

2、

import cv2

import numpy as np

import matplotlib.pyplot as plt

def imgfilter2D(img, filter, ratio):

    img\_x, img\_y = img.shape

    filter\_x, filter\_y = filter.shape

    result\_mtx = np.zeros(((img\_x - filter\_x + 1), (img\_y - filter\_y + 1)))

    for i in range(result\_mtx.shape[0]):

        for j in range(result\_mtx.shape[1]):

            result\_mtx[i][j] = np.abs(np.sum(img[i:i + filter\_x, j:j + filter\_y] \* filter) / ratio)

    return result\_mtx.astype('uint8').reshape(result\_mtx.shape[0], result\_mtx.shape[1], 1)

def gaussian\_filter(filter\_size, sigma):

    filter = np.zeros((filter\_size, filter\_size))

    img\_x, img\_y = filter.shape

    center = (img\_x//2, img\_y//2)

    for x\_loc in range(img\_x):

        for y\_loc in range(img\_y):

            dist\_x\_center = x\_loc-center[0]

            dist\_y\_center = y\_loc-center[1]

            r\_sqr = ((dist\_x\_center)\*\*2 + (dist\_y\_center)\*\*2)

            filter[x\_loc][y\_loc] = np.exp(-r\_sqr/(2\*(sigma\*\*2)))

    return filter

img=cv2.imread("479632\_checkerboard1024-shaded.tif", cv2.IMREAD\_GRAYSCALE)

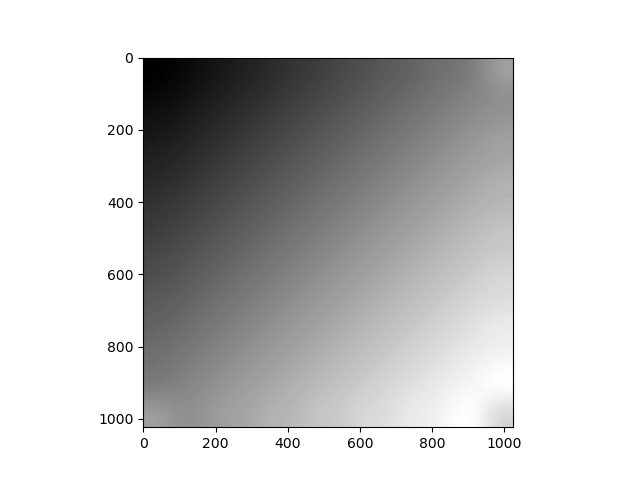
padding\_img = np.pad(img, (128, 128), 'reflect')

g\_filter = gaussian\_filter(257, 64)

gaussian\_blur\_img = imgfilter2D(padding\_img, g\_filter, np.sum(g\_filter))

plt.imshow(gaussian\_blur\_img, "gray")

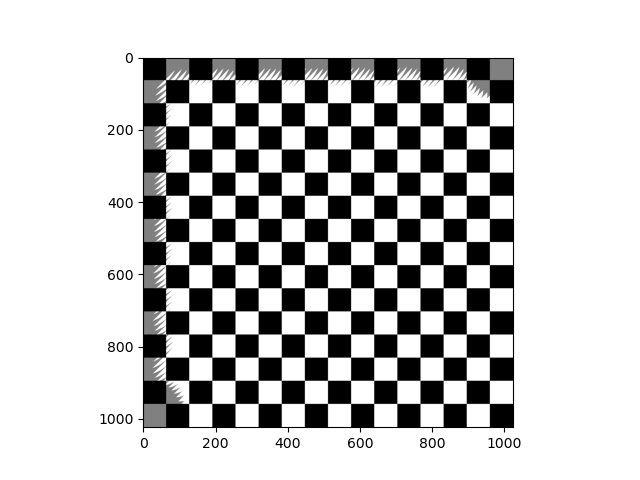
plt.show()



process\_img = (img/gaussian\_blur\_img)

plt.imshow(process\_img, "gray")

plt.show()



In another way we use

FFT base low pass filter will rect kernel radius 8

def low\_pass\_filter(dia, img\_shape):

    filter = np.zeros(img\_shape[:2])

    img\_x, img\_y = filter.shape

    center = (img\_x//2, img\_y//2)

    for x\_loc in range(img\_x):

        for y\_loc in range(img\_y):

            if dia > ((x\_loc-center[0])\*\*2 + (y\_loc-center[1])\*\*2)\*\*0.5:

                filter[x\_loc][y\_loc] = 1

    return filter

def rever\_fft\_with\_filter(fftshift\_img, filter\_size):

        passfilter\_fftshift\_img = fftshift\_img \* low\_pass\_gaussian\_filter(filter\_size, sigma, (fftshift\_img.shape))

    passfilter\_ifftshift\_img = np.fft.ifftshift(passfilter\_fftshift\_img)

    passfilter\_img = np.fft.ifft2(passfilter\_ifftshift\_img)

    return passfilter\_img

fft\_img = np.fft.fft2(img)

fftshift\_img = np.fft.fftshift(fft\_img)

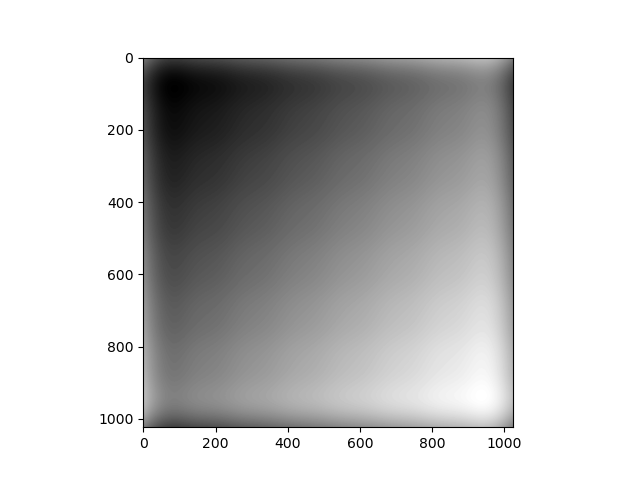
low\_pass\_8 = np.abs(rever\_fft\_with\_filter(fftshift\_img, 8))

low\_pass\_img = np.clip(np.abs(rever\_fft\_with\_filter(fftshift\_img, 8, 4)), 0, 255)

process\_img = (img-low\_pass\_img).astype("uint8")

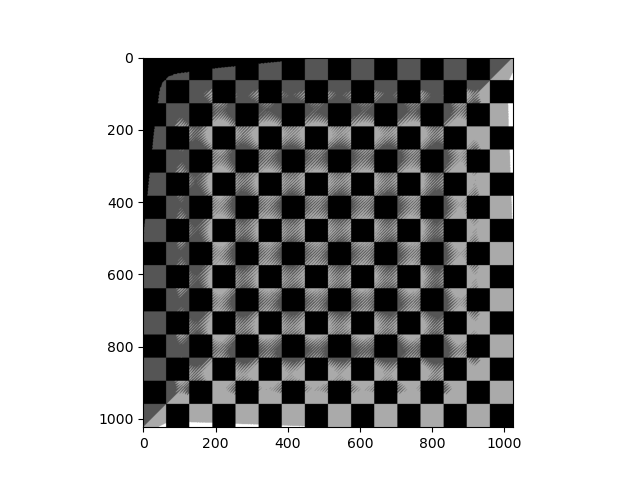
plt.imshow(low\_pass\_img, "gray")

plt.show()



plt.imshow(process\_img, "gray")

plt.show()



2-2、

Due to our use reflecting method padding so the right bottom block padding will add another 3 black block around his neighbor.

3、

import cv2

import numpy as np

import matplotlib.pyplot as plt

def bgr2gray(img):

    return ( img[:, :, 0]\*0.114 + img[:, :, 1]\*0.587 + img[:, :, 2]\*0.299).astype('uint8').reshape(img.shape[0],img.shape[1],1)

def hist\_equal(img\_array):

    flat = img\_array.flatten()

    n = len(flat)

    img\_bincount = np.bincount(flat)

    T = 190 \* np.cumsum(img\_bincount)/n

    T = T.astype('uint8')

    return T[img\_array.astype('uint8')]

def custom\_equal(img\_array):

    flat = img\_array.flatten()

    n = len(flat)

    img\_bincount = np.bincount(flat)

    new\_mapping\_table = np.arange(len(img\_bincount))

    T = ((new\_mapping\_table-73) \* 0.5) + np.clip(((new\_mapping\_table-85) \* 6.5), 0, 2222) - np.clip(((new\_mapping\_table-103) \* 5.5), 0, 2222)

    print(T)

    T = T.astype('uint8')

    return T[img\_array.astype('uint8')]

img=cv2.imread("479632\_einstein-low-contrast.tif", cv2.IMREAD\_GRAYSCALE)

after\_hist = hist\_equal(img)

# gray\_img = bgr2gray(img)

array\_img = np.asarray(img)

result = custom\_equal(array\_img)

flat = after\_hist.flatten()

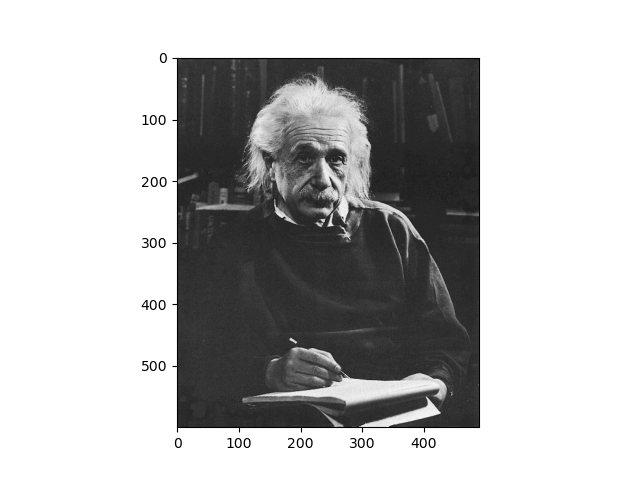
plt.hist(flat, bins=50)

plt.show()

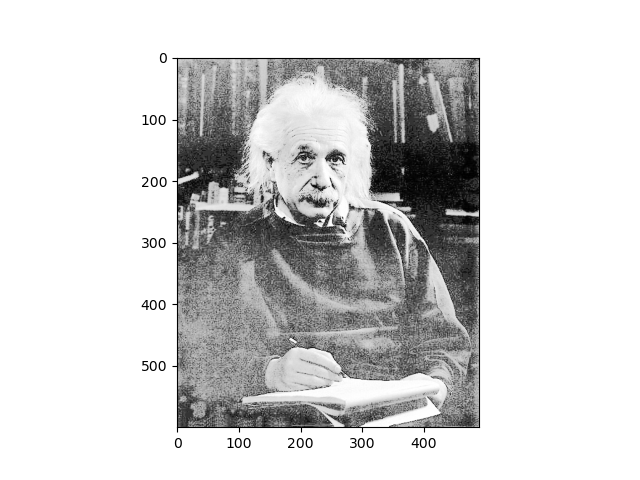
plt.imshow(after\_hist, cmap = 'gray')

plt.show()

we read as gray image will show a not bad result

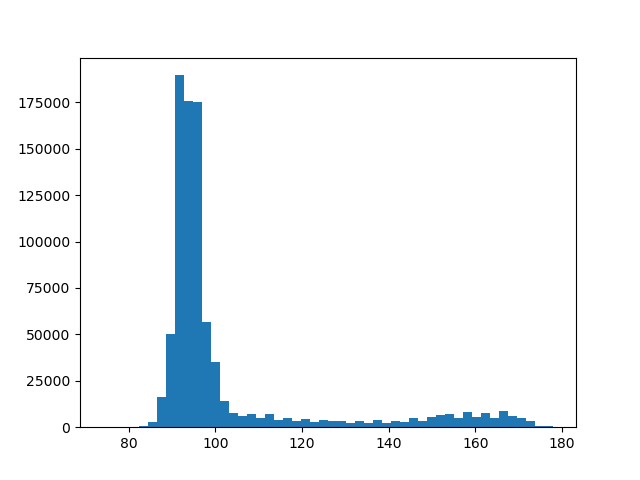


use normal histogram equalizer we will get

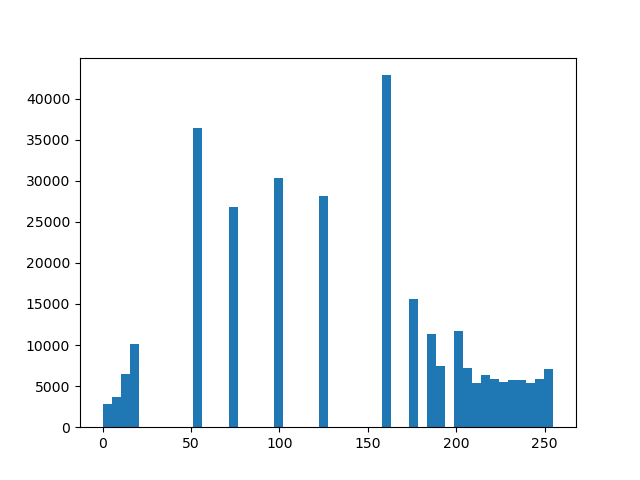


And we use our customization way to enhance the data distribution

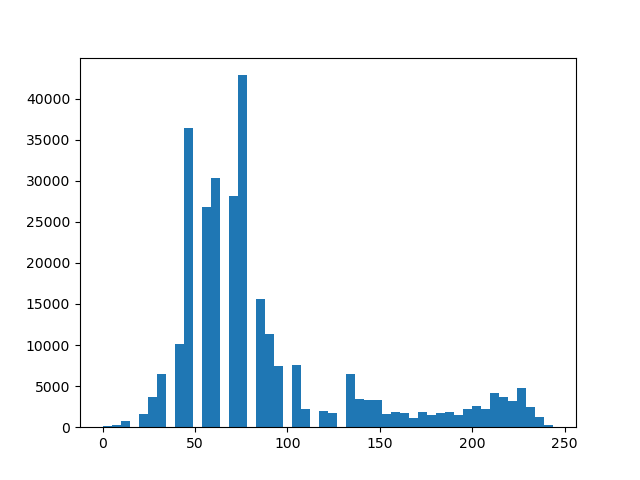
Raw data



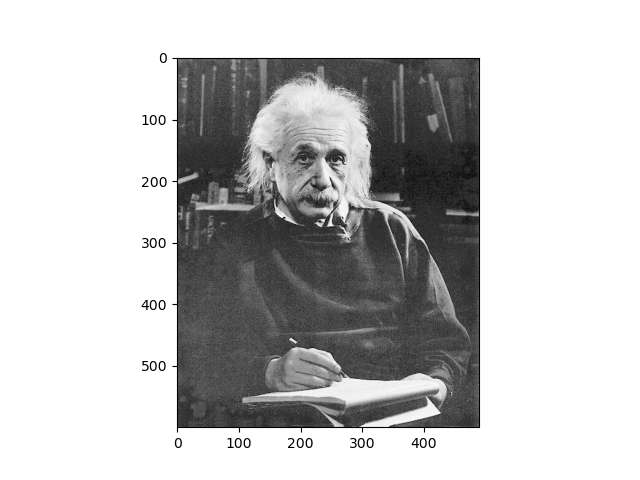
After histogram equalization



In above result we have to enhance 80-110 intensity resolution in raw picture we use our customization way



Get better result in distribution



Also in our processed image