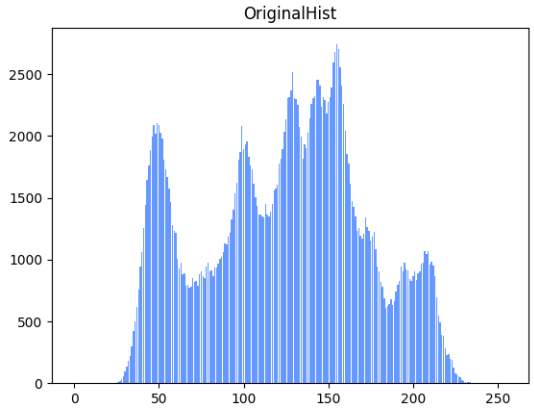
Computer Vision HW3

R08922a27 資工系 人工智慧碩士班 李吉昌

I use python 3.7 to implement all image processing requirements. Reading .bmp file by **PIL**, plotting histogram by **matplotlib.pyplot** and then processing through **NumPy** array.

- ## (a) original image and its histogram

1. Results



2. Code fragment

```python
# define function
def get_hist(img):
    counter = np.zeros(256).astype(int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            counter[int(img[i, j])] += 1
    return counter


def write_hist(hist, name=None):
    fig = plt.figure()
    plt.bar(np.arange(256), hist, color=(0.4, 0.6, 1))
    if name is not None:
        plt.title(name)
        plt.savefig(f'results/{name}.png')
```

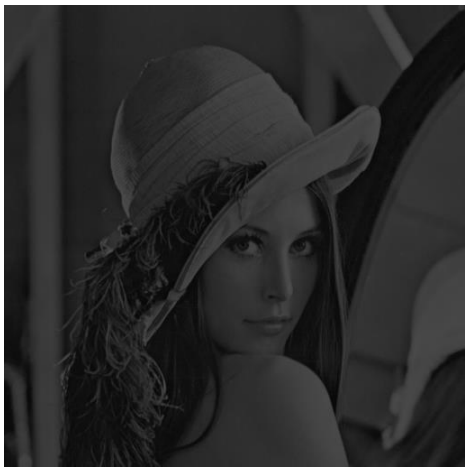
```
org_hist = get_hist(sample_arr)
np.savetxt('results/OriginalHist.csv', org_hist, fmt='%d',
delimiter=",")
write_hist(org_hist, 'OriginalHist')
PIL_image = Image.fromarray(sample_arr.astype('uint8'))
PIL_image.save('results/Original.bmp')
```

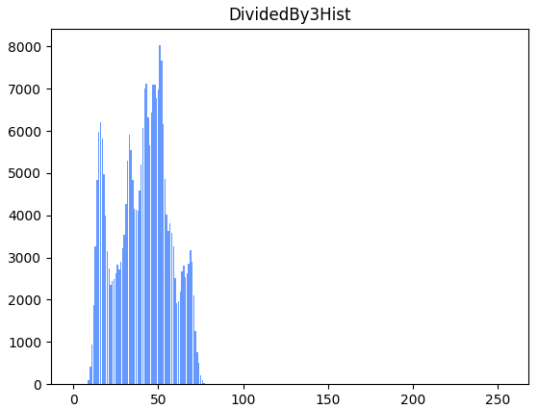
3. Brief description

Firstly, two functions are defined for deriving and writing histograms, respectively. Both of the two functions are also used for other problems. When the histogram is computed, an integer array is used to count the numbers of each grey level; each level value corresponds to the index of each element.

• (b) image with intensity divided by 3 and its histogram

1. Results





2. Code fragment

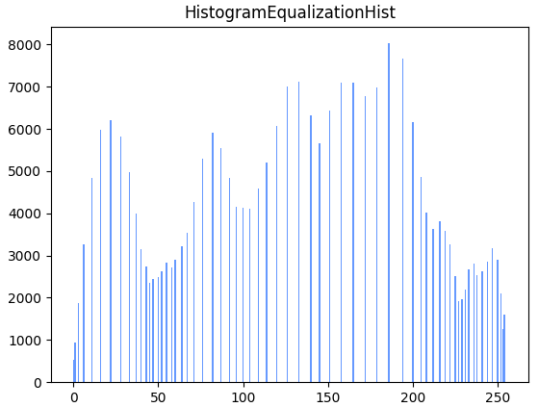
```
div_by_3 = sample_arr / 3
div_by_3_hist = get_hist(div_by_3)
np.savetxt('results/DividedBy3Hist.csv',
           div_by_3_hist, fmt='%d', delimiter=",")
write_hist(div_by_3_hist, 'DividedBy3Hist')
PIL_image = Image.fromarray(div_by_3.astype('uint8'))
PIL_image.save('results/DividedBy3.bmp')
```

3. Brief description

All the intensities are scaled-down by 3 and then run the process the same with (a).

- ## (c) image after applying histogram equalization to (b) and its histogram

1. Results



2. Code fragment

```
CMF[0] = div_by_3_hist[0]
for i in range(1, 256):
    CMF[i] = CMF[i - 1] + div_by_3_hist[i]

CMF = CMF * 255 / div_by_3_hist.sum()
hist_equal = np.zeros(div_by_3.shape).astype(int)
for i in range(div_by_3.shape[0]):
    for j in range(div_by_3.shape[1]):
        hist_equal[i, j] = CMF[int(div_by_3[i, j])]

hist_equal_hist = get_hist(hist_equal)
np.savetxt('results/HistogramEqualizationHist.csv', hist_equal_hist,
fmt='%d', delimiter=",")
write_hist(hist_equal_hist, 'HistogramEqualizationHist')
PIL_image = Image.fromarray(hist_equal.astype('uint8'))
PIL_image.save('results/HistogramEqualization.bmp')
```

3. Brief description

The histogram equalization mapping function, Cumulative Distribution Function(CDF), is derived by the histogram derived in (b). CDF is derived by accumulating the histogram value and then mapping each pixel value to the result.