**1. Read binary SAR image (real valued channels) using python struct**

**import** struct

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

fname = **r**'C:\Users\sdey2\Downloads\PolSAR-tools-master\PolSAR-tools-master\sample\_data\full\_pol\T3\T12\_real.bin'

f = **open**(fname, "rb")

byte\_gap = 4

init\_byte = -4 # We keep it -4 because we want to start reading from 0

data = []

tot\_byte = 81204 # Check the total size of file in bytes

**while** init\_byte <= (tot\_kb-8): # substract 8 because it reads 4 bytes at once

f.**seek**(init\_byte+byte\_gap)

byte = f.**read**(4)

data\_pix = struct.unpack("f", byte)

data.append(data\_pix)

init\_byte = init\_byte+byte\_gap *# We want to move cursor 4 byte and then read 4 byte again*

all\_data = np.array(data)

all\_data = all\_data.reshape([201, 101*])# Check the size of image from hdr or from the config file*

plt.imshow(all\_data, **cmap** = 'gray')

**2. Now, we can read the same using gdal**

**from** osgeo **import** gdal

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

file = **r**'C:\Users\sdey2\Downloads\PolSAR-tools-master\PolSAR-tools-master\sample\_data\full\_pol\T3\T12\_real.bin'

ds = gdal.**Open**(file) # Opens the file as dataset

band = ds.**GetRasterBand**(1) # Get band number 1

arr = band.**ReadAsArray**() # Read the band as an array

plt.**imshow**(arr, **cmap** = 'gray')

**3. Write bin file using gdal**

**from** osgeo **import** gdal

**import** numpy **as** np

# First read a reference data whose geometric features we will copy in our newly obtained data

refData = **r**'C:\Users\sdey2\Downloads\PolSAR-tools-master\PolSAR-tools-master\sample\_data\full\_pol\T3\T12\_real.bin'

# Generate a new data. In this case a 2D data is generated using numpy

wdata = np.**random**.**random**((201,101))

# Set the filename for the new data to be saved in .bin format

file = **r**'C:\Users\sdey2\Downloads\PolSAR-tools-master\PolSAR-tools-master\sample\_data\full\_pol\T3\New\_data.bin'

ds = gdal.**Open**(refData)

[cols, rows] = wdata.**shape**

driver = gdal.**GetDriverByName**("ENVI")

outdata = driver.**Create**(file, rows, cols, 1, gdal.GDT\_Float32)

#sets same geotransform as reference data

outdata.**SetGeoTransform**(ds.GetGeoTransform())

#sets same projection as reference data

outdata.**SetProjection**(ds.**GetProjection**())

outdata.**SetDescription**(file)

outdata.**GetRasterBand**(1).**WriteArray**(wdata)

outdata.**FlushCache**()

**4. Enhance image visualization**

Copy the complete code from 1

**Min-Max stretching**

plt.**imshow**(all\_data, **vmin** = **float**(all\_data.**min**()), **vmax** = **float**(all\_data.**max**()), **cmap** = 'gray')

**Meanstandard deviation x n stretching**

For example, n = 2,

mean = np.**mean**(all\_data)

std = np.**std**(all\_data)

vmin = mean - std \* 2

vmax = mean + std \* 2

plt.**imshow**(all\_data, **vmin** = vmin, **vmax** = vmax, **cmap** = 'gray')

**Cumulative count cut (lower percent, upper percent)**

low\_frac = 0.02

upper\_frac = 0.98

flat\_data = all\_data.**ravel**()

flat\_data = flat\_data[np.**isfinite**(flat\_data)]

sorted\_vals = np.**sort**(flat\_data)

n = **len**(sorted\_vals)

vmin = sorted\_vals[**int**(low\_frac \* n)]

vmax = sorted\_vals[**int**(upper\_frac \* n)]

plt.**imshow**(all\_data, **vmin** = vmin, **vmax** = vmax, **cmap** = 'gray')