Digital Image Processing HW2 Report

B01902008 資工五 傅敏桓

I. Environment

Programming language: C++ with OpenCV 2.4.13.

Execution Environment: Ubuntu 16.04

II. Execution

Compilation: `$ mkdir build && cd build && cmake .. && make`

For Question 1 to 4: `$ ./hist <image_path>`

For Question 6: `$ ./unsharp <image_path> <factor_k>`

III. Details and Results

Let f be the input and g be the output image, and let L be the intensity level.

T is the transformation function such that g(x,y)=T(f(x,y)).

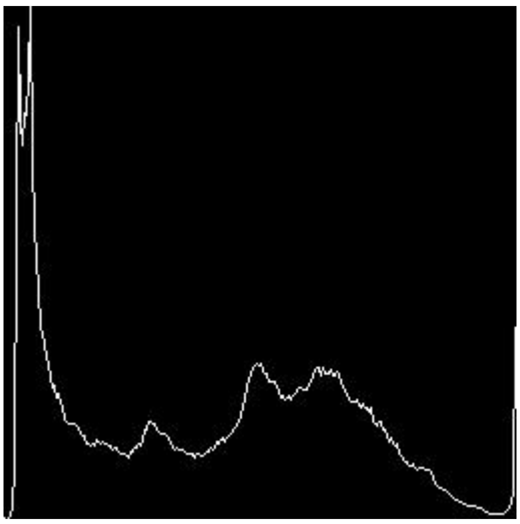
In the following problems, function `cv::LUT()` is used. We first build the 1-D look-up table lut. Each entry of the table maps input intensity level r to output intensity T(r).

For each pixel (x,y) , `LUT()` maps f(x,y) to g(x,y) according to the table.

1. Draw histogram

`cv::calcHist()`: calculates a histogram of given input image.

`cv::line()` is used to draw the histogram.

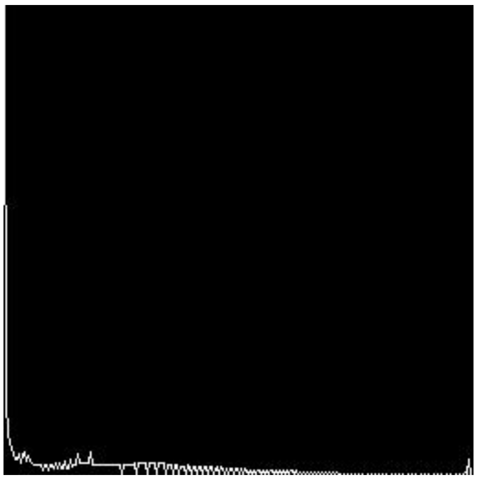


2. Image Degration

2.1 Gamma transformation

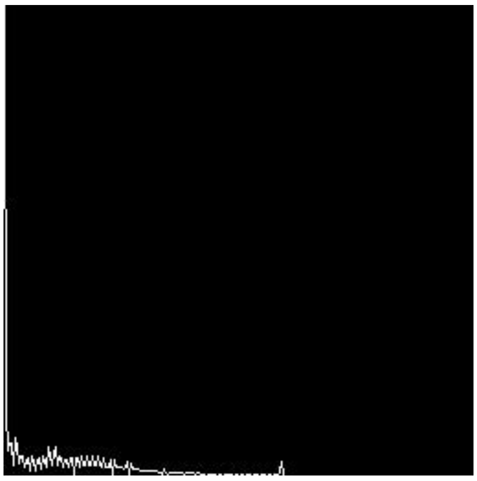
The gamma transformation function is given by $T(r) = cr^{\gamma}, 0 \le r \le 1$.

Look-up table lut can be built by: lut[r]=T(r)= $c(r/L)^{\gamma}L$.



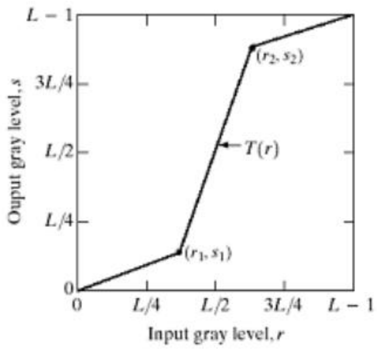
2.2 Reduce the dynamic range to 60%

Let lut[r]=T(r)=0.6r and apply `cv::LUT()`.



3. Full-scale histogram stretch

The contrast stretching function can be determined by four points: (0,0), (r1,s1), (r2,s2), (L-1,L-1).



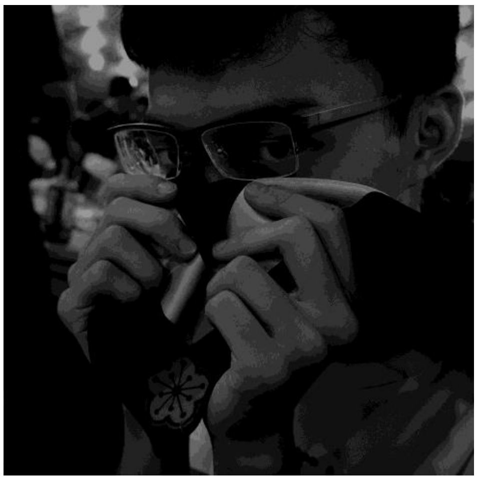
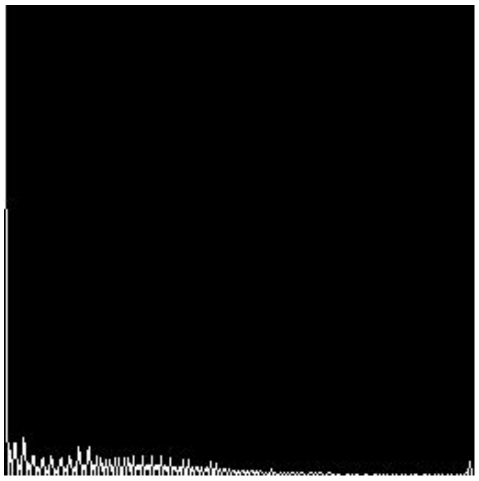
In the case of full-scale histogram:

$$(r_1, s_1) = (r_{min}, 0)$$
$$(r_2, s_2) = (r_{max}, L-1)$$
, where

r: input gray level

s: output gray level

Thus T can be written as $T(r) = \begin{cases} 0, & 0 \le r \le r_{min} \\ \frac{L-1}{(r_{max}-r_{min})}(r - r_{min}), & r_{max} \le r \le r_{min} \\ L-1, & r_{max} \le r \le L-1 \end{cases}$.

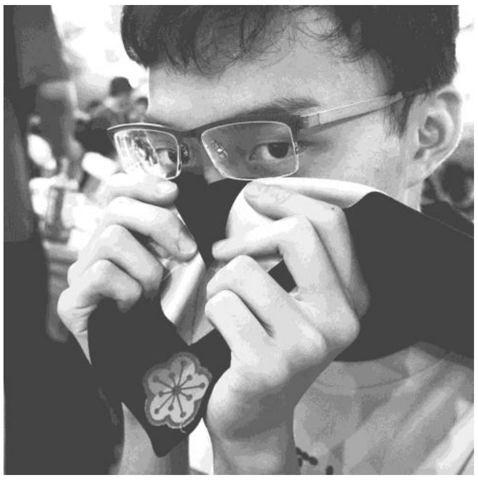
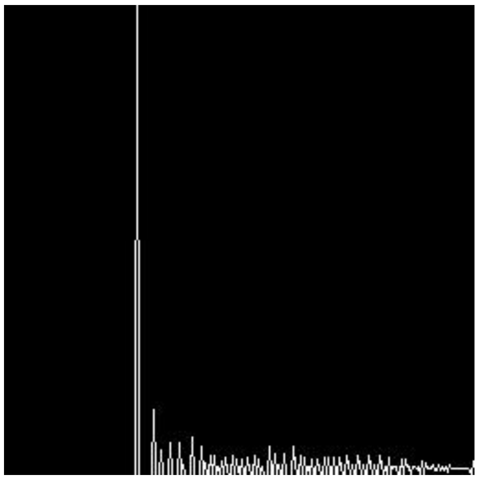
4. Histogram equalization

The transformation function can be written as

$$T(k) = 255\sum_{j=0}^{k}\frac{h(j)}{n}$$

, where n is number of pixels and h(j) is the j-th entry of the histogram.

T is the CDF function of input image (multiplied by (L-1)).

5. Questions

5.1 Explain the reason why the result you get from histogram equalization (Question 4) is not a uniform histogram.

In the proof of histogram equalization, we assume that CDF functions are functions of continuous variable and have continuous output. This works only when we have infinite number of pixels and gray levels.

However, practically we have only fixed number of values in the intensity range [0, L-1], which does not allow us to remap values to the accurate position (in real number) but the nearest discrete gray level.

5.2 Compare the result of Question 3 and Question 4.

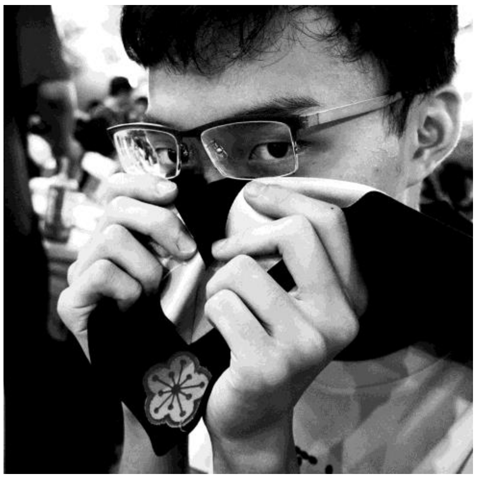
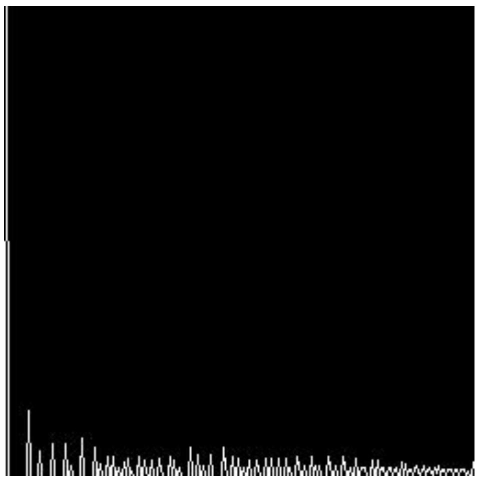
It is obvious that the output image of Question 4 is brighter than the other one, and the histogram is slightly biased to the right side.

Consider the transformation function

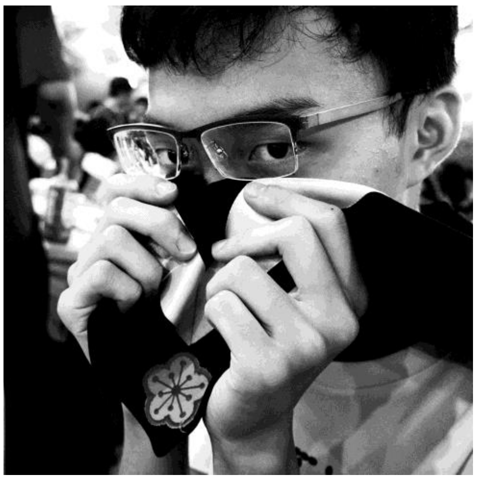
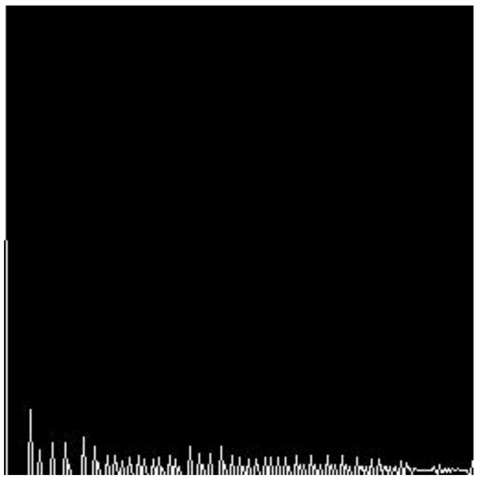
$T(k) = 255 \sum_{j=0}^{k} \frac{h(j)}{n}$, we may easily observe that T(0) is not 0 for most of the case, and thus we can hardly map any value to low intensity part.

We can apply histogram stretch to the equalized histogram.

The following one is obtained by equalization + stretching.

And following result obtained by `cv::equalizeHist()`.

6. Unsharp masking

6.1 Smoothing: 5x5 box filter

6.2 Unsharp masking image

Subtract the smoothed image from the original image to obtain the unsharp masking image.

6.3 Unsharp masking: find best k

Unsharp masking:

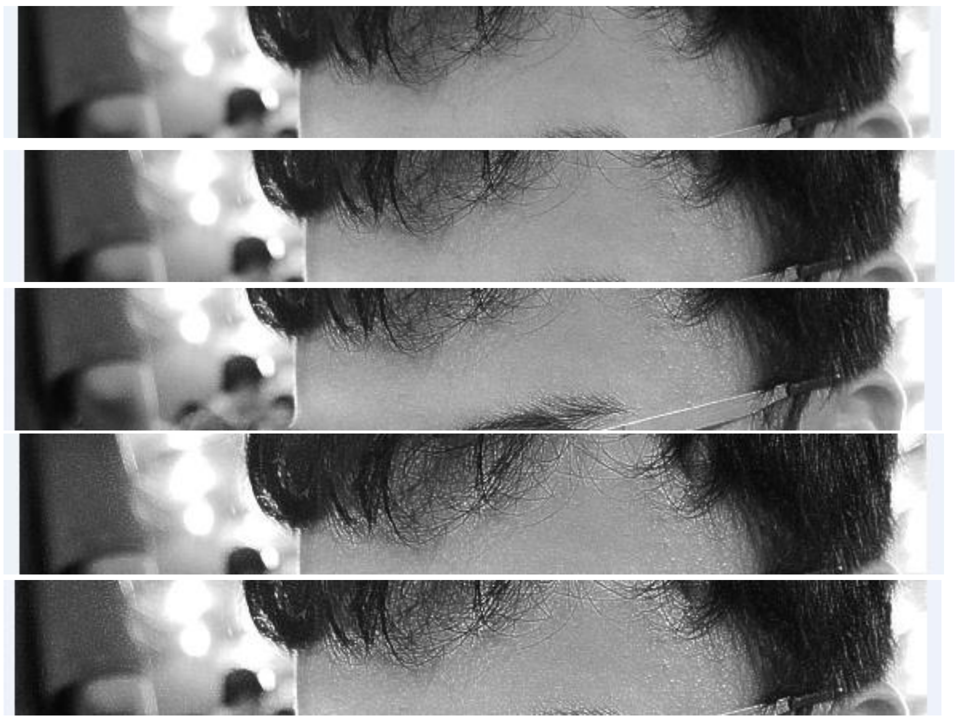
$$\begin{aligned} f_s(x,y) &= f(x,y) + k g_{mask}(x,y) \\ &= f(x,y) + k[f(x,y) - \bar{f}(x,y)] \end{aligned}$$

where

$f_s(x, y)$: sharpened image $f(x, y)$: original image

$g_{mask}(x, y)$: unsharp masking image   k: a constant

Unsharp masking with different k values (k = 1 to 5). Edges of the image become sharper when k grows. Here I think k=4 best sharpen the image.



IV. Results

1-1: src_hist.jpg

2-1: gamma.jpg, gamma_hist.jpg

2-1: degrad.jpg, degrad_hist.jpg

3-1: stretch.jpg, stretch_hist.jpg

4-1: equal.jpg, equal_hist.jpg

6-1: smooth.jpg

6-2: mask.jpg

6-3: unsharp_4.jpg (k=4)