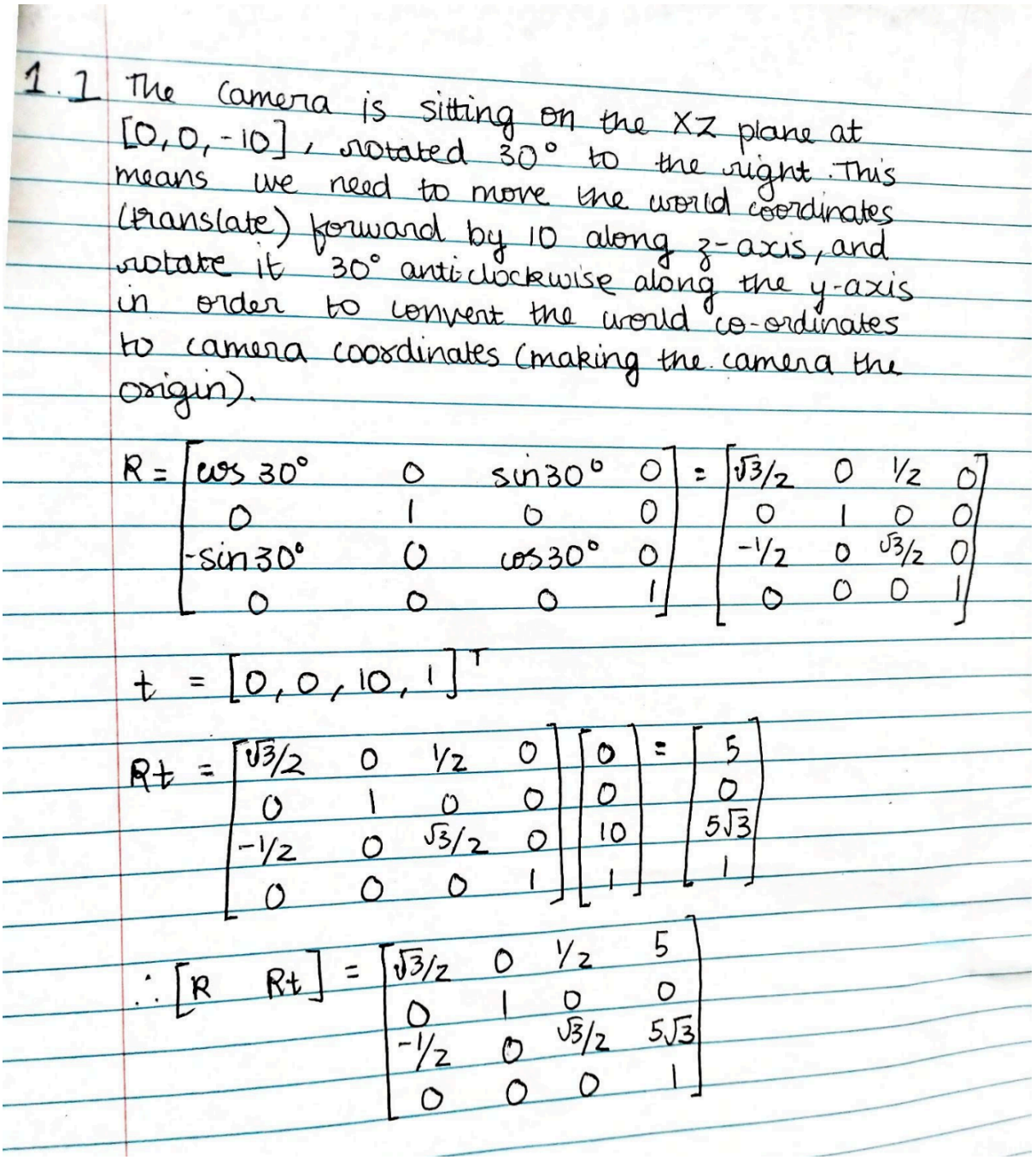
AIML 331 - ASSIGNMENT 1

Merisa Joseph - 300630477

Link to Code:
https://github.com/merisarj/AIML-331.git

1. Camera Problem

1.1 [R Rt] matrix converting world coordinates to camera coordinates.



1.1 The camera is sitting on the XZ plane at [0, 0, -10], rotated 30° to the right. This means we need to move the world coordinates (translate) forward by 10 along z-axis, and rotate it 30° anticlockwise along the y-axis in order to convert the world co-ordinates to camera coordinates (making the camera the origin).

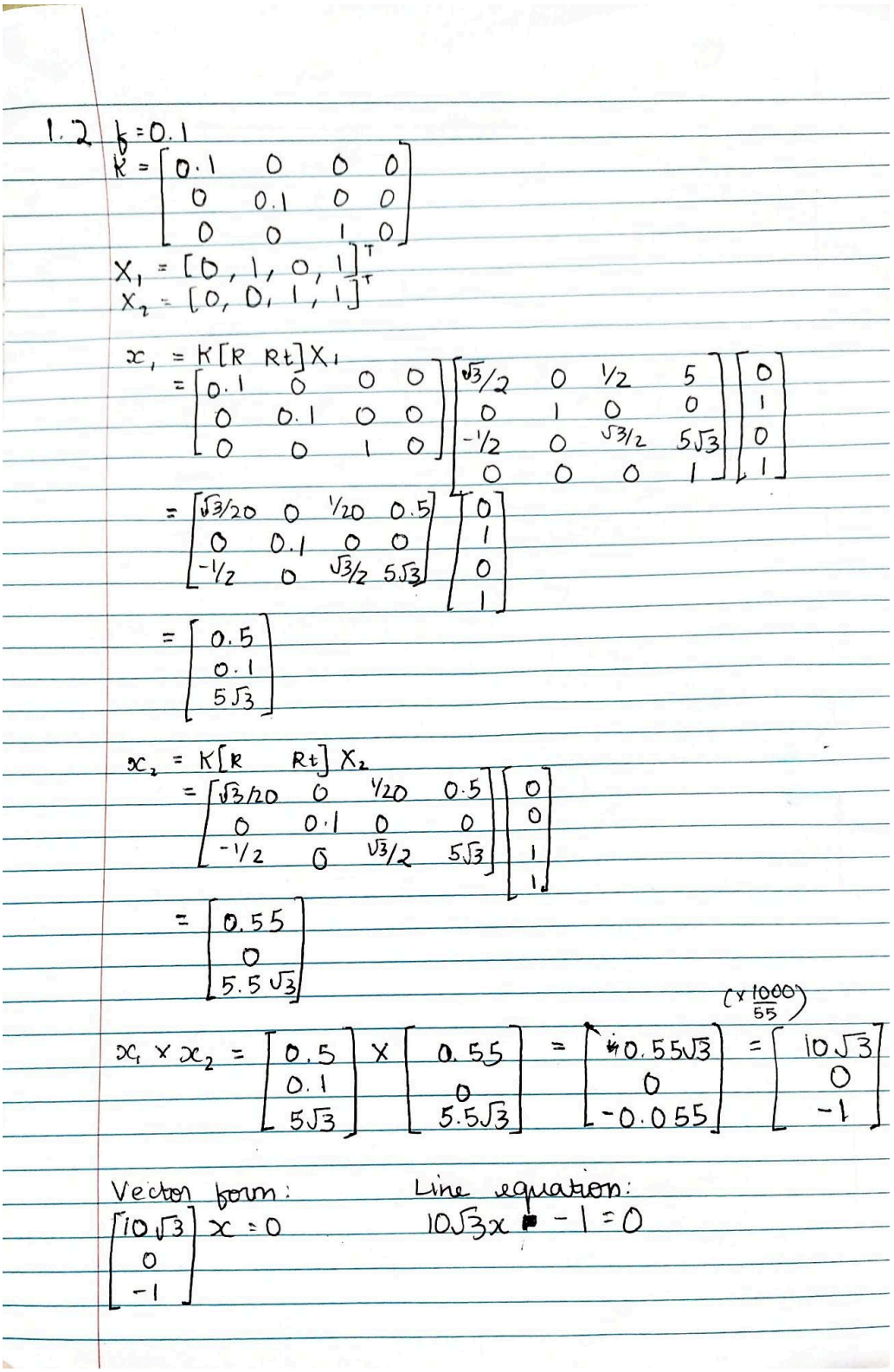
$$R = \begin{bmatrix} \cos 30° & 0 & \sin 30° & 0 \\ 0 & 1 & 0 & 0 \\ -\sin 30° & 0 & \cos 30° & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$t = [0, 0, 10, 1]^T$$

$$Rt = \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 0 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \\ 5\sqrt{3} \\ 1 \end{bmatrix}$$

$$\therefore [R \quad Rt] = \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 5 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 5\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

1.2 The equation of a line on the projective plane that goes through the points that correspond to [0,1,0] and [0,0,1] in the world coordinates.



1.2 $f = 0.1$

$$K = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$X_1 = [0, 1, 0, 1]^T$

$X_2 = [0, 0, 1, 1]^T$

$$x_1 = K[R\ Rt]X_1 = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 5 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 5\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} \sqrt{3}/20 & 0 & 1/20 & 0.5 \\ 0 & 0.1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 5\sqrt{3} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 \\ 0.1 \\ 5\sqrt{3} \end{bmatrix}$$

$$x_2 = K[R\ \ Rt]X_2 = \begin{bmatrix} \sqrt{3}/20 & 0 & 1/20 & 0.5 \\ 0 & 0.1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 5\sqrt{3} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.55 \\ 0 \\ 5.5\sqrt{3} \end{bmatrix}$$

$$x_1 \times x_2 = \begin{bmatrix} 0.5 \\ 0.1 \\ 5\sqrt{3} \end{bmatrix} \times \begin{bmatrix} 0.55 \\ 0 \\ 5.5\sqrt{3} \end{bmatrix} = \begin{bmatrix} 0.55\sqrt{3} \\ 0 \\ -0.055 \end{bmatrix} \xrightarrow{\times \frac{1000}{55}} = \begin{bmatrix} 10\sqrt{3} \\ 0 \\ -1 \end{bmatrix}$$

Vector form:

$$\begin{bmatrix} 10\sqrt{3} \\ 0 \\ -1 \end{bmatrix} x = 0$$

Line equation:

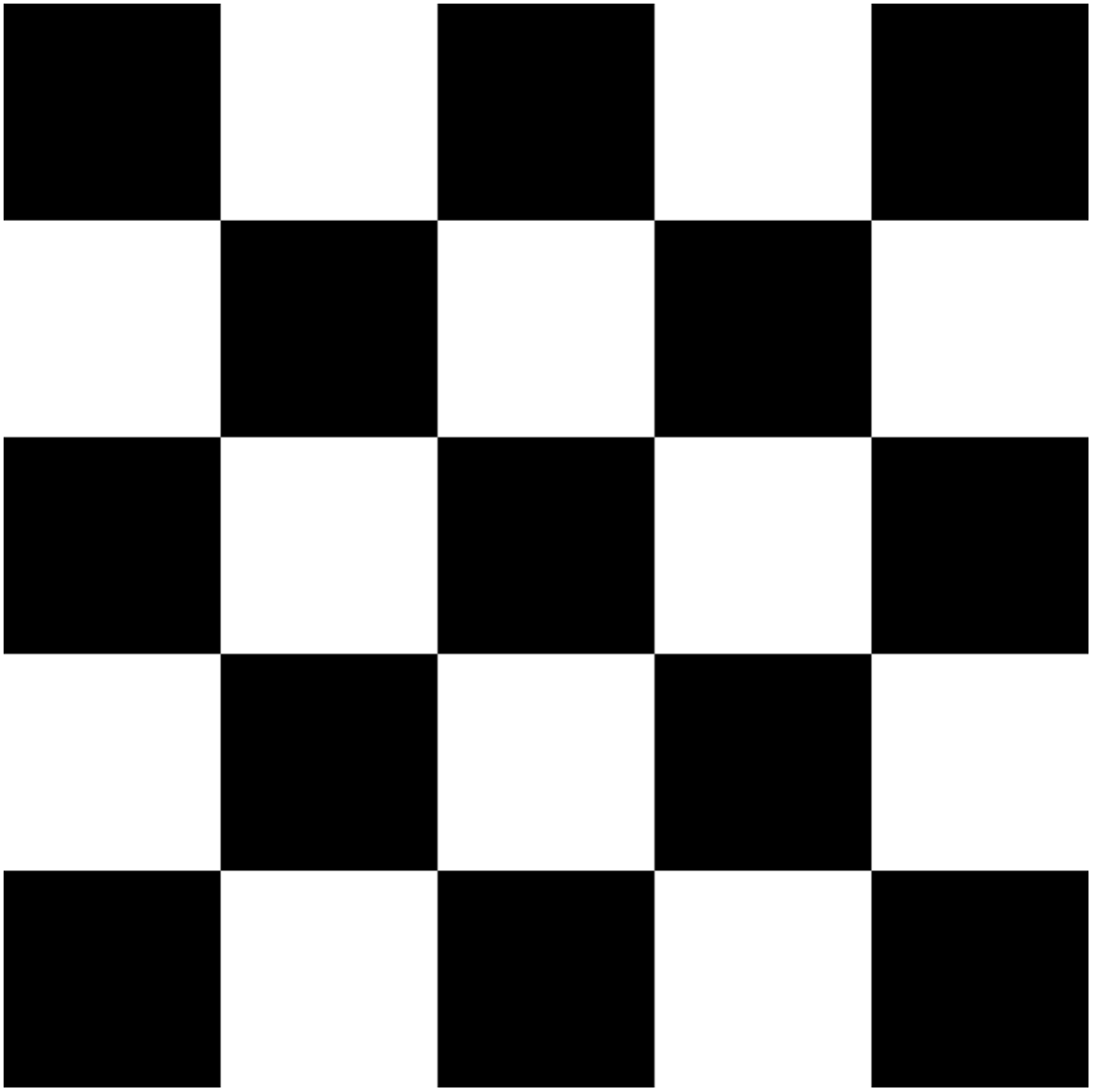
$$10\sqrt{3}x - 1 = 0$$

2 Checkerboard problem

I have used numpy - for all calculations, matplotlib - for plotting histograms and displaying images and PIL- for importing the image.

2.1

I downloaded a checkerboard image(1024 x 1024):

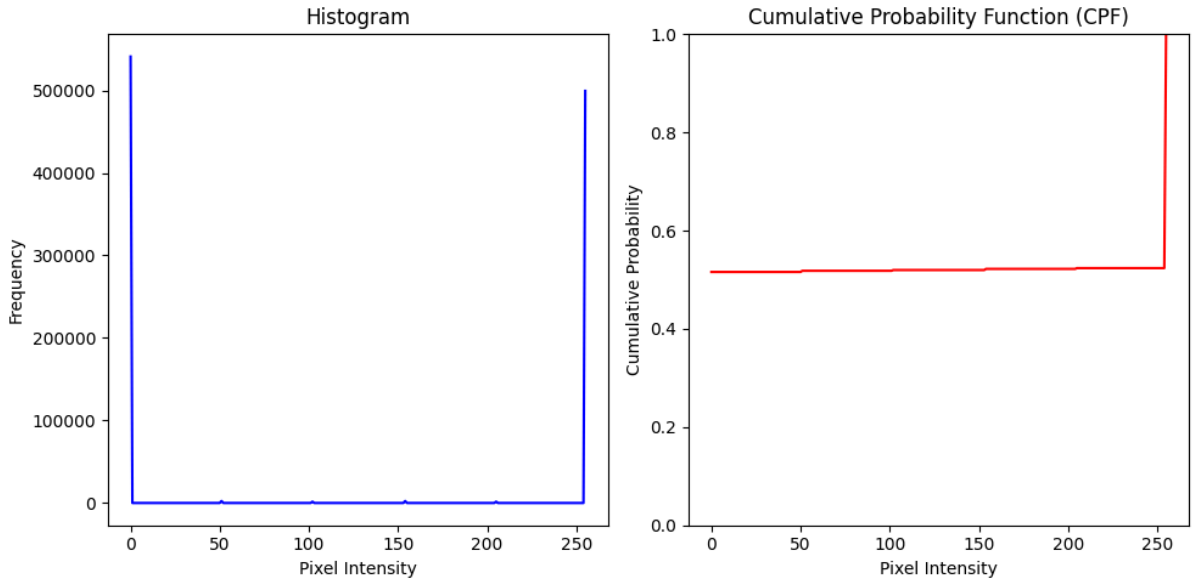


Src : https://upload.wikimedia.org/wikipedia/commons/thumb/7/70/Checkerboard_pattern.svg/1024px-Checkerboard_pattern.svg.png

I converted it to grayscale using the .convert('L') method from the PIL library. It removes color information, keeping only the intensity of light (brightness). Each pixel in the output image has a

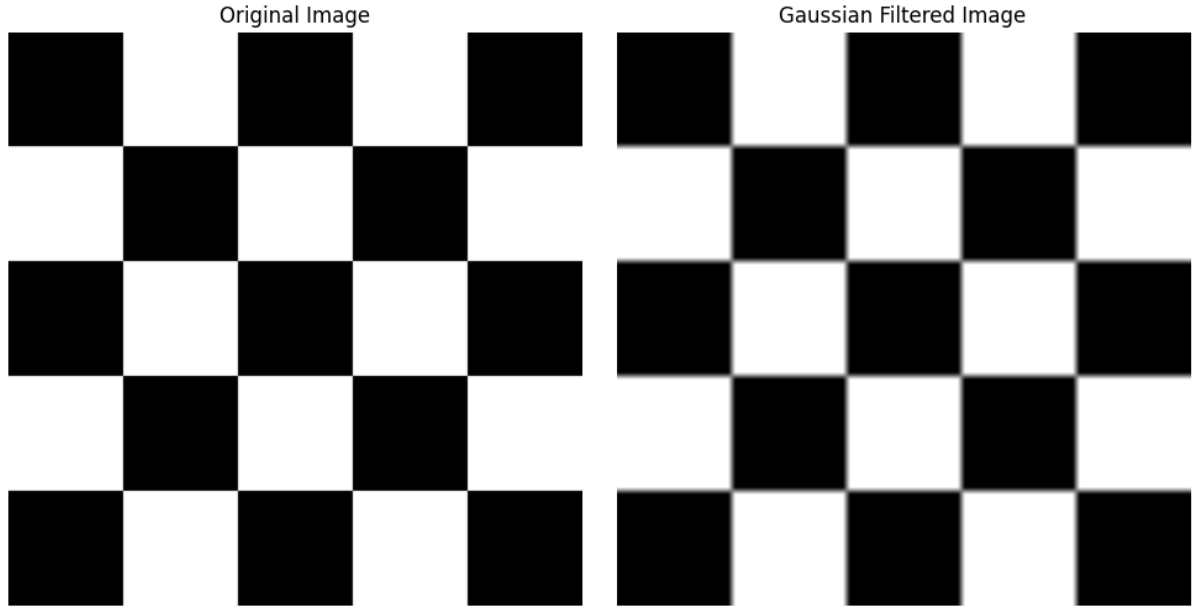
value between 0 (black) and 255 (white), with intermediate values representing shades of gray. Then I turned it into a numpy array for easy usability.

2.2



My image has 13 black squares and 12 white squares which is why there are more 0 pixels.

2.3



Created using the 2D Gaussian function with numpy. Handles edges using reflected padding to avoid border artifacts. Used nested loops to apply the filter to each pixel.

2.4

Screenshot from jupyter notebook:

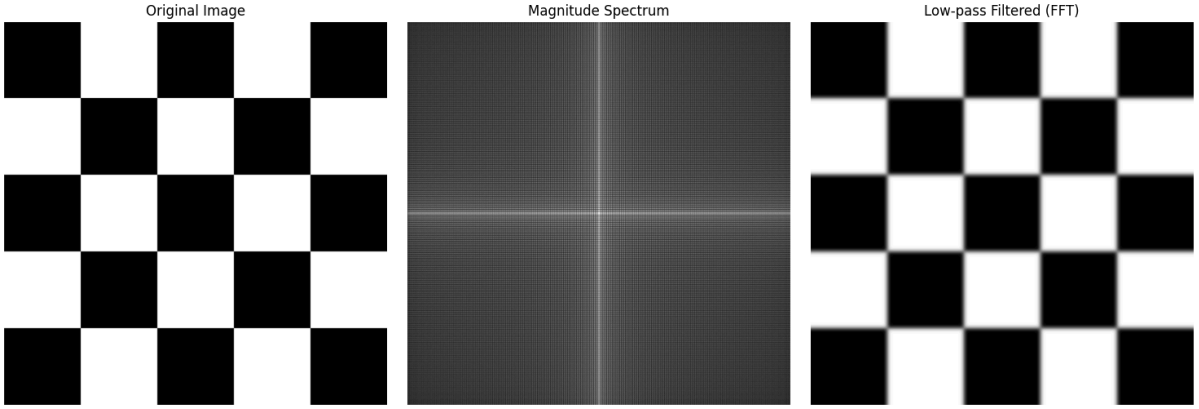
Yes it is seperable because the equation for the gaussian filter for 2D is given by: $w(i,j) = Ke^{\frac{x^2+y^2}{2\sigma^2}}$

This can be factored or broken down to be processed in 1D as

$w(i,j) = \sqrt{K}e^{\frac{x^2}{2\sigma^2}}.\sqrt{K}e^{\frac{y^2}{2\sigma^2}}$

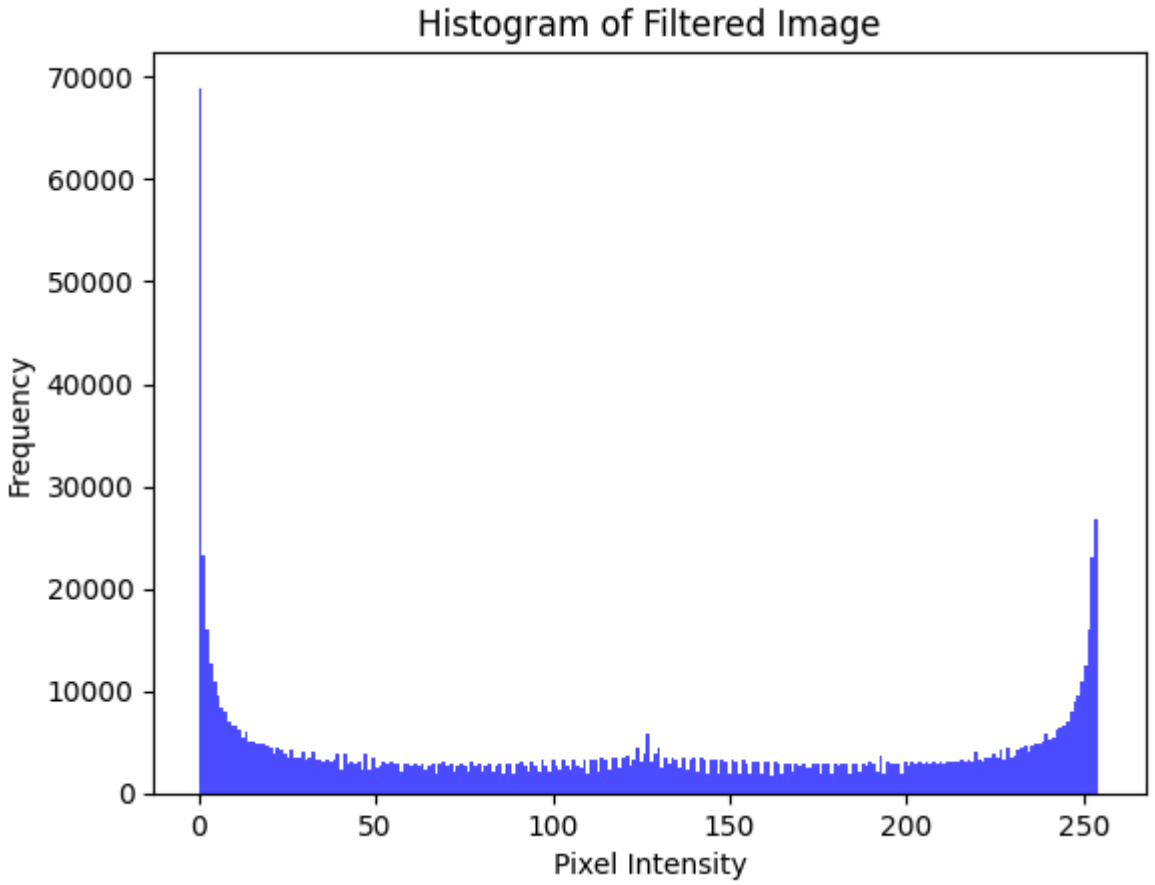
Instead of applying a 2D convolution, you can apply two 1D convolutions (one along rows, one along columns), reducing the complexity from $(O(N^2)$ to $O(2N)$.The output remains identical to using the full 2D kernel.

2.5

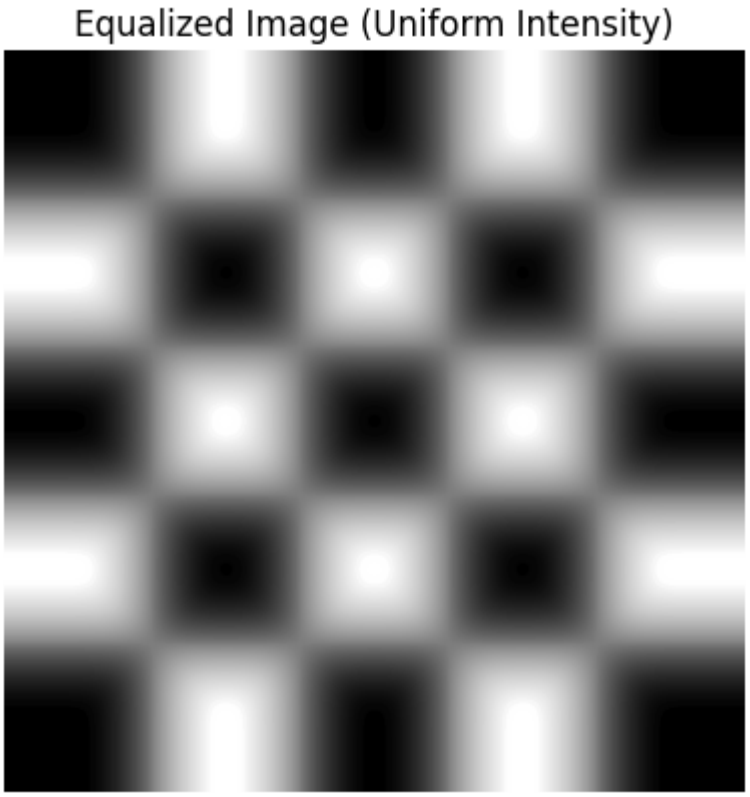


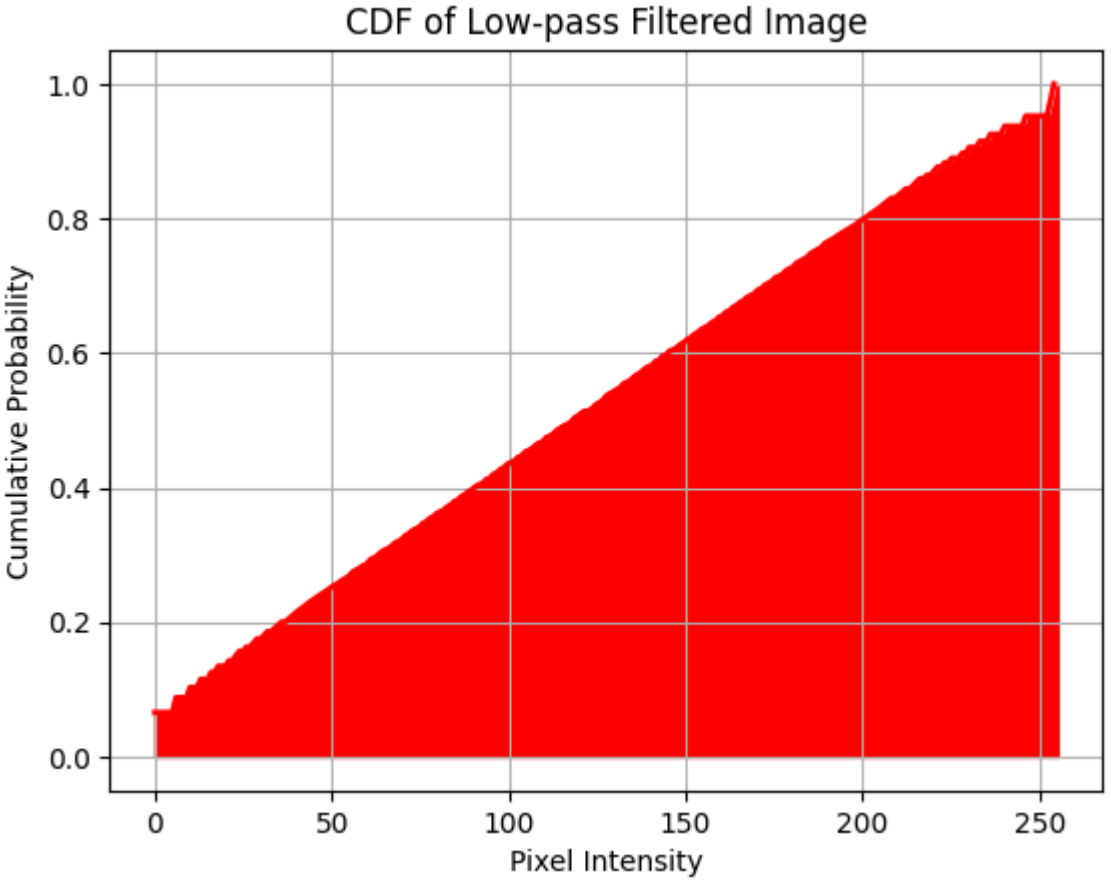
I apply a low-pass filter to an image in the frequency domain using the Fourier Transform (FFT). It first converts the image from the spatial domain to the frequency domain and shifts the low frequencies to the center for visualization. The magnitude spectrum is computed and displayed to observe the frequency components. A Gaussian low-pass filter is then created. This filter is applied to the FFT of the image, and the Inverse FFT (IFFT) is used to convert the filtered data back to the spatial domain, resulting in a blurred version of the original image.

2.6


Histogram of Filtered Image
Frequency
Pixel Intensity
0
10000
20000
30000
40000
50000
60000
70000
0
50
100
150
200
250

2.7


Equalized Image (Uniform Intensity)


CDF of Low-pass Filtered Image
Cumulative Probability
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
Pixel Intensity

I applied histogram equalization to the low-pass filtered image to achieve a uniform intensity distribution. First, it ensures the filtered image is in grayscale uint8 format and computes its histogram and cumulative distribution function (CDF).