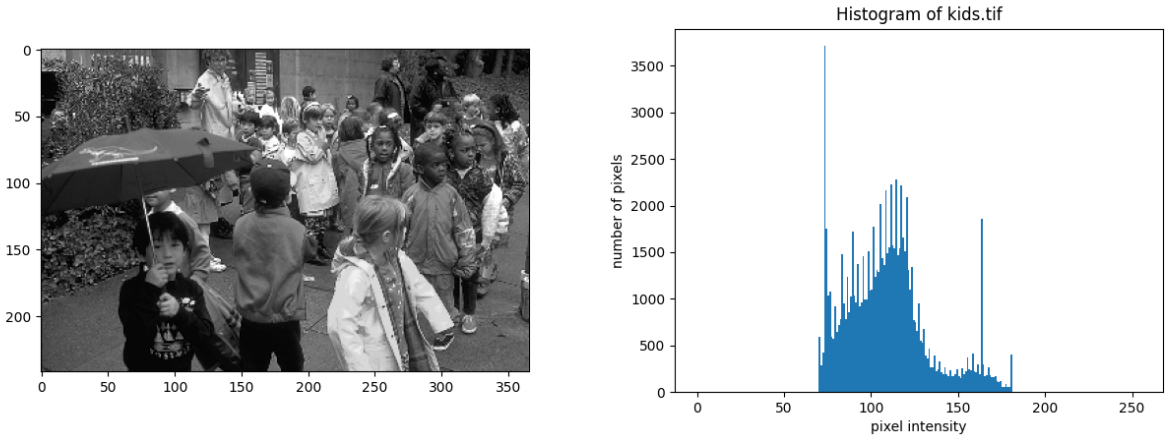
Ganga Meghanath
gmeghana@purdue.edu

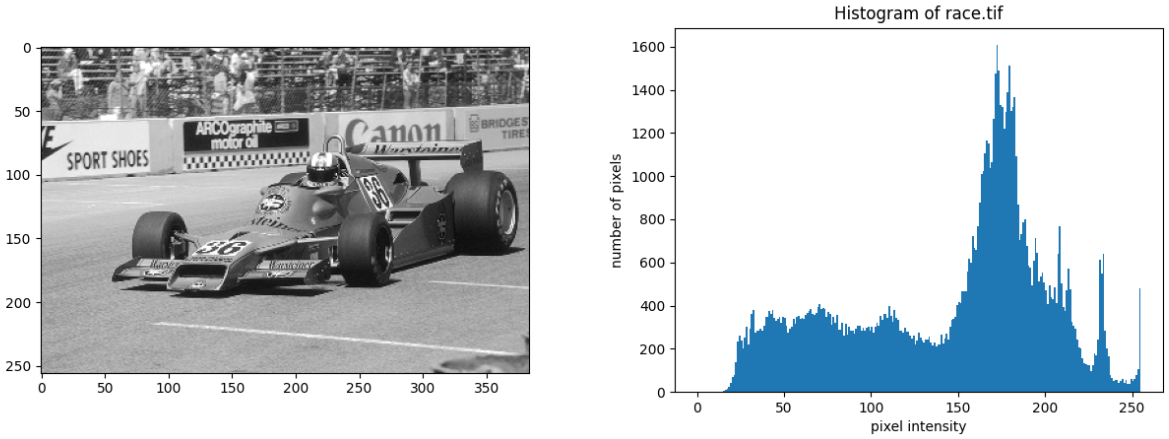
1 Histogram of an Image



(a) Original image kids.tif

(b) Histogram of kids.tif

Figure 1: Visualizing image and its histogram for kids.tif

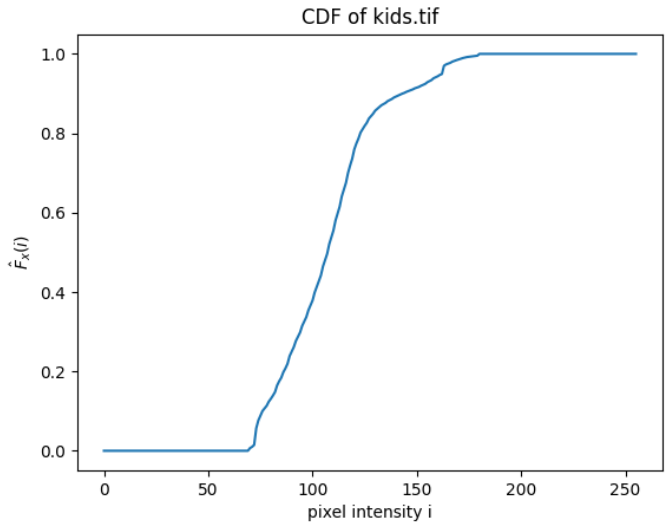


(a) Original image race.tif

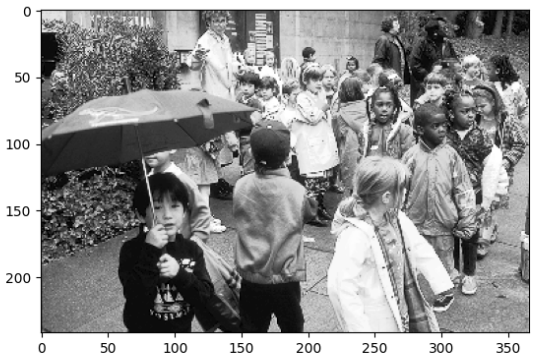
(b) Histogram of race.tif

Figure 2: Visualizing image and its histogram for race.tif

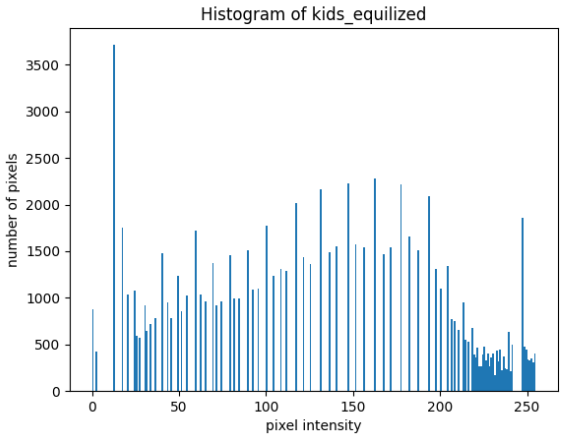
2 Histogram Equalization



(a) Cumulative distribution function $\hat{F}_x(i)$ for kids.tif



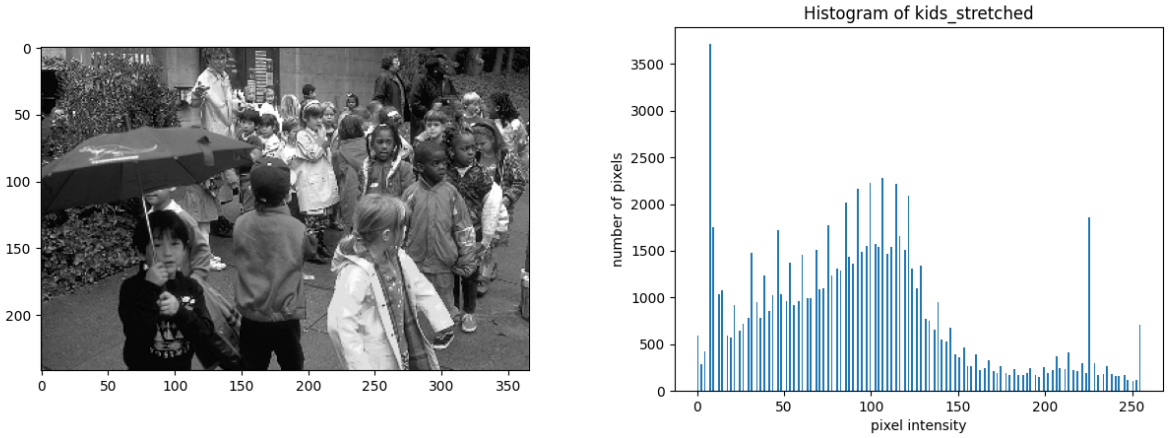
(b) Equalized image of kids.tif



(c) Histogram of the equalized image kids.tif

Figure 3: Histogram equilization on kids.tif

3 Contrast Stretching



(a) kids.tif after contrast stretching

(b) Histogram of the image after contrast stretching

Figure 4: Constrast stretching of image kids.tif using threshold values T1=70 and T2=175

4 Gamma (γ)

4.1 Setting the Black Level and Picture of Your Monitor

The Black Point of my monitor has been set as described in the procedure

4.2 Determining the Gamma of Your Computer Monitor

4.2.1 Matching gray level

The gray level of 160 produced the best intensity match between the stripes on my monitor.

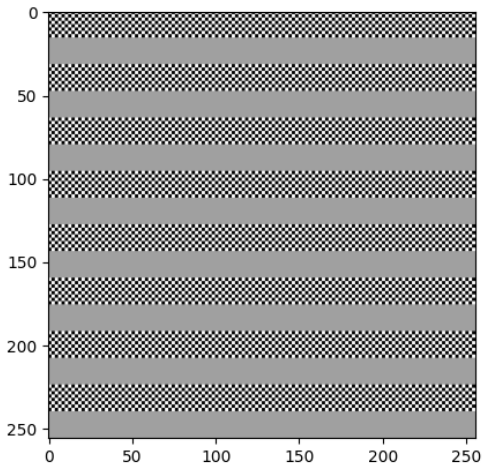


Figure 5: Gray level = 160

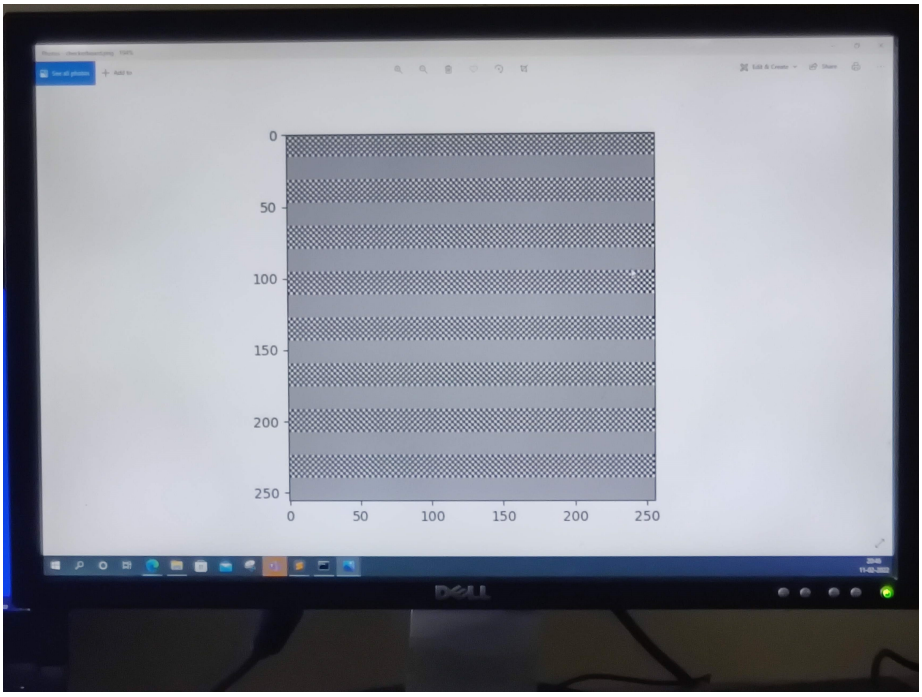


Figure 6: View of image from the monitor. If the distance is farther away, using the naked eye, the intensity between the stripes are matched for gray level of 160. The picture was taken using a smart phone camera.

4.2.2 Derive an analytical expression for γ in terms of the matching gray level

The perceived intensity of the checkerboard is given by

$$I_c = \frac{I_{255} + 0}{2}$$

Assuming a standard gamma model for the behavior of the monitor, the gray level g produces a perceived intensity of

$$I_g = I_{255}\left(\frac{g}{255}\right)^{\gamma}$$

We can then calculate the value of γ for the monitor by determining the gray level g which makes

$$I_g = I_c$$

$$ie, \quad I_{255}\left(\frac{g}{255}\right)^{\gamma} = \frac{I_{255}}{2}$$

$$ie., \quad \gamma \log\left(\frac{g}{255}\right) = \log\left(\frac{1}{2}\right)$$

$$\text{Therefore,} \quad \gamma = -\frac{\log 2}{\log\left(\frac{g}{255}\right)}$$

4.2.3 Measured gray level and the measured γ

$$\text{Measured gray level} = 160$$

$$\text{Measured } \gamma = -\frac{\log 2}{\log\left(\frac{160}{255}\right)} \approx 1.487$$

4.3 Gamma Correction

4.3.1 Derive the expression for gamma correction

We know that,

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

where x is the original pixel value, and y is the pixel light intensity produced by the display. Hence, to compensate for this effect, we gamma correct the image using the inverse of the above equation.
ie.,

$$
\begin{aligned}
y &= 255\left(\frac{x}{255}\right)^{\gamma} \\
\implies \frac{y}{255} &= \left(\frac{x}{255}\right)^{\gamma} \\
\implies \frac{x}{255} &= \left(\frac{y}{255}\right)^{\frac{1}{\gamma}} \\
\implies x &= 255\left(\frac{y}{255}\right)^{\frac{1}{\gamma}}
\end{aligned}
$$

4.3.2 Gamma Corrected Image

(a) Original linear.tif

(b) Gamma corrected for $\gamma = 1.487$

Figure 7: Comparison of image stored with a linear scaling and corrected version of the image using determined γ value of the monitor.

Figure 8: The gamma corrected image as seen on the computer monitor photographed by mobile phone. I've tried to capture the picture as close to the one observed by naked eye.

4.4 Derive the expression for gamma correction for a gamma corrected image

If the image is already gamma corrected using γ_1, then our input image is,

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

Therefore, the original image pixel value y is given by

$$y = 255\Big(\frac{x}{255}\Big)^{\gamma_1}$$

To gamma correct the image for our display monitor, we use gamma correction using γ_2 which was determined by us previously as 1.487. Let the corrected image be z.
ie.,

$$\begin{aligned} z &= 255\Big(\frac{y}{255}\Big)^{\frac{1}{\gamma_2}} \\ &= 255\left(\frac{255\big(\frac{x}{255}\big)^{\gamma_1}}{255}\right)^{\frac{1}{\gamma_2}} \\ &= 255\Big(\frac{x}{255}\Big)^{\frac{\gamma_1}{\gamma_2}} \end{aligned}$$

Hence, resultant γ value is given by,

$$\gamma = \frac{\gamma_1}{\gamma_2}$$

Substituting $\gamma_1 = 1.5$ and $\gamma_2 = 1.487$ gives us

$$\gamma \approx 1.009$$

4.5 Gamma corrected image (taking into account prior correction)

(a) Original gamma15.tif

(b) Gamma corrected for $\gamma = 1.009$

Figure 9: Comparison of originally gamma corrected image and corrected version of the image using determined γ value of the monitor.

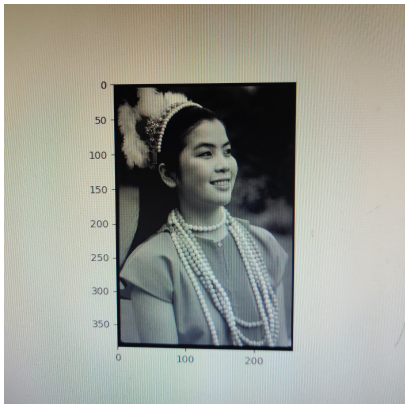


Figure 10: The gamma corrected image as seen on the computer monitor photographed by mobile phone. I've tried to capture the picture as close to the one observed by naked eye.

5 Code snippets

5.1 histogram.py

```
import numpy as np
import matplotlib.pyplot as plt

from PIL import Image


def read_img(img_file):
  im = Image.open(img_file)
  x = np.array(im)
  return x

def display_gray_img(img_file, out_dir):
  x = read_img(img_file)
  plt.clf()
  plt.imshow(x, cmap="gray")
  plt.savefig(out_dir + img_file.split("\\")[-1].strip(".tif")+".png")

def get_histogram(x):
  bins = np.zeros(256)
  for pixel in x.flatten():
    bins[pixel] += 1
  return bins

def plot_histogram(x, img_file, out_dir):
  plt.clf()
  plt.hist(x.flatten(), bins=np.linspace(0,255,256))
  plt.title("Histogram of {}".format(img_file.split("\\")[-1]))
  plt.xlabel("pixel intensity")
  plt.ylabel("number of pixels")
  plt.savefig(out_dir + img_file.split("\\")[-1].strip(".tif")+"_histogram.
   png")

def plot_cdf(F, img_file, out_dir):
  plt.clf()
  plt.plot(np.arange(256), F)
  plt.title("CDF of {}".format(img_file.split("\\")[-1]))
  plt.xlabel("pixel intensity i")
  plt.ylabel(r"$\^{F}_x(i)$")
  plt.savefig(out_dir + img_file.split("\\")[-1].strip(".tif")+"_CDF.png")

def equalize(img_file, out_dir):
  x = read_img(img_file)
  h = get_histogram(x)

  Fx = np.cumsum(h)/sum(h)

  Y = Fx[x]

  Y_max = Fx[np.max(x)]
  Y_min = Fx[np.min(x)]

  Z = 255 * (Y - Y_min)/(Y_max - Y_min)

  plot_cdf(Fx, img_file, out_dir)

  # Save equilized Image Z
  plt.clf()
```

```
  plt.imshow(Z, cmap="gray")
  plt.savefig(out_dir + img_file.split("\\")[-1].strip(".tif")+"_equilized.
   png")

  # Plot histogram of equilized image
  plot_histogram(Z.astype(int), img_file.split("\\")[-1].strip(".tif")+"
   _equilized", out_dir)

def stretch(img_file, out_dir, T1, T2):
  x = read_img(img_file)

  for i in range(x.shape[0]):
    for j in range(x.shape[1]):
      if x[i, j]<=T1:
        x[i, j]=0
      elif x[i, j]>=T2:
        x[i, j]=255
      else:
        x[i, j] = 255 * (x[i, j]-T1)/(T2-T1)

  # Save stretched Image x
  plt.clf()
  plt.imshow(x, cmap="gray")
  plt.savefig(out_dir + img_file.split("\\")[-1].strip(".tif")+"_stretched.
   png")

  # Plot histogram of stretched image
  plot_histogram(x.astype(int), img_file.split("\\")[-1].strip(".tif")+"
   _stretched", out_dir)

def create_checkerboard_pattern(out_dir, h, w, gray=160, stripe=16, block
   =4):
  img = np.ones((h, w))*gray # 160
  for s in range(0, h, stripe):
    if (s//stripe)%2==0:
      for j in range(s, s+stripe, block):
          # initialize block rows as black
          img[j:j+block] = 0
          for k in range(0, block):
            skip = k//(block//2)*(block//2)
            # Modify every 4th (size of block) element from skip and skip+1
    as white for row j+k
            img[j+k, skip::block] = 255
            img[j+k, skip+1::block] = 255
  # print(img)
  plt.clf()
  plt.imshow(img, cmap="gray")
  plt.savefig(out_dir +"checkerboard.png")

def gamma_correct_img(img_file, out_dir, gamma=1.487):
  y = read_img(img_file)

  # Gamma correction
  x = 255 * ((y/255)**(1/gamma))

  # Save Gamma corrected Image x
  plt.clf()
  plt.imshow(x, cmap="gray")
  plt.savefig(out_dir + img_file.split("\\")[-1].strip(".tif")+"
   _gamma_corrected.png")
```

Listing 1: histogram.py : Contains all the functions required for creating histograms

5.2 main.py

```
from histogram import *

import sys

def main(img_file, out_dir, choice):
  if choice=="histogram":
    x = read_img(img_file)
    display_gray_img(img_file, out_dir)
    plot_histogram(x, img_file, out_dir)
  elif choice=="equilize":
    equalize(img_file, out_dir)
  elif choice=="stretch":
    stretch(img_file, out_dir, T1=70, T2=175)
  elif choice=="gamma":
    create_checkerboard_pattern(out_dir, 256, 256, gray=160, stripe=16,
   block=4)
    display_gray_img(img_file, out_dir)
    gamma_correct_img(img_file, out_dir, gamma=sys.argv[3])


if __name__=="__main__":
  img_file = "..\\images\\" + sys.argv[1]
  out_dir = "..\\output\\"
  choice = sys.argv[2]


  main(img_file, out_dir, choice)
```

Listing 2: main.py : Contains the main function that calls histogram.py

5.3 run.sh

```
# ------- Qn 1 : Histogram of an Image ----------
python main.py kids.tif histogram
python main.py race.tif histogram

# ------- Qn 2 : Histogram Equalization ----------
python main.py kids.tif equilize

# ------- Qn 3 : Contrast Stretching ----------
python main.py kids.tif stretch

# ------- Qn 4 : Gamma Correction ----------
python main.py linear.tif gamma 1.487
python main.py gamma15.tif gamma 1.009
```

Listing 3: run.sh : Generates all outputs in the report