d11922022 資工所 楊寓鈞

Environment

- python3.9
- Pillow 10.0.0, numpy 1.25.2, pandas 2.0.3

basic setups and utility functions

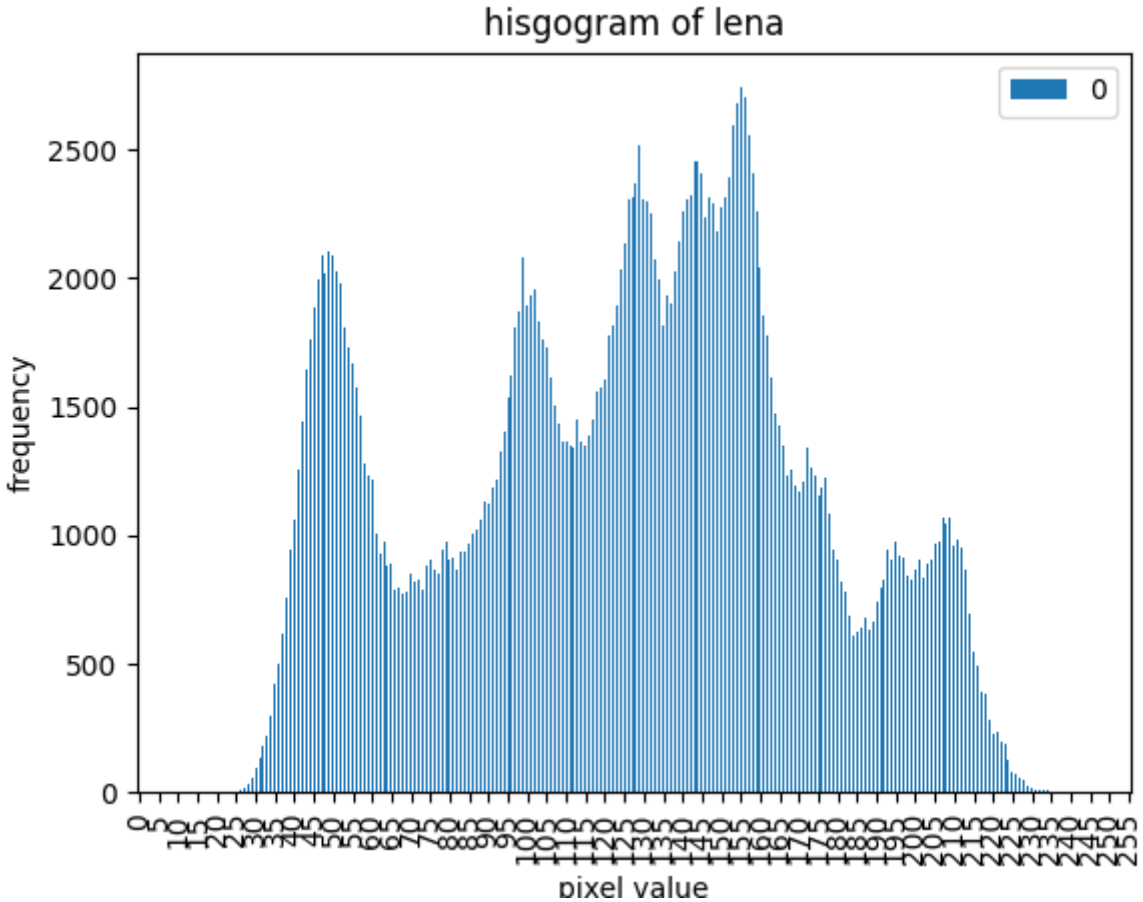
```
from PIL import Image
import numpy as np
import copy

img = Image.open('./lena.bmp') # load lena.bmp
img_array = np.array(img) # pixel content saved in np.array
width, height = img_array.shape # get `width` and `height`
img_list = img_array.tolist() # transform pixel content into list

def save_image(img, path='./lena.bmp'):
        img_ = Image.fromarray(np.array(img, dtype='uint8'), mode='L')
        img_.save(path)
        return img_
```

a. histogram

```
import matplotlib.ticker as ticker
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline

def get_histogram(img, hieght=height, width=width):
        histogram = {i:0 for i in range(0, 256)}
        for y in range(height):
                for x in range(width):
                        bin = histogram.get(img[y][x], 0)
                        histogram[img[y][x]] = bin + 1
        return histogram

result = copy.deepcopy(img_list)
histogram = get_histogram(result)
histogram = dict(sorted(histogram.items(), key=lambda x: x[0]))
df = pd.DataFrame({k:[v] for k,v in histogram.items()}).T
```

```
ax = df.plot.bar(title='hisgogram of lena', xlabel='pixel value',
ylabel='frequency')
ax.xaxis.set_major_locator(ticker.MultipleLocator(base=5))
plt.savefig('histogram.png')
```

1. the function traverse through all pixels of the binarized image
2. create an dict to record gray level and it's corresponding pixel amounts
3. take pixel's gray level as key, accumulate the counts of the pixels (at the same level) as key's value
4. plot histogram via pandas API



b. image intensity div by 3, then histogram

```
def div_by_3(img, height=height, width=width):
        for y in range(height):
                for x in range(width):
                        img[y][x] = img[y][x]//3
        return img

result = copy.deepcopy(img_list)
```

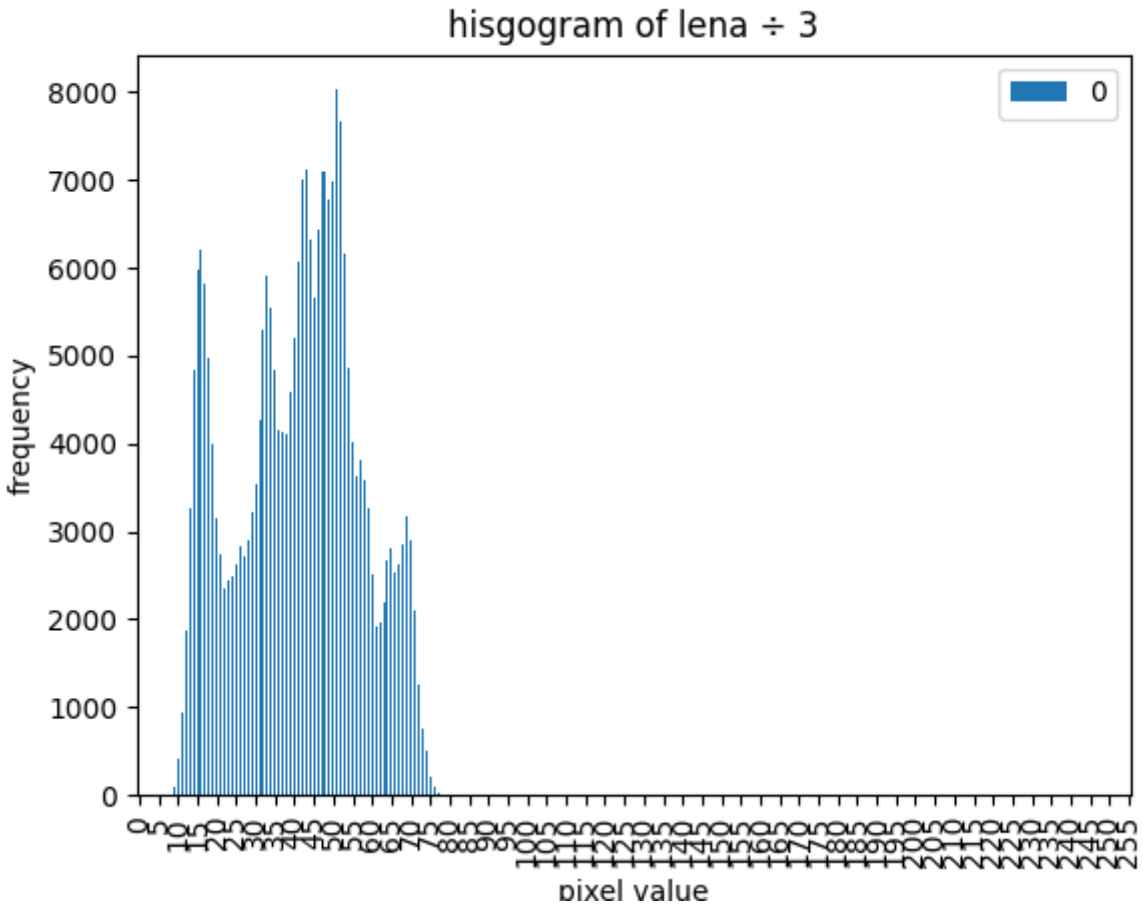
```
result = div_by_3(result)
save_image(result, './lena_div3.bmp')
```

1. simply divide every pixel's value by 3
2. the image appears darker than the origin image

```
histogram = get_histogram(result)
histogram = dict(sorted(histogram.items(), key=lambda x: x[0]))
df = pd.DataFrame({k:[v] for k,v in histogram.items()}).T
ax = df.plot.bar(title='hisgogram of lena ÷ 3', xlabel='pixel value',
ylabel='frequency')
ax.xaxis.set_major_locator(ticker.MultipleLocator(base=5))
plt.savefig('histogram_div3.png')
```

1. apply `get_histogram` function as part a. on divided-by-3 image

2. you can now observe that all the bins have shifted to the left of the spectrum.



c. histogram equalization on b.

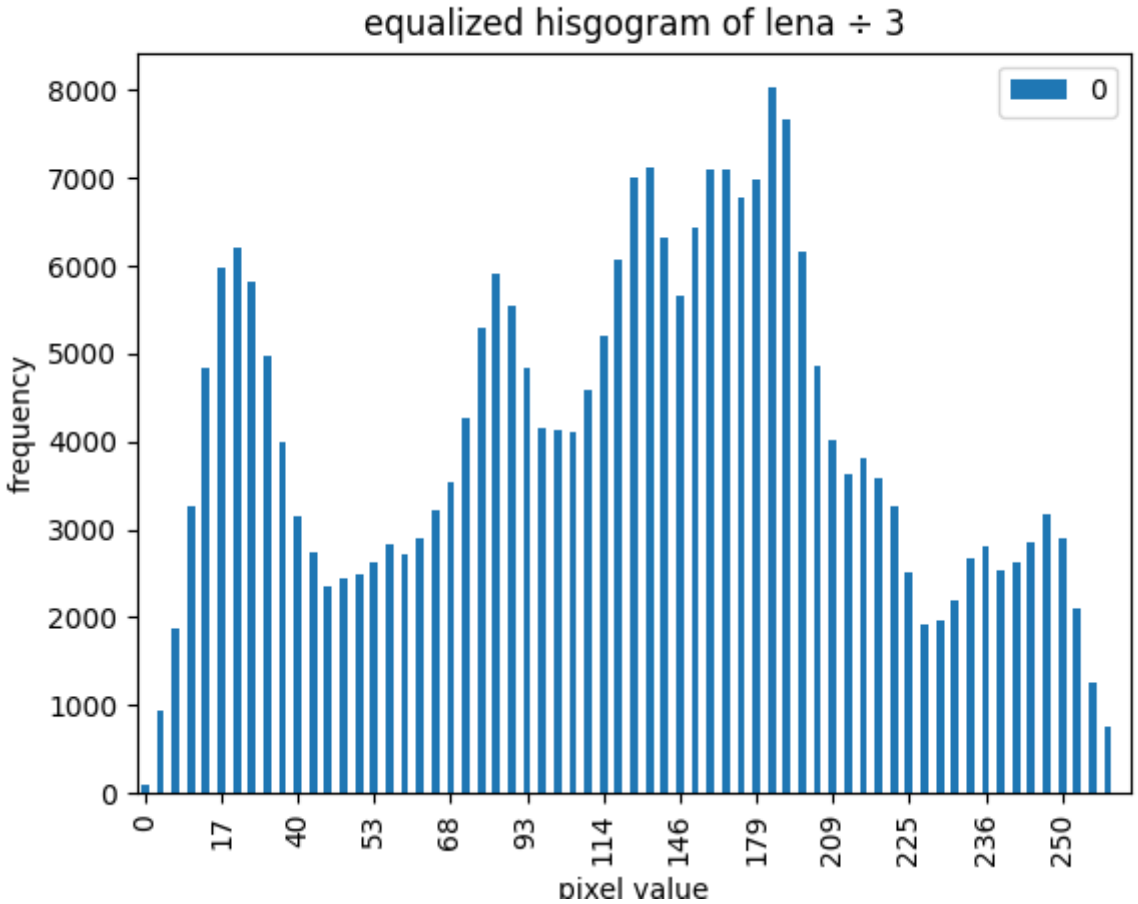
```
# remove value = 0's key-value pair
for k, v in list(histogram.items()):
        if v == 0:
                del histogram[k]

# sort by key
histogram = sorted(histogram.items(), key = lambda x: x[0])
min_bin, min_bin_count = histogram[0]
histogram = dict(histogram)

# direct map min_bin to 0
equalized_histogram = {i:0 for i in range(0, 256)}
equalized_histogram = {0:min_bin_count}
equalize = {}

cdf = 0
for i, (k, v) in enumerate(histogram.items()):
# min_bin has already mapped to 0
```

```
        if i == 0:
                equalize[k] = 0
                continue
        cdf += v
        hv = round(
                ((cdf - min_bin_count)/(height*width - min_bin_count))*255
        )
        equalized_histogram[hv] = v
        equalize[k] = hv

equalized_histogram = dict(sorted(equalized_histogram.items(), key=lambda x:
x[0]))
df = pd.DataFrame({k:[v] for k,v in equalized_histogram.items()}).T
ax = df.plot.bar(title='equalized hisgogram of lena ÷ 3', xlabel='pixel
value', ylabel='frequency')

ax.xaxis.set_major_locator(ticker.MultipleLocator(base=5))
plt.savefig('equalized_histogram_div3.png')
```

1. following to histogram equalization formula:

$$h(v) = \text{round}\left(\frac{\text{cdf}(v) - \text{cdf}_{\text{min}}}{\text{height} \times \text{width}} \times 255\right)$$

   equalize the histogram to make the bin spread across the spectrum uniformly
2. also create a pixel value mapper `eqalize` to map the pixel value to the corresponding equalized value




```
for y in range(height):
        for x in range(width):
                result[y][x] = equalize[result[y][x]]

save_image(result, './lena_div3_equalized.bmp')
```

1. apply the pixel value mapper `eqalize` to map the divide by 3 value