NTU DIP 2020 Spring HW1 Report

B06902002　資工三　沈郁鈞

Problem 0: WARM-UP

(a)

- Read the raw file into a 1D numpy array, then reshape it to the given image size. Write the reshaped array into the jpg format file. The result image `result1.jpg` is same as the given image in spec.

`result1.jpg` (`sample1.raw`)

(b)

- Read the color image in 3-channel, with each represents R, G, B. The convert function is refer to this link (https://bit.ly/33EZyNZ). The result image `result2.jpg` is shown as below.

	
`sample2.jpg`	`result2.jpg`

(c)

- By changing the coordinate of every pixel, we can generate the 90 degrees counterclockwise image `result3.jpg` and diagonally mirrored image `result4.jpg` . The transfer function from `result2.jpg` to `result3.jpg` is $I3(i,j) = I2(j, w-1-i)$ (w represents the image width), and the transfer function from `result2.jpg` to `result4.jpg` is $I4(i,j) = I2(j,i)$. The result images `result3.jpg` and `result4.jpg` are shown as below.

	
`result3.jpg`	`result4.jpg`

Problem 1: IMAGE ENHANCEMENT

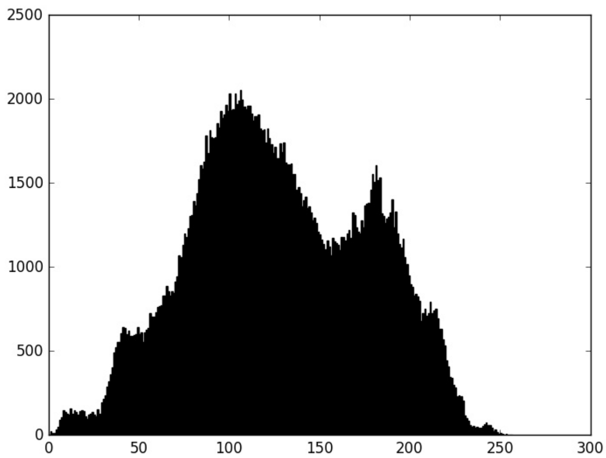
> In this problem, the function `cv2.calcHist()` is used for plotting histogram.

(a)

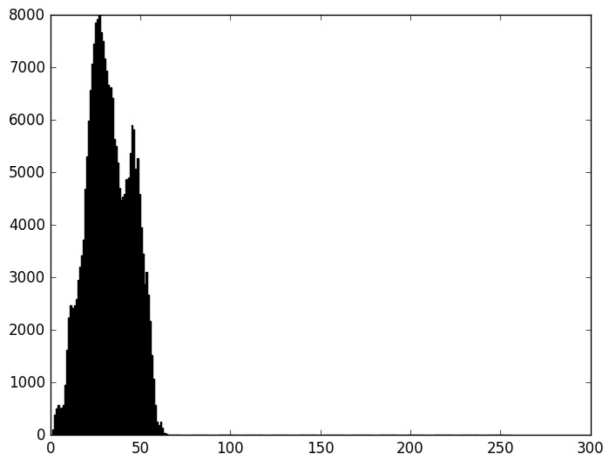
- The histograms of `result1.jpg` and `sample3.jpg` are shown as below. Oberve that the max gray-value in `result1.jpg` is 255, while the max gray-value in `sample3.jpg` is 63. Thus, we can use the transfer function $S3'(i,j) = S3(i,j)$ to make `sample3.jpg` look like `result1.jpg` . ($S3$ represents the original `sample3.jpg` , while $S3'$ represents the result image.)

`result1.jpg`

`sample3.jpg`



`result1_histo.jpg`

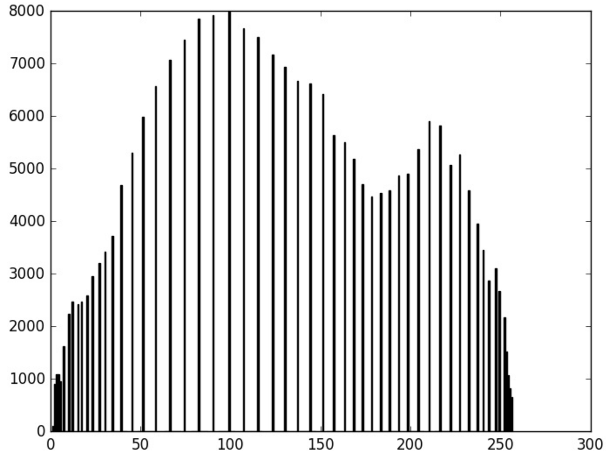


`sample3_histo.jpg`

(b)

- Apply global histogram equilization to `sample3.jpg` .
  The result image `result5.jpg` and its histogram `result5_histo.jpg` is shown as below.

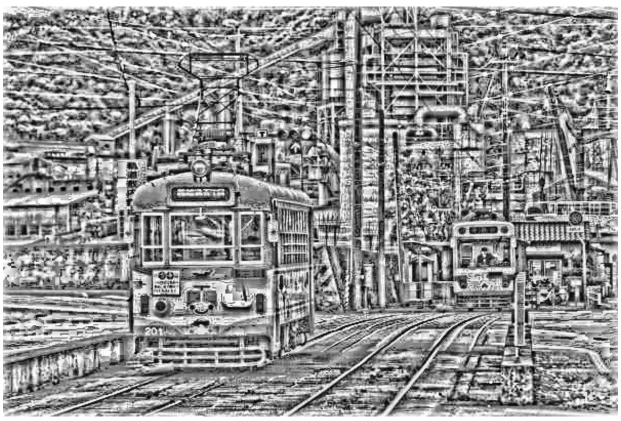
result5.jpg



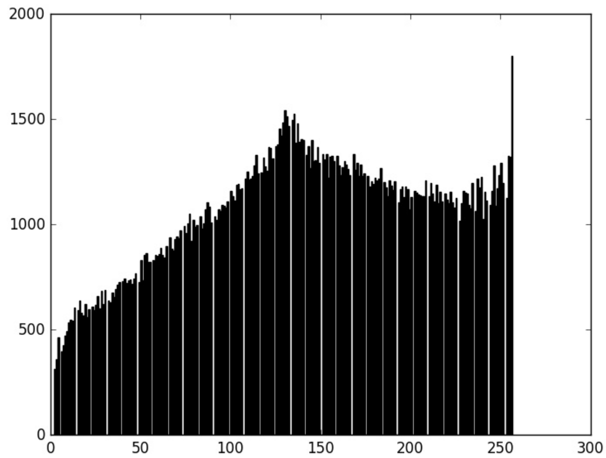
result5_histo.jpg

(c)

- Apply local histogram equilization to `sample3.jpg` with window size 15×15.
  The result image `result6.jpg` and its histogram `result6_histo.jpg` is shown as below.



result6.jpg



result6_histo.jpg

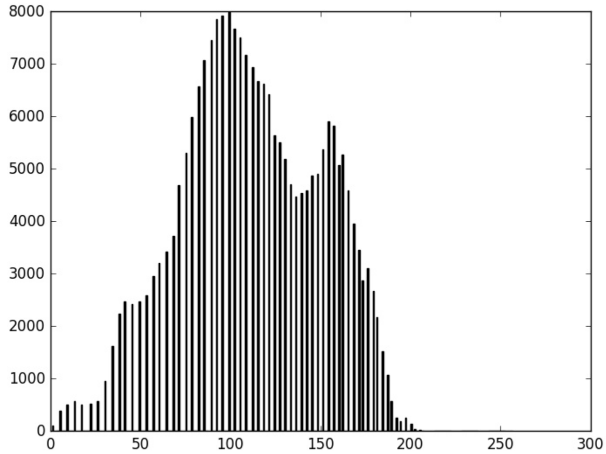
(d)

- Comparing two images above, we can found that the global histogram equalized image enhances the contrast and the average brightness of the original dark image, while local histogram equalized image emphasizes more small details in every small region.

(e)

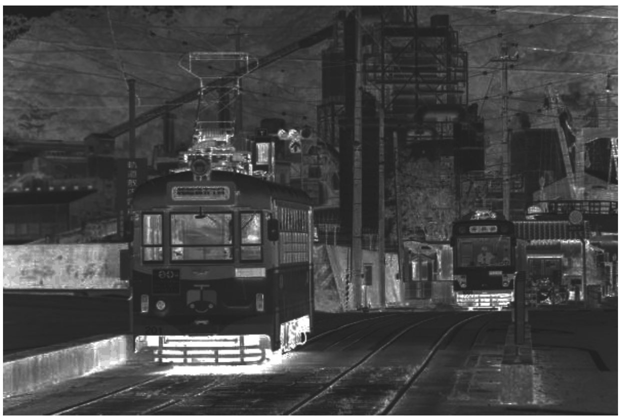
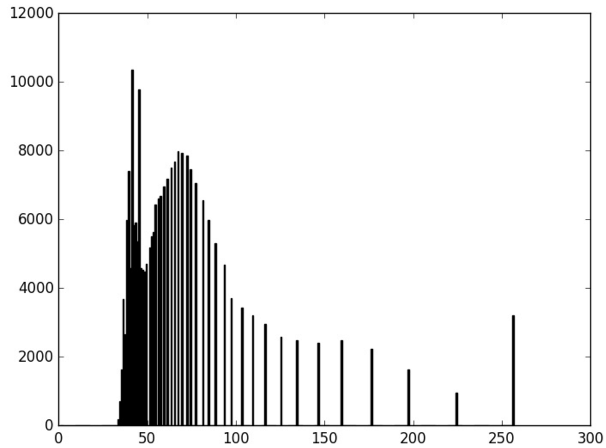
- **Log Transform:**

Apply log transform to `sample3.jpg` with transfer function
$R7(i,j) = \frac{ln(1+aS3(i,j))}{ln2}$ $(a = 3)$.
The result image `result7.jpg` and its histogram `result7_histo.jpg` are shown as below.

	
`result7.jpg`	`result7_histo.jpg`

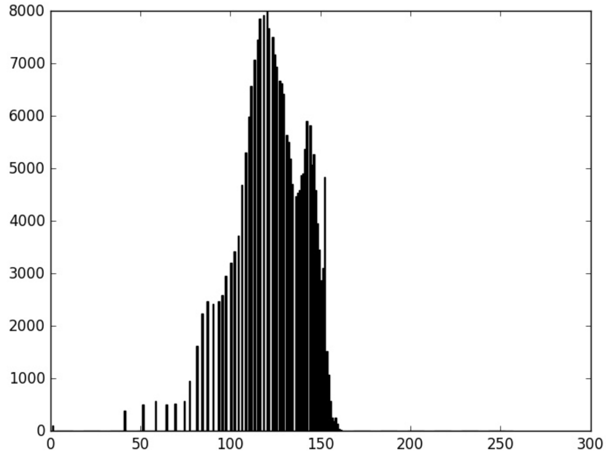
- **Inverse Log Transform:**

  Apply inverse log transform to `sample3.jpg` with transfer function
  $R8(i,j) = \frac{0.1}{R7(i,j)}$ $(0.1 \leq R7(i,j) < 1) \vee R8(i,j) = 1$ $(R7(i,j) < 0.1)$
  The result image `result8.jpg` and its histogram `result8_histo.jpg` are shown as below.

	
`result8.jpg`	`result8_histo.jpg`

- **Power-Law Transform:**

  Apply power-law transform to `sample3.jpg` with transfer function
  $R9(i,j) = [S3(i,j)]^p$ $(p = 1/3)$.
  The result image `result9.jpg` and its histogram `result9_histo.jpg` are shown as below.

	
`result9.jpg`	`result9_histo.jpg`

- **Summary** :
  - Among these three enhanced images, `result7.jpg` seems to be the most similar to `result1.jpg` . The distribution of its histogram is also similar to `result1.jpg` .
  - `result8.jpg` looks dark in most part of the entire image, but it seems to have a few bright regions. These bright regions are definitely the darker regions in `result7.jpg` .
  - Though applying Power-law transform (`result9.jpg`) increases average brightness, the improvement of overall contrast is not apparent.

Problem 2: NOISE REMOVAL

> Since noises are added randomly, execution result may differ.

- Below is `sample4.jpg` , which is the original image and the standard in calculating $PSNR$ value in （e）.

`sample4.jpg`

(a)

- The function $Gau(i,j) = I(i,j) + \delta \cdot N(\mu, \sigma)$ adds **Gaussian** Noise to the given image. (I: original image, Gau: image with gaussian noise, δ: amplitude, $N(\mu, \sigma)$: normal distribution with mean μ and variance σ, here $\mu = 0$ and $\sigma = 1$)

- The two images below are corrupted with Gaussian noise, with amplitude $\delta = 10$ and 30.

	
`resultG1.jpg` ($\delta = 10$)	`resultG2.jpg` ($\delta = 20$)

(b)

- Define a threshold t ($0 \leq t \leq 1$), the function $SNP(i,j)$ adds **salt and pepper** noise to the given image, determined by probability p with uniform distribution $U(\mu, \sigma)$. (Here $\mu = 0$ and $\sigma = 1$)

- For each pixel:
  If $p < t$, then set $SNP(i,j)$ to 0.
  If $t \leq p \leq 1 - t$, then set $SNP(i,j)$ to $I(i,j)$.
  If $p > 1 - t$, then set $SNP(i,j)$ to 255.
  ($I(i,j)$ represents the original image, while $SNP(i,j)$ stands for image contaminated with salt and pepper noise.)

- The two images below are contaminated with salt and pepper noise, with threshold $t = 0.005$ and 0.01.

	
`resultS1.jpg` ($t = 0.005$)	`resultS2.jpg` ($t = 0.01$)

(c)

- Since Gaussian noise is a kind of Uniform noise, performing low-pass filtering is a better choice to remove noise. We set the 3×3 mask as $\frac{1}{(b+2)^2} \begin{bmatrix} 1 & b & 1 \\ b & b^2 & b \\ 1 & b & 1 \end{bmatrix}$. Here I apply $b = 3$.

- The two images below are results of images in **(a)** after low-pass filtering with mask above.

	
`resultR1.jpg`	`resultR2.jpg`

(d)

- Since Salt and pepper noise is a kind of Impulse noise, performing non-linear filtering is a better choice to remove noise. Here I choose outlier detection as the way to remove two noise images in **(b)**, and I set $\varepsilon = 65$

- The two images below are results of images in **(a)** after low-pass filtering with mask above.

	
`resultR3.jpg`	`resultR4.jpg`

(e)

- The $PSNR$ value of `resultR1.jpg` and `resultR2.jpg` are listed below:

Image	**`resultR1.jpg`**	**`resultR2.jpg`**
$PSNR$	33.79	26.48

> Apparently, `resultR1.jpg` is better than `resultR2.jpg` , since `resultR1.jpg` has lower MSE, which means the mean of each pixel in `resultR1.jpg` has smaller
> error to the original image.

- The $PSNR$ value of `resultR3.jpg` and `resultR4.jpg` are listed below:

Image	**`resultR3.jpg`**	**`resultR4.jpg`**
$PSNR$	37.56	34.75

> Most of the **pepper** noise has been removed in both images. However, some **salt** noise (especially in `resultR4.jpg`) still exists in the background. I guess that the gray value of background is close to 255 (gray value of white pixel)

- According to [Wikipedia (https://bit.ly/39frNEI)](https://bit.ly/39frNEI), typical $PSNR$ value of 8-bit grayscale image is between $30db \sim 50db$.

Bonus

- First, I apply median filtering with window size 7×5, then I perform the low-pass filtering using the same mask in （c）. The image below is the result image `result_bonus.jpg` after two-step process.

	
`sample5.jpg`	`result_bonus.jpg`