ECE 637: Lab 4

Rahul Deshmukh

February 19, 2021

Section 1 Report

1. Print out of kids.tif

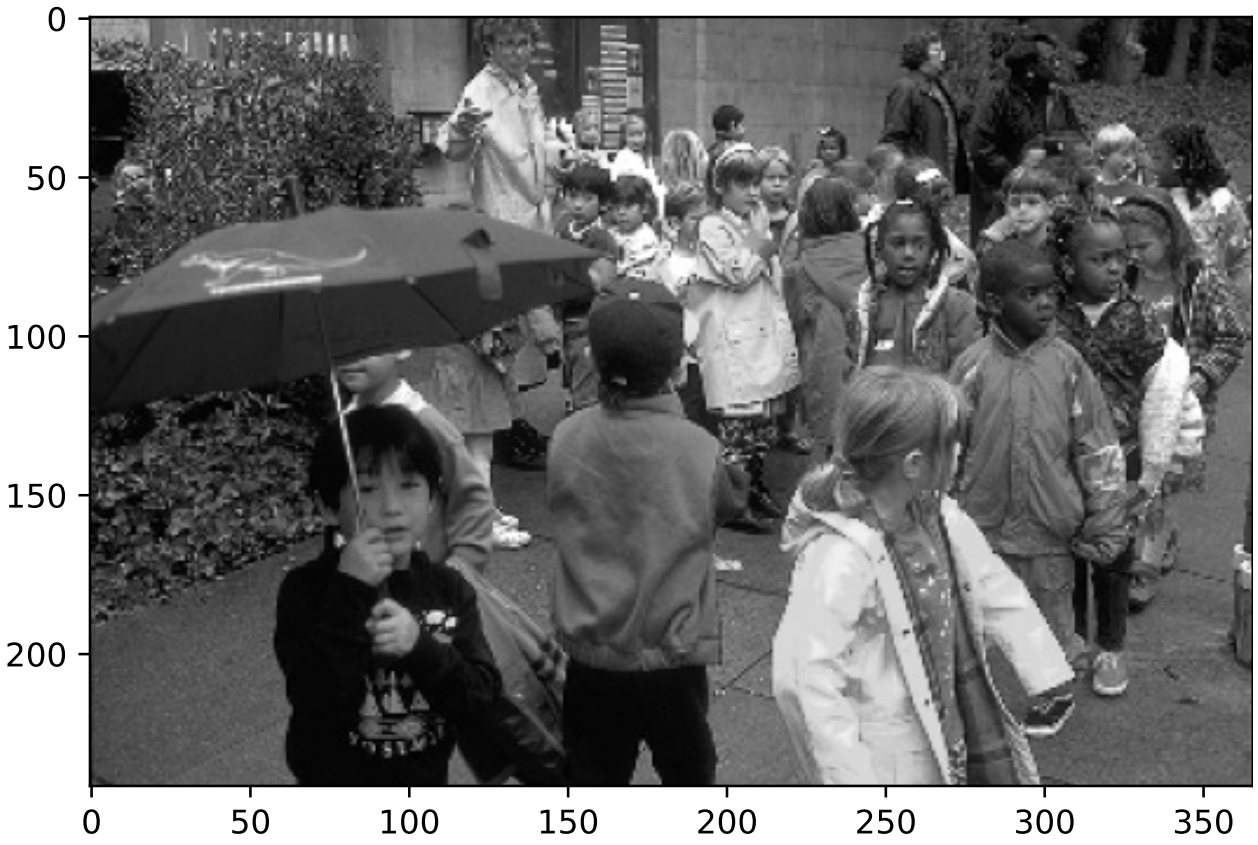


Figure 1: kids.tif as gray scale image

2. Print out of histogram of kids.tif

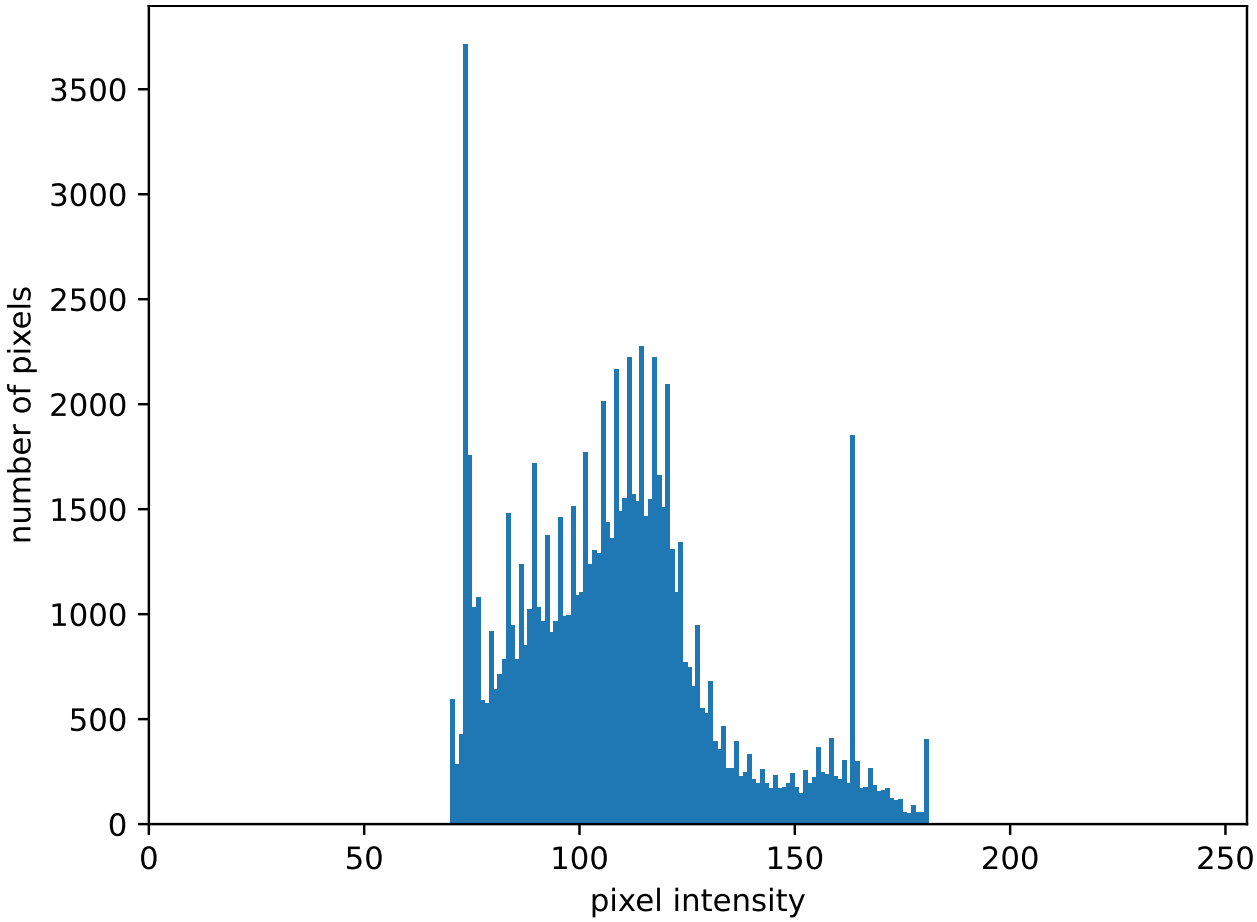


Figure 2: Histogram of kids.tif

3. Print out of race.tif

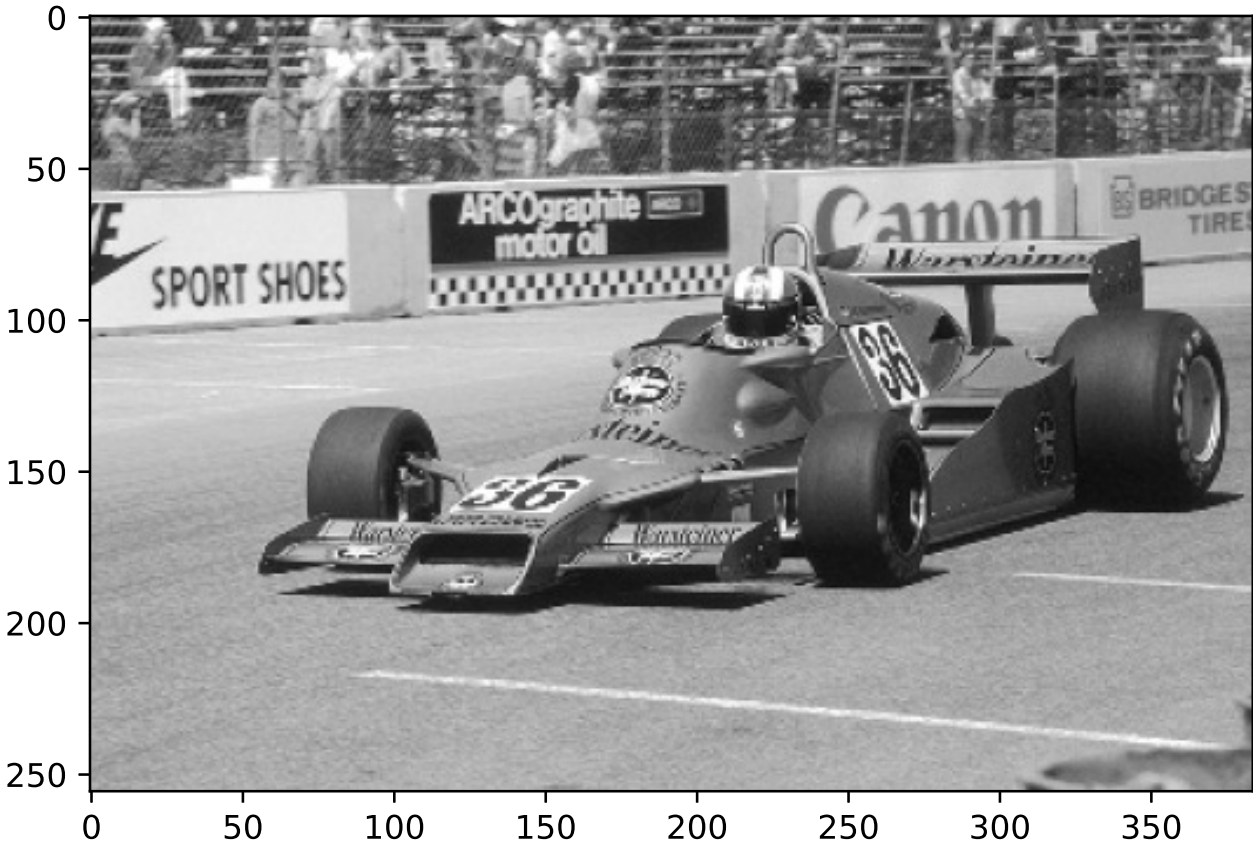


Figure 3: race.tif as gray scale image

4. Print out of histogram of race.tif

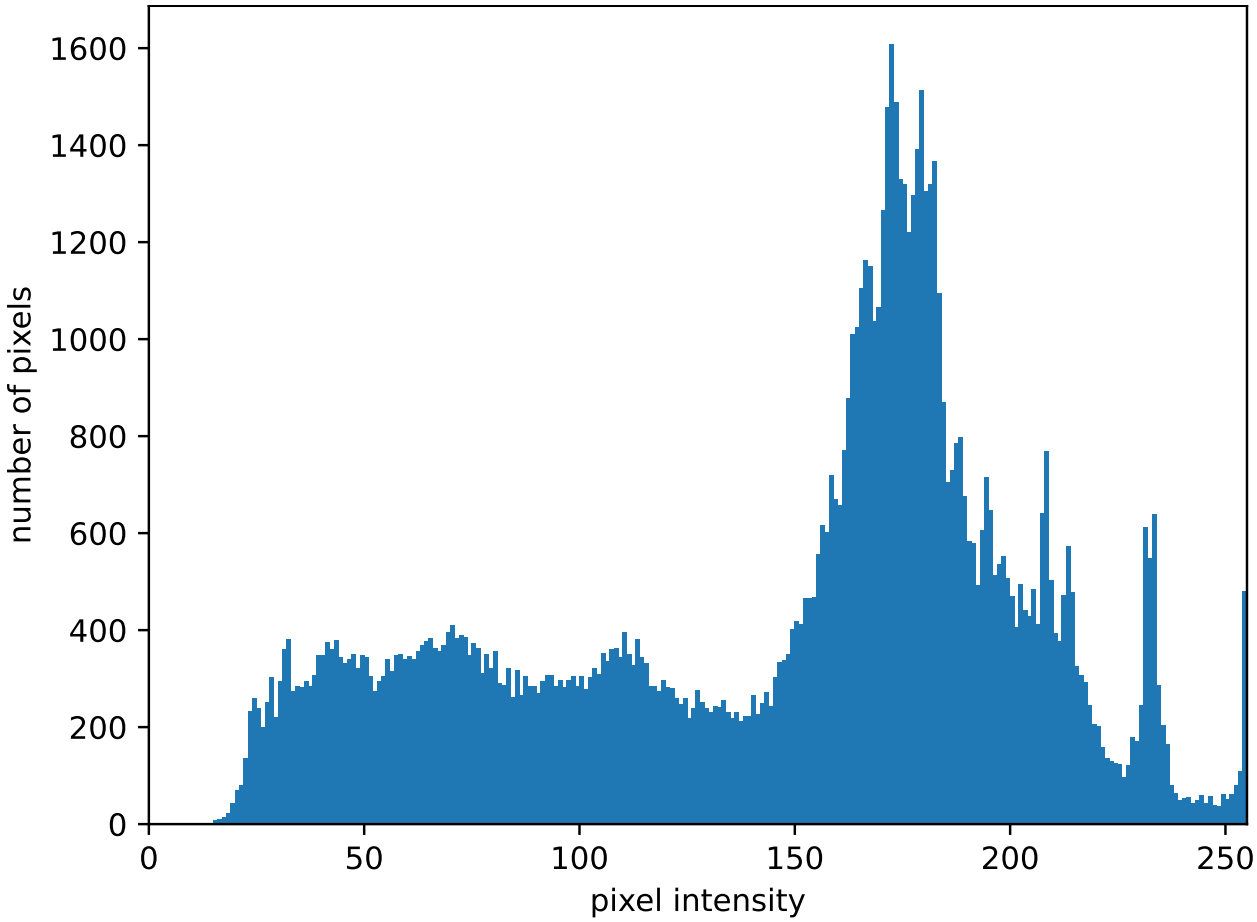


Figure 4: Histogram of race.tif

5. For python code refer to Listing 3 at page 15.

Section 2 Report

1. For python code refer to Listing 1 at page 13.
2. Plot of $\hat{F}_x(i)$ for *kids.tif*

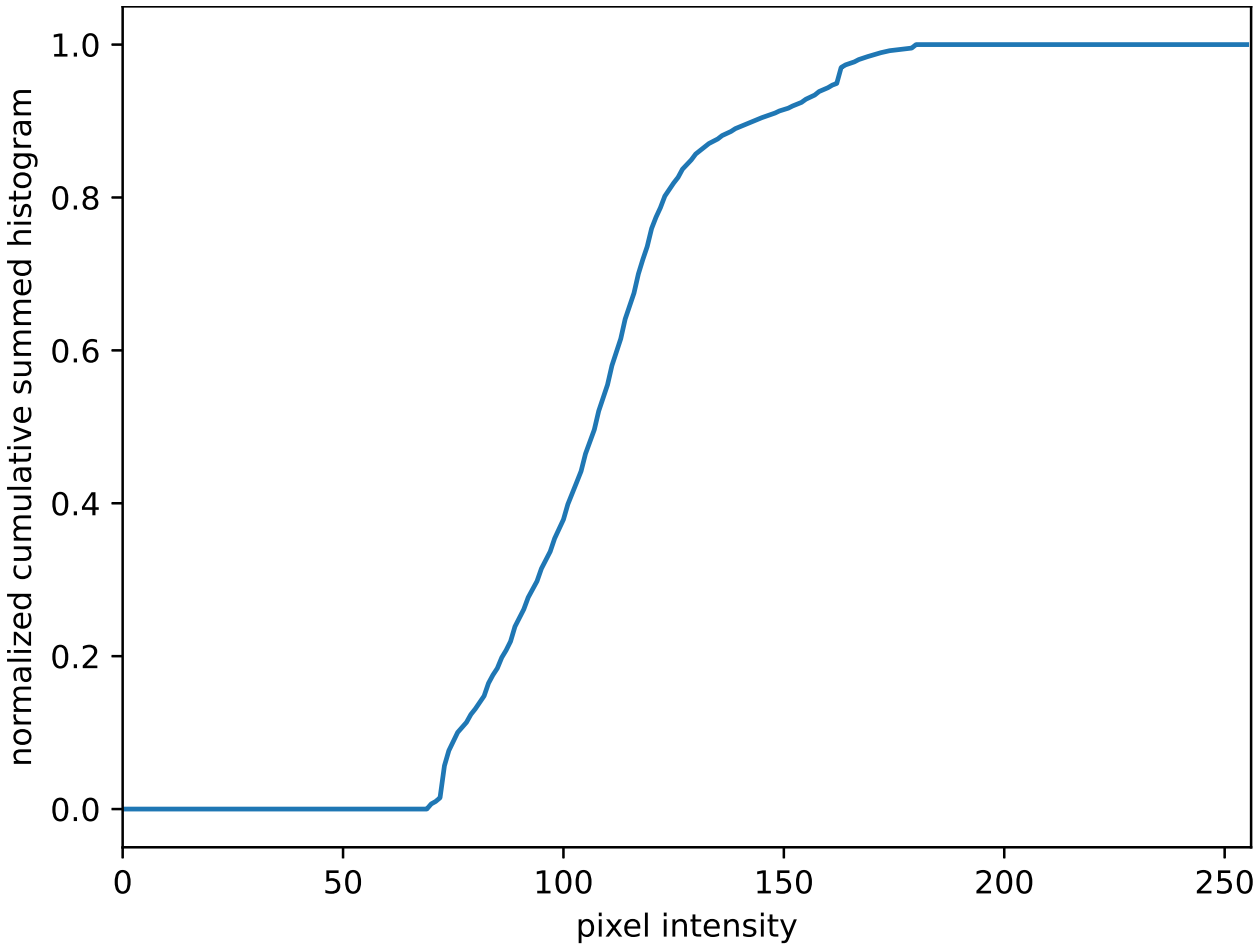


Figure 5: $\hat{F}_x(i)$ for *kids.tif*

3. Plot of equalized image's histogram

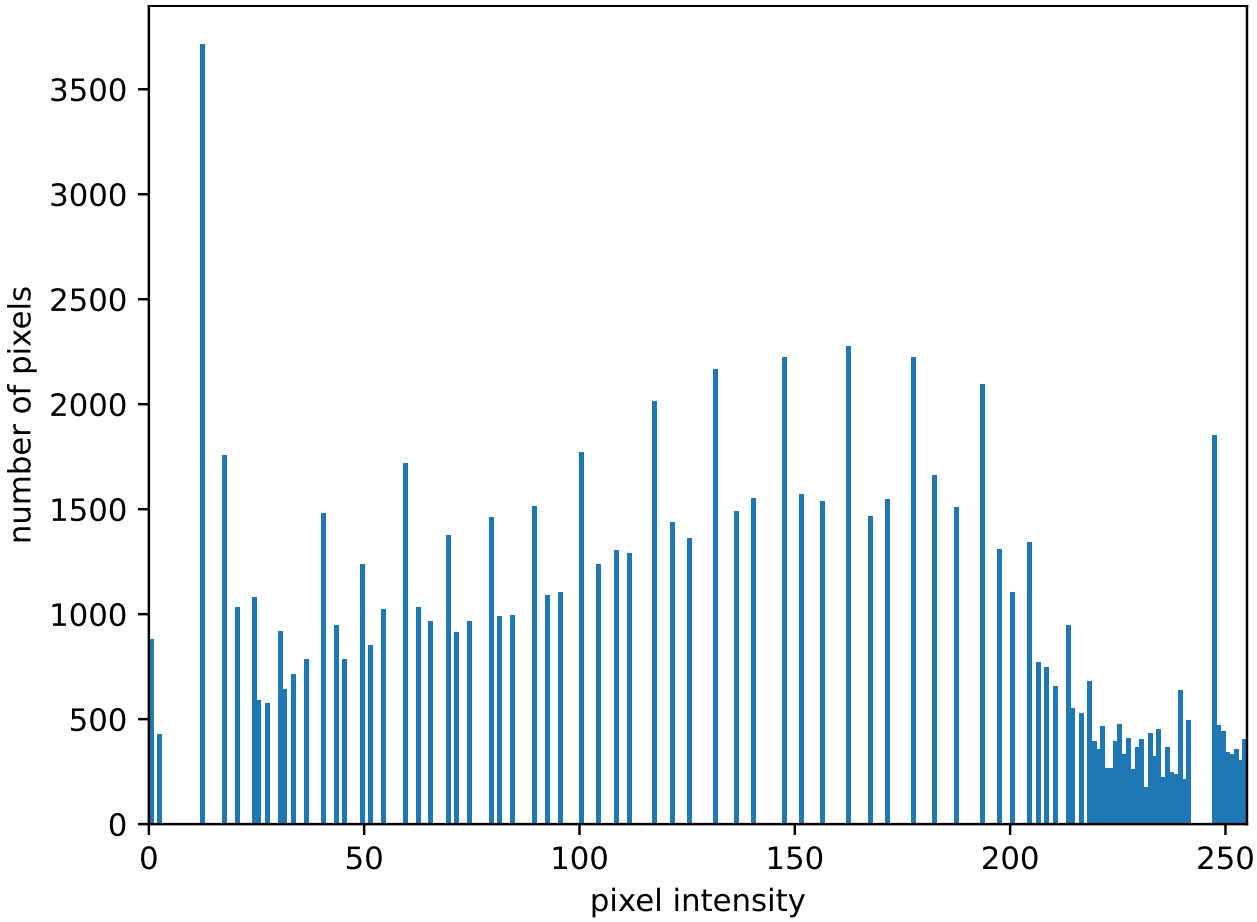


Figure 6: equalized image's histogram for *kids.tif*

4. Equalized image

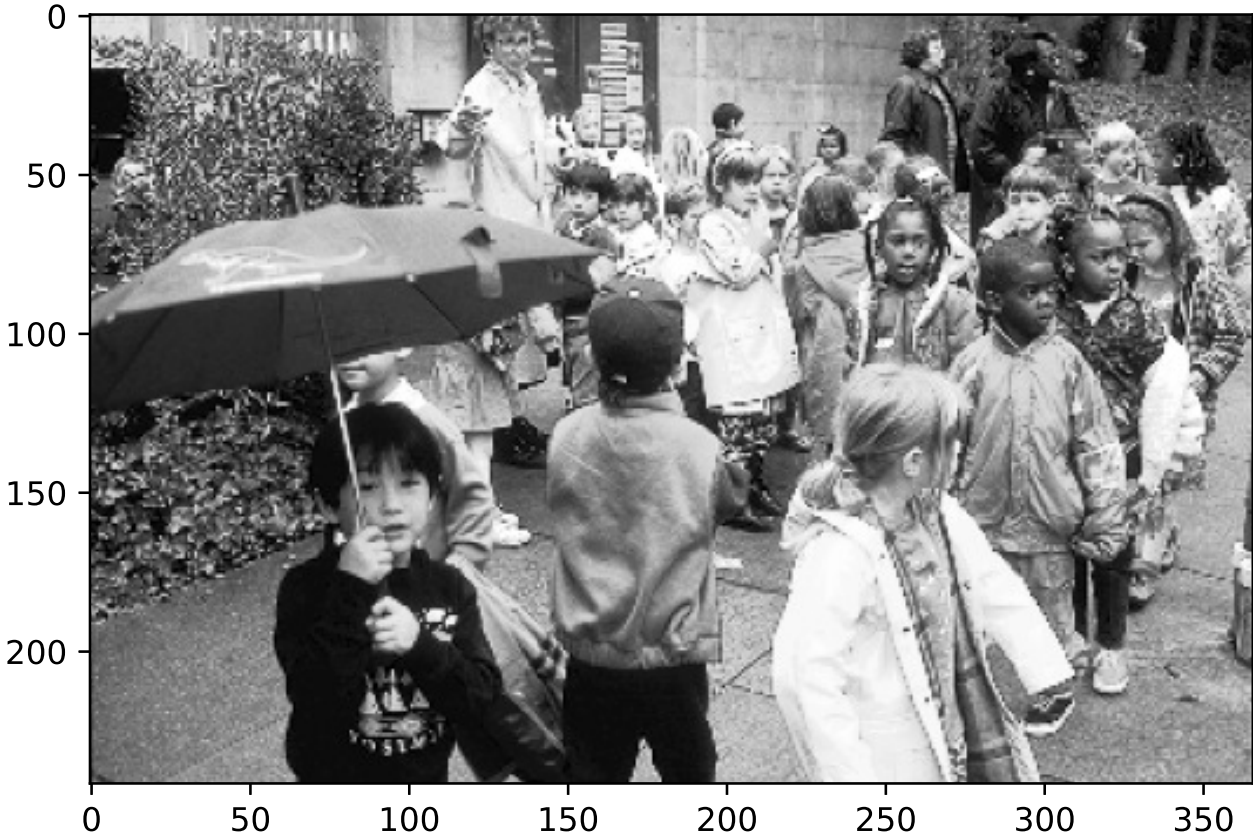


Figure 7: equalized image's histogram for *kids.tif*

Section 3 Report

1. For python code refer to Listing 1 at page 13.

2. Plot of transformed image and its histogram for *kids.tif* with $T1 = 80, \quad T2 = 160$

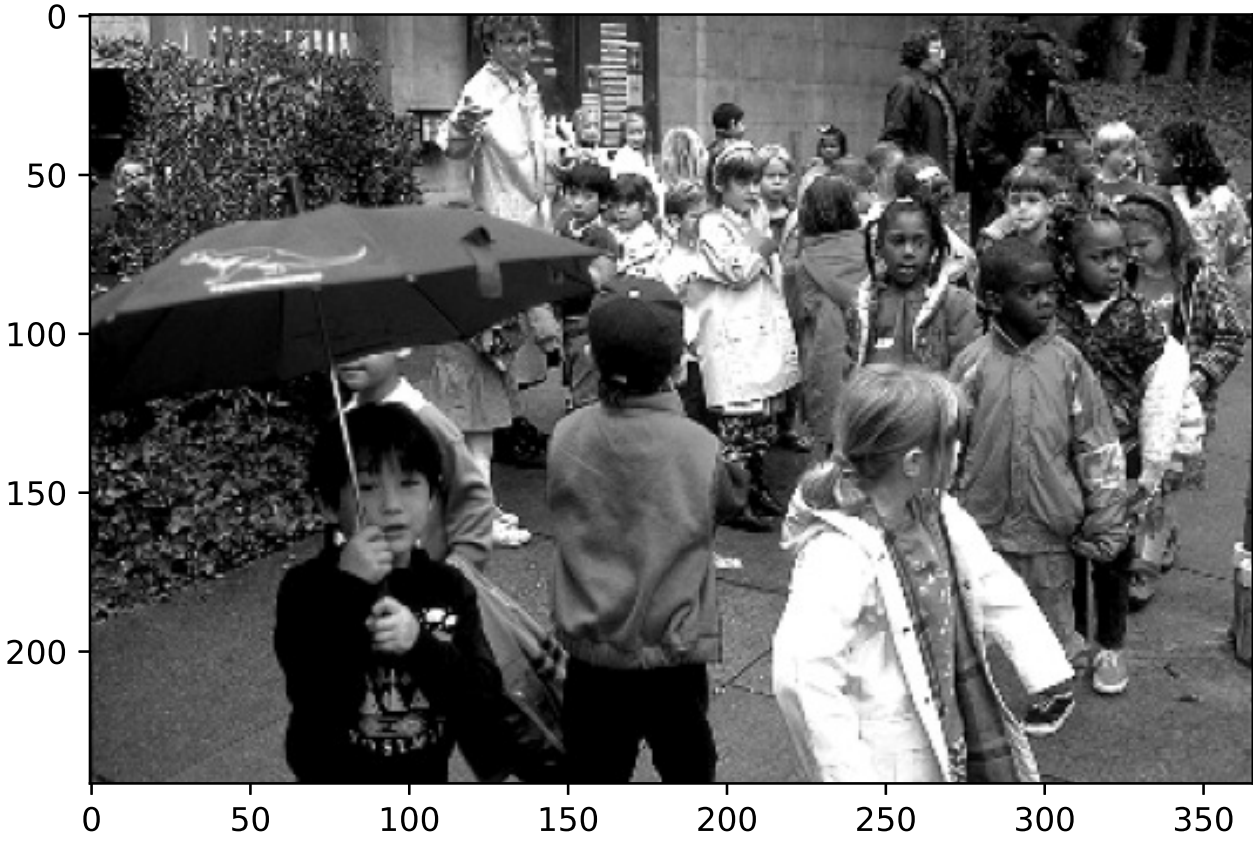


Figure 8: Contrast stretched image for *kids.tif*

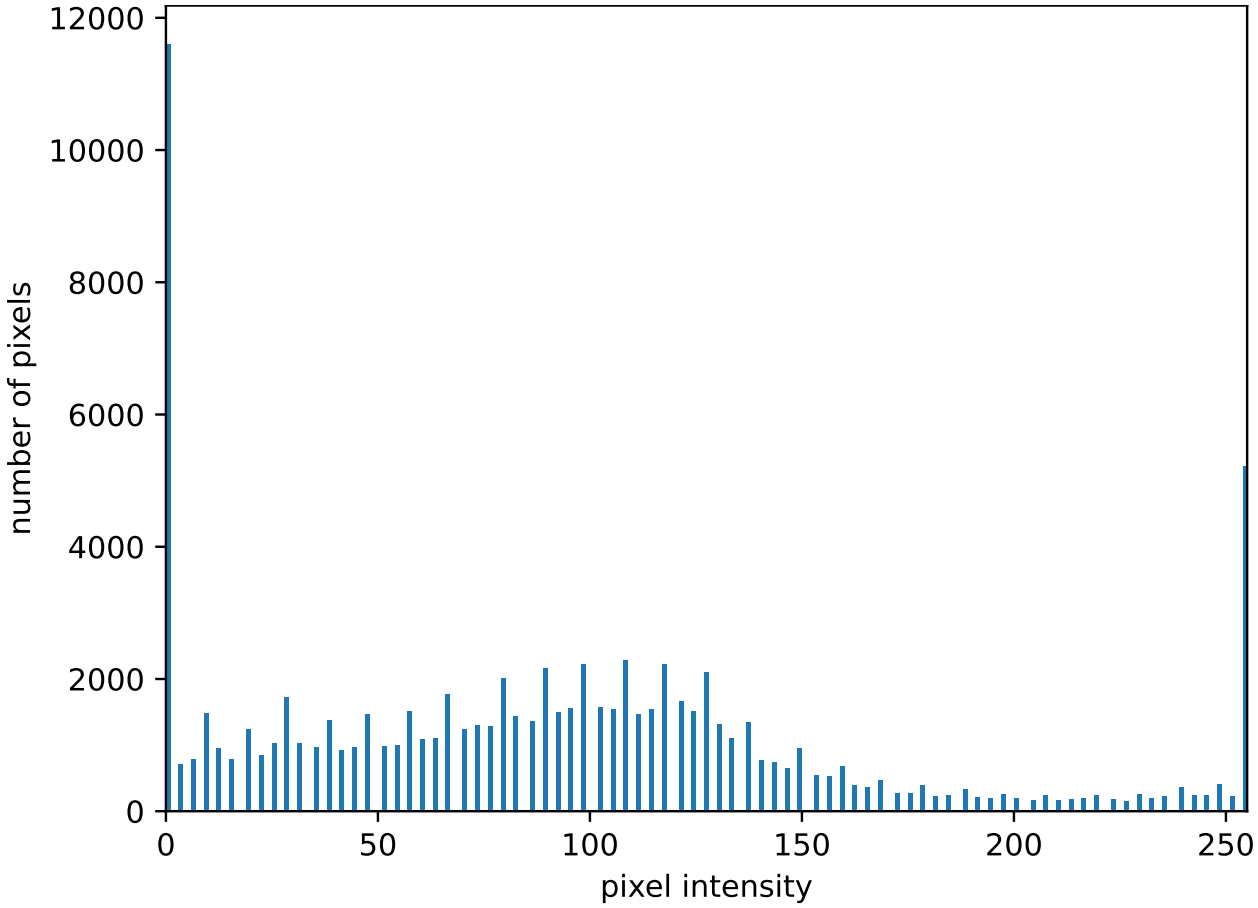


Figure 9: Histogram of contrast stretched image for *kids.tif*

Section 4.2 Report

1. Image corresponding to the matching gray level

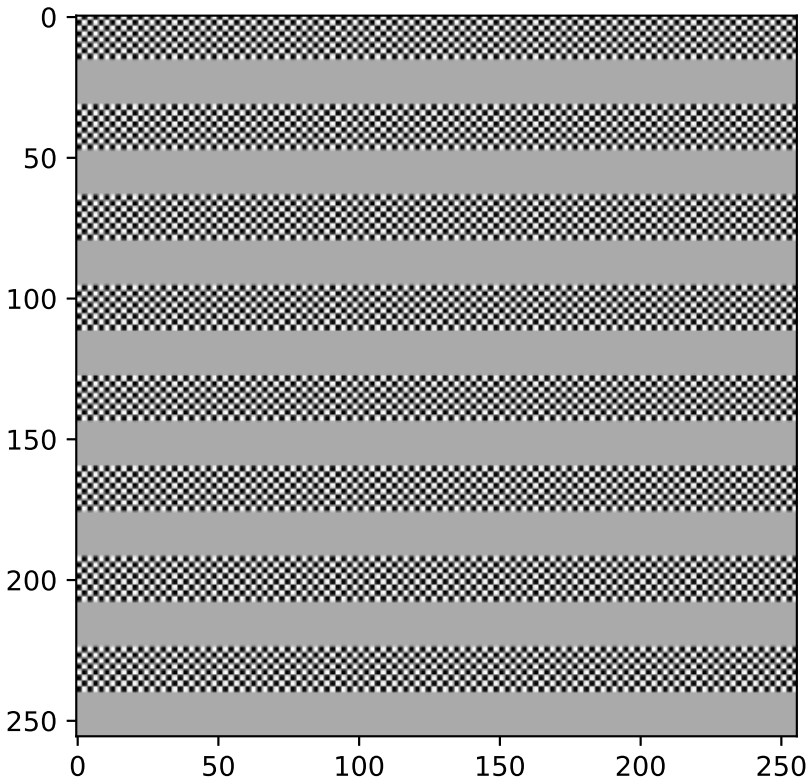


Figure 10: Array pattern corresponding to matching gray level

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

   For matching gray level we have the following equation:

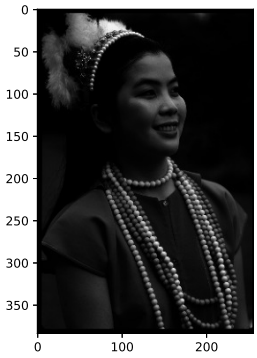
$$\begin{aligned} I_c &= I_g \\ \Rightarrow \frac{I_{255}}{2} &= I_{255}(\frac{g}{255})^{\gamma} \\ \Rightarrow g &= \frac{255}{\gamma} log(0.5) \end{aligned} \tag{1}$$

   Using Eq. 1 we can calculte the matching gray level.

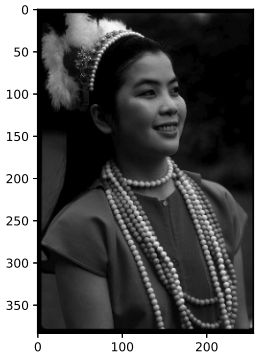
3. The measured gray level was $g = 170$ and measured $\gamma = 1.709511$. (For more details refer to the python code at Listing 4 at page 15 and bash script at Listing 6 at page 17 with corresponding output log at Listing 7 at page 18)

Section 4.3.1 Report

1. Original and corrected images *linear.tif*



(a) Original image



(b) Corrected image with $\gamma = 1.709511$

Figure 11: plots for linear.tif

2. Formula used to transform the original image

For CRT we have the following transformation:

$$y(x) = 255(\frac{x}{255})^{\gamma} \tag{2}$$

We want to find correction function $f(x)$ such that $y(f(x)) = x$

$$y(x) = 255(\frac{f(x)}{255})^{\gamma} = x$$
$$\Rightarrow \gamma log(\frac{f(x)}{255}) = log(\frac{x}{255})$$
$$\Rightarrow f(x) = 255e^{\frac{1}{\gamma}log(\frac{x}{255})} \tag{3}$$

Using Eq. 3 we can transform the original image for gamma correction.

Section 4.3.2 Report

1. Original and corrected images for *gamma15.tif*

(a) Original image with $\gamma = 1.5$ (b) Corrected image with $\gamma = 1.709511$

Figure 12: plots for gamma15.tif

2. Procedure used to change gamma correction of the original image, for each pixel value do the following:

   (i) Convert γ_1 image (x) to linear image (y) using Eq. 2

   (ii) Do gamma correction with γ_2 using Eq. 3

   For implementation details refer to Listing 5 at page 16

Source Code

Listing 1: Python code for histogram equalization of image

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab4- section 2 histogram eq
"""
import sys,os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

Nbins=256

def equalize(X,L=256):
    h,w = X.shape
    x_hist,_ = np.histogram(X.flatten(),bins=np.arange(L+1))
    cumsum_norm = np.cumsum(x_hist).astype(np.float32)
    cumsum_norm /= cumsum_norm[-1]
    Y = np.zeros_like(X).astype(np.float32)
    for i in range(h):
        for j in range(w):
            Y[i][j]  = cumsum_norm[X[i][j]]
    #normalize
    y_max = np.max(Y)
    y_min = np.min(Y)
    Z = (L-1.0)*(Y-y_min)/(y_max-y_min)
    return Z,cumsum_norm

def main(filename):
    basename = os.path.basename(filename).split('.')[0]
    im = Image.open(filename)
    img = np.array(im)
    img_hist_eq, cumsum_norm = equalize(img,L=Nbins)

    #save image
    im_save = Image.fromarray(img_hist_eq)
    im_save.save(basename+'_hist_eq.tif')

    #plot
    gray = mpl.cm.get_cmap('gray',256)
    plt.figure(1)
    plt.imshow(img_hist_eq,cmap=gray)
    plt.savefig(basename+'_hist_eq.eps',format='eps',
                bbox_inches='tight', pad_inches = 0)

    plt.figure(2)
    plt.plot(np.arange(Nbins),cumsum_norm)
    plt.xlabel('pixel intensity')
    plt.ylabel('normalized cumulative summed histogram')
    plt.xlim([0,Nbins])
    plt.savefig(basename+'_cumsum_norm.eps',format='eps',
                bbox_inches='tight', pad_inches = 0)

```

```
if __name__=="__main__":
    filename = sys.argv[1]
    main(filename)
```

Listing 2: Python code for contrast stretching of image

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab4- section 3 contrast stretching
"""
import sys,os
from PIL import Image
import numpy as np

Nbins=256

def stretch(X,T1,T2):
    h,w = X.shape
    slope = (255.)/(T2-T1)
    Y = np.zeros_like(X).astype(np.float32)
    for i in range(h):
        for j in range(w):
            if(X[i][j]<=T1):
                Y[i][j]=0.
            elif(X[i][j]>=T2):
                Y[i][j]=255.
            else:
                Y[i][j] = slope*(X[i][j]-T1)
    return Y

def find_T1_T2(X):
    "T1 and T2 such that o/p histogram spans 0-255"
    T1 = X.min()
    T2 = X.max()
    return T1,T2

def main(filename,T1,T2):
    basename = os.path.basename(filename).split('.')[0]
    im = Image.open(filename)
    img = np.array(im)
    # find T1 and T2
    if(T1<0 or T2<0 or T2<T1):
        T1,T2 = find_T1_T2(img)
    print('T1='+str(T1)+' T2='+str(T2))
    img_cont_st = stretch(img,T1,T2)
    #save image
    im_save = Image.fromarray(img_cont_st)
    im_save.save(basename+'_cont_st.tif')

if __name__=="__main__":
    filename = sys.argv[1]
    T1 = np.float32(sys.argv[2])
    T2 = np.float32(sys.argv[3])
    main(filename,T1,T2)
```

Appendix

Got to git repo for complete code.

Listing 3: Python code for printing histogram of image

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab4- section 1 histogram
"""
import sys,os
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

def main(filename):
    basename = os.path.basename(filename).split('.')[0]
    im = Image.open(filename)
    img = np.array(im)
    gray = mpl.cm.get_cmap('gray',256)
    plt.figure(1)
    plt.imshow(img,cmap=gray)
    plt.savefig(basename+'_gray.eps',format='eps',
                bbox_inches='tight', pad_inches = 0)

    plt.figure(2)
    plt.hist(img.flatten(),bins=np.linspace(0,255,256))
    plt.xlim([0,255])
    plt.xlabel('pixel intensity')
    plt.ylabel('number of pixels')
    plt.savefig(basename+'_histogram.eps',format='eps',
                bbox_inches='tight', pad_inches = 0)

if __name__=="__main__":
    filename = sys.argv[1]
    main(filename)
```

Listing 4: Python code for calculating gamma of display

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab4- section 4.2 Gamma of monitor
"""
import sys
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

img_size=256
stripe_ht=16
chkr_blk = np.array(([[255,255,0,0],
                      [255,255,0,0],
```

```
                        [0,0,255,255],
                        [0,0,255,255]]))

def compute_gamma(g):
    return np.log(0.5)/np.log(g/255.0)

def main(gray):
    img = gray*np.ones((img_size,img_size))
    h,w = chkr_blk.shape
    for row in range(img_size//(2*stripe_ht)):
        pivot_row = row*(2*stripe_ht)
        for i in range(stripe_ht//h):
            for j in range(img_size//w):
                img[pivot_row+i*h:pivot_row+(i+1)*h, j*w:(j+1)*w] = chkr_blk

    gamma = compute_gamma(gray)
    gamma_str = "{:.3f}".format(gamma)
    gamma_str = gamma_str.replace('.','_')
    #print computed gamma
    print("Computed gamma: %0.6f"%(gamma))

    #display image
    plt.imshow(img,cmap=mpl.cm.gray)
    plt.show(block=False)

    #ask if want to save
    print("Do you want to print the image?(y/n): ",end='')
    print_it=input()
    print(print_it)
    if(print_it=='y'):
        plt.savefig('Array_pattern_gamma_'+gamma_str+'.eps', format='eps')

if __name__=="__main__":
    gray = np.int(sys.argv[1])
    main(gray)
```

Listing 5: Python code for gamma correction of image

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: rahul
course: ECE637-DIP-I
lab4- section 4.2 Gamma of monitor
"""
import argparse
import os
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
import matplotlib as mpl

gamma_monitor=1.709511 # hard coded

def gamma_correct_from_linear(img,gamma_out):
    out_img = np.zeros_like(img)
    h,w = img.shape
    for i in range(h):
        for j in range(w):
```

```
            #inverse of eq (5)
            out_img[i,j] = 255.0*np.exp((1.0/gamma_out)*np.log(img[i,j]/255.0))
    return out_img

def gamma_correct_from_gamma(img,gamma_out,gamma_in):
    out_img = np.zeros_like(img)
    h,w = img.shape
    for i in range(h):
        for j in range(w):
            x = 255.0*(img[i,j]/255.0)**gamma_in
            out_img[i,j] = 255.0*np.exp((1.0/gamma_out)*np.log(x/255.0))
    return out_img

if __name__=="__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument("image_file",type=str, help="path to input image file")
    parser.add_argument("-l","--linear", help="Linear scaled input flag",action="
        store_true")
    parser.add_argument("-gin","--gamma_input",type=np.float, help="gamma value of
        input")
    parser.add_argument("-gout","--gamma_output",type=np.float, help="gamma value of
         output")

    args = parser.parse_args()

    filename = args.image_file

    gamma_out = args.gamma_output  if args.gamma_output else gamma_monitor
    if args.gamma_input: gamma_in = args.gamma_input

    basename = os.path.basename(filename).split('.')[0]
    im = Image.open(filename)
    img = np.array(im)

    #display input image
    plt.imshow(img,cmap=mpl.cm.gray)
    plt.savefig(basename+'.eps',format='eps')
    plt.close()

    if args.linear:
        out_img = gamma_correct_from_linear(img,gamma_out)
    else:
        print("gamma_input: "+str(gamma_in))
        print("gamma_output: "+str(gamma_out))
        out_img = gamma_correct_from_gamma(img,gamma_out,gamma_in)

    #save output image
    plt.imshow(out_img,cmap=mpl.cm.gray)
    plt.savefig(basename+'_gamma_corrected.eps',format='eps')
    plt.close()
```

Listing 6: Bash code for running python code for lab4

```
#!/bin/bash

#section 1
python histogram.py ../../kids.tif
python histogram.py ../../race.tif
mv ./*.eps output/section1/
```

```
7
8  #section 2
9  python hist_eq.py ../../kids.tif
10 python histogram.py kids_hist_eq.tif
11 mv ./*.eps output/section2/
12
13 #section 3
14 python contrast_stretch.py ../../kids.tif 80 160
15 python histogram.py kids_cont_st.tif
16 mv ./*.eps output/section3/
17
18 #section 4.2
19 #gamma=$1
20 gamma=170
21 python gamma_monitor_4.2.py $gamma
22 mv ./*.eps output/section4/
23
24 #section 4.3
25 python gamma_correction.py ../../linear.tif --linear
26 #python gamma_correction.py ../../gamma15.tif -gin 1.5 -gout 2.5
27 python gamma_correction.py ../../gamma15.tif -gin 1.5
28 mv ./*.eps output/section4/
29
30 echo "done"
```

Listing 7: output log for task for section4.2

```
1 Computed gamma: 1.709511
2 Do you want to print the image?(y/n): y
3 done
```