[Image Pre-process]

0. normalize() vs. equalizeHist()

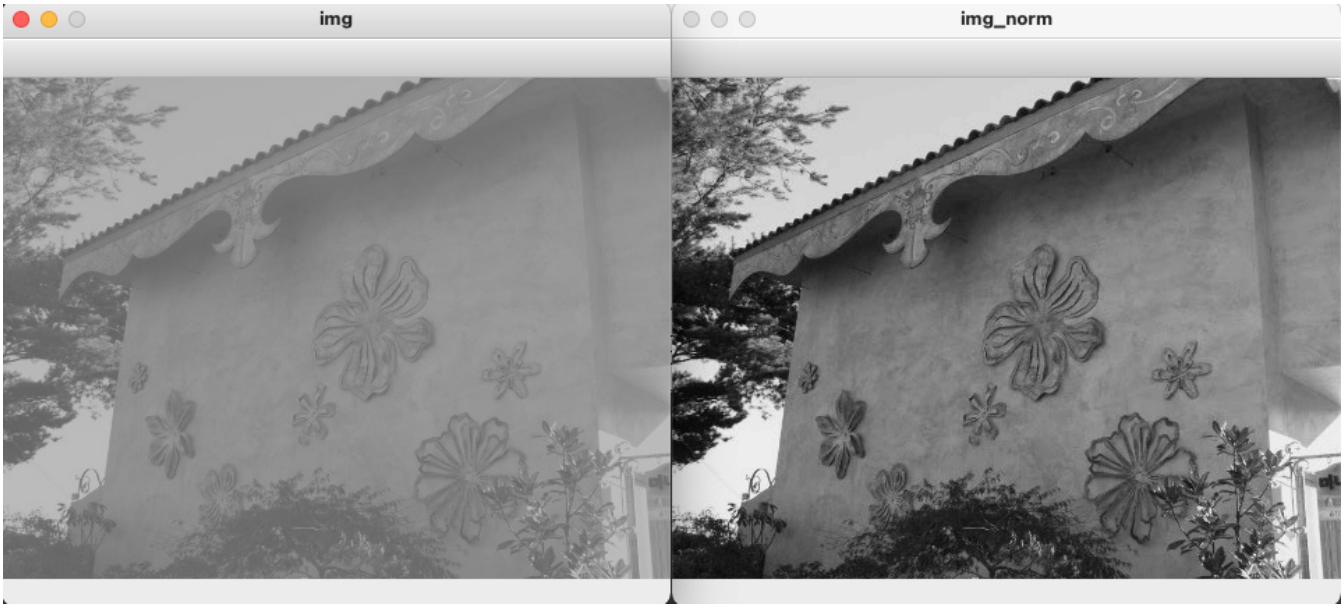
normalize()

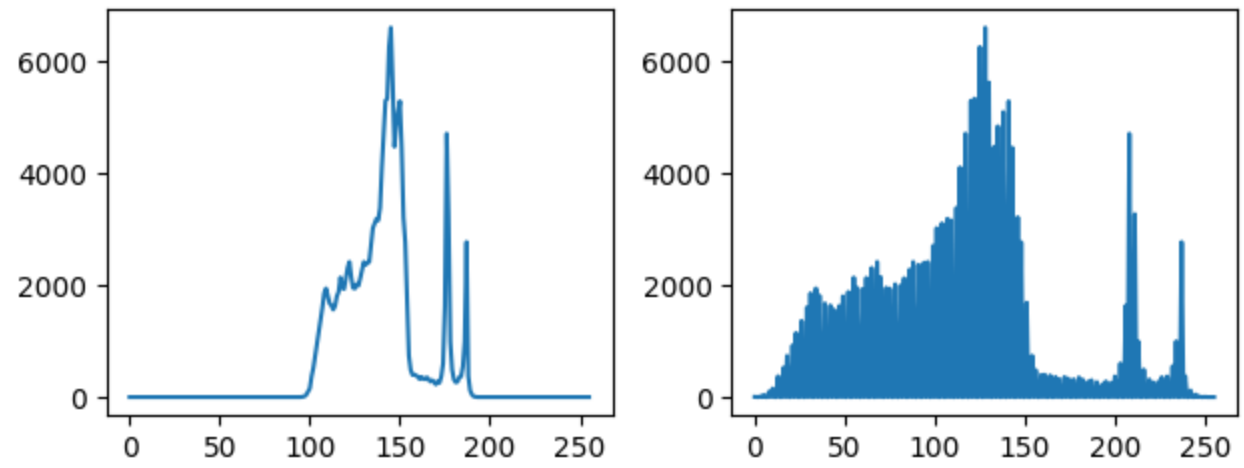
$$I_n = (I - Min)\frac{Max_{new} - Min_{new}}{Max - Min} + Min_{new}$$

I_n : Pixel after normalize(), I : Pixel before normalize()

Min : Minimun pixel before normalize(), Max : Maximum pixel before normalize()

Min_{new} : Minimun pixel after normalize(), Max_{new} : Maximum pixel after normalize()





equalizeHist()

$$h(v) = round\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

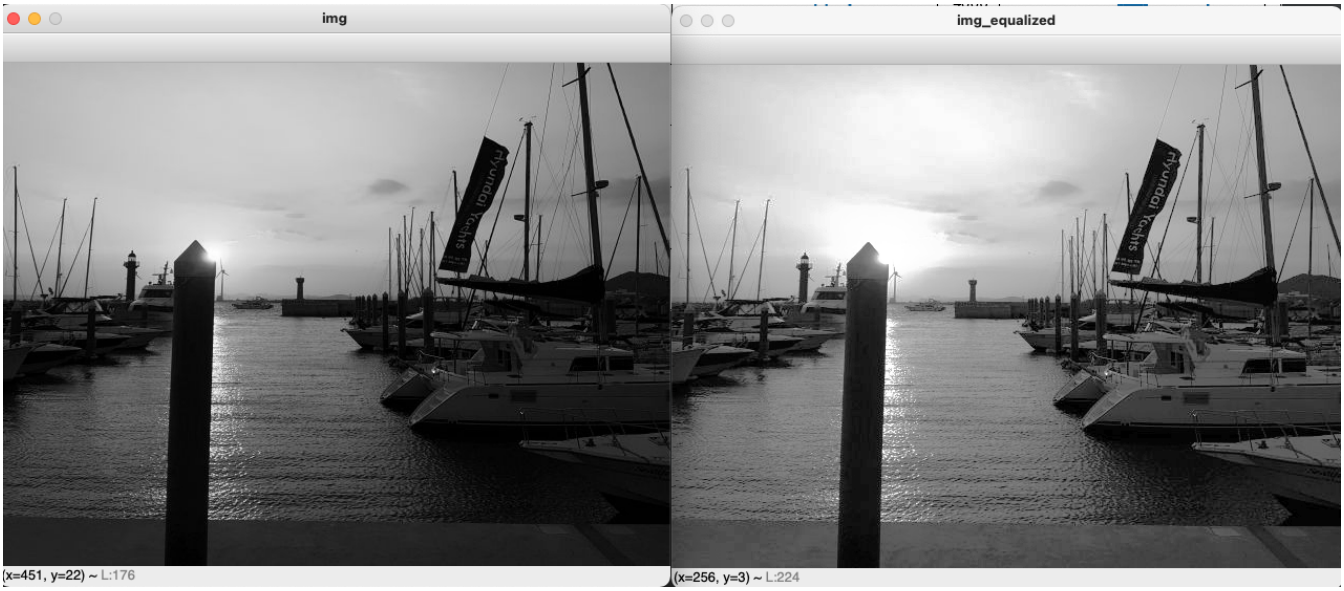
$h(v)$: Image pixel after equalizeHist()

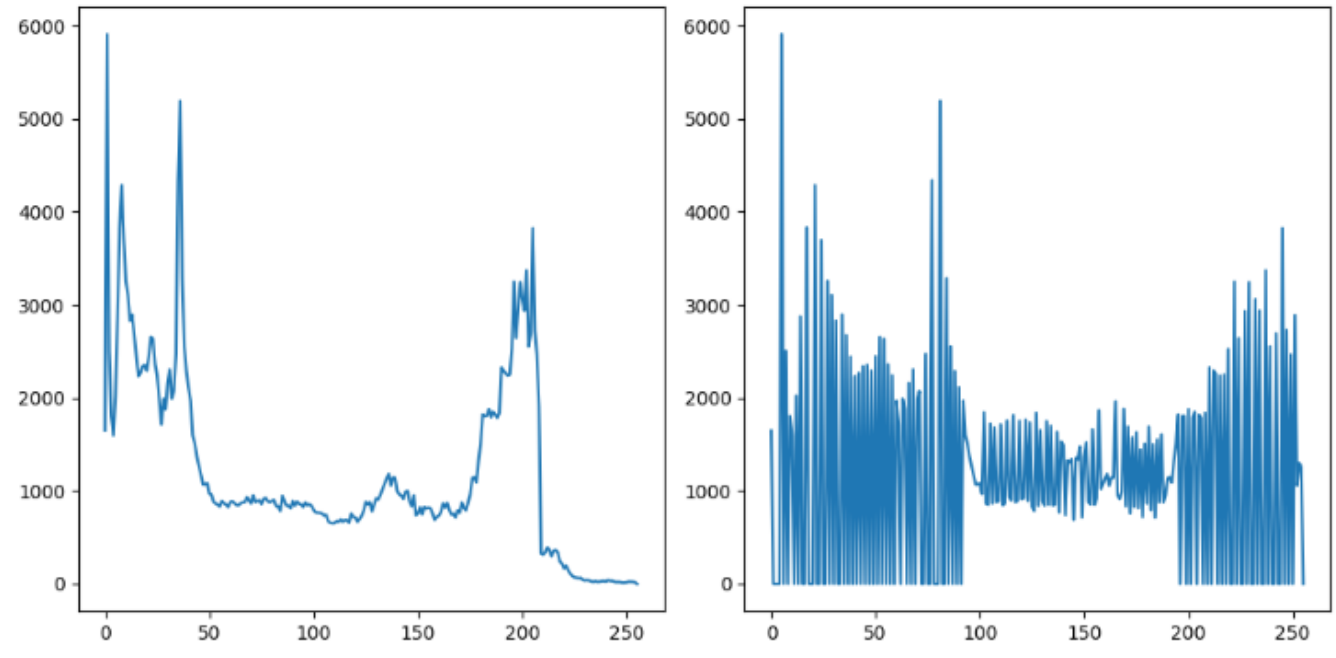
$cdf(v)$: Cumulative distribution function(CDF) before equalizeHist()

cdf_{min} : Min(CDF) before equalizeHist()

$M \times N$: Image size (M: width, N: height)

L : the number of pixel levels (eg: 256 for grayscale)





1. Import module

In [1]:
```
# import basic modules
import os
import copy
import random

# import mudules & version check
import cv2
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

print(f'OpenCV: {cv2.__version__}')
print(f'Numpy: {np.__version__}')
```

```
OpenCV: 4.6.0
Numpy: 1.21.5
```

2. Define functions

In [1]:
```
def show_img(img):
    plt.figure(figsize=(20,30))
    plt.imshow(img)
    plt.axis('off')
    img.shape
    return

def crop_img(img, crop_size_y, crop_size_x=0):
    y, x, _ = img.shape
    return img[:y - crop_size_y, :x - crop_size_x]

def norm_img(img):
    return cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)

def eq_color_img(img):
    # convert BGR to YCrCb Y: luminance, Cr: red-difference chroma component, Cb: blue-d
    img_ycrcb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
    ycrcb_planes = cv2.split(img_ycrcb)
    ycrcb_planes = list(ycrcb_planes)

    # Histogram Equalization in luminance
    ycrcb_planes[0] = cv2.equalizeHist(ycrcb_planes[0])
    ycrcb_planes = tuple(ycrcb_planes)

    dst_ycrcb = cv2.merge(ycrcb_planes)
    dst = cv2.cvtColor(dst_ycrcb, cv2.COLOR_YCrCb2BGR)
    return dst
```

In [3]:
```
def get_subdir(path):
    items = os.listdir(path)

    subdirs = []
    for item in items:
        if item[0] == '.':
            continue
        path_item = os.path.join(path, item)
        if os.path.isdir(path_item):
            subdirs.append(path_item)

    return subdirs

def get_tif_files(path):
```

```
    items = os.listdir(path)

    subfiles = []
    for item in items:
        if item[0] == '.':
            continue
        if item.endswith('.tif'):
            subfiles.append(item)

    return subfiles

def get_path_files(path, fns):
    p_fns = []

    for item in fns:
        p_fns.append(os.path.join(path, item))

    return p_fns

def get_imgs(files):
    return [cv2.imread(f) for f in files]

# check deep copy
# def return_var(v):
#     var = copy.deepcopy(v)
#     return var
```

3. Get path

In [4]:
```
# get path
# os.chdir('../')
path_here = os.getcwd()

subdirs_here = get_subdir(path_here)

print(f'Current Dir: {path_here}')
print(f'Sub Dirs: {subdirs_here}')
```

```
Current Dir: /home/ubuntu/ML_dev/SEM
Sub Dirs: ['/home/ubuntu/ML_dev/SEM/20220823', '/home/ubuntu/ML_dev/SEM/20220826', '/hom
e/ubuntu/ML_dev/SEM/20220831', '/home/ubuntu/ML_dev/SEM/20220902', '/home/ubuntu/ML_dev/
SEM/20220720']
```

4. Image Pre-process: Crop, Normalize, Save image files

In [5]:
```
for dir_iter in subdirs_here:
    os.chdir(dir_iter)

    path_here = os.getcwd()
    crop_path_here = os.path.join(path_here, 'crop')
    norm_path_here = os.path.join(path_here, 'norm')
    eq_path_here = os.path.join(path_here, 'eq')

    if os.path.exists(crop_path_here) == False:
        os.makedirs(crop_path_here)
    if os.path.exists(norm_path_here) == False:
        os.makedirs(norm_path_here)
    if os.path.exists(eq_path_here) == False:
        os.makedirs(eq_path_here)

    files_here = get_tif_files(path_here)

    imgs_here = get_imgs(get_path_files(path_here, files_here))
```

```
    crop_imgs_here = []
    norm_imgs_here = []
    eq_imgs_here = []

    for i in imgs_here:
        crp = crop_img(i, 60, 0)
        crop_imgs_here.append(crp)
        nrm = norm_img(crp)
        norm_imgs_here.append(nrm)
        eq = eq_color_img(crp)
        eq_imgs_here.append(eq)

    crop_fnames = get_path_files(crop_path_here, files_here)
    norm_fnames = get_path_files(norm_path_here, files_here)
    eq_fnames = get_path_files(eq_path_here, files_here)

    for cf, nf, ef, c, n, e in zip(crop_fnames, norm_fnames, eq_fnames, crop_imgs_here,
        cv2.imwrite(cf, c)
        cv2.imwrite(nf, n)
        cv2.imwrite(ef, e)

    # files_dict[return_var(path_here)] = return_var(files_here)
    # crop_files_dict[return_var(path_here)] = return_var(crop_fnames)
    # norm_files_dict[return_var(path_here)] = return_var(norm_fnames)

    # src_imgs_dict[return_var(path_here)] = return_var(imgs_here)
    # crop_imgs_dict[return_var(path_here)] = return_var(crop_imgs_here)
    # norm_imgs_dict[return_var(path_here)] = return_var(norm_imgs_here)

    print(f'{path_here} : {len(files_here)} source files: {len(crop_imgs_here)} files cr

    os.chdir('../')
```

```
/home/ubuntu/ML_dev/SEM/20220823 : 70 source files: 70 files cropped, 70 files normalize
d, 70 files equalized.
/home/ubuntu/ML_dev/SEM/20220826 : 210 source files: 210 files cropped, 210 files normal
ized, 210 files equalized.
/home/ubuntu/ML_dev/SEM/20220831 : 105 source files: 105 files cropped, 105 files normal
ized, 105 files equalized.
/home/ubuntu/ML_dev/SEM/20220902 : 140 source files: 140 files cropped, 140 files normal
ized, 140 files equalized.
/home/ubuntu/ML_dev/SEM/20220720 : 6 source files: 6 files cropped, 6 files normalized,
6 files equalized.
```

5. Image & Histogram Check (validation)

In [12]:

```
# get path
# os.chdir('../')
path_here = os.getcwd()

subdirs_here = get_subdir(path_here)

print(f'Current Dir: {path_here}')
print(f'Sub Dirs: {subdirs_here}')
```

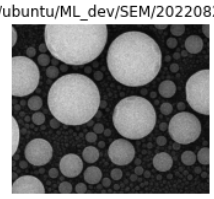
```
Current Dir: /home/ubuntu/ML_dev/SEM
Sub Dirs: ['/home/ubuntu/ML_dev/SEM/20220823', '/home/ubuntu/ML_dev/SEM/20220826', '/hom
e/ubuntu/ML_dev/SEM/20220831', '/home/ubuntu/ML_dev/SEM/20220902', '/home/ubuntu/ML_dev/
SEM/20220720']
```

In [13]:

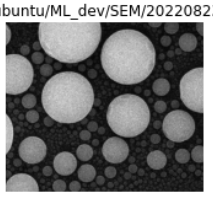
```
plt.figure(figsize=(20,30))
row = 10
col = 3
i = 1
```

```
for dir_iter in subdirs_here:
    os.chdir(dir_iter)

    path_here = os.getcwd()

    crop_path_here = os.path.join(path_here, 'crop')
    norm_path_here = os.path.join(path_here, 'norm')
    eq_path_here = os.path.join(path_here, 'eq')

    files_here = get_tif_files(path_here)

    crop_imgs_here = get_imgs(get_path_files(crop_path_here, files_here))
    norm_imgs_here = get_imgs(get_path_files(norm_path_here, files_here))
    eq_imgs_here = get_imgs(get_path_files(eq_path_here, files_here))

    x = random.randint(0, len(crop_imgs_here) - 1)

    plt.subplot(row, col, i)
    plt.imshow(crop_imgs_here[x])
    plt.title(path_here+'_crop')
    plt.axis('off')

    hist = cv2.calcHist([crop_imgs_here[x]], [0], None, [256], [0, 256])
    plt.subplot(row, col, i + 3)
    plt.plot(hist, color='k')
    plt.title(path_here+'_crop_hist')
    plt.axis('off')

    plt.subplot(row, col, i + 1)
    plt.imshow(norm_imgs_here[x])
    plt.title(path_here+'_norm')
    plt.axis('off')

    hist = cv2.calcHist([norm_imgs_here[x]], [0], None, [256], [0, 256])
    plt.subplot(row, col, i + 4)
    plt.plot(hist, color='k')
    plt.title(path_here+'_norm_hist')
    plt.axis('off')

    plt.subplot(row, col, i + 2)
    plt.imshow(eq_imgs_here[x])
    plt.title(path_here+'_eq')
    plt.axis('off')

    hist = cv2.calcHist([eq_imgs_here[x]], [0], None, [256], [0, 256])
    plt.subplot(row, col, i + 5)
    plt.plot(hist, color='k')
    plt.title(path_here+'_eq_hist')
    plt.axis('off')

    i += 6

    os.chdir('../')
```

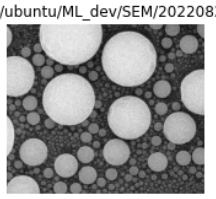
/home/ubuntu/ML_dev/SEM/20220823_crop



/home/ubuntu/ML_dev/SEM/20220823_norm



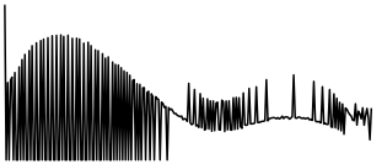
/home/ubuntu/ML_dev/SEM/20220823_eq



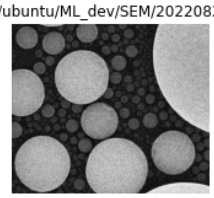
/home/ubuntu/ML_dev/SEM/20220823_crop_hist

/home/ubuntu/ML_dev/SEM/20220823_norm_hist

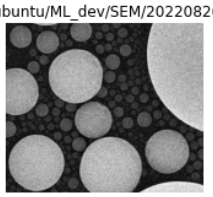
/home/ubuntu/ML_dev/SEM/20220823_eq_hist



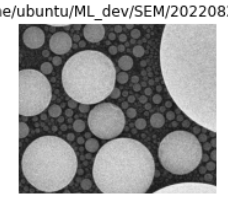
/home/ubuntu/ML_dev/SEM/20220826_crop



/home/ubuntu/ML_dev/SEM/20220826_norm



/home/ubuntu/ML_dev/SEM/20220826_eq

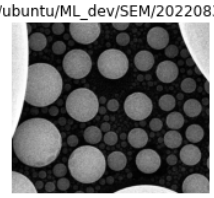


/home/ubuntu/ML_dev/SEM/20220826_crop_hist

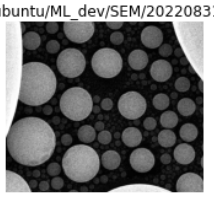
/home/ubuntu/ML_dev/SEM/20220826_norm_hist

/home/ubuntu/ML_dev/SEM/20220826_eq_hist

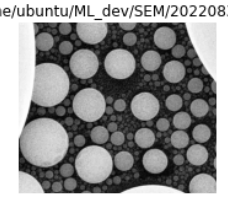
/home/ubuntu/ML_dev/SEM/20220831_crop



/home/ubuntu/ML_dev/SEM/20220831_norm



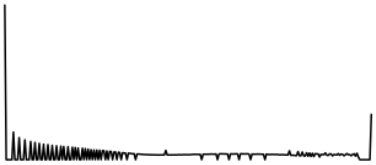
/home/ubuntu/ML_dev/SEM/20220831_eq



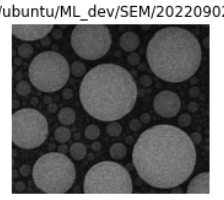
/home/ubuntu/ML_dev/SEM/20220831_crop_hist

/home/ubuntu/ML_dev/SEM/20220831_norm_hist

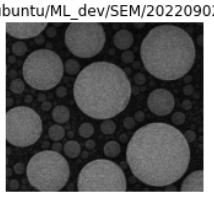
/home/ubuntu/ML_dev/SEM/20220831_eq_hist



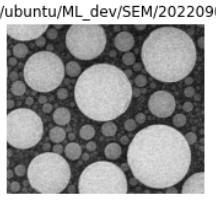
/home/ubuntu/ML_dev/SEM/20220902_crop



/home/ubuntu/ML_dev/SEM/20220902_norm

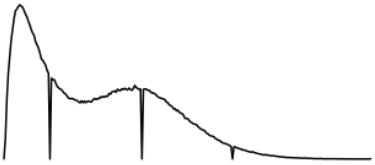


/home/ubuntu/ML_dev/SEM/20220902_eq



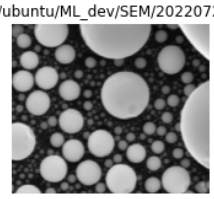
/home/ubuntu/ML_dev/SEM/20220902_crop_hist

/home/ubuntu/ML_dev/SEM/20220902_norm_hist

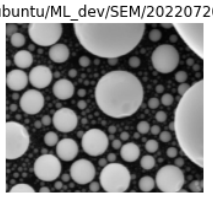


/home/ubuntu/ML_dev/SEM/20220902_eq_hist

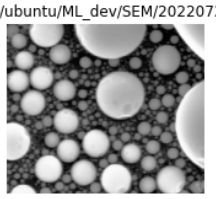
/home/ubuntu/ML_dev/SEM/20220720_crop



/home/ubuntu/ML_dev/SEM/20220720_norm

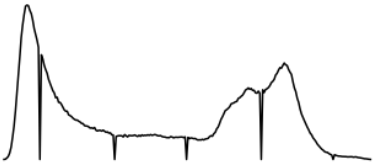


/home/ubuntu/ML_dev/SEM/20220720_eq



/home/ubuntu/ML_dev/SEM/20220720_crop_hist

/home/ubuntu/ML_dev/SEM/20220720_norm_hist



/home/ubuntu/ML_dev/SEM/20220720_eq_hist

References

Wikipedia: Normalization (image processing))
Histogram equalization
Wikipedia: YCbCr
Wikipedia Kr: YCbCr
OpenCV - 10. 히스토그램과 정규화(Normalize), 평탄화(Equalization), CLAHE
Python OpenCV Equalization(BGRscale)
OpenCV: Color Space Conversions
OpenCV: Color Conversions

In [ ]: