```
# TEST ON GPU
import os,sys

import numpy as np
import torch
sys.path.append(os.path.abspath(os.getcwd()))
from make_surface.lbfgs2_routine import *
import matplotlib.pyplot as plt
from skimage.transform import rescale, resize
import pandas as pd
from
make_surface.kymatio.phaseharmonics2d.phase_harmonics_k_bump_non_isotropic \
    import PhaseHarmonics2d
from kymatio.phaseharmonics2d.phase_harmonics_k_bump_fftshift2d \
    import PhaseHarmonics2d as wphshift2d

def generate_surface(folder_index, im = None):
    size = 512
    Krec = 1

    profildir = './make_surface
/original_profilometry_{}.npy'.format(folder_index)
    FOLOUT = './make_surface/results/sample_number_' + str(0) +
'original_folder_' + str(folder_index) + '/'
    if im is None:
        im =  np.array(np.load(profildir.format(folder_index), allow_pickle =
True), dtype = 'float')

    new_im = resize(im, (size, size), anti_aliasing = False)

    dict_image = {'max':[np.max(new_im)], 'min':[np.min(new_im)]}
    new_im = (new_im -np.min(new_im))/(np.max(new_im) - np.min(new_im))

    original_im = new_im
    minmaxdf = pd.DataFrame.from_dict(dict_image)
    minmaxdf.to_csv(FOLOUT+ 'minmax_values{}.csv'.format(folder_index))
    ymean = np.repeat(np.mean(new_im, axis = 1)[:, None],size, axis = 1)

    plt.imshow(new_im - ymean) #, vmin = 0, vmax = 1
    np.savetxt(FOLOUT+'ymean{}'.format(8), ymean)
    plt.colorbar()
    plt.title('Y mean subtracted')
    plt.savefig(FOLOUT+'ymean_subtracted{}.png'.format(folder_index))

    plt.clf()
    np.savetxt(FOLOUT+'ymean{}'.format(folder_index),ymean)

    new_im = new_im - ymean
    xmean = np.repeat(np.mean(new_im, axis = 0)[None, :],size, axis = 0)
    new_im = new_im  - xmean
    np.savetxt(FOLOUT+'xmean{}'.format(folder_index), xmean)
    print('done saving to ' + FOLOUT)
    # breakpoint()
```

```python
    im = torch.tensor(new_im, dtype=torch.float).unsqueeze(0).unsqueeze(0).cuda()
    # Parameters for transforms
    J = 4
    L = 4
    M, N = im.shape[-2], im.shape[-1]
    delta_j = 1
    delta_l = 4
    delta_n = 2
    delta_k = 0
    maxk_shift = 1
    nb_chunks = 4
    nb_restarts = 1
    factr = 10
    maxite = 500
    maxcor = 20
    init = 'normalstdbarx'
    stdn = 1

    information  = 'meanremoved_bump_lbfgs2_gpu_N' + str(N) + 'J' + str(J) + 'L' + str(L) + 'dj' +\
        str(delta_j) + 'dl' + str(delta_l) + 'dk' + str(delta_k) + 'dn' + str(delta_n) +\
        '_maxkshift' + str(maxk_shift) +\
        '_factr' + str(int(factr)) + 'maxite' + str(maxite) +\
        'maxcor' + str(maxcor) + '_init' + init +\
        'ns' + str(nb_restarts)
    os.makedirs(FOLOUT, exist_ok=True)
    text_file = open(FOLOUT + "/information.txt", "w")
    n = text_file.write(information)
    n = text_file.write('\n')
    n = text_file.write('model C')
    text_file.close()
    labelname = 'modelC'

    # kymatio scattering


    Sims = []
    wph_ops = []
    factr_ops = []
    nCov = 0
    total_nbcov = 0
    for chunk_id in range(J+1):
        wph_op = wphshift2d(M,N,J,L,delta_n,maxk_shift,J+1,chunk_id,submean=1,stdnorm=stdn)
        if chunk_id ==0:
            total_nbcov += wph_op.nbcov

        wph_op = wph_op.cuda()
        wph_ops.append(wph_op)
        Sim_ = wph_op(im)
        nCov += Sim_.shape[2]
        print('wph coefficients',Sim_.shape[2])
        Sims.append(Sim_)
        factr_ops.append(factr)

    for chunk_id in range(nb_chunks):
        wph_op = PhaseHarmonics2d(M, N, J, L, delta_j, delta_l, delta_k,
                                nb_chunks, chunk_id, submean=1, stdnorm=stdn)
        if chunk_id ==0:
            total_nbcov += wph_op.nbcov
        wph_op = wph_op.cuda()
        wph_ops.append(wph_op)
        Sim_ = wph_op(im) # output size: (nb,nc,nb_channels,1,1,2)
        nCov += Sim_.shape[2]
        print('wph coefficients',Sim_.shape[2])
        Sims.append(Sim_)
        factr_ops.append(factr)

    print('total nbcov is',total_nbcov)

    generated = call_lbfgs2_routine(FOLOUT,labelname,im,wph_ops,Sims,N,Krec,nb_restarts,maxite,factr,factr_ops,init=init)
    return generated, xmean, ymean, dict_image
if __name__ == "__main__":
    if 'generate_surface.py' in os.listdir():
        os.chdir('../')
    generate_surface(0)

```