```
import numpy as np
import visual_map
import os, re, sys
import time, math, string
from matplotlib import pyplot as plt

def main(off_zero_frame_file, on_zero_frame_file, plot_save_path):

    off_zero_frame_load = np.loadtxt(off_zero_frame_file)
    on_zero_frame_load = np.loadtxt(on_zero_frame_file)

    off_zero_frame_load = negative_frame_load / np.mean(off_zero_frame_load)
    on_zero_frame_load = zero_frame_load / np.mean(on_zero_frame_load)

    j_max = 11664

    off_zero_length = np.size(off_zero_frame_load)
    on_zero_length = np.size(on_zero_frame_load)

    if (off_zero_length != j_max * 2):
        print ('off zero frame file is incorrect length, should contain %d elements' ) % j_max
        return 0
    if (on_zero_length != j_max * 2):
        print ('on zero frame file is incorrect length, should contain %d elements' ) % j_max
        return 0

    crystal_strength_log = on_zero_frame_file

    x_list, y_list, z_list = [], [], []
    # [addr] = i, [i] = addr
    #collect_addr_dict, collect_ordr_dict = visual_map.collect_dicts()
    normal_addr_dict, normal_ordr_dict = visual_map.normal_dicts()
    fid_list, corners_list = visual_map.index11664_fiducials()

    for j in range(0, j_max):
        #addr = collect_ordr_dict[j]
        addr = normal_ordr_dict[j]
        x, y = visual_map.get_xy(addr)

        if (crystal_strength_log[j][0] >= 10):
            z = 0
        else :
            z = relative_intensity_log[j][0]

        x_list.append(float(x))
        y_list.append(float(y))
        z_list.append(float(z))

    X = np.array(x_list)
    Y = np.array(y_list)
    Z = np.array(z_list)
    xr = X.ravel()
    yr = Y.ravel()
    zr = Z.ravel()

    print ('before plot' )

    fig = plt.figure(num=None, figsize=(9,9), facecolor='0.6' , edgecolor='k' )
    fig.subplots_adjust(left=0.03,bottom=0.03,right=0.97,top=0.97,wspace=0,hspace=0)
    ax1 = fig.add_subplot(111, aspect='equal' , axisbg='0.7' )
    ax1.scatter(xr, yr, c=zr, s=14, alpha=1, marker='s' , linewidth=0.1)#,cmap='PuOr')
    ax1.set_xticks([2.2*x for x in range(11)])
    ax1.set_yticks([2.5*x for x in range(11)])
    ax1.set_xlim(xr.min()-0.2, xr.max()+0.2)
    ax1.set_ylim(yr.min()-0.2, yr.max()+0.2)
    ax1.invert_yaxis()

    plt.show

    plt.savefig(plot_save_path, dpi=600, bbox_inches='tight' , pad_inches=0.05)


    return 1
```