Warm-up

(a) sample1.raw (I_1)

(b) Grayscale

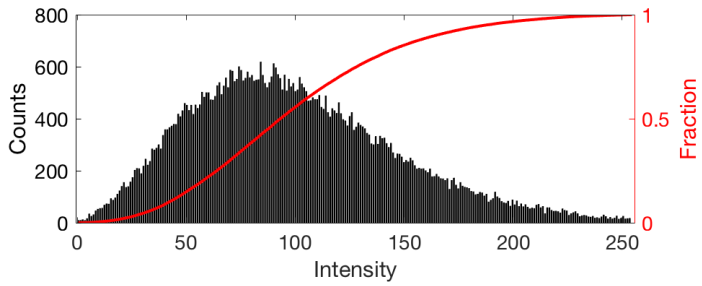
(c) Diagonally flipped (B)

Figure 1: Simple manipulations

Problem 1

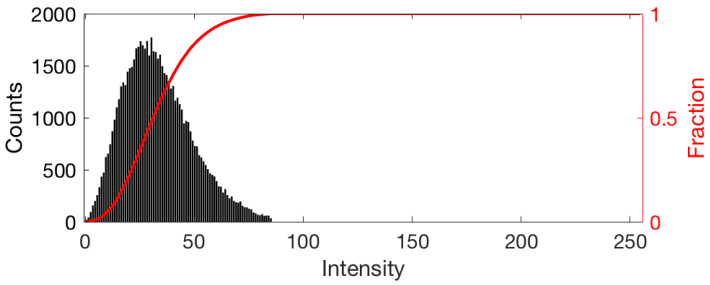
Divide-by-3

(a) sample2.raw (I_2)



(b) Histogram of I_2

(c) Brightness decreased (D)

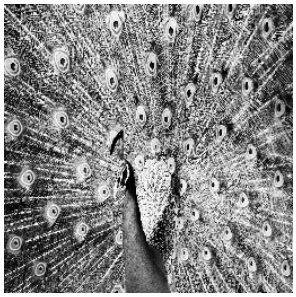


(d) Histogram of D

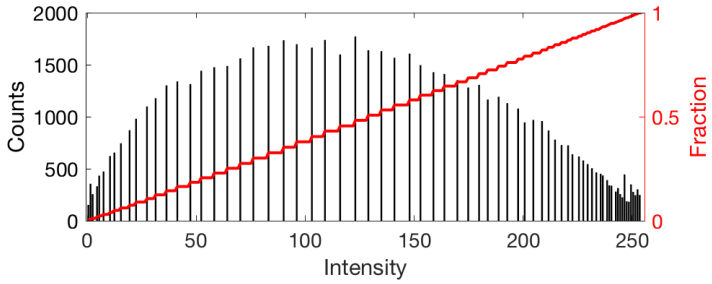
Figure 2: Histogram after intensity divisions

From Figure 2, due to division, all the pixels are accumulated at low intensity portion of the curve, yielding the right-skewed histogram. The division also causes pixels to have relatively fewer levels to present their contrast, hence the fraction curve rises in histogram of D.

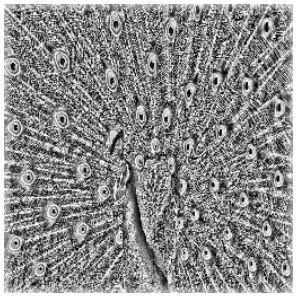
Histogram equalization



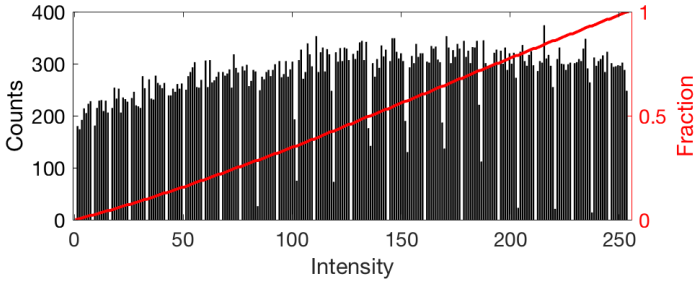
(a) Global (H)



(b) Histogram of H



(c) Local (L), $k = 15$



(d) Histogram of L

Figure 3: Histogram after histogram equalization

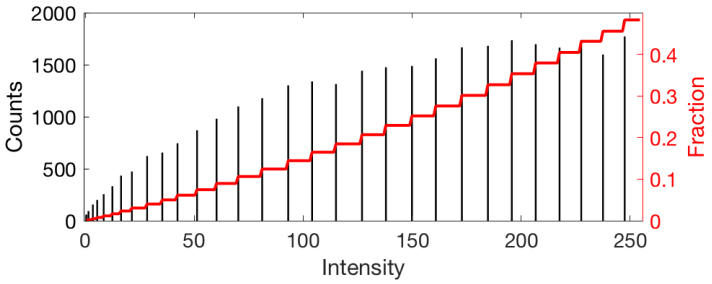
By building a simple transfer function to convert the distribution of pixels in D as uniformly as possible, we have H. The stretching is visible in its histogram (Figure 3b) as gaps due to information loss during the division (pixels in similar levels are forced to group as the same one).

Since local histogram equalization only consider intensity distribution in a defined, small region (kernel size is 15), it potentially allows one to visualize more details, but also being prone to background noise. Hence, while L is visually clear in the orientation of feathers, noise is also visible after zoom-in. Convolving the equalization filter also allows L to distributes pixels more uniformly, since the transfer function do not have to accommodate for wide intensity variations (globally speaking).

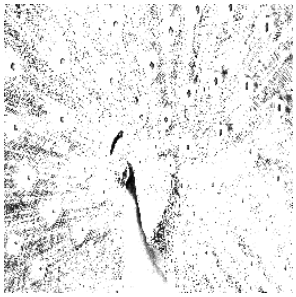
When one compares the result with Figure 2d, envelope from global histogram equalization highly resemble the envelope of the original image's histogram, which is not visible in the local histogram equalization.

Transformations

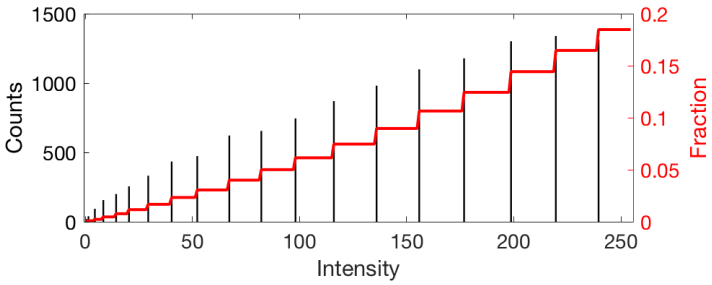
(a) $y = log_2(1 + 2x)$



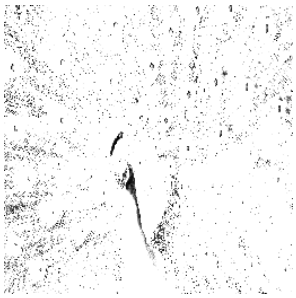
(b) Histogram of Figure 4a



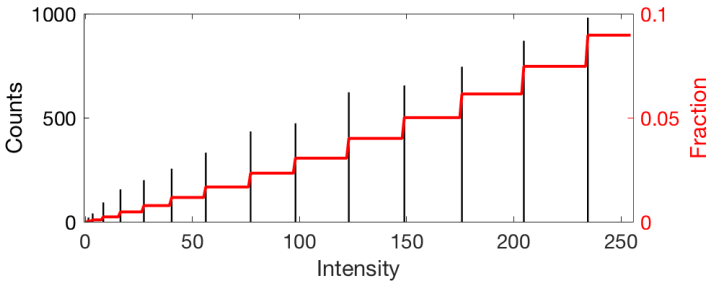
(c) $y = log_2(1 + 5x)$



(d) Histogram of Figure 4c



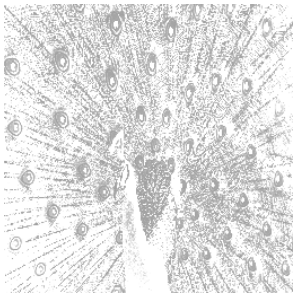
(e) $y = log_2(1 + 10x)$



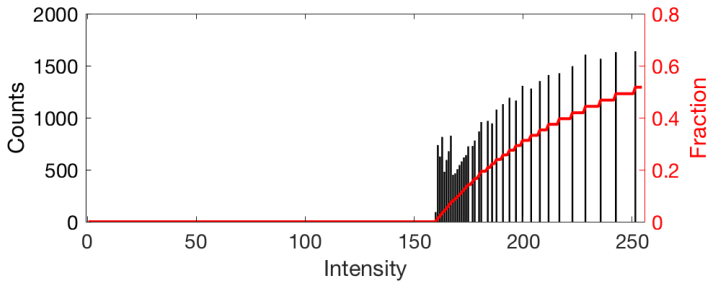
(f) Histogram of Figure 4e

Figure 4: *log* transform

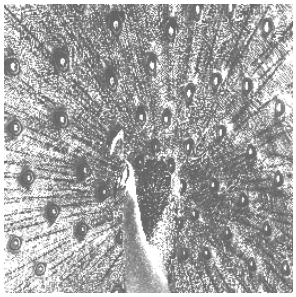
The logarithm function expend the brighter regions while compress the darker regions. As the factor increases, it darker regions are push forward to the brighter regions, causing them to get expand faster, this causes the dramatic increase both in gaps and quantities at the right-hand side of the histograms.



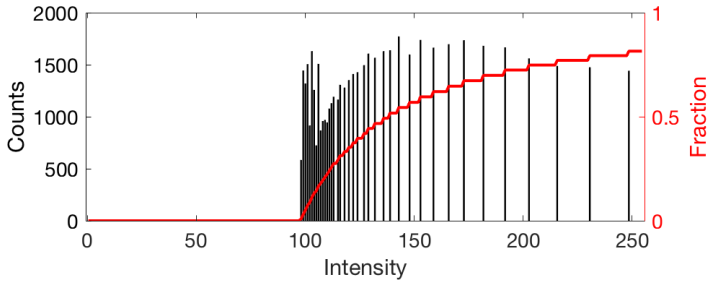
(a) $y = \frac{1}{log_2(1+2x)}$



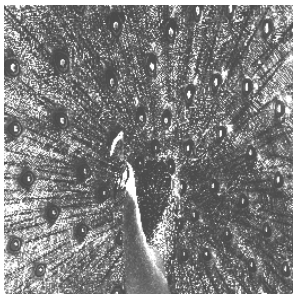
(b) Histogram of Figure 5a



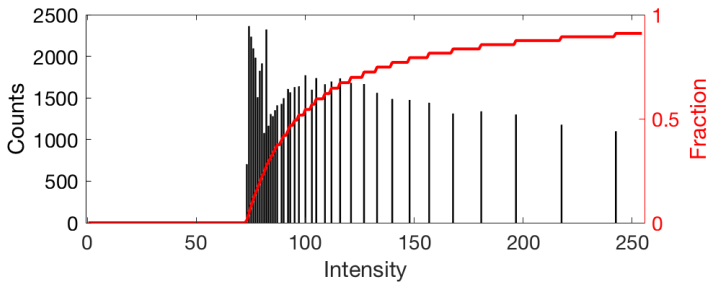
(c) $y = \frac{1}{log_2(1+5x)}$



(d) Histogram of Figure 5c



(e) $y = \frac{1}{log_2(1+10x)}$

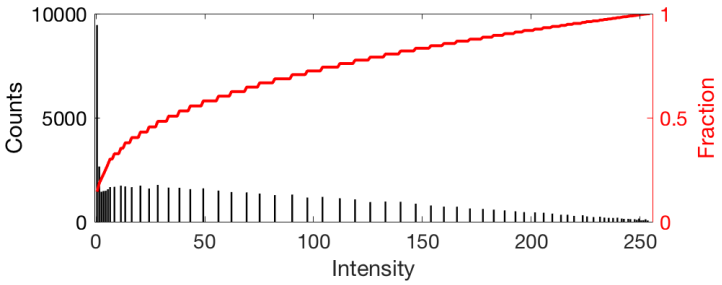


(f) Histogram of Figure 5e

Figure 5: Inverse *log* transform

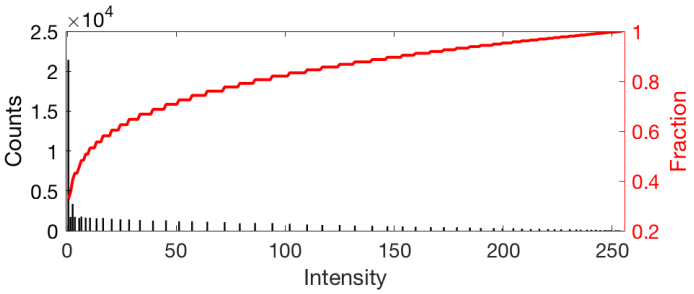
Inverse of the logarithm function turns all the bright pixels to dark one and vice versa, hence the reversed behavior in the histograms – in both intensity distribution density and gaps.

(a) $y = x^3$



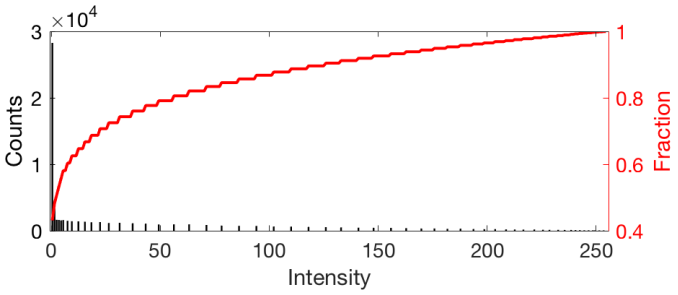
(b) Histogram of Figure 6a

(c) $y = x^5$



(d) Histogram of Figure 6c

(e) $y = x^7$



(f) Histogram of Figure 6e

Figure 6: Power-law transform

Though the power-law behaves similar to the inverse logarithm function, it is continuous at near-zero, therefore, the distribution of the histogram can go all the way to the left hand side instead of having a asymptotic line (Figure 5).

Problem 2

Noise generation

(a) $\sigma = 16$

(b) $\sigma = 32$

Figure 7: Gaussian noise

As σ increases, intensity distribution of the noise spreads wider, causing more variety of intensity values to appear in the image.

(a) threshold = 0.01

(b) threshold = 0.1

Figure 8: Salt-n-pepper noise

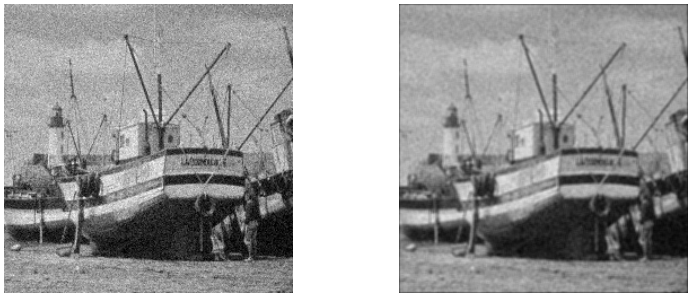
When the threshold increases, it hints that more sampled random points are allowed to show up as salt or pepper in the image, therefore, we observed that Figure 8b has higher density of the impulse noise.

Denoise and PSNR

Since Gaussian noise is uniformly distributed across entire image, using a low-pass filter (LPF) one of the methods to smooth out the flaky effect. The LPF used has $b = 1$, which is effectively an average filter

$$H = \frac{1}{m^2} \begin{bmatrix} 1_{11} & 1_{12} & \dots & 1_{1m} \\ 1_{21} & 1_{22} & \dots & 1_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1_{m1} & 1_{m2} & \dots & 1_{mm} \end{bmatrix}$$

where m denotes the kernel size.



(a) Before (b) After ($m = 3$)

Figure 9: Low-pass filter

If we increase the kernel size (m), which is currently minimal, the image only deteriorate further, illustrated in Figure 10, determined by the PSNR.

(a) $m = 3$, PSNR = 25.9 dB (b) $m = 5$, PSNR = 23.4 dB (c) $m = 7$, PSNR = 21.9 dB

Figure 10: Different kernel size.

In this implementation, images are zero-padded instead of mirror-padded, hence the surrounding of the images will gradually show black boundaries as the kernel size increases.

(a) Before (b) After

Figure 11: Median filter

Since impulse noise are localized to specific regions (in low density), using average filter to smooth them out is an overkill. The extremity nature of the salt-n-pepper noise makes it an excellent candidate for the median filter, which chooses the most average-behaved intensity to keep, yet, rest of the pixels remain untouched.

(a) $m = 3$, PSNR = 29.1 dB (b) $m = 5$, PSNR = 25.2 dB (c) $m = 7$, PSNR = 23.4 dB

Figure 12: Different kernel size.

Optimal kernel size is still minimal ($m = 3$) in this case, determined by the PSNR.

Wrinkle removal

(a) Before

(b) After

Using kernel size of 14 to suppress the fine lines, but the mermaid lines will require edge detection for better result [1].

[1] I have run out of time.