ECE637 Digital Image Processing I
Laboratory work 4:
Pointwise Operations and Gamma

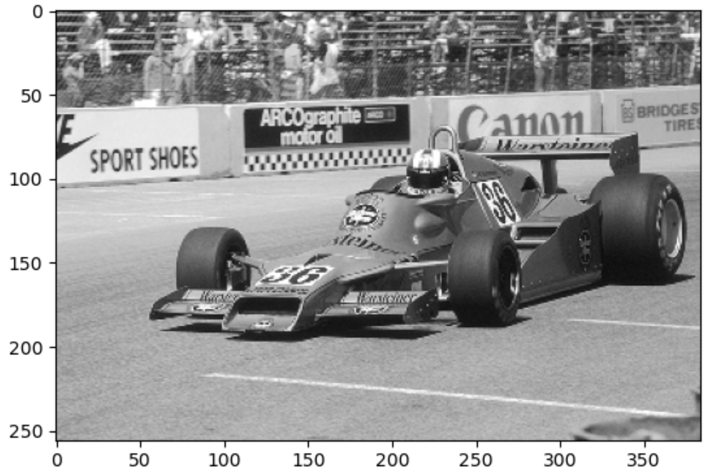
Boris Diner

February 20, 2021

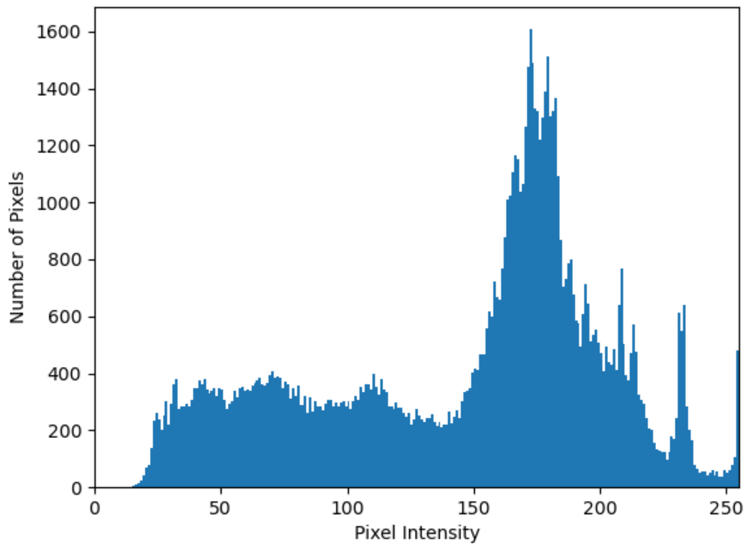
1 Histogram of an Image

In this section, we obtain histograms of images *race.tif* and *kids.tif*. The results are depicted in Figures 1 and 2

1.1 The image *race.tif* and its histogram



(a) The image *race.tif*

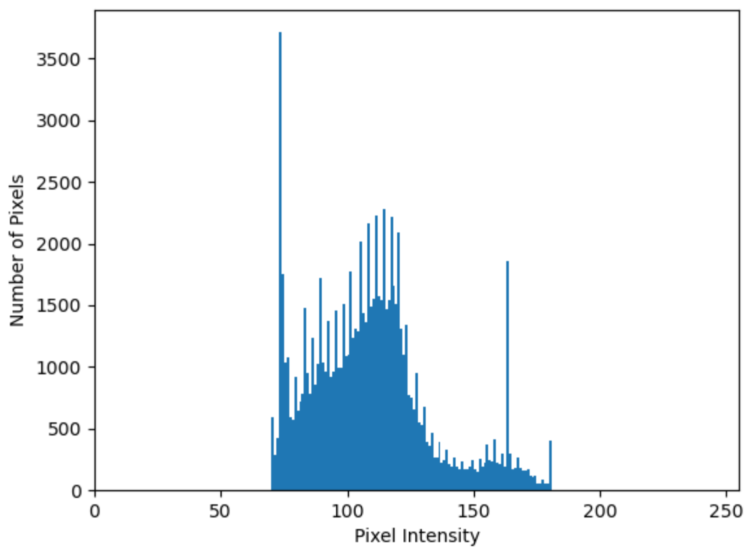


(b) The histogram of *race.tif*

Figure 1: The image *race.tif* and its histogram

1.2 The image *kids.tif* and its histogram

(a) The image *kids.tif*



(b) The histogram of *kids.tif*

Figure 2: The image *kids.tif* and its histogram

2 Histogram Equalization

In this section, the histogram of the image *kids.tif* is equalized. As we can see in 2, the histogram of this image does not span the full range of gray level values, so equalization of the histogram will allow us to enhance the quality of the image.

2.1 The CDF estimate $\hat{F}_x(i)$ for the image *kids.tif*

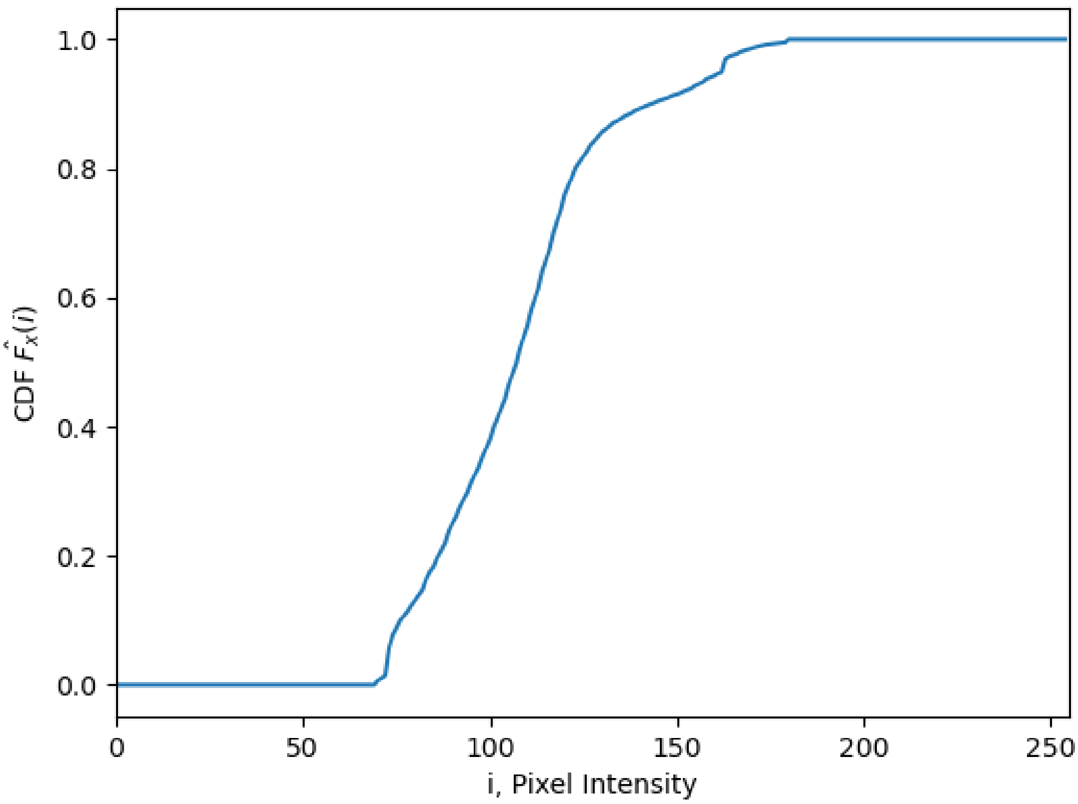
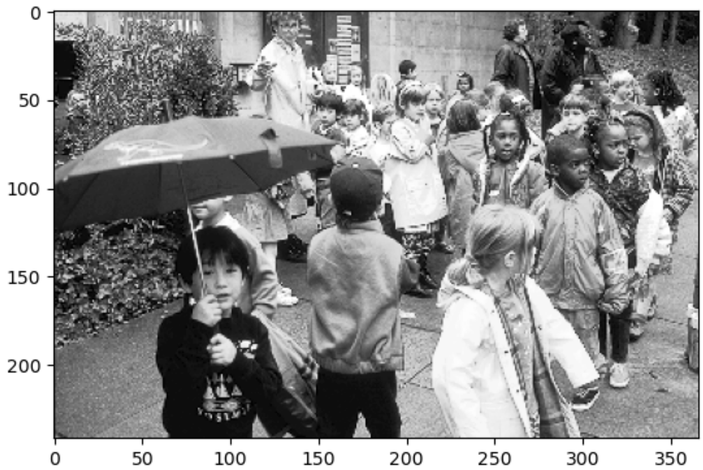
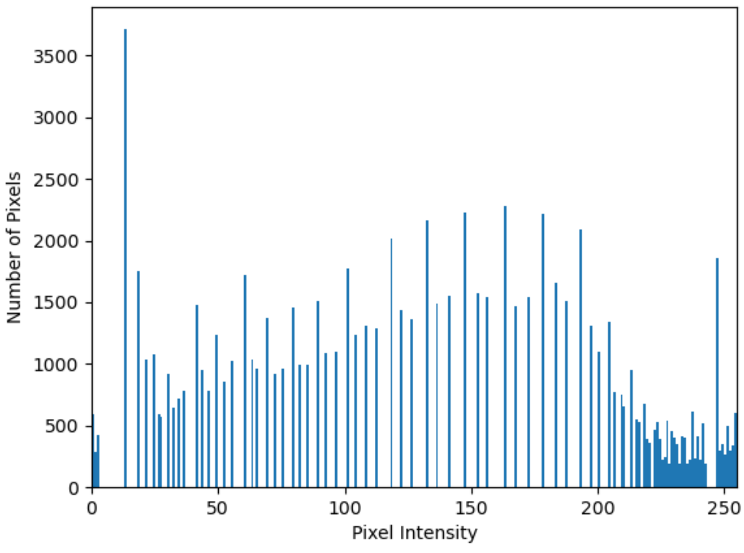


Figure 3: The CDF $\hat{F}_x(i)$ of the image *kids.tif*

2.2 The equalized image *kids.tif* and its histogram



(a) The equalized image *kids.tif*



(b) The equalized histogram of *kids.tif*

Figure 4: The image *kids.tif* and its histogram

2.3 Python code for *equalize*

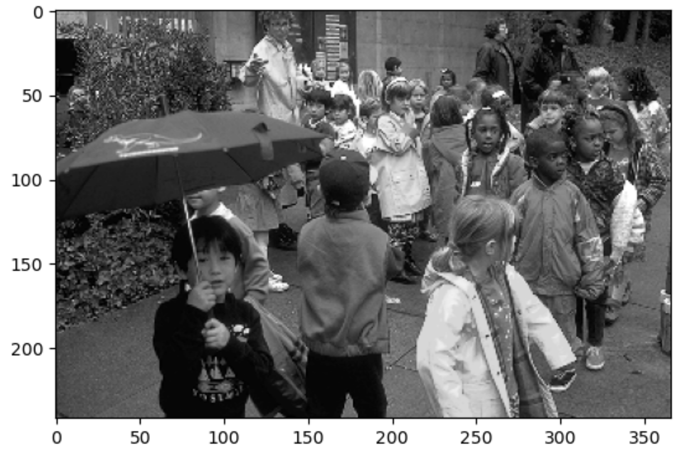
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from PIL import Image

def equalize(X):

    # Get the histogram of an image X
    hist , bins = np.histogram(X.flatten (), bins=np.linspace(0,255,
                                            256))

    # Get the estimate of the cdf of the image X
    cdf_X = np.cumsum(hist)/np.sum(hist)

    plt.figure ()
    plt.xlabel("i, Pixel Intensity")
    plt.ylabel(r'CDF $\hat{F_{x}}(i)}$')
    plt.xlim([0,255])
    plt.ylim([-0.05,1.05])
    plt.plot(cdf_X)
    plt.show()

    # Pass the image through the CDF of X
    y_s = cdf_X[X]

    # Get min and max values of the new image
    y_min = np.min(y_s)
    y_max = np.max(y_s)

    # Get the equalized image
    z = np.round(255*(y_s - y_min)/(y_max - y_min))

    return z.astype(np.uint8)

# Open an image
gray = cm.get_cmap('gray', 256)
im = Image.open('kids.tif')
x = np.array(im)

# Equalize the histogram
z = equalize(x)

# Show the equalized image
plt.imshow(z, cmap=gray , vmin=0, vmax=255)

# Show the equalized histogram
```

```
plt.figure ()
plt.xlim([0,255])
plt.xlabel("Pixel Intensity")
plt.ylabel("Number of Pixels")
plt.xlim([0,255])
plt.hist(z.flatten (),bins=np.linspace(0,255 ,256))
plt.show()
```

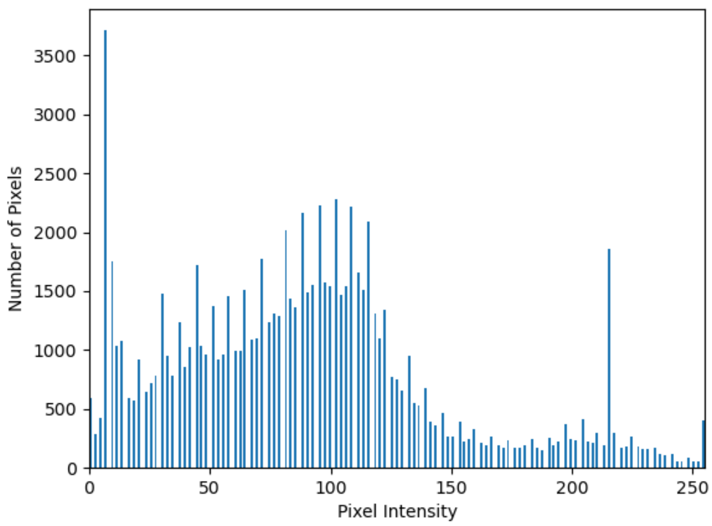
3 Contrast Stretching

In this section, the technique of contrast stretching is applied to the image *kids.tif*. Below is the result obtained for thresholds $T_1 = 70$ and $T_2 = 180$.

3.1 The transformed image ***kids.tif*** and its histogram



(a) The "stretched" image *kids.tif*



(b) The stretched histogram *kids.tif*

Figure 5: The "stretched" image *kids.tif* and its histogram

3.2 Python code for *stretch*

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from PIL import Image

def stretch(input_img, T1, T2):

    output_img = np.zeros(input_img.shape)
    row, col = input_img.shape

    for i in range(row):
        for j in range(col):
            if input_img[i,j] <= T1:
                output_img[i,j] = 0
            elif input_img[i,j] >= T2:
                output_img[i,j] = 255
            else:
                output_img[i,j] = 255 * (input_img[i,j] - T1)/(T2 -
                                                          T1)

    return output_img.astype(np.uint8)

# Open an image
gray = cm.get_cmap('gray', 256)
im = Image.open('kids.tif')
x = np.array(im)

# Apply stretching
y = stretch(x, 70, 180)

# Show the results
plt.imshow(y, cmap=gray, vmin=0, vmax=255)

plt.figure()
plt.xlim([0,255])
plt.xlabel("Pixel Intensity")
plt.ylabel("Number of Pixels")
plt.xlim([0,255])
plt.hist(y.flatten(),bins=np.linspace(0,255,256))
plt.show()
```

4 Gamma (γ)

4.1 The image corresponding to the matching gray level

For my monitor, the pattern with the gray level of 175 produced the best intensity match between the stripes.

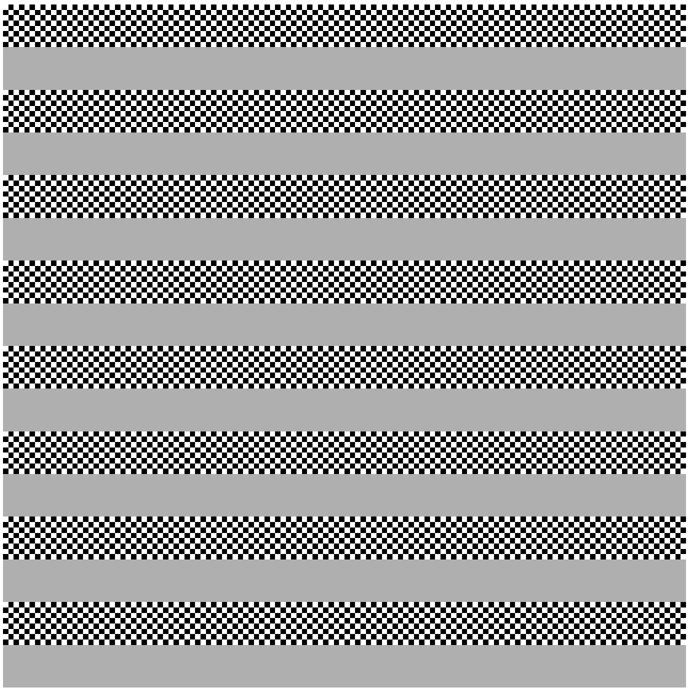


Figure 6: Array pattern for determining γ with gray level 175

4.2 Derivation of the expression which relates the matching gray level to the value of γ

We are given the following facts

$$\begin{aligned} I_c &= \frac{I_{255}}{2}, \\ I_g &= I_{255}\left(\frac{g}{255}\right)^{\gamma} \end{aligned} \tag{1}$$

To match the gray level, the following condition should hold

$$
\begin{aligned}
I_c &= I_g, \\
\frac{I_{255}}{2} &= I_{255} \left(\frac{g}{255}\right)^{\gamma}
\end{aligned}
$$

Therefore,

$$\gamma = -\frac{\log 2}{\log \frac{g}{255}}$$

For $g = 175$, we have $\gamma \approx 1.84$.

We know the relation between the pixel light intensity produced by the display y and the original pixel value x

$$y = 255 \left(\frac{x}{255}\right)^{\gamma}$$

Therefore,

$$x = 255 \left(\frac{y}{255}\right)^{\frac{1}{\gamma}}$$

(a) The original image *linear.tif* (b) The corrected image *linear.tif*

Figure 7: The original and the corrected versions of the image *linear.tif*

4.3 Python code for gamma correction of the image *linear.tif*

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from PIL import Image

gray = cm.get_cmap('gray', 256)
im = Image.open('linear.tif')
x = np.array(im)

cor_x = 255 * (np.double(x)/255) ** (1/1.84)

cor_x_uint8 = cor_x.astype(np.uint8)
cor_x_im = Image.fromarray(cor_x_uint8)
cor_x_im = cor_x_im.save('cor_linear_img.png')
```

4.4 Gamma correction for a specific monitor

We can model the process of gamma correction of an image that has already been corrected as a two stage process.

Let y be the gamma corrected image for $\gamma_1 = 1.5$. Then we know that

$$y = 255 \left(\frac{x}{255} \right)^{\frac{1}{\gamma_1}} \tag{2}$$

What we observe is the image z on the display with γ_2.

$$z = 255 \left(\frac{y}{255} \right)^{\gamma_2} \tag{3}$$

Plugging 2 in 3, we get the following correction relation

$$x = 255 \left(\frac{z}{255} \right)^{\frac{\gamma_1}{\gamma_2}}$$

The result of gamma correction ($\gamma_2 = 1.84$) of the given image *gamma15.tif* is shown in Figure 8.

(a) The original image *gamma15.tif*

(b) The corrected image *gamma15.tif*

Figure 8: The original and the corrected versions of the image *gamma15.tif*

4.5 Python code for gamma correction of the image *gamma15.tif*

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from PIL import Image

gray = cm.get_cmap('gray', 256)
im = Image.open('gamma15.tif')
x = np.array(im)

cor_x = 255 * (np.double(x)/255) ** (1.5/1.84)

cor_x_uint8 = cor_x.astype(np.uint8)
cor_x_im = Image.fromarray(cor_x_uint8)
cor_x_im = cor_x_im.save('cor_gamma15_img.png')
```