1. *Consider the two image subsets S_1 and S_2. For $V = \{1\}$, determine whether these two subsets are 4-adjacent, 8-adjacent, and m-adjacent. Show justification in detail.*

   The sets are shown in the HW instructions.

   (a) Sets S_1 and S_2 **are not 4-adjacent** due to no shared elements of V perpendicular to one another.

   (b) Sets S_1 and S_2 **are 8-adjacent** due to elements of V being in diagonal neighborhoods of one another.

   (c) Sets S_1 and S_2 **are not m-adjacent** since no conditions apply for m-adjacency to occur (4 adjacent OR $\notin V$, in diagonal neighborhood, and have common 4-adjacent neighbors)

2. *Provide a single, composite transformation functions for performing the following operations*

   (a) *Scaling → translation*

   Any linear transformation on pairs of variables can be represented as a linear combination of said variables to some result. This is known as a linear mapping, where

$$\begin{bmatrix} x \\ y \end{bmatrix} \xrightarrow{L'} \begin{bmatrix} u \\ v \end{bmatrix} \tag{1}$$

   where x and u correspond to the **horizontal** axes of the image while y and v correspond to the **vertical** axes of the image (this is evident when defining vertical/horizonal shear). However, it is known that some linear transformations (such as translations) require constants, so they are not purely linear combinations. Because of this, we can include a unit scalar as well as an additional term post transformation which results in

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \xrightarrow{L} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{2}$$

   in order to scale, our transformation L_s can be represented as a matrix of values which result in such a system of equations such that

$$\alpha_x x = u$$

$$\alpha_y y = v$$

   Therefore

$$L_s(m) = Sm = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix} m \tag{3}$$

   As for translations, our transformation L_t can be represented as a matrix of values which result in such a system of equations such that

$$x + \delta_x = u$$

$$y + \delta_y = v$$

   Therefore

$$L_t(m) = Tm = \begin{bmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{bmatrix} m \tag{4}$$

Together, performing a scaling transformation followed by a translation results in

$$L_t(L_s(m)) = L_{st}(m) = TSm = \begin{bmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix} m \tag{5}$$

The two transformations can be combined into a composite matrix by calculating TS, is

$$TS = \begin{bmatrix} \alpha_x & 0 & \delta_x \\ 0 & \alpha_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

(b) *Scaling → translation → rotation*

Using the logic built up in the previous method, we can define a rotation transformation L_r and then simply append it to the total list of transformations. I am not going to derive the transformation matrix for rotation, but the mapping is as follows (for a counter-clockwise rotation)

$$x\cos(\theta) - y\sin(\theta) = u$$

$$x\sin(\theta) + y\cos(\theta) = v$$

Therefore

$$L_r(m) = Rm = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} m \tag{7}$$

So that

$$L_{str}(m) = RTSm = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_x & 0 & \delta_x \\ 0 & \alpha_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} m \tag{8}$$

Where the single composite matrix RTS is

$$RTS = \begin{bmatrix} \alpha_x\cos(\theta) & -\alpha_y\sin(\theta) & \delta_x\cos(\theta) - \delta_y\sin(\theta) \\ \alpha_x\sin(\theta) & \alpha_y\cos(\theta) & \delta_x\sin(\theta) + \delta_y\cos(\theta) \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

(c) *Vertical shear → scaling → translation → rotation*

Using the logic built up in the previous methods, we can define a vertical shear transformation L_v and then simply append it to the total list of transformations. A vertical shear is described as distorting the image in the **vertical** (y) direction w.r.t. a scalar multiple the other dimensions (in this case only **horizontal**, x). Therefore the mapping is as follows

$$x = u$$

$$y + \gamma_y x = v$$

Therefore

$$L_v(m) = Vm = \begin{bmatrix} 1 & 0 & 0 \\ \gamma_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} m \tag{10}$$

So that

$$L_{vstr}(m) = RTSVm = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_x & 0 & \delta_x \\ 0 & \alpha_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \gamma_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} m \quad (11)$$

Where the single composite matrix $RTSV$ is

$$RTSV = \begin{bmatrix} \alpha_x \cos(\theta) - \alpha_y \gamma_y \sin(\theta) & -\alpha_y \sin(\theta) & \delta_x \cos(\theta) - \delta_y \sin(\theta) \\ \alpha_x \sin(\theta) + \alpha_y \gamma_y \cos(\theta) & \alpha_y \cos(\theta) & \delta_x \sin(\theta) + \delta_y \cos(\theta) \\ 0 & 0 & 1 \end{bmatrix} \quad (12)$$

(d) *Does the order of multiplication of the individual matrices to produce a single transformation make a difference? Give an example based on a scaling/translation transformation to support your answer*

Order of multiplication does matter. As seen above, the order of which the transformations are applied, the matrices are listed in a reverse order left-to-right. This is because the first transformation applied must be next to the coordinate system vector m, and so on and so forth.

Intuitively, if some image was first scaled, then translated, the system of equations would result in

$$(\alpha_x x) + \delta_x = u$$
$$(\alpha_y y) + \delta_y = v$$

Which results in

$$L_{st}(m) = TSm = \begin{bmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix} m = \begin{bmatrix} \alpha_x & 0 & \delta_x \\ 0 & \alpha_y & \delta_y \\ 0 & 0 & 1 \end{bmatrix} m \quad (13)$$

Again, if some image was first translated, then scaled, the system of equations would result in

$$\alpha_x(x + \delta_x) = u$$
$$\alpha_y(y + \delta_y) = v$$

Which can clearly be seen, that now the translation has ALSO been scaled. This is because each successive transformation is w.r.t. the altered mapping (u, v) invoked by the previous transformations, and not the original map (x, y). This is shown here

$$L_{ts}(m) = STm = \begin{bmatrix} \alpha_x & 0 & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \delta_x \\ 0 & 1 & \delta_y \\ 0 & 0 & 1 \end{bmatrix} m = \begin{bmatrix} \alpha_x & 0 & \alpha_x\delta_x \\ 0 & \alpha_y & \alpha_y\delta_y \\ 0 & 0 & 1 \end{bmatrix} m \quad (14)$$

It is clear that $L_{st}(m) \neq L_{ts}(m)$ because $TS \neq ST$, meaning that the order of the matrices matters when performing transformations.

3. *An experiment in particle physics is set up with an imaging system whose (processed) output consists of two types of images. Images of Type I have at least one particle collision present, while images of Type II are 'blank' (i.e. they contain no particle collisions). The images are stored sequentially, as they come out of the experiment. The occurrence of collisions is random, and it is known that the bank images occur 80% of the time. In a particular run, the experiment generates 1000 images*

(a) *What is the probability that if we look at the first three images they would all be of Type I?*

It was externally given that the imaging of these particle collisions are i.i.d. Therefore, the number of images as well as any order of the images is irrelevant. Rather, the only thing that's affected is the number of times Type I or II occurrences can happen sequentially. A Type I occurrence happens 20% of the time, therefore for the first 3 images to be Type I would simply be $(0.2)^3$ or 0.08%

(b) *Will the result be different if we look at the last three images?*

Since the imaging of these particles is i.i.d., the result will **not** be different when looking at the last three images.

4. *Give a single intensity transformation function for spreading the intensities of an image so the lowest intensity is 0 and the highest is $L-1$ (e.g. $L = 100$)*

(a) *Derive the transformation function*

i. **Method I: Simple Transformation**

Let's take the simplest possible example and use a 2-pixel image. For a 2-pixel greyscale image I where $I = [p_1, p_2]$, we know that there is some transformation T where $T(I) = [0,\, L-1]$ or $T(I) = [L-1,\, 0]$. We know that

$$\min(I) \xrightarrow{T} 0$$

and

$$\max(I) \xrightarrow{T} L-1$$

This can simply be done by removing an offset to decrease the lower-bound of I to 0. The range of I changes from $[\min(I), \max(I)]$ to $[0, \max(I) - \min(I)]$. Then, we simply have to scale it so that the upper-bound equals $L-1$, so we simply multiply it by the $L-1$ and the reciprocal of the current quantity. Therefore the overall transformation function for any image I where $I \in \mathbb{R}^{n\times m}$ is given by

$$T(I) = \frac{(I - \min(I))\,(L-1)}{\max(I) - \min(I)} \tag{15}$$

This simply takes the distribution, shifts it down to the minimum, and stretches it until $L-1$ is reached. Therefore, is no change in the shape/envelope of the distribution. More can be seen in the images below.

ii. **Method II: Histogram Equalization**

In order to achieve histogram-equalization, we must have a change in variables using some transformation function T. Let us say we are transforming the probability density function of pixels S to a new PDF R. We can define this as

$$S \xrightarrow{T} R, \quad S = T(R) \tag{16}$$

where S and R are random variables. We can note that the resulting equalized histogram should be defined as a uniform distribution with PDF

$$S \sim U[0,\, L-1], \quad p_S(s) = \begin{cases} \frac{1}{(L-1)}, & 0 \le s \le L-1 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

Therefore, we can easily relate the ranges of S and R

$$0 \leq S \leq s \Longleftrightarrow 0 \leq R \leq T^{-1}(s) \tag{18}$$

and define equivalence using their PDFs as such

$$\int_0^s p_S(\sigma) d\sigma = \int_0^{T^{-1}(s)} p_R(\rho) d\rho \tag{19}$$

If we differentiate both sides w.r.t. s, then LHS becomes

$$\frac{d}{ds} \int_0^s p_S(\sigma) d\sigma = p_S(s) \tag{20}$$

and RHS becomes

$$\frac{d}{ds} \int_0^{T^{-1}(s)} p_R(\rho) d\rho = \frac{d}{ds} P_R(T^{-1}(s)) \tag{21}$$

therefore

$$p_S(s) = \frac{d}{ds} P_R(T^{-1}(s)) \tag{22}$$

Using the chain rule on the RHS, we get

$$p_S(s) = P_R'(T^{-1}(s)) \frac{dT^{-1}(s)}{ds} \tag{23}$$

and since $T^{-1}(s)$ equals r, then

$$p_S(s) = p_R(r) \frac{dr}{ds} \tag{24}$$

We know that $p_S(s) = \frac{1}{L-1}$ given its range, so we can rearrange the equation above to get

$$\frac{ds}{dr} = (L-1) p_R(r) \tag{25}$$

integrate both sides w.r.t. r to get the final result

$$s = \int_0^r (L-1) p_R(\rho) d\rho \tag{26}$$

Discretely, this is equivalent to

$$s_{i,j} = (L-1) \sum_{n=0}^{n_{i,j}} p_R(n) \tag{27}$$

where

$$p_R(n) = \frac{\sum_{k=0}^{K} 1_{n=k}}{M}, \text{ where } 1_z = \begin{cases} 1, \text{ if } z \text{ is true} \\ 0, \text{ if } z \text{ is false} \end{cases} \tag{28}$$

where i and j indicate indices of the image, n is the intensity of a pixel, K is the count of pixels w.r.t. intensity, and M is the total number of pixels within the image.

(b) *Check your derived function by writing code and test the code on the provided* `lena.png` *image (convert to grayscale using the built-in functions)*

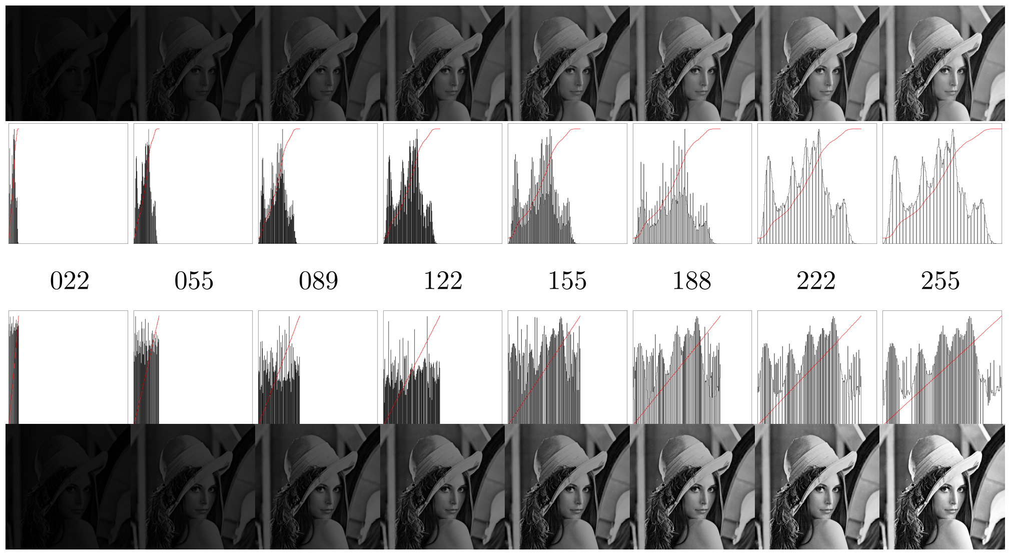


Figure 1: Method I & Method II: `lena.png` with the various $L-1$ values and histograms

$L-1 \in [0,\ 255]$ because these are `uint8`. The top row correspond to Method I and the bottom row to Method II. As it can be observed, Method 1 preserves the distribution while shifting and stretching it. Method II creates a uniform distribution, which can clearly be seen by the linear red line that is the calculated CDF. The listing for the script can be found at the appendix at the end.

5. *Obtain the un-normalized and the normalized histograms of the following 8-bit, $M \times N$ image. Given your histogram either in a table or a graph, labeling clearly the value and the location of each histogram component in terms of M and N.*

If the top left of the figure is $(0, 0)$ and the downward direction corresponds to M and the rightward direction corresponds to N, then the bottom left corner is (M, N). Then, we can find the points which bound each area in terms of M and N

bounds	value	area	%area
$(0.25M, 0.75N), (0.5M, 0.5N), (0.5M, N), (M, 0.5N), (M, N)$	0	5	0.3125
$(0M, 0.25N), (0M, 0.5N), (0.75M, 0.25N), (0.75M, 0.5N)$	16	3	0.1875
$(0M, 0.5N), (0M, N), (0.25M, 0.75N)$	32	1	0.0625
$(0.75M, 0N), (0.75M, 0.25N), (M, 0N), (M, 0.25N)$	127	1	0.0625
$(0.75M, 0.25N), (0.75M, 0.5N), (M, 0.25N), (M, 0.5N)$	191	1	0.0625
$(0M, 0.5N), (0.25M, 0.75N), (0.5M, 0.5N)$	228	1	0.0625
$(0M, 0N), (0M, 0.25N), (0.75M, 0N), (0.75M, 0.25N)$	240	3	0.1875
$(0M, N), (0.25M, 0.75N), (0.5M, N)$	255	1	0.0625

Table 1: Unnormalized & Normalized Histogram in table format

Where all values in the %area correspond to the normalized histogram; column sums to 1.

6. *Assume continuous intensity values, and suppose that the intensity values of an image have the following PDF (probability density function)*

$$p_R(r) = \begin{cases} \frac{2r}{(L-1)^2}, & 0 \leq r \leq L-1 \\ 0, & \text{otherwise} \end{cases} \tag{29}$$

   (a) *Find the transformation function that will map the input intensity value r into values s of a histogram-equalized image*

   We derived histogram-equalization in Equation 26, so substituting $p_R(r)$ with the value in Equation 29, we get

$$s = \int_0^r (L-1)\frac{2\rho}{(L-1)^2}d\rho \tag{30}$$

$$s = \frac{2}{L-1}\left[\frac{\rho^2}{2}\right]_0^r = \frac{r^2}{L-1} \tag{31}$$

   i. **Monte-Carlo Simulation**
   For confirmation, a Monte-Carlo simulation was performed. An image is randomly populated pixels which follow the PDF given in Equation 29. The MATLAB function **histeq** is used to equalize the histogram of the image, and it is compared to the simple transformation derived in Equation 31. In these simulations, $L-1$ is always 255. As you can see in the figures below, the PDF is linear and the histogram shows it. We compare the figure and histogram of the discrete histogram-equalization algorithm (built-in to MATLAB, **histeq**) with the figure and histogram of the transformation defined in Equation 31.

   (b) *Find the transformation function that (when applied to the histogram-equalized intensity values, s), will produce an image whose intensity PDF is given by*

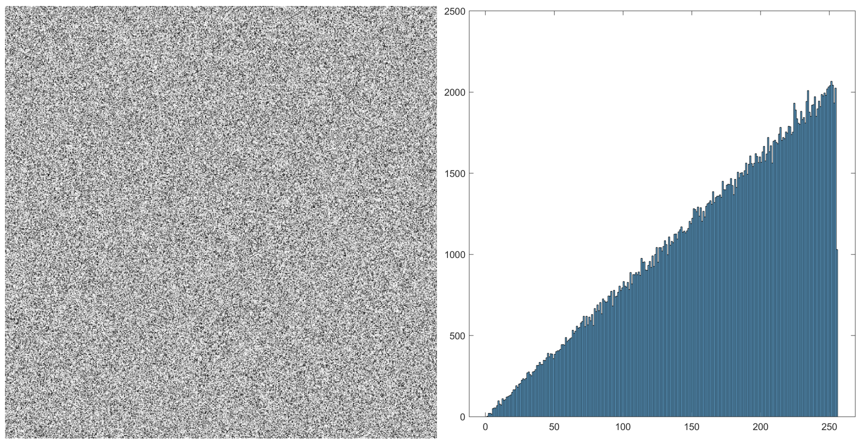
$$p_Z(z) = \begin{cases} \frac{3z^2}{(L-1)^3}, & 0 \leq z \leq L-1 \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

   Similarly to part (a), we can substitute the value of $p_R(r)$ from Equation 26 with the value given in Equation 32 to get

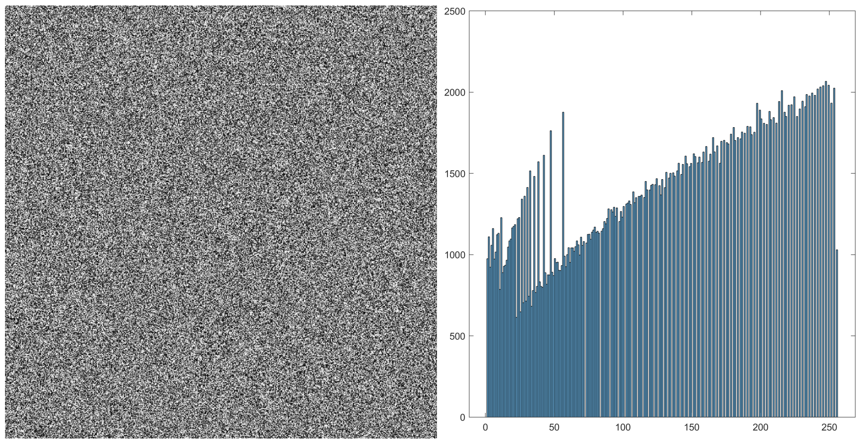
$$s = \int_0^z (L-1)\frac{3\rho^2}{(L-1)^3}d\rho \tag{33}$$

$$s = \frac{3}{(L-1)^2}\left[\frac{\rho^3}{3}\right]_0^z = \frac{z^3}{(L-1)^2} \tag{34}$$

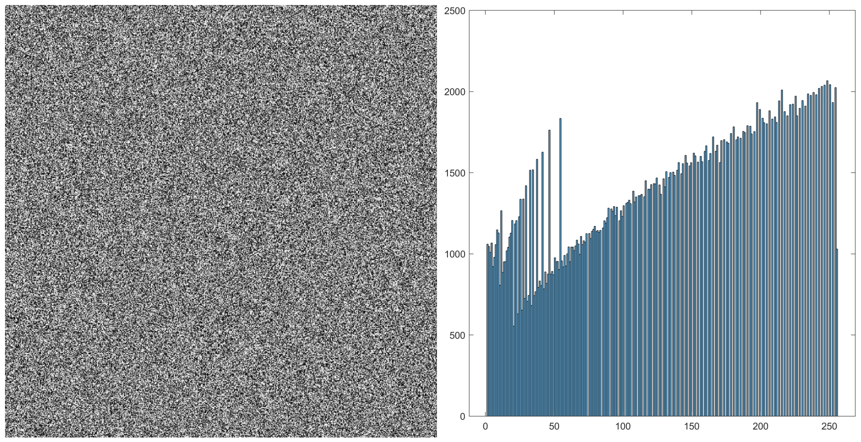
   i. **Monte-Carlo Simulation**
   Again, for confirmation, a Monte-Carlo simulation was performed. An image is randomly populated pixels which follow the PDF given in Equation 32, using similar methods to the previous simulation. The results are shown below. As it can be observed, compared to the previous simulation the new PDF is non-linear, and the histogram confirms this



Default PDF and Histogram

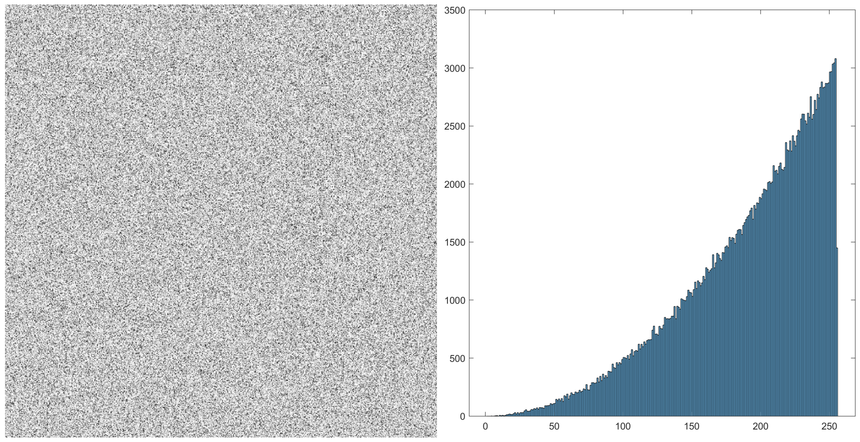


Histogram-equalization using transformation function defined in Equation 31

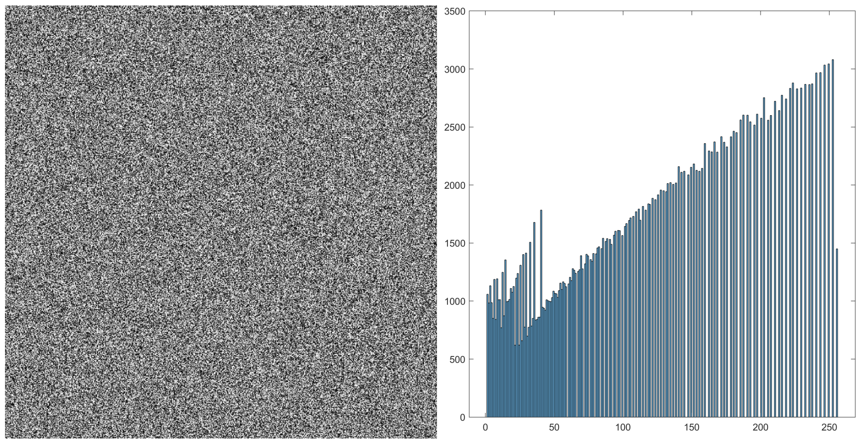


Histogram-equalization built-in MATLAB function **histeq**

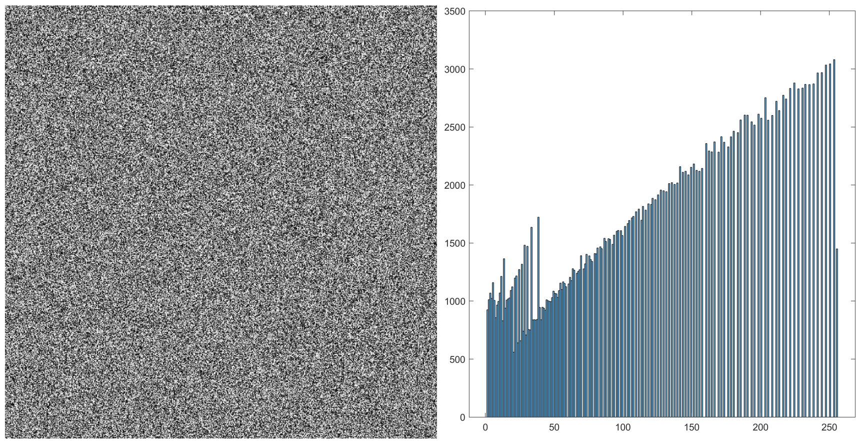
Figure 2: Monte-Carlo simulations using intensities PDF is $\frac{2r}{(L-1)^2}$ (Equation 29)



Default PDF and Histogram



Histogram-equalization using transformation function defined in Equation 34



Histogram-equalization built-in MATLAB function **histeq**

Figure 3: Monte-Carlo simulations using intensities PDF is $\frac{3z^2}{(L-1)^3}$ (Equation 32)

(c) *Express the transformation function from (b) directly in terms of r, the intensities of the output image.*

To go from one domain to the other, we use the uniform distribution (where s is a uniformly distributed PDF) as a medium of transformation. Therefore we can simply set the two results to one another as such

$$\frac{z^3}{(L-1)^2} = s = \frac{r^2}{L-1} \tag{35}$$

and then solve for z to get

$$z = \sqrt[3]{r^2(L-1)} \tag{36}$$

1 Appendix

```
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.axes import Axes

import numpy as np

from PIL import Image, ImageOps
from skimage import exposure
from skimage import img_as_float

# modified from https://scikit-image.org/docs/dev/auto_examples/...
# ...color_exposure/plot_log_gamma.html
# further modified from Dr. Tianfu Wu - NC State University - ECE 558
def plot_hist_cdf(image: np.ndarray, axes: Axes, bins=256) -> (Axes, ...
                                        Axes, Axes):
    """Plot an image along with its histogram and cumulative histogram.
        image: a float image """
    ax_hist = axes
    ax_cdf = ax_hist.twinx()

    # skimage get image as float
    image = img_as_float(image)

    # Display histogram
    ax_hist.hist(image.ravel(), bins=bins, histtype='step', color='black')
    ax_hist.set_xlim(0, 1)
    ax_hist.set_xticks([])
    ax_hist.set_yticks([])

    # Display cumulative distribution
    img_cdf, bins = exposure.cumulative_distribution(image, bins)
    ax_cdf.plot(bins, img_cdf, 'r')
    ax_cdf.set_xticks([])
    ax_cdf.set_yticks([])

    return ax_hist, ax_cdf

# ============================================================

# get the lena image
folder_dir = 'C:/Users/Arpad/Documents/Academic/NCSU/E. NCSU Grad Sem ...
                                        1/ECE 558/HW02/'
```

```python
img_dir = folder_dir + 'hw2/lena.tiff'

# greyscale
img = np.array(ImageOps.grayscale(Image.open(img_dir)), dtype=np.float32)

# image shape
num_row, num_col = np.shape(img)

# min and max pixel values
v_max = np.max(img)
v_min = np.min(img)

# uint8 range
uint8_min = 0
uint8_max = 255

# # variable number for L - 1
# num_show = 24
# L = np.linspace(uint8_min + 1, uint8_max + 1, num=num_show)

# values used in example for HW
L = np.array([22, 55, 89, 122, 155, 188, 222, 255]) + 1

# cumulative count + scale, used in histogram equalization
cum_count = np.zeros([uint8_max + 1])
for v in range(uint8_max + 1):
    if not v:
        cum_count[v] = np.sum(img == v)
    else:
        cum_count[v] = cum_count[v - 1] + np.sum(img == v)
cum_count /= (num_row * num_col)

# plot each result
for l in L:
    # simple transformation (METHOD I)
    timg = np.copy(img)
    ratio = (l - 1) / (v_max - v_min)
    timg -= v_min
    timg *= ratio
    timg = np.uint8(timg)

    # ============================================================
    # PLOTTING LENA WITH HISTOGRAM (METHOD I)

    fig, plts = plt.subplots(2, 1, tight_layout=True, figsize=(4, 7.78))
    plts[1].set_box_aspect()
    ax_hist, ax_cdf = plot_hist_cdf(timg, plts[1])
    # image part
    plts[0].set_xticks([])
    plts[0].set_yticks([])
    plts[0].imshow(timg, cmap="gray", vmin=uint8_min, vmax=uint8_max)
    fig.tight_layout()

    # # SAVE LENA WITH HISTOGRAM FIGURE
    # fig.savefig(folder_dir + 'hw2/' + str(round(l - 1)) + ...
                                          'lena_and_hist.png')
    #
    # # ============================================================
    # # PLOTTING HISTOGRAM ONLY (METHOD I)
    #
    # hist_fig, hist_plts = plt.subplots(1, 1, tight_layout=True, ...
                                          figsize=(8, 8))
    # ax_hist, ax_cdf = plot_hist_cdf(timg, hist_plts)
    # hist_fig.tight_layout()
    #
    # # SAVE HISTOGRAM ONLY FIGURE
    # hist_fig.savefig(folder_dir + 'hw2/' + str(round(l - 1)) + 'hist.png')
    #
    # # ============================================================
    # SAVE LENA ONLY IMAGE (METHOD I)
    #
    # Image.fromarray(np.uint8(timg)).save(folder_dir + 'hw2/' + ...
                                        str(round(l - 1)) + ...
                                        'lena_max.png')
    #
    # # ============================================================
    # # ============================================================
    # # ============================================================
    # histogram equalization transformation (METHOD II)
    ttimg = np.copy(img)
    for v in range(uint8_max + 1):
        ttimg[(img == v)] = (l - 1) * cum_count[v]
    ttimg = np.uint8(ttimg)
    # ============================================================
    # PLOTTING LENA WITH HISTOGRAM (METHOD II)

    fig, plts = plt.subplots(2, 1, tight_layout=True, figsize=(4, 7.78))
    plts[1].set_box_aspect()
    ax_hist, ax_cdf = plot_hist_cdf(ttimg, plts[1])
    # image part
    plts[0].set_xticks([])
    plts[0].set_yticks([])
    plts[0].imshow(ttimg, cmap="gray", vmin=uint8_min, vmax=uint8_max)
    fig.tight_layout()

    # # SAVE LENA WITH HISTOGRAM FIGURE
    # fig.savefig(folder_dir + 'hw2/eq_' + str(round(l - 1)) + ...
                                        'lena_and_hist.png')
    #
    # # ============================================================
    # # PLOTTING HISTOGRAM ONLY (METHOD II)
    #
    # hist_fig, hist_plts = plt.subplots(1, 1, tight_layout=True, ...
                                        figsize=(8, 8))
    # ax_hist, ax_cdf = plot_hist_cdf(ttimg, hist_plts)
    # hist_fig.tight_layout()
    #
    # # SAVE HISTOGRAM ONLY FIGURE
    # hist_fig.savefig(folder_dir + 'hw2/eq_' + str(round(l - 1)) + ...
                                        'hist.png')
    #
    # # ============================================================
    # # SAVE LENA ONLY IMAGE (METHOD II)
    #
    # Image.fromarray(np.uint8(ttimg)).save(folder_dir + 'hw2/eq_' + ...
                                        str(round(l - 1)) + ...
                                        'lena_max.png')
    #
    # # ============================================================
    # # ============================================================

    # show the plot
    plt.show()
```

Monte-Carlo Listing. Just for reference, otherwise ignore

```
%% monte carlo image
uint8_max = 255;
uint8_min = 0;

L = uint8_max + 1;
%%
r_res = 1;

r = (uint8_min + 1):r_res:(L - 1);

% r distribution, linear pdf
% pdf = (2 * r) / ((L - 1) ^ 2);
% pdf = [r', pdf'];

% z distribution, non-linear pdf
z = r;
pdf = (3 * (z.^2)) / ((L - 1) ^ 3);
pdf = [z', pdf'];

cdf = cumsum(pdf(:, 2));

%% lena
img = double(rgb2gray(imread("C:\Users\Arpad\Documents\Academic\NCSU\E. NCSU ...
    Grad Sem 1\ECE 558\HW02\hw2\lena.tiff")));
img_size = size(img);
img_size = img_size(1);
total_pix = img_size * img_size;

%% random trials
img_size = 512;
total_pix = img_size * img_size;

trials = rand(total_pix, 1);
p = @(r) find(r < cdf, 1, 'first');

res = arrayfun(p, trials);
% histogram(res, 1:256);

img = reshape(res, img_size, img_size);
imshow(img, [uint8_min - 1, uint8_max - 1]);

%% hist eq
uint8_range = uint8_min:uint8_max;
maxv = max(img, [], 'all');
count_pix_cdf = zeros(size(uint8_range));

imgT = zeros(size(img));

idx = 1;
for v = uint8_range
    img_bool = (img == v);
    if idx == 1
        count_pix_cdf(idx) = sum(img_bool, 'all');
    else
        count_pix_cdf(idx) = count_pix_cdf(idx - 1) + sum(img_bool, 'all');
    end
    imgT(img_bool) = floor((L - 1) * (count_pix_cdf(idx) / total_pix));
    idx = idx + 1;
end
```

```

%% own transform
% % for pdf = 2r/(L-1)^2
% imgTT = round((img .^ 2) ./ (L - 1));

% for pdf = 3(z^2)/(L-1)^3
imgTT = round((img .^ 3) ./ ((L - 1) ^ 2));

%% matlab hist eq
imgMT = round(histeq(img ./ uint8_max, uint8_max + 1) * uint8_max);

%%
f = figure(1);
imshow(img, [uint8_min - 1, uint8_max - 1]);
% title({sprintf("Simulated %d x %d image (N = %d), intensities %d to %d", ...
    img_size, img_size, total_pix, uint8_min, uint8_max), ...
%         sprintf("Intensities PDF = $\\frac{2r}{(L - 1)^{2}}$ from $0 ...
    \\rightarrow (L - 1)$")}, 'interpreter', 'latex');

f = figure(2);
imshow(imgMT, [uint8_min - 1, uint8_max - 1]);
% title({sprintf("Histogram-equalization applied, using $\\textbf{histeq}$"), ...
%         sprintf("Intensities PDF = $\\frac{2r}{(L - 1)^{2}}$ from $0 ...
    \\rightarrow (L - 1)$")}, 'interpreter', 'latex');

% figure(3);
% imshow(imgT, [uint8_min - 1, uint8_max - 1]);

f = figure(4);
imshow(imgTT, [uint8_min - 1, uint8_max - 1]);
% title({sprintf("Histogram-equalization applied, using derived transformation ...
    $s = \\frac{r^{2}}{L - 1}$"), ...
%         sprintf("Intensities PDF = $\\frac{2r}{(L - 1)^{2}}$ from $0 ...
    \\rightarrow (L - 1)$")}, 'interpreter', 'latex');

bins_range = 1:(uint8_max + 1);

f = figure(5);
histogram(reshape(img, total_pix, 1), bins_range);
% title(sprintf("Intensities PDF = $\\frac{2r}{(L - 1)^{2}} \\longrightarrow$ ...
    default - Histogram view"), 'interpreter', 'latex');

f = figure(6);
histogram(reshape(imgMT, total_pix, 1), bins_range);
% title(sprintf("Intensities PDF = $\\frac{2r}{(L - 1)^{2}} \\longrightarrow ...
    \\textbf{histeq}$ - Histogram view"), 'interpreter', 'latex');

% figure(7);
% histogram(reshape(imgT, total_pix, 1), bins_range);

f = figure(8);
histogram(reshape(imgTT, total_pix, 1), bins_range);
% title(sprintf("Intensities PDF = $\\frac{2r}{(L - 1)^{2}} \\longrightarrow ...
    \\frac{r^{2}}{L - 1}$ - Histogram view"), 'interpreter', 'latex');
```