DIP Homework Assignment #1

資工三 b05902115 陳建丞

WARM-UP: SIMPLE MANIPULATIONS

(a) Perform *horizontal flipping*

- **Original image**

Sample1.raw

- **Approach**

    a. Store the pixel values in a 2d array.
    b. swap the values of eahc pair of $A(j,k)$ & $A(j,255-k)$

- **Result**

B.raw

(b) Power-law transform

- **Approach**
    - $G(j,k) = [F(j,k)]^{\gamma}$
    - $F(j,k) =$ the original pixel value / 255
    - Multiply $G(j,k)$ with 255 and round to the nearest integer.
    - Modify the parameter γ to generate different output images.

- **Result**

$\gamma = 0.5$ $\gamma = 3$ $\gamma = 5$ $\gamma = 10$

- **Discussion**

  The original image is a kind of too light. With a proper γ value we can generate a new image with a stronger constrast. $\gamma = 3$ is appropritae while higher γ value might generate an image which is too dark.

- $\gamma < 1 \Rightarrow$ Brighten the image
- $\gamma > 1 \Rightarrow$ Darken the image

PROBLEM 1: IMAGE ENHANCEMENT

Original Image

sample2.raw

(a) Decrease brightness (Divide the intensity by 2)

D.raw

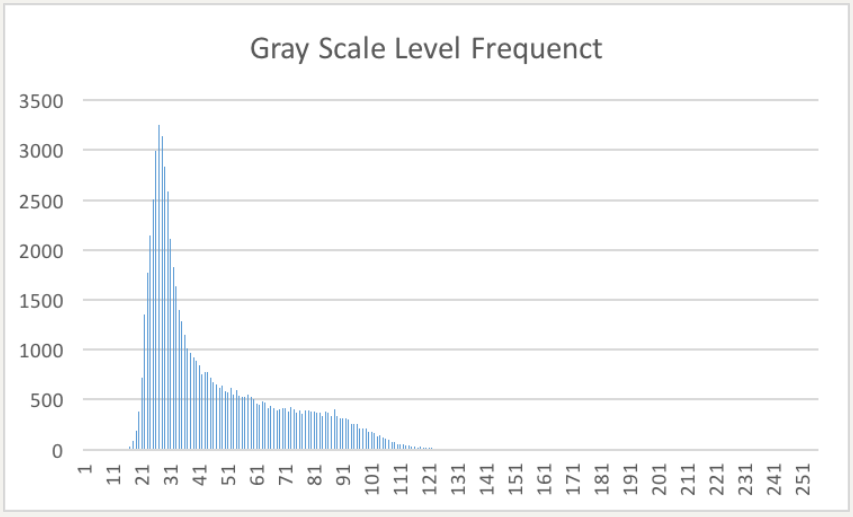
(b) Decrease brightness (Divide the intensity by 3)

E.raw

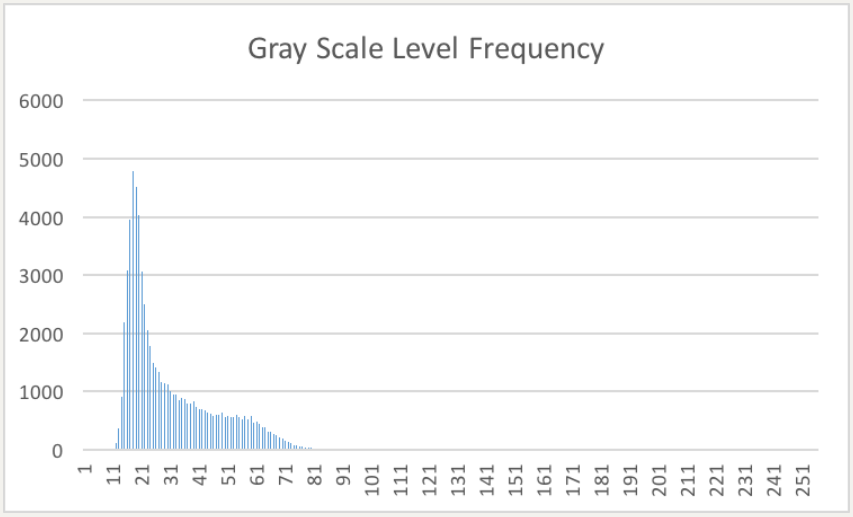
(c) Plot the histograms of D & E

- **Approach**
    a. Use a loop to calculate the frequency of each gray scale level in an array.
    b. Write the frequency array into a CSV file.
    c. Use Microsoft Excel to open the CSV file and generate the histogram.

- **Result**



Histogram of D



Histogram of E

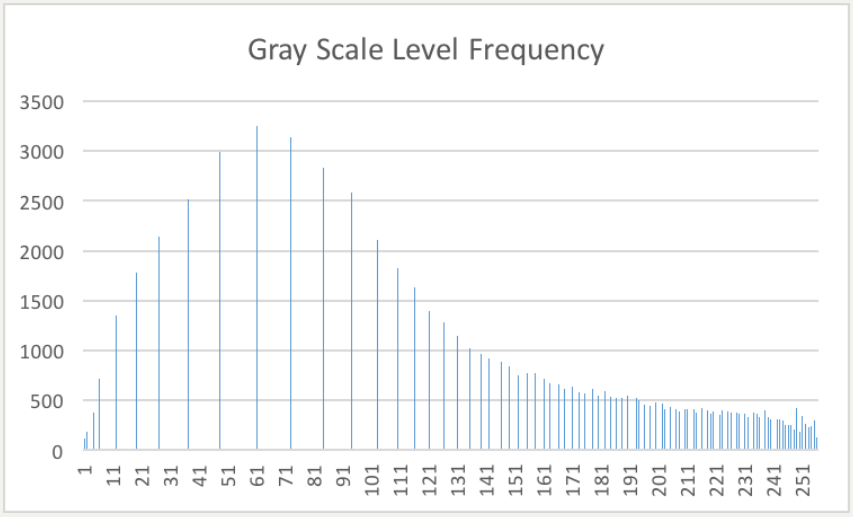
- **Observation**

  From the histograms above, we see that the gray scale level of both images are centralized at the dark side. It's obvious that the gray scale level distribution of E is more centralized. That is, the image E is darker than the image D.

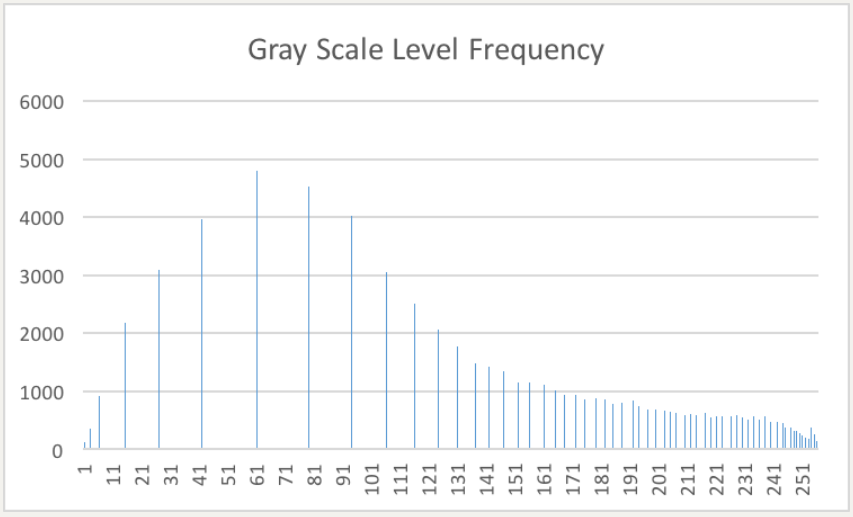
(d) Global histogram equalization

- **Approach**
  a. Use the frequency array above to calculate CDF
  b. New gray scale level = original gray scale level $\times$ CDF / Total pixels
- **Result**

$H_D.raw$ $H_E.raw$



Histogram of H_D



Histogram of H_E

- **Discussion**

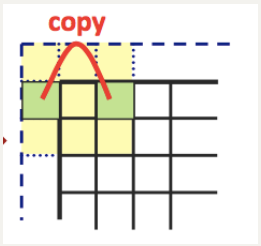
  The two output images look identical even though the original images D & E have a clear difference in the brightness. Since the original image E is more centralized, which means that there may be a lots of pixels with the same gray scale level. Therefore, after the global equalization, the numbers of different gray scale level being used will be very small. As the histograms above, histogram of H_E has less bars than H_D does. Take a deep look at those two output images we we can find that the color of H_E is a little more diverse than that of H_D.

(e) Local histogram equalization

- **Approach**

  a. Set up a window size to do local histogram equalization.

b. Expand the original image to deal with the border condition.



c. Take each small window×window size mask as a new image.

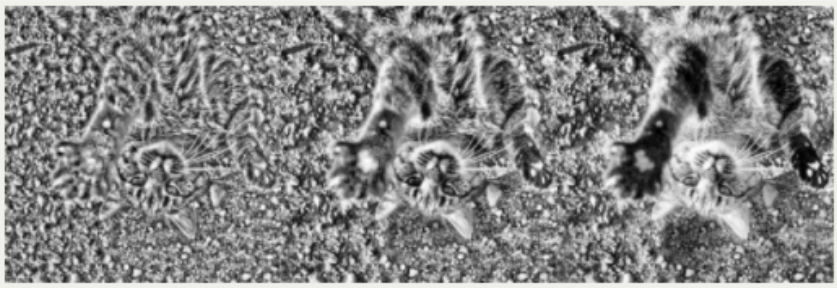
d. Do the same process as global histogram equalizaiton on the small new image segmentation.

e. Only revise the center pixel's gray scale level in each mask iteration.

f. Overlap each mask to make the output image smoother.

- **Result**

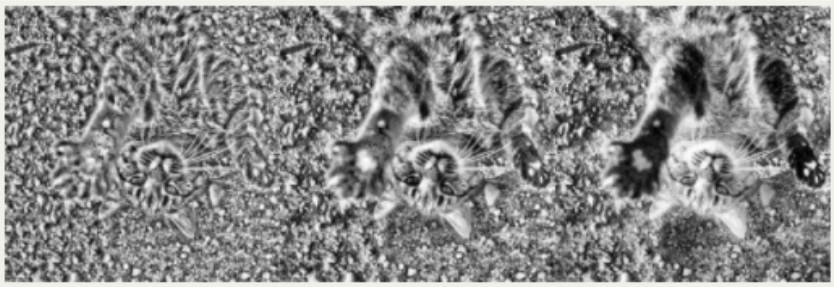
  L_D with different window size



window = 15 window = 35 window = 95

window = 155 window = 191 window = 247

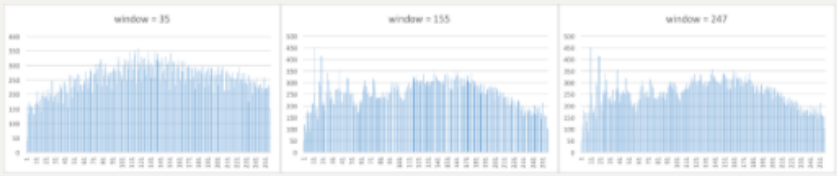
  L_E with different window size



window = 15 window = 35 window = 95

window = 155 window = 191 window = 247

Histogram of L_D



window = 35 window = 155 window = 247

Histogram of L_E

window = 35 window = 155 window = 247

- **Discussion**

  If the window size is too small, the image will become a freak. That is because if we want to equalize in a small mask, we must plug a lot of scale levels in a small mask. Therefore, all the mask will look like the same shape with many scale levels. A real image might contain some blocks of pixels share the same scale level like the paw of the cats. If we just put a lot of scale levels in that paw, hardly can we see what it is.

(f) Difference between global and local histogram equalization

Global equalization performs great, but it can't enhance the local areas of the image. Local equalization is more flexible. We can set up a suitable window size to generate an output image. When window size gets closer to 255, the output image gets more identical to the output image of global equalization.

Problem 2: NOISE REMOVAL

(a) Design filter to remove noise

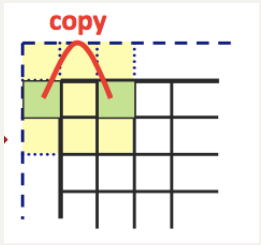
- **Motivation**

  For sample4.raw, I use low-pass filtering to remove noise since it's uniform noise. And for sample5.raw, I use median filtering to remove noise since it's impulse noise. Also median filtering is easy to computed but effective, too.

- **Approach**
  - Low-pass filtering
    i. Set up filter with parameter b.

    $$H = \frac{1}{(b+2)^2}\begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix}$$

    ii. Expand the original image to deal with the border condition.

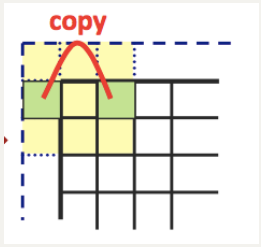
    iii. Multiply each small window×window mask with filter.

    iv. Revise the center pixel's gray scale level in each mask iteration with the sum of the step3.

    v. Overlap each mask to make the output image smoother.

  - Median filtering
    i. Set up filter window size.

    ii. Expand the original image to deal with the border condition.



iii. Sort the values in the mask.

iv. Revise the center pixel's gray scale level in each mask iteration with the median value.

v. Overlap each mask to make the output image smoother.

- **Result**
  - Low-pass filtering (3×3 filter, b = 2)

sample4.raw N_1.raw

  - Median filtering (3×3 filter)

sample5.raw N_2.raw

- **Discussion**

  Uniform noise is hard to remove clearly. Still a lot of noise remains in image N_1. Also, the image becomes blurred after the low-pass filtering. Impulse noise can be removed greatly. The edges in the image still remain clear.

(b) PSNR

- **Approach**

$$PSNR = 10 \times \log\left(\frac{255^2}{MSE}\right)$$

$$MSE = \frac{\sum_{n=1}^{FrameSize} (I_n - P_n)^2}{FrameSize}$$

- **Result**
  - Low-pass filtering (3×3 filter)

```
b = 1, PSNR: 28.246235
b = 2, PSNR: 28.751093
b = 3, PSNR: 28.569782
b = 4, PSNR: 28.173734
b = 5, PSNR: 27.743011
b = 6, PSNR: 27.345499
b = 7, PSNR: 26.995616
b = 8, PSNR: 26.696256
b = 9, PSNR: 26.438589
b = 10, PSNR: 26.216541
b = 11, PSNR: 26.021428
b = 12, PSNR: 25.853626
b = 13, PSNR: 25.704016
b = 14, PSNR: 25.573165
```

  - Median filtering

```
Window Size: 3, PSNR: 28.792838
Window Size: 5, PSNR: 28.708047
Window Size: 7, PSNR: 27.426969
Window Size: 9, PSNR: 26.203034
Window Size: 11, PSNR: 25.148153
Window Size: 13, PSNR: 24.258672
Window Size: 15, PSNR: 23.521943
Window Size: 17, PSNR: 22.868365
Window Size: 19, PSNR: 22.288959
Window Size: 21, PSNR: 21.816641
Window Size: 23, PSNR: 21.392170
```

- **Discussion**

  When the fiter size is largen, the PSNR values will lower. And the computing time will rise considerably. The conclusion is that when b = 2 in the low-pass filtering program. N_1 image is the best. On the other hand, 3×3 Window size of median filtering program performs the best and the running time is pretty fast.

Electronic version README

- **The input image should be on the same directory of source code**
- call **make –f README**
- The programs will be compiled and executed automatically after the command.
- Output images will be stored in the folder **output** and the csv files for histogram use will be stored in the folder **csv**. These two folders will be created automatically, too.