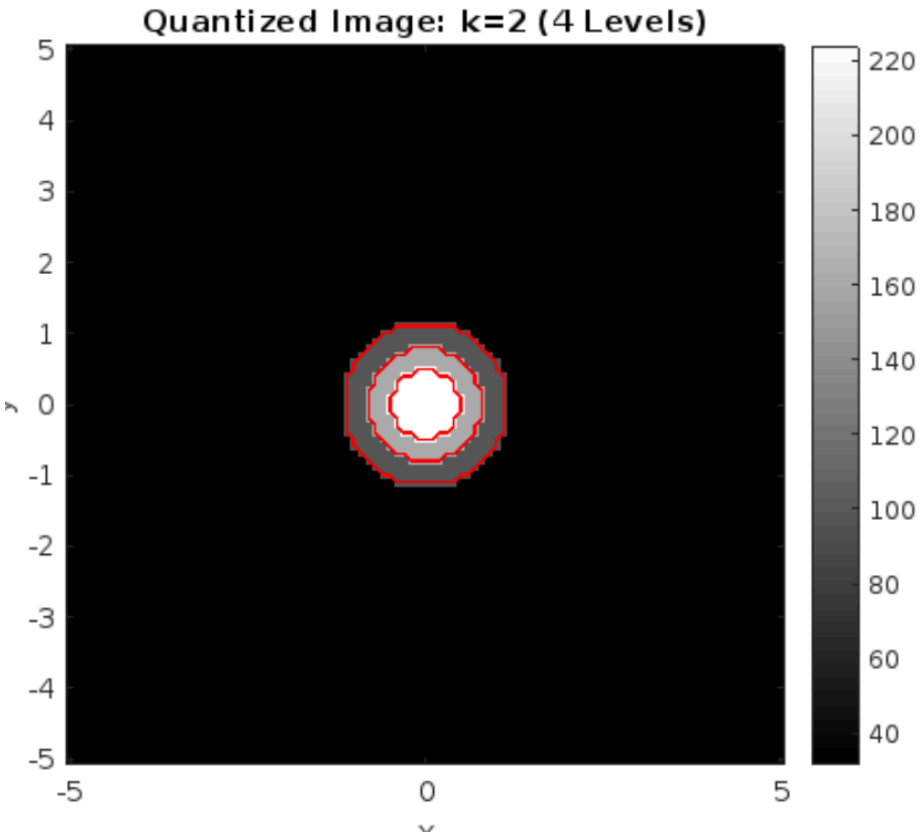
DEPARTMENT OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING
CONCORDIA UNIVERSITY
COMP 478/6771 Image Processing
Fall 2024
ASSIGNMENT 1
Omer Sayem: 40226505
Due: February 11, 2025

1.

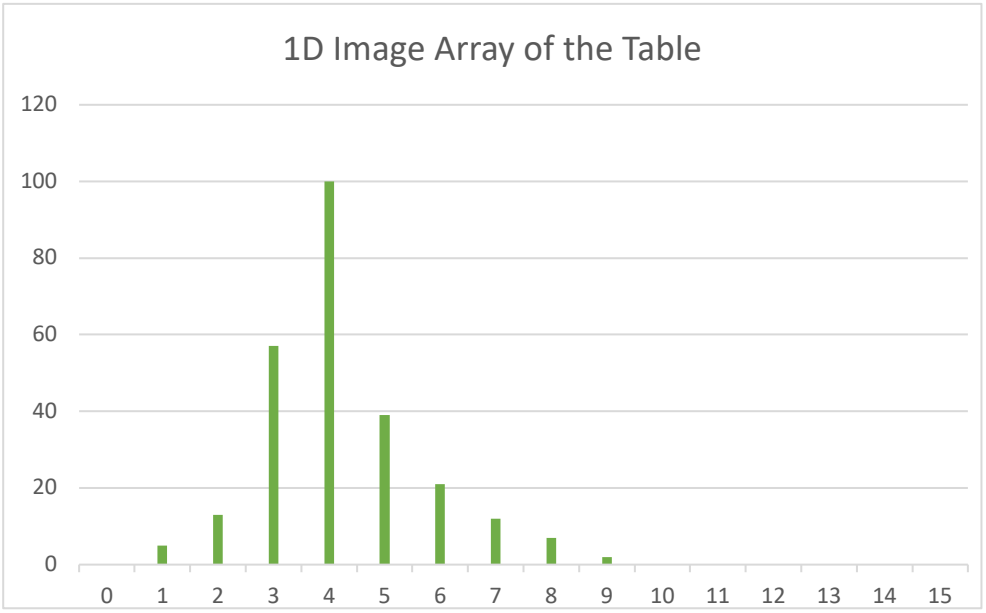
Given intensity function $i(x,y) = Ke^{-[(x-x_0)^2+(y-y_0)^2]}$, and also given that the eye can detect a jump of about 8 grey levels between adjacent pixels. With a maximum of k= 255 intensity and 2^K equally spaced digital value, the increment is approximately $\frac{255}{2^k-1}$. The required bit depth k at which the step is large enough that a jump of one quantization level is $\frac{255}{2^k-1} \geq 8$ => $2^k \leq 32.87$ => $k \approx 5$.

a.



2.

a. Histogram plot of 1D image Array:

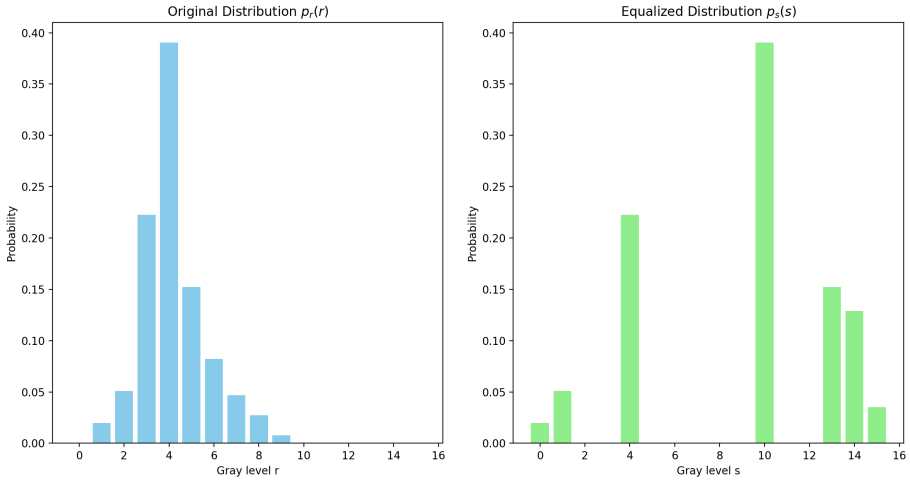


b.

i. Total number of pixels : $\sum_{k=0}^{15} n_k = 256$ the s_k table is the following

GL	NP	$p_r(r_k) = \frac{N_k}{256}$	CDF S(K)	$S_k = 15S(K)$	Round (S_K)
0	0	0	0	0	0
1	5	0.01953	0.01953	0.2930	0
2	13	0.05078	0.07031	1.0547	1
3	57	0.22266	0.29297	4.3945	4
4	100	0.39063	0.68359	10.2539	10
5	39	0.15234	0.83594	12.5391	12
6	21	0.08203	0.91797	13.7695	13
7	12	0.04688	0.96484	14.4726	14
8	7	0.02734	0.99219	14.8829	14
9	2	0.00781	1	15	15
10-15	0	0	1	15	15

ii. Plot of probability distribution function $p_r(r_k) and\ p_s(s_k)$:



c. Plot of the new histogram:

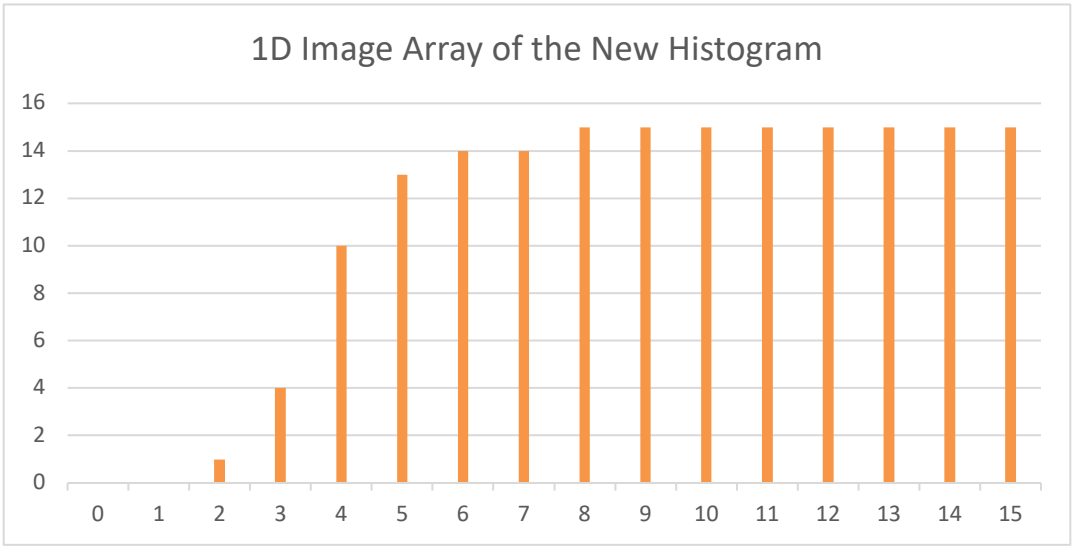


Figure: After performing the histogram equalization

d. Discrete histogram equalization does not yield a flat histogram because we have to perfectly map each original pixel count to a uniform distribution. The new GL's s_k are rounded so that multiple ranges of original intensities fall into the same output bin, which results in an imperfect equalized histogram.

e. A second pass of the histogram equalization will map the already equalized image almost onto itself. We will get the same result because the first pass will make the pixel value distribution uniform. The second pass of the same transformation will not make any difference. Moreover, if any small changes occur, rounding will result in a close result of the first image.

3.

a. Derivation of the mapping function $v = T(u)$ that maps any arbitrary u to output v with the desired cube-root hyperbolic distribution.

Given that,

$$p(v) = \begin{cases} \frac{1}{3} v^{-\frac{2}{3}} / v_{max}^{\frac{1}{3}} - v_{min}^{\frac{1}{3}}, & v_{min} \le v \le v_{max} \\ 0, & \text{otherwise} \end{cases}$$

Let,

$$\alpha = v_{max}^{\frac{1}{3}}, \quad \beta = v_{min}^{\frac{1}{3}}$$

Then for $v \in [v_{min}, v_{max}]$,

$$p(v) = \frac{1}{3} v^{-\frac{2}{3}} / (\alpha - \beta)$$

The CDF $F(v)$, integrate v_{min} to v.

$$F(v) = \int_{v_{min}}^{v} p(t)\, dt = \int_{v_{min}}^{v} \frac{1}{3} t^{-\frac{2}{3}} \frac{1}{\alpha - \beta}\, dt.$$

$$F(v) = \frac{1}{3(\alpha - \beta)} \int_{v_{min}}^{v} t^{-\frac{2}{3}}\, dt$$

$$= \frac{1}{3(\alpha - \beta)} \left[3 t^{\frac{1}{3}} \right]_{t = v_{min}}^{t = v}$$

$$= \frac{1}{\alpha - \beta} \left[v^{\frac{1}{3}} - v_{min}^{\frac{1}{3}} \right]$$

Thus,

$$u = F(v) = \frac{v^{\frac{1}{3}} - \beta}{\alpha - \beta} \Rightarrow v^{\frac{1}{3}} = \beta + u(\alpha - \beta)$$

$$\Rightarrow v = \left[\beta + u(\alpha - \beta) \right]^3$$

Hence, the transformation,

$$T(u) = \left[v_{min}^{\frac{1}{3}} + u \left(v_{max}^{\frac{1}{3}} - v_{min}^{\frac{1}{3}} \right) \right]^3$$

b. Plotting the input-output (u-v) mapping curve,

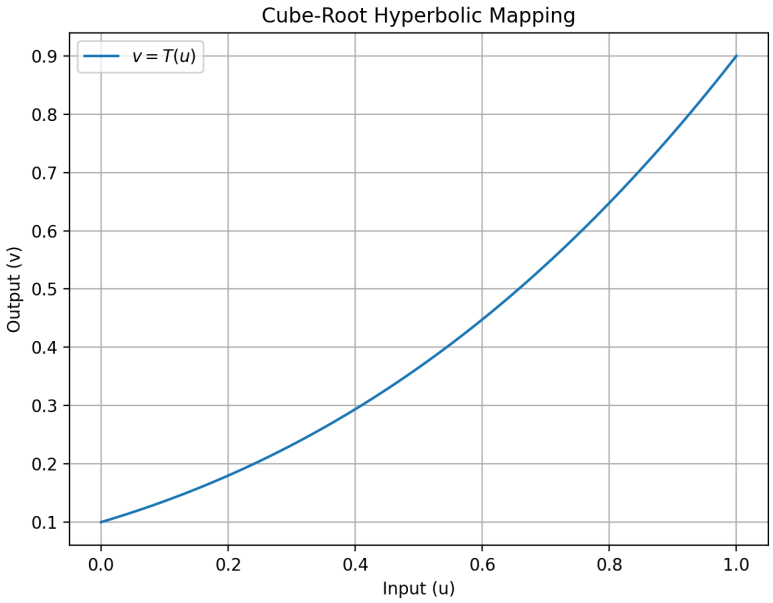


Figure: $v = T(u)$ curve assuming, u as a uniform pdf over [0,1]

4. The probability of the exposure time is less than 4 seconds:

Given that,

The shape parameter, $k = 2$

" rate parameter, $\lambda = 0.5$

we know,

mean $\mu = \frac{k}{\lambda} = \frac{2}{0.5} = 4$

variance $\sigma^2 = \frac{k}{\lambda^2} = 8$

Also,

$$f(x;k;\lambda) = \frac{\lambda^k \cdot x^{k-1} \cdot e^{-\lambda x}}{(k-1)!}$$

$$= \frac{(0.5)^2 \cdot x^{2-1} \cdot e^{-0.5x}}{(2-1)!}$$

So, $P(x \le 4) = \int_{x_1}^{x_2} (0.5)^2 x \cdot e^{-0.5x} dx = (0.5)^2 \cdot x \cdot e^{-0.5x}$

$$= (0.5)^2 \int_{x_1}^{x_2} x \cdot e^{-0.5x} dx$$

$$= (0.5)^2 \int_{x_1}^{x_2} \frac{1}{0.5} e^u du \quad [u = -0.5x]$$

$$= \int e^u du = \left(e^u u - \int e^u du\right)$$

$$= \left(e^u u - e^u\right) \quad [\because \int e^u du = e^u]$$

$$= \left[-0.5 e^{-0.5x} x - e^{-0.5x}\right]_0^4$$

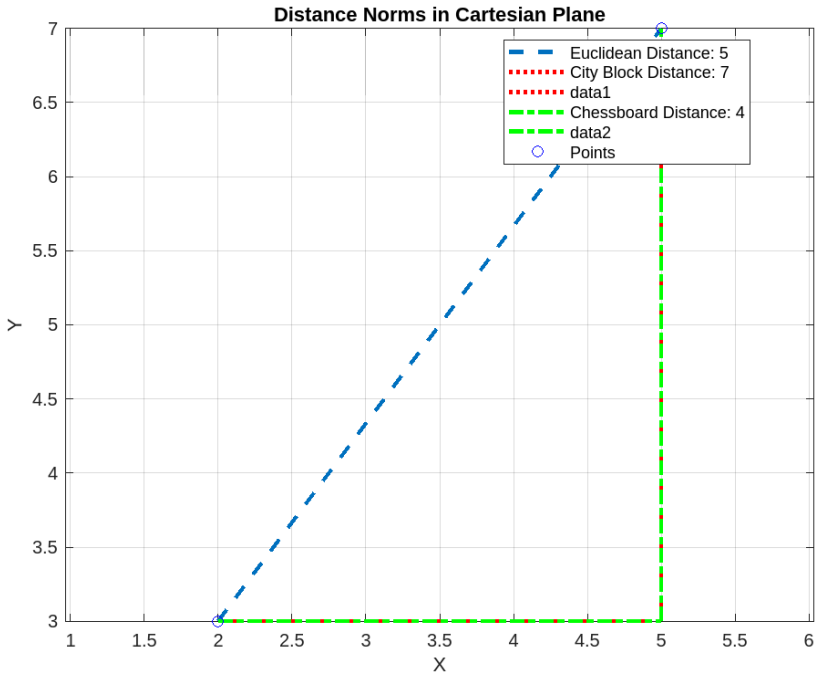
$$= 0.594$$

$\therefore P(x \le 4) = 0.594$

5.

a. Below is a chart showing the difference between Euclidian norm, City block norm, and Chessboard norm:

Norm	Distance Formula	Usage	How it works	Shape
Euclidian (L2)	$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$	It measures the straight-line distance	Pythagorean formula	A circle of radius 1
City block/Manhattan/(L1)	$\lvert x_2 - x_1 \rvert + \lvert y_2 - y_1 \rvert$	Grid path	Absolute horizontal and vertical distances	Diamond
Chessboard (L_∞)	$\max(\lvert x_2 - x_1 \rvert, \lvert y_2 - y_1 \rvert)$	boundingbox comparisons	Largest distance between the absolute difference in X and Y axis	A square aligned with the axes

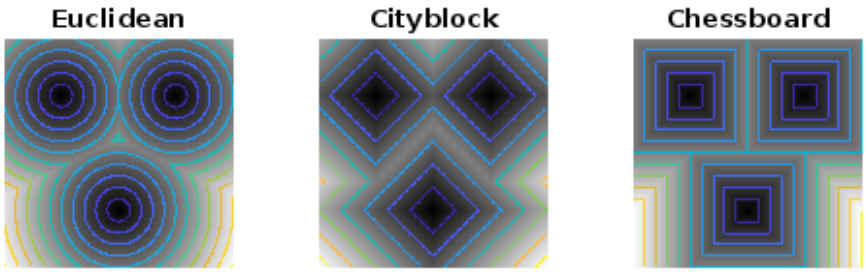


b. Below is the MATLAB code to show the differences:

```
% Creating the cartesian plane
bw = zeros(200,200);
% Creating 3 centers
bw(50,50) = 1; bw(50,150) = 1; bw(150,100) = 1;
% Implementing bwdist function with Euclidean, cityblock and
chessboard norm
D1 = bwdist(bw,'euclidean');
D2 = bwdist(bw,'cityblock');
```

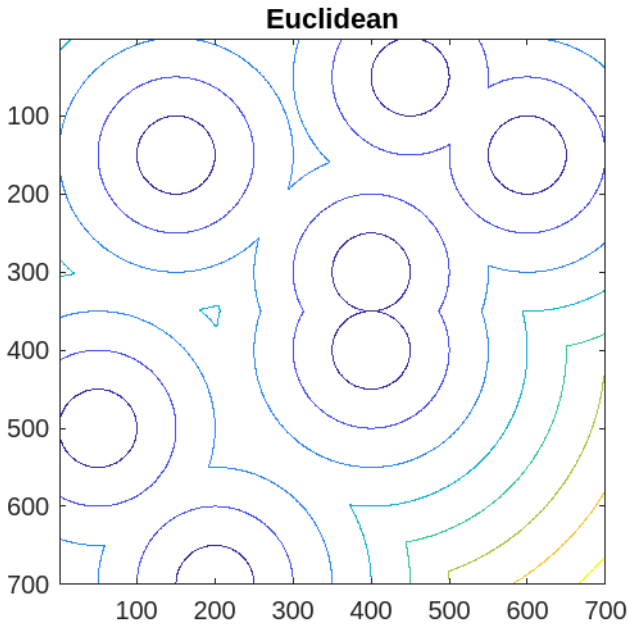
```
D3 = bwdist(bw,'chessboard');
% Scaling the distances
img1 = repmat(rescale(D1), [1 1 3]);
img2 = repmat(rescale(D2), [1 1 3]);
img3 = repmat(rescale(D3), [1 1 3]);
%Plotting the graphs for each norm
subplot(1,3,1), imshow(img1), title('Euclidean')
hold on, imcontour(D1)
subplot(1,3,2), imshow(img2), title('Cityblock')
hold on, imcontour(D2)
subplot(1,3,3), imshow(img3), title('Chessboard')
hold on, imcontour(D3)
```

Visualization:



Explanation: Firstly, for Euclidean norm, we get a circle shape. Secondly, for the city block norm, we get the distance in the X direction and Y direction. Hence making a diamond shape. Finally, for chessboard norm we take the maximum of X or Y direction differences. So, moving in each quadrant until one coordinate hits the distance limit gives an "L" path in each quadrant. Combining them forms an axis-aligned square**.**

c. Here is the visualization of 7 circles made with Euclidean norm:



Below is the MATLAB code to generate the graph:

```
% Creating 2D Cartesian plane
bw = zeros(700,700);
%Creating 7 centers
bw(150,150) = 1;
bw(50,450) = 1;
bw(150,600) = 1;
bw(700, 200) =1;
bw(300, 400) = 1;
bw(400, 400) = 1;
bw(500, 50) = 1;
%Implementing the bwdist function
D1 = bwdist(bw,'euclidean');
% Plotting the grpah
figure, imcontour(D1), title('Euclidean')
```

d. Below is the visualization and MATLAB code:

```
% Define image size and create a grid
image_size = 200;
[x, y] = meshgrid(linspace(-90, 90, image_size));
% Define circle centers
center1 = [-10, -10];
center2 = [10, 10];
% Define circle radii
radius = 20;
% Calculate Euclidean distances from circle centers
distances1 = sqrt((x - center1(1)).^2 + (y - center1(2)).^2);
distances2 = sqrt((x - center2(1)).^2 + (y - center2(2)).^2);
% Create a black-white image with overlapped circles
image = zeros(size(x));
image(distances1 <= radius | distances2 <= radius) = 1;
% Display the image
imshow(image, 'colormap', [0, 0, 0; 1, 1, 1]);
title('Overlapped Circles Using Euclidean Norm');
axis on;
```