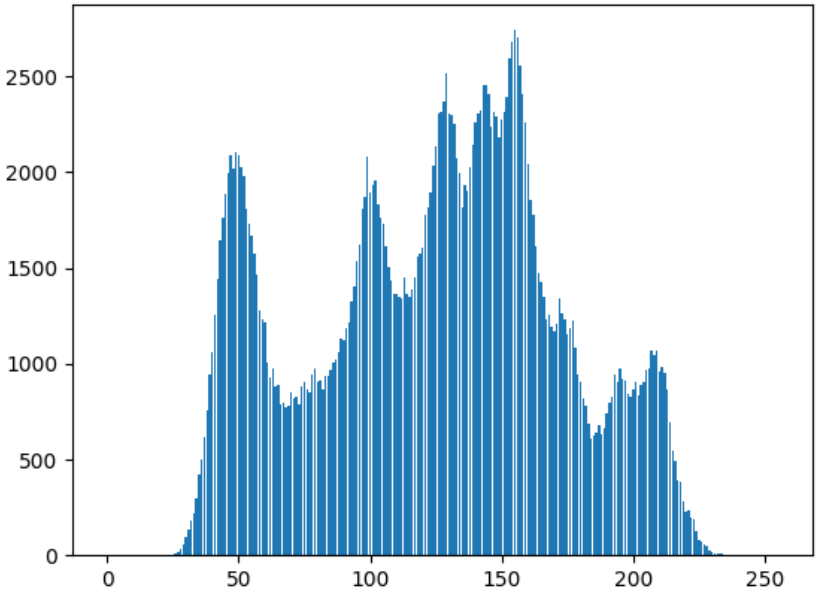
Computer Vision Homework #3

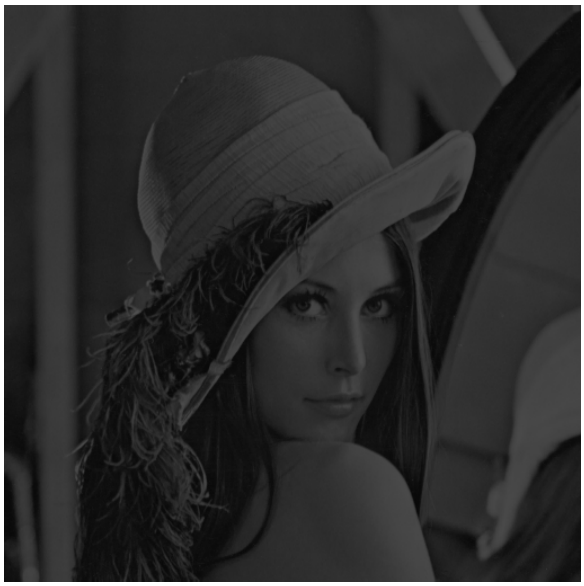
Student-ID: r10944020; Name: 林顯倫; Department: GINM

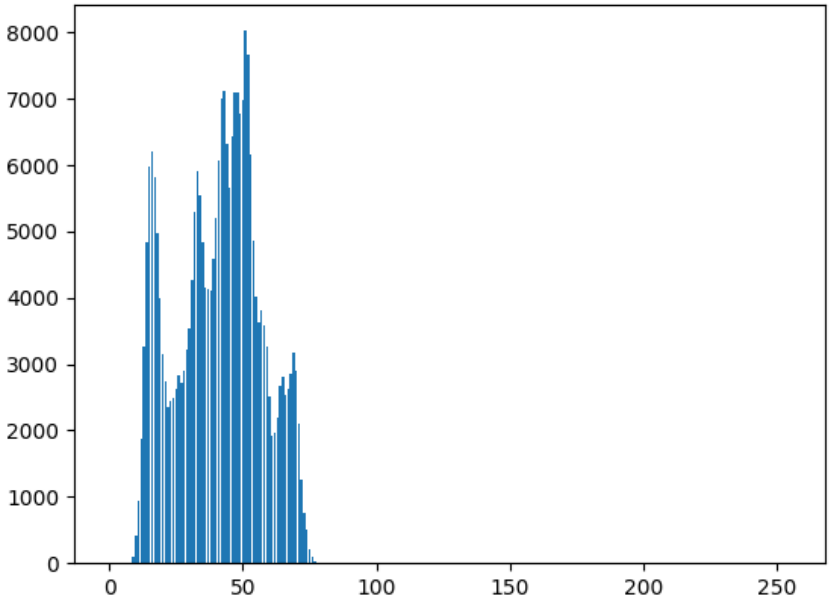
[Results]

(a) original image and its histogram

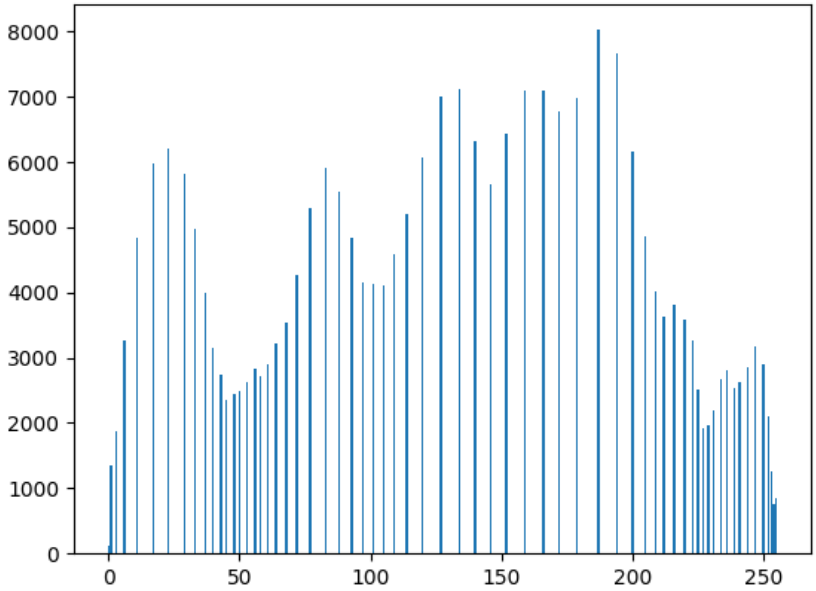


(b) image with intensity divided by 3 and its histogram





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



[Code Fragment & Explanation]

Part-1 Image to Histogram

```
 6 def img2histogram(img):
 7     hist = [0 for i in range(256)]
 8     imageW, imageH = img.width, img.height
 9     for x in range(imageW):
10         for y in range(imageH):
11             hist[img.getpixel((x,y))] += 1
12     plt.bar(range(0, 256), hist)
13     plt.savefig('./histogram_lena.png')
14     plt.clf()
```

1-1. Create a hash vector with dimension 256 to record the pixel value's occurrence.
1-2. Traverse all pixels to record their value occurrence.
1-3. Plot histogram and save it.

Part-2 Intensity Divided by 3

```
16 def img_div3_and_2histogram(img):
17     imageW, imageH = img.width, img.height
18     new_img = img.copy()
19     new_img_pixel = new_img.load()
20     hist = [0 for i in range(256)]
21     for x in range(imageW):
22         for y in range(imageH):
23             new_img_pixel[x, y] = new_img_pixel[x, y] // 3
24             hist[new_img_pixel[x, y]] += 1
25     new_img.save('./div3_lena.bmp')
26     plt.bar(range(0, 256), hist)
27     plt.savefig('./histogram_div3_lena.png')
28     plt.clf()
```

2-1. Create a hash vector with dimension 256 to record the pixel value's occurrence.
2-2. Traverse all pixels to divide its intensity by 3 and record their new pixel value occurrence.
2-3. Plot histogram and save images.

Part-3 Applying Histogram Equalization

```
30 def img_eq_and_2histogram(img):
31     # (1) Count original histogram
32     hist = [0 for i in range(256)]
33     imageW, imageH = img.width, img.height
34     for x in range(imageW):
35         for y in range(imageH):
36             hist[img.getpixel((x,y))] += 1
37     # (2) Compute cdf, cdf_max, cdf_min
38     cdf, cdf_val, cdf_max_pixel, cdf_min_pixel = [0 for i in range(256)], 0, 0, 256
39     for pixel in range(256):
40         if hist[pixel] > 0 :
41             cdf_max_pixel = max(cdf_max_pixel, pixel)
42             cdf_min_pixel = min(cdf_min_pixel, pixel)
43             cdf[pixel] = cdf_val + hist[pixel]
44             cdf_val = cdf[pixel]
45     # (3) Compute pixel values after transformation
46     #            cdf(v) - cdf_min
47     #   h(v) = ------------------- * (Gray Scale Level - 1)
48     #           cdf_max - cdf_min
49     trans_pixel = [0 for i in range(256)]
50     for pixel in range(256):
51         trans_pixel[pixel] = round( float(cdf[pixel] - cdf[cdf_min_pixel]) \
52                                 / float(cdf[cdf_max_pixel] - cdf[cdf_min_pixel]) * 255)
53                                 # / float(imageW*imageH - cdf[cdf_min_pixel]) * 255)
54     # (4) Assign transformed values
55     new_img = img.copy()
56     new_img_pixel = new_img.load()
57     eq_hist = [0 for i in range(256)]
58     for x in range(imageW):
59         for y in range(imageH):
60             new_img_pixel[x, y] = trans_pixel[img.getpixel((x, y))]
61             eq_hist[new_img_pixel[x, y]] += 1
62     new_img.save('./eq_lena.bmp')
63     plt.bar(range(0, 256), eq_hist)
64     plt.savefig('./histogram_eq_lena.png')
65     plt.clf()
```

3-1. Create a hash vector with dimension 256 to record the pixel value's occurrence.
3-2. Compute the information needed for equalization, including cdf, cdf_max, cdf_min.
3-3. Compute the histogram equalization.
3-4. Create new image and assign transformed value to each pixel. At the same time, record the new pixel value's occurrence.
3-5. Plot histogram and save images.