (**a)**.The reconstruction’s root mean square errors (RMSE) (**b)** were normalized by an object distorted with a 5º missing wedge (**c**) to provide directly interpretable quantitative results. Pixels in (**b)** with values below 1 indicate satisfactory the reconstructions. The outputs are visualized in (**i-iii**) to demonstrate that descent parameters of a < 0.25 is optimal. Furthermore, the method converges around 50 iterations.

****

**Fig S2: Convergence rate and parameters determination with experimental micrographs**. **a-b**, Dark field (DF)-STEM test images at 300 keV. **c,** BSE-SEM test image at 20 keV. **d-f,** The normalized RMSE for the test images reconstructed with a 5º horizontal missing wedge starting at a minimum frequency of 15 pixels (field of view)-1 in Fourier space. (**a-c).** Unlike the nanoparticles, experimental images converge around 100 iterations. An increase in the number of iterations may be due to the difference in image complexity and lack of sparsity compared to the simulated image. **g-i**, Reconstructions are visualized after 250 iterations with the descent parameter set to *a 0.1*.

****

**Fig S3: Additional Reconstructions with Experimental Micrographs.** Removing vertical scratches from mechanical polishing (**a-c**) and beam instability (**d-f**) from Pearl SE-SEM micrographs at 20 keV. **g-i,** Curtaining artifacts from BSE FIB-tomography data is removed.

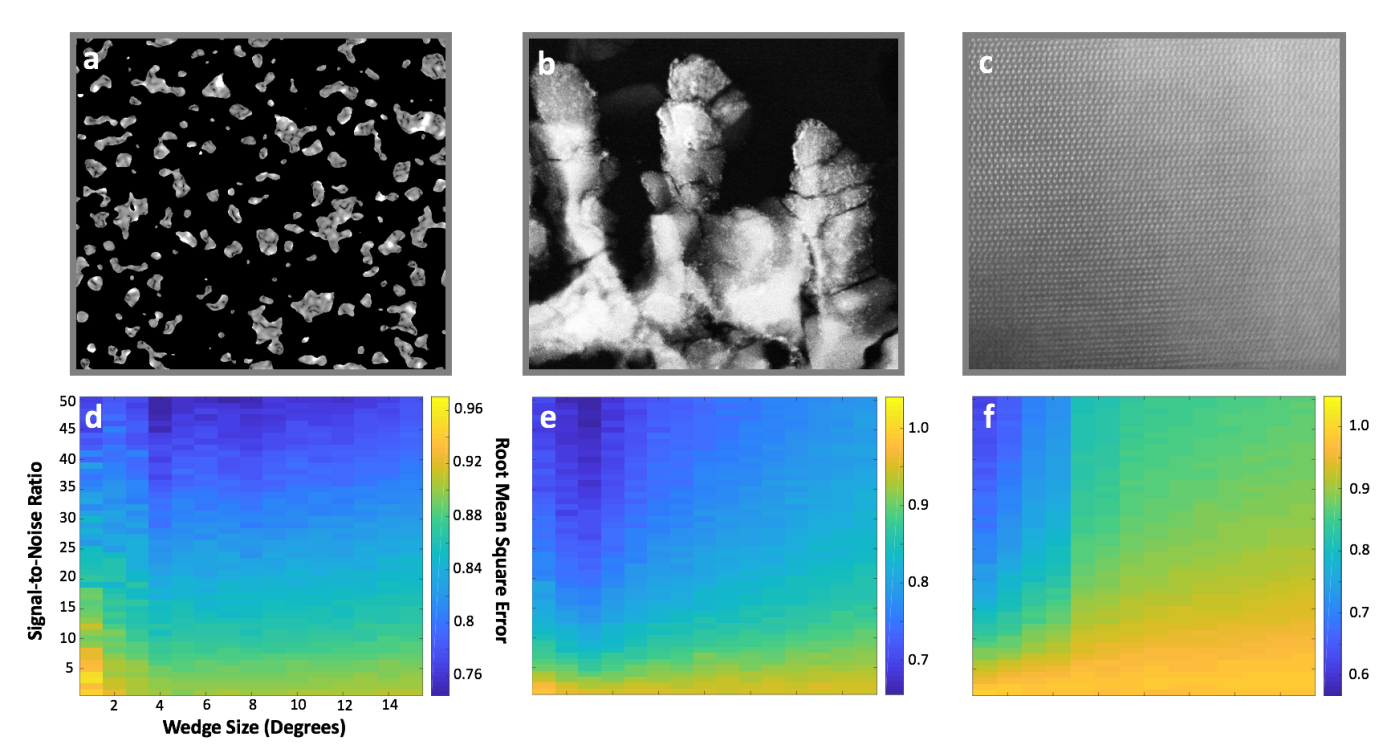


**Fig S4:** **Quantitative Study of SNR and Wedge Size.** **a,** A plot of the RMSE normalized by the error from various missing wedge sizes. Values below 1 indicate the reconstruction outperforms loss of information. Wedges below 8º, consistently achieves satisfactory performance at all SNR values above 10. **b,** DF-S/TEM micrograph (at 300 keV) of MBE grown InGaN nanowires with platinum nanoparticles coated on the surface. **c,** A distorted image with a horizontal 5º ‘missing’ wedge. **d-e,** The outputs for horizontal missing wedges at SNR values of 50 and 15 and missing wedge sizes of 5º and 15º, respectively. FFT insets shown lower right.

A screenshot of a cell phone

Description automatically generated

**Fig S5:** **Quantitative Study of SNR and Wedge Size**. **a,** A BSE-SEM micrograph (at 20 keV) of percolated MgB2 —a superconductor. **b,** A plot of the RMSE normalized by the error from various missing wedge sizes. Values below 1 indicate the reconstruction outperforms loss of information. Wedges below 8º, consistently achieves satisfactory performance at all SNR values above 10. **c,** A distorted image with a horizontal 3º ‘missing’ wedge. **i-iii,** The outputs for a 3º horizontal missing wedge at SNR values of 50, 30, and 10 respectively.

****

**Fig S6:** **Quantitative Study of SNR and Wedge Size with Synthetic and Experimental Data**. **a,** Simulated nanoparticles generated on *tomviz.*  **b-c**, Dark field (DF)-STEM test images at 300 keV. **d-f,** Plots of RMSE normalized by RMSE by wedge sizes between 1-15º. Test images **a** and **b** are sparse (composed of pixels with a value of zero or low intensity). Therefore satisfactory reconstructions occur across larger missing wedge sizes, as shown in **d** and **e**. In comparison, image **c** is not sparse and consequently the reconstructions perform best only for smaller missing wedges, as shown in **f**. There is a transition in **f** at 5º which coincides with additional brag peaks that fall within the missing wedge.

**Python Code**

import numpy as np

from numpy.fft import fftn, ifftn, fftshift, ifftshift

def TV\_reconstruction(dataset, Niter = 100, a = 0.1, wedgeSize = 5, theta =0, kmin = 15):

#dataset – real-space image (pixels)

#Niter – Number of iterations for reconstruction

# a – Decent parameter (unitless, typically 0 – 0.3)

# wedgeSize – angular range of the missing wedge (degrees)

# theta – orientation of the missing wedge (degrees)

# kmin – Minimum frequency to start the missing wedge (1/px)

#Import dimensions from dataset

(nx, ny) = dataset.shape

# Convert angle from Degrees to Radians.

theta = (theta+90)\*(np.pi/180)

dtheta = wedgeSize\*(np.pi/180)

#Create coordinate grid in polar

x = np.arange(-nx/2, nx/2-1,dtype=np.float64)

y = np.arange(-ny/2, ny/2-1,dtype=np.float64)

[x,y] = np.meshgrid(x,y,indexing ='ij')

rr = (np.square(x) + np.square(y))

phi = np.arctan2(x,y)

#Create the Mask

mask = np.ones( (nx, ny), dtype = np.int8 )

mask[np.where((phi >= (theta-dtheta/2)) & (phi <= (theta+dtheta/2)))] = 0

mask[np.where((phi >= (np.pi+theta-dtheta/2)) & (phi <= (np.pi+theta+dtheta/2)))] = 0

mask[np.where((phi >= (-np.pi+theta-dtheta/2)) & (phi <= (-np.pi+theta+dtheta/2)))] = 0

mask[np.where(rr < np.square(kmin))] = 1 # Keep values below rmin.

mask = np.array(mask, dtype = bool)

#FFT of the Original Image

FFT\_image = fftshift(fftn(dataset))

# Reconstruction starts as random image.

recon\_init = np.random.rand(nx,ny)

#Artifact Removal Loop

for i in range(Niter):

#FFT of Reconstructed Image.

FFT\_recon = fftshift(fftn(recon\_init))

#Data Constraint

FFT\_recon[mask] = FFT\_image[mask]

#Inverse FFT

recon\_constraint = np.real(ifftn(ifftshift(FFT\_recon)))

#Positivity Constraint

recon\_constraint[ recon\_constraint < 0 ] = 0

if i < Niter -1: #ignore on last iteration

#TV Minimization

# The basis for TVDerivative (20 iterations and epsilon = 1e-8) was determined by Sidky (2006).

recon\_minTV = recon\_constraint

d = np.linalg.norm(recon\_minTV - recon\_init)

for j in range(20):

Vst = TVDerivative(recon\_minTV, nx, ny)

recon\_minTV = recon\_minTV - a\*d\*Vst

#Initialize the Next Loop.

reconinit = recon\_minTV

#Return the reconstruction.

return recon\_constraint

def TVDerivative(img):

fxy = np.pad(img, (1,1), 'constant', constant\_values = np.mean(img))

fxnegy = np.roll(fxy, -1, axis = 0)

fxposy = np.roll(fxy, 1, axis = 0)

fnegxy = np.roll(fxy, -1, axis = 1)

fposxy = np.roll(fxy, 1, axis = 1)

fposxnegy = np.roll( np.roll(fxy, 1, axis = 1), -1, axis = 0 )

fnegxposy = np.roll( np.roll(fxy, -1, axis = 1), 1, axis = 0)

vst1 = (2\*(fxy - fnegxy) +

2\*(fxy - fxnegy))/np.sqrt(1e-8 + (fxy - fnegxy)\*\*2 + (fxy - fxnegy)\*\*2)

vst2 = (2\*(fposxy - fxy))/np.sqrt(1e-8 + (fposxy - fxy)\*\*2 + (fposxy - fposxnegy)\*\*2)

vst3 = (2\*(fxposy - fxy))/np.sqrt(1e-8 + (fxposy - fxy)\*\*2 + (fxposy - fnegxposy)\*\*2)

vst = vst1 - vst2 - vst3

vst = vst[1:-1, 1:-1]

vst = vst/np.linalg.norm(vst)

return vst