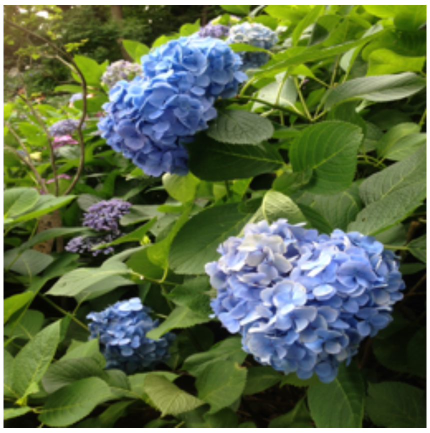
Homework Assignment

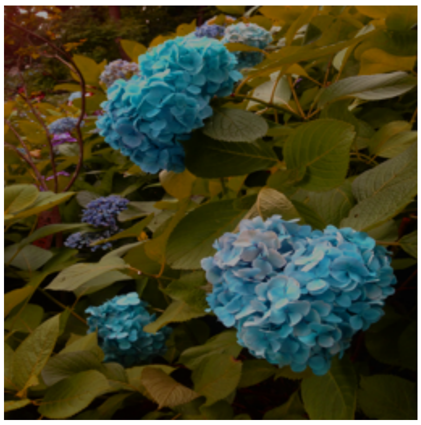
b03902001駱定暄

Programming Language:
C++
Environment:
CSIE Workstation
How to Compile the Programs:
$make

Problem 1.

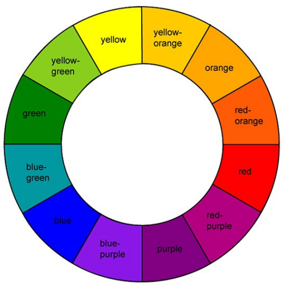
Original Image: Modified Image:





Effect:
(a)Hue:

Assume the hue chart looks like the image below:



Hence, if we only modify the hue channel with thirty-degree clockwise, the image will become figure 1-A.

(b)Saturation:

It is easier to see the effect of saturation modification in the image figure 1-B. (only increase the saturation by 50%):

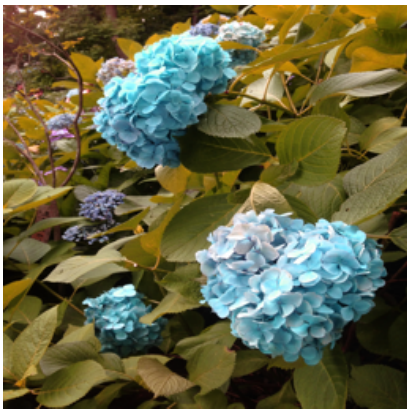


figure 1-A

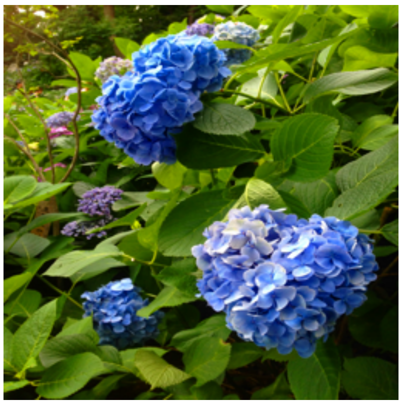


figure 1-B

(c)Lightness:

The image becomes darker, because lightness is reduced by 35%.

Implementation:
Code: rgb.cpp
Executable File: prob1
Usage: $./prob1 filepath

Functions:

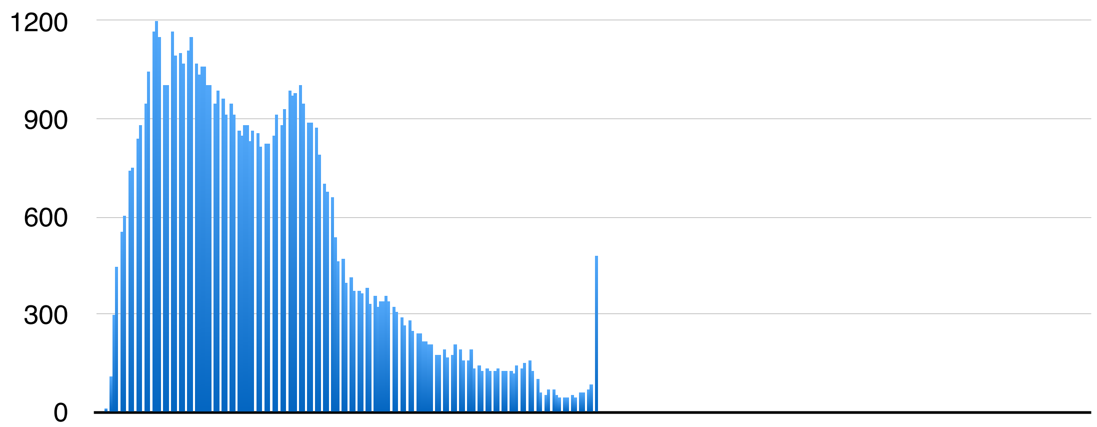
(a) int main():
Read the input image.
Call rgb_to_hsv()
Adjust hue, saturation, and lightness.
Call hsv_to_rgb()
Write to the output file.

(b) rgb_to_hsv():
Change pixel encoding from RGB to HSV.

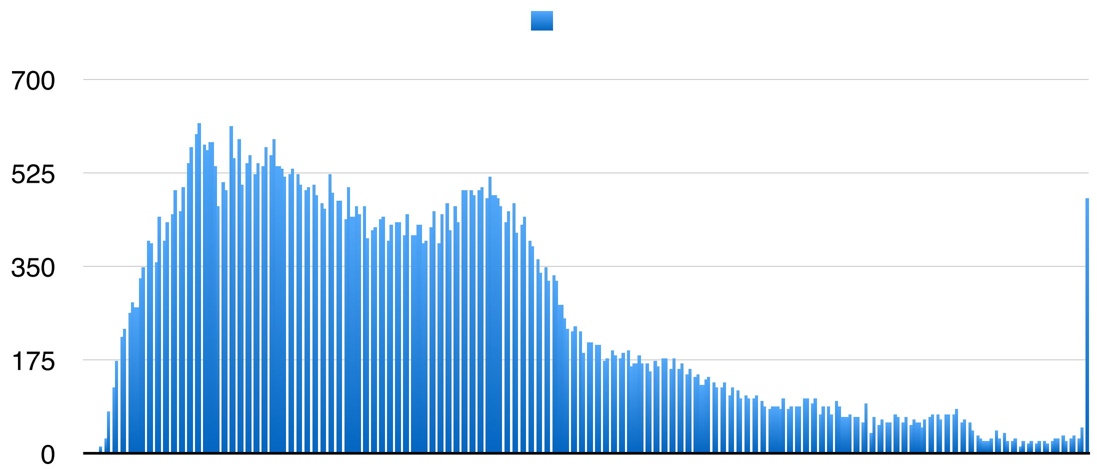
(c) hsv_to_rgb():
Change pixel encoding from HSV to RGB.

Problem 2.

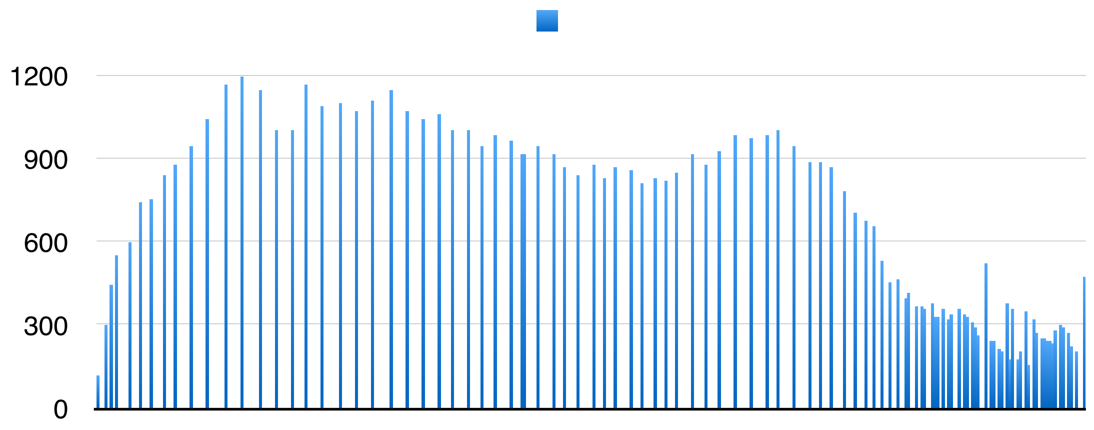
The Histogram of ‘sample2.raw’:



The Histogram of ‘sample3.raw’:



The Histogram of ‘sample2.raw’ after Histogram Equalisation:



(a) The grey-scale range of ‘sample2.raw’(figure 2-A) is smaller than that of ‘sample3.raw’(figure 2-B). Hence, if I want to make ‘sample2.raw’ look like ‘sample3.raw’, I can enhance the contrast of ‘sample2.raw’ by the technique called histogram equalisation. And the result is shown in figure 2-C.

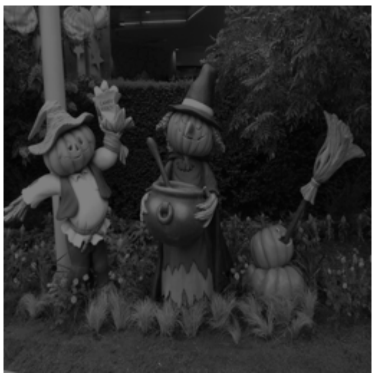


figure 2-A

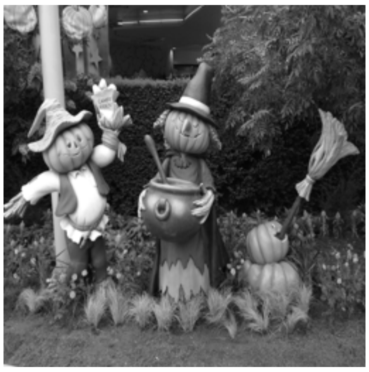


figure 2-B

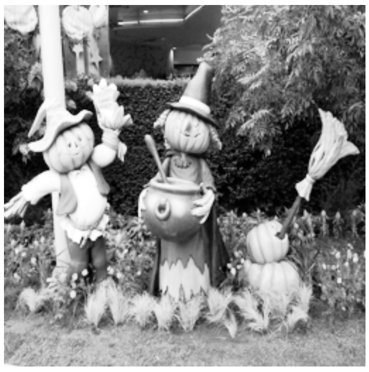


figure 2-C

(c) The original intensities on the histogram only range from greyscale 2 to greyscale 128. After histogram equalisation, the intensities are distributed more evenly on the entire histogram. As for the effect on the image, the whole image looks brighter and clearer to human beings’ eyes.

Implementation:
Code: histogram.cpp
Executable File: prob2
Usage: $./prob2 filepath

Functions:

(a) int main():

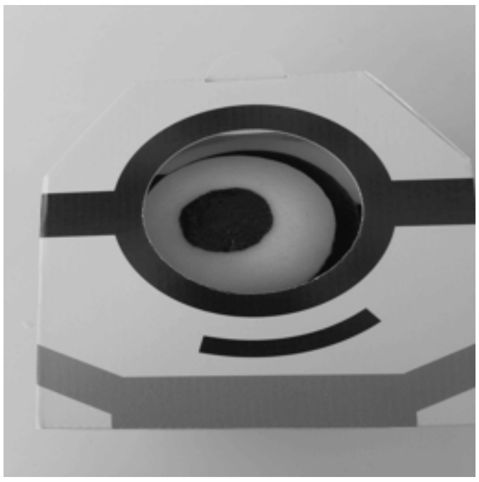
Load the input image and count the number of occurrence of each intensity.
Construct the CDF of original intensities
Map each original intensity to a new intensity value by histogram equalisation, and the mapping function looks like below:

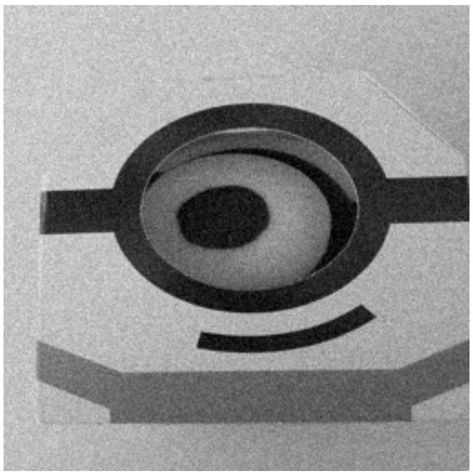
$h(v) = \text{round}((\text{cdf}(v) - \text{cdfmin}) / (M * N - 1) * (L - 1))$

cdf(v) is the cdf of old intensity.
cdfmin is the minimum cdf value.
h(v) is the new intensity.
M * N is the number of pixels.
L is the number of grey levels.

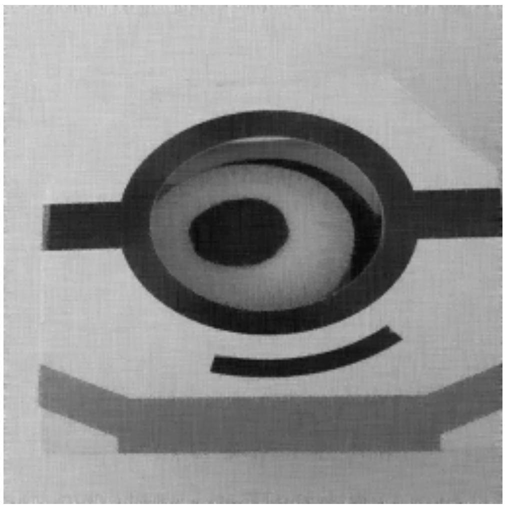
Store the new image according to the new intensities.

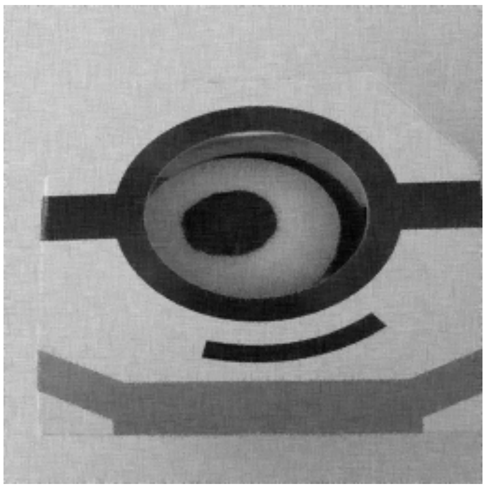
Problem 3.

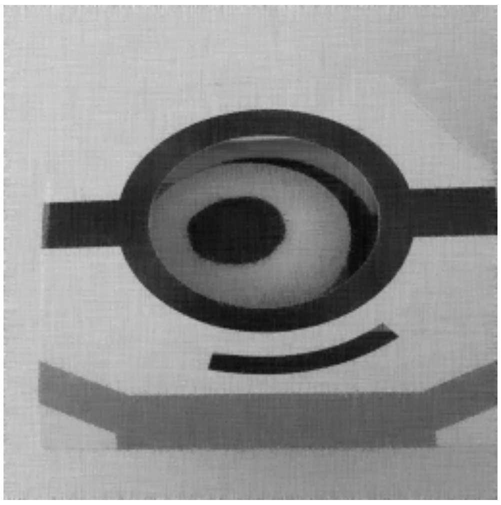

(sample4.raw)


(Noise with Sigma = 10)

Removing Noise:


(window size = 10*2+1, PSNR = 33.7587dB)


(window size = 5*2+1, PSNR = 33.3286dB)


(window size = 12*2+1, PSNR = 33.5391dB)

Implementation:
Code: noise.cpp
Executable File: prob3
Usage: $./prob3 filepath

Idea:

I choose two-directional pseudo median filter to eliminate the noise in the image. The filter is in the shape of a cross which is similar to the method described in 20161216 lecture note p.17. The window size denotes the width and height of the filter.

Functions:

(a)int main():

Load the input image.
Call add_noise() to add noise to the image.
Store the image with noise to ‘sample4_noise.raw’.
Call pseudo_median() to eliminate noise on the image.
Call getPSNR() to get the PSNR value of the image after pseudo median.
Store the resulting image to ‘sample4_pseudo.raw’.

(b)add_noise():

Generate the noise by calling a normal distribution random generator in C++.

```
normal_distribution<double> distribution (0.0, 10.0);
```

(c)pseudo_median():

For a filter with a window size of 2*n+1, the pseudo median is the average of the maximum of the minimum and the minimum of the maximum of the sub-window with a size of n+1 sliding through the vertical and horizontal region of the cross.

(d)getPSNR():

Calculate the MSE between the original image and the image performed pseudo median.

$$MSE = \frac{1}{m \cdot n} \Sigma\Sigma(I(i, j) - I'(i, j))^2$$

m, n is the width and height of the image.
I(i, j) is the pixel value of the original image at position (i, j).
I’(i, j) is the pixel value of the new image at position (i, j).

Calculate the PSNR depending on the MSE.

PSNR = 10 * log (255^2/MSE)

Result Discussion:
In the beginning, the PSNR value grows when the window sizes grows. However, after the window size exceeds 10*2+1, the PSNR value starts to decrease. In my opinion, at first, the PSNR value improves, because a larger filter can eliminate more errors. Nevertheless, when the filter size becomes too large, the pseudo median method will smooth the image. Thus, the PSNR value declines.

Bonus:

Code: png.py
Usage:

```
$python png.py [rgb|grey] input_file output_file
```

Idea:
Load the raw data and store it into another file with .png data type.

Reference:

[1] https://en.wikipedia.org/wiki/HSL_and_HSV
[2] https://www.programmingalgorithms.com/algorithm/rgb-to-hsv?lang=C%2B%2B
[3] https://www.programmingalgorithms.com/algorithm/hsv-to-rgb?lang=C%2B%2B
[4] https://en.wikipedia.org/wiki/Gaussian_noise
[5] http://www.cplusplus.com/reference/random/normal_distribution/normal_distirbution/
[6] https://en.wikipedia.org/wiki/Pseudomedian
[7] https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio