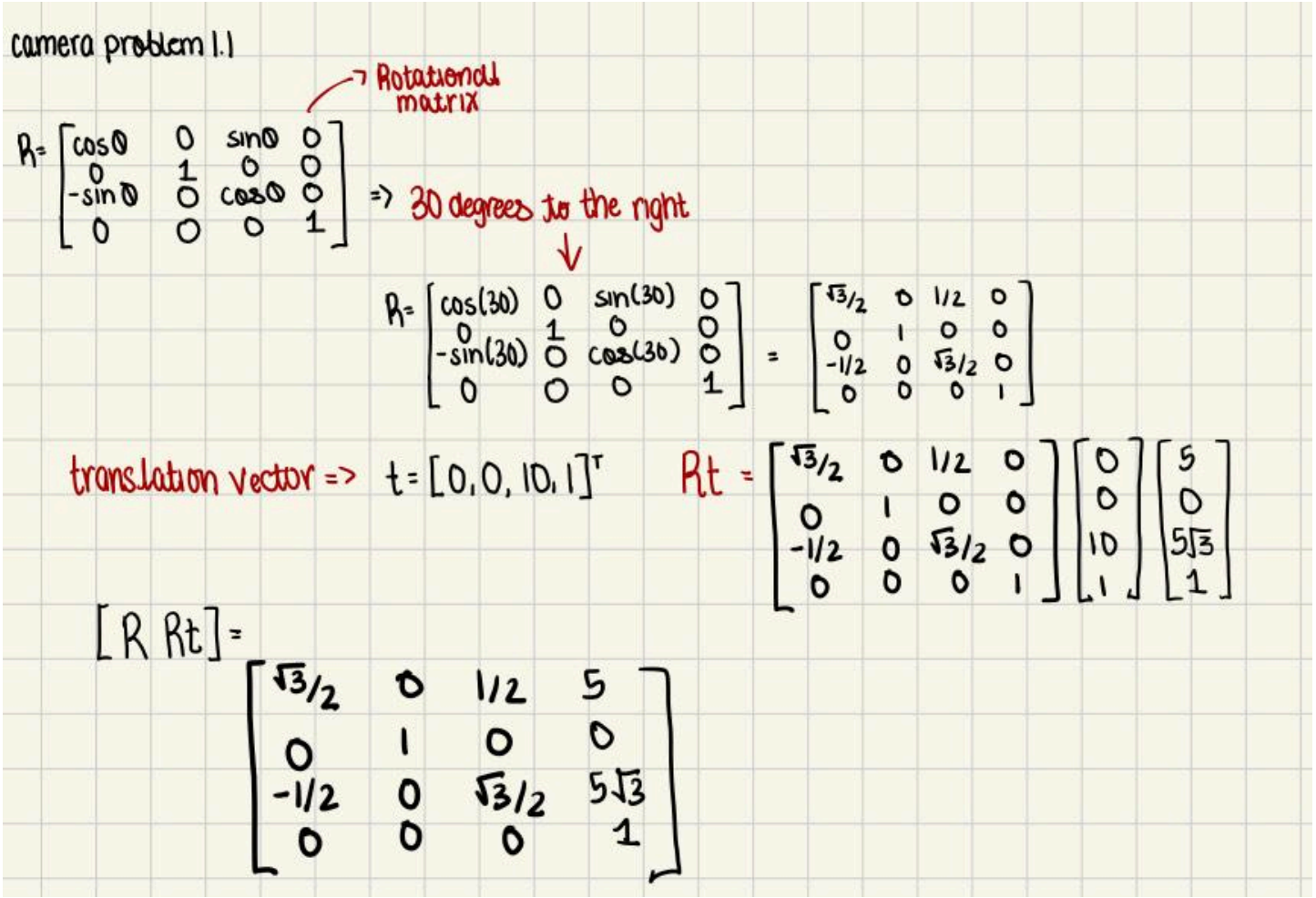
AIML331 A1:

name: Nikita Xalxo
ID: 300651670
code for checkerboard: attached to submission

1.1 Camera Problem



1.2 Camera Problem

camera problem 1.2

focal length = 0.1

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

$x = K[R\ Rt]X$

Projection 1: $[0,1,0] \rightarrow \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$ $\quad x_1 = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 5 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 5\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/10 \\ 5\sqrt{3} \end{bmatrix}$

Projection 2: $(0,1,0) \rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = X_2 = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \sqrt{3}/2 & 0 & 1/2 & 5 \\ 0 & 1 & 0 & 0 \\ -1/2 & 0 & \sqrt{3}/2 & 5\sqrt{3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 11/20 \\ 0 \\ 11\sqrt{3}/2 \end{bmatrix}$

$$X_1 \times X_2 = \begin{bmatrix} 1/2 \\ 1/10 \\ 5\sqrt{3} \end{bmatrix} \times \begin{bmatrix} 11/20 \\ 0 \\ 11\sqrt{3}/2 \end{bmatrix} = \begin{bmatrix} \frac{11\sqrt{3}}{20} \\ 0 \\ \frac{-11}{200} \end{bmatrix}$$

$$\frac{200}{11} \times \begin{bmatrix} \frac{11\sqrt{3}}{20} \\ 0 \\ \frac{-11}{200} \end{bmatrix} = \begin{bmatrix} 10\sqrt{3} \\ 0 \\ -1 \end{bmatrix} = \therefore \begin{bmatrix} 10\sqrt{3} \\ 0 \\ -1 \end{bmatrix} x = 0$$

Checkerboard

2.1 Find an image on-line that displays a checkerboard (provide source website). Convert to gray-scale if it is colour. [1]

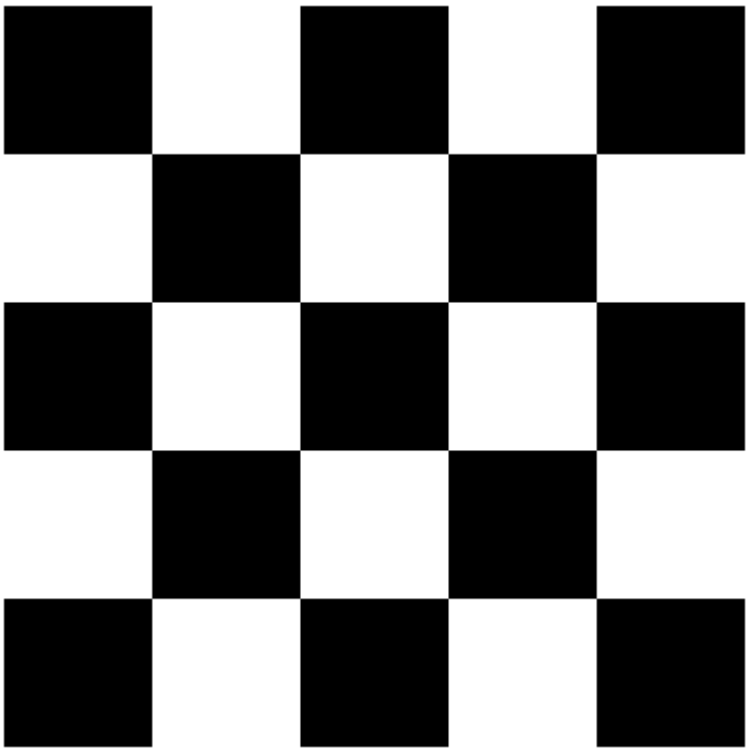
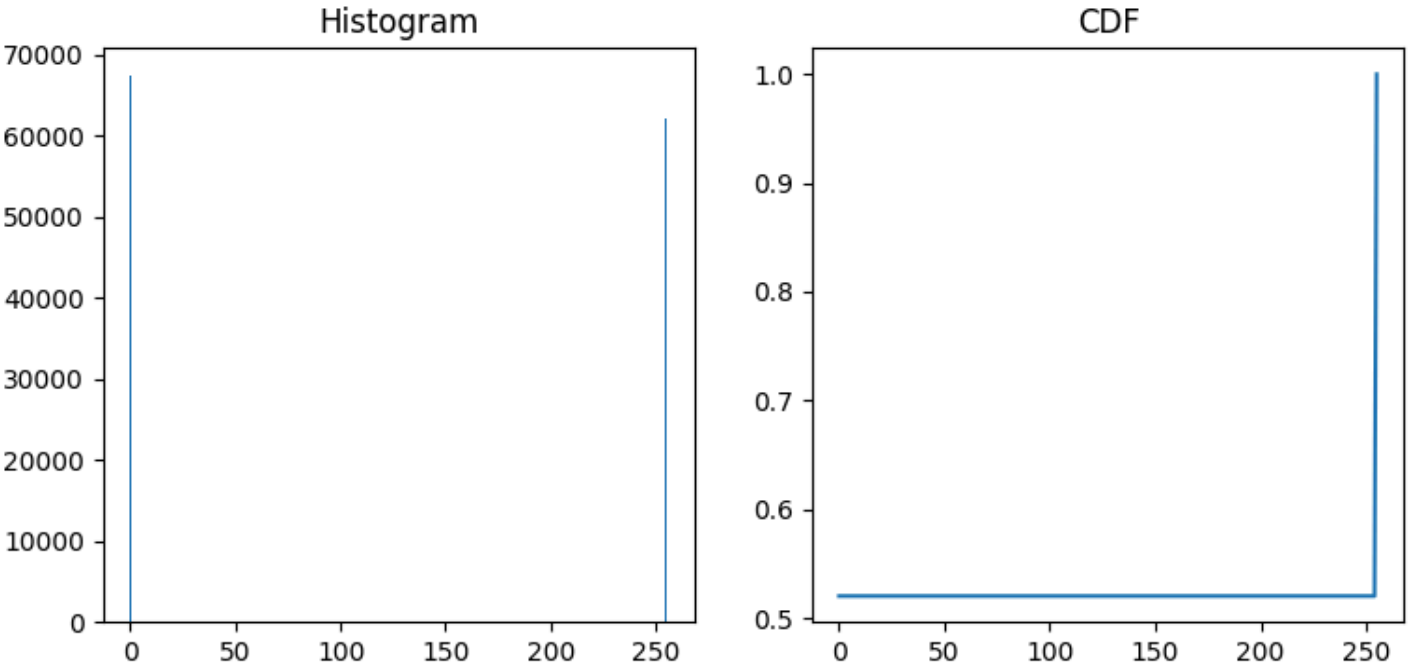


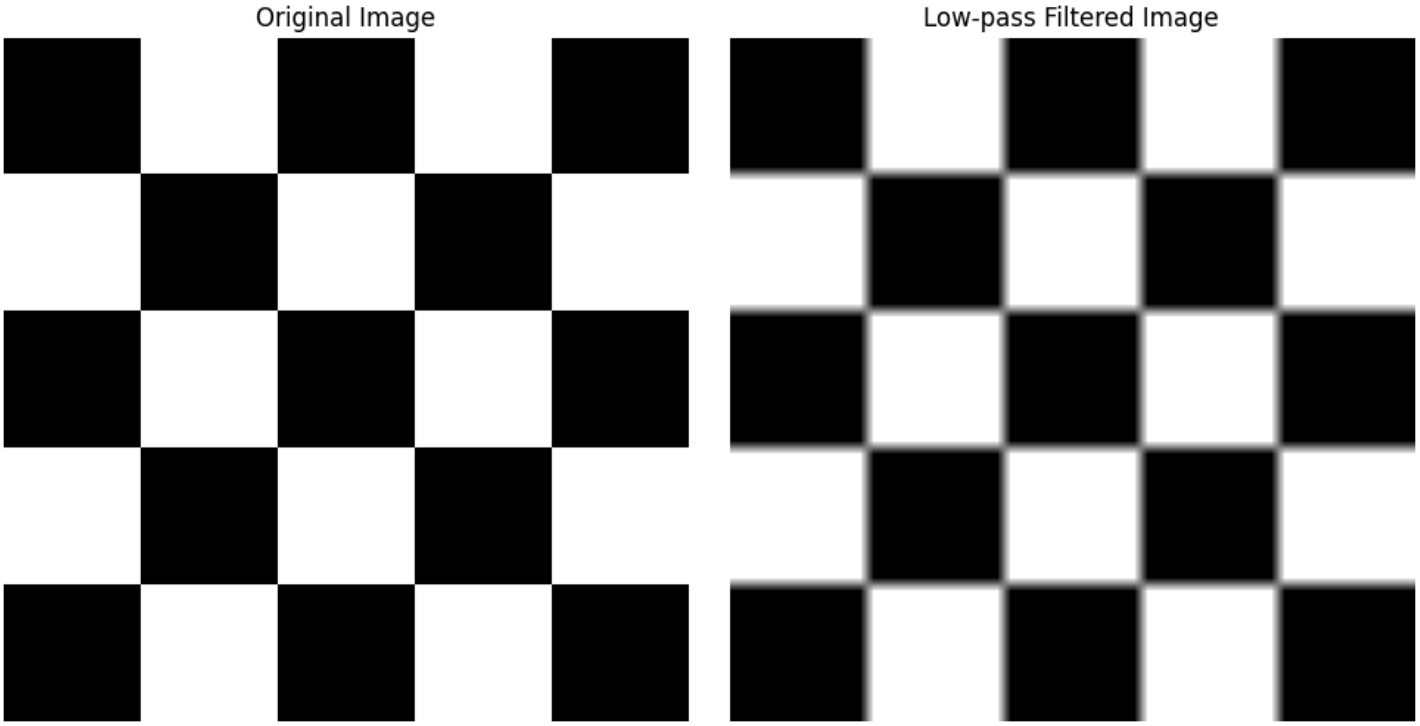
Image source: https://upload.wikimedia.org/wikipedia/commons/thumb/7/70/Checkerboard_pattern.svg/360px-Checkerboard_pattern.svg.png For this part, I obtained a checkerboard image from Wikipedia. The image URL is https://upload.wikimedia.org/wikipedia/commons/thumb/7/70/Checkerboard_pattern.svg/360px-Checkerboard_pattern.svg.png. The image is already in grayscale format with clear black and white squares, making it ideal for our analysis. I used the urllib.request library to download the image and matplotlib to display it. The resulting image shows a standard 8×8 checkerboard pattern with alternating black and white square**s.**

2.2 Write a program to compute the histogram and cumulative probability function of the image and display result. [5]Find an image on-line that displays a checkerboard (provide source website). Convert to gray-scale if it is colour. [1]



I implemented a program to compute the histogram and cumulative distribution function (CDF) of a checkerboard image. The histogram captures the frequency of pixel intensities, while the CDF represents the probability of a pixel having an intensity less than or equal to a given value. I achieved this by flattening the image, computing intensity frequencies using np.bincount, and normalizing the cumulative sum for the CDF. The histogram shows two distinct peaks at black and white intensity levels, while the CDF confirms the binary nature of the image with sharp transitions at these values.

2.3 Write a program to Write a program to low-pass filter (smooth) the image in the spatial domain. You can select the filter. [5]

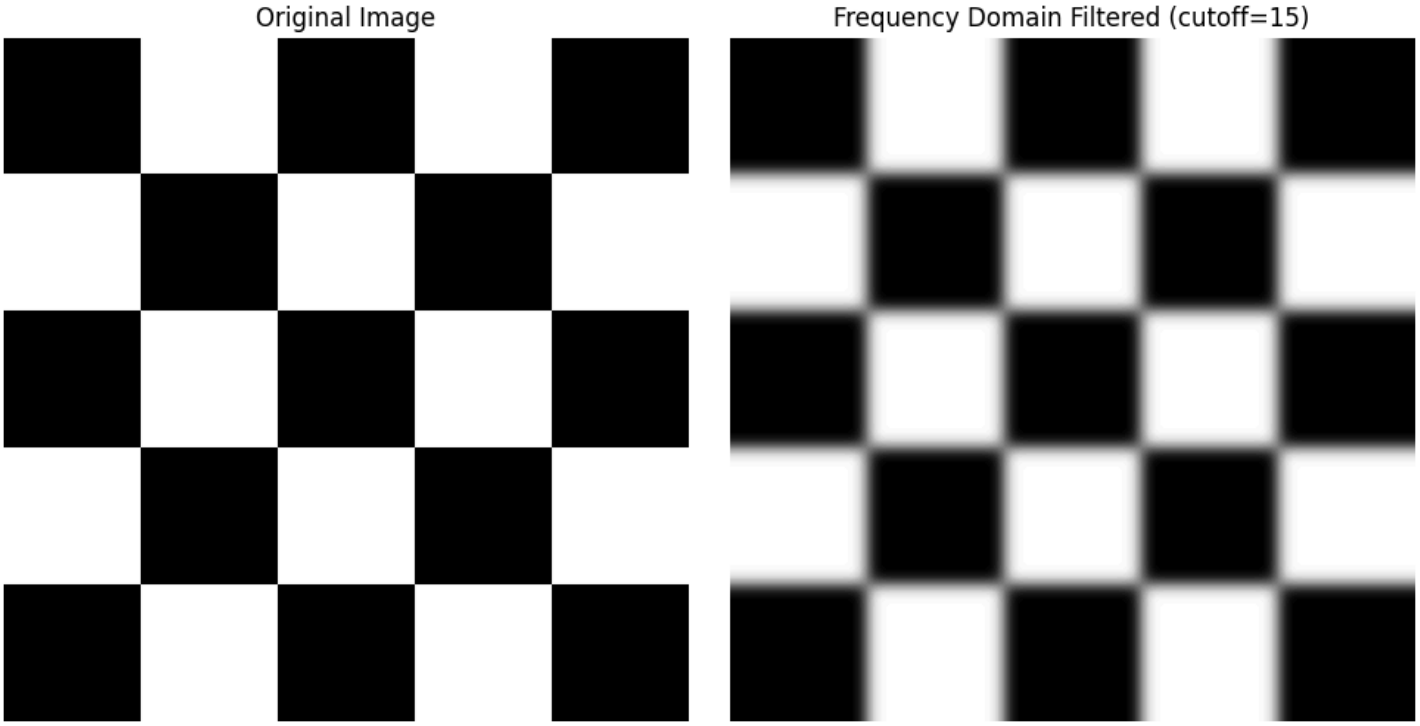


I implemented a low-pass filter (smoothing operation) in the spatial domain using a box filter (averaging filter) with a 7×7 kernel. This filter replaces each pixel with the average of its surrounding pixels within the filter window. The implementation involved creating a kernel of size 7×7 filled with values of 1/(7×7), then applying this kernel through spatial convolution. To handle edge effects, I used reflected padding around the image borders. The resulting filtered image shows significantly reduced sharp transitions between the black and white squares, creating a blurred appearance where the edges between squares are smoothed out.

2.4 Explain if the low-pass filter you selected is separable. [2]

The box filter used in this implementation is separable. This means it can be broken down into two separate 1D filters - one applied horizontally and one applied vertically. Rather than performing 49 operations per pixel with a 7×7 filter, separability allows us to perform only 14 operations (7 horizontal + 7 vertical). This reduces the computational complexity from $O(M \times N \times k^2)$ to $O(M \times N \times 2k)$, where M and N are the image dimensions and k is the filter size. This property makes the box filter computationally efficient for image processing tasks, especially with larger filter sizes

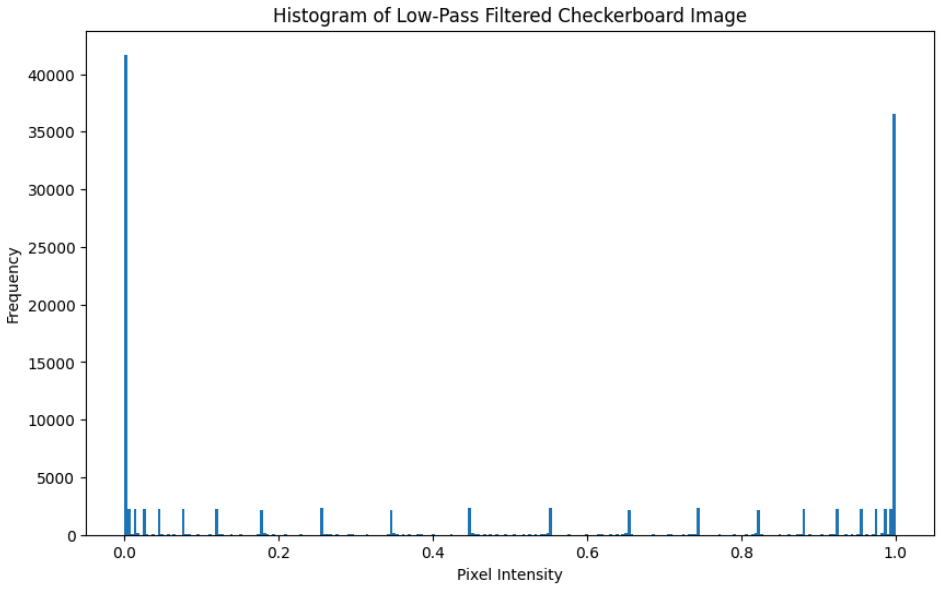
2.5 Write a program to perform the same low-pass filtering in the frequency domain (may use a library for the DFT/FFT operator). [7]



I implemented the same low-pass filtering operation in the frequency domain using Fourier transforms. This approach involved converting the image to the frequency domain using Fast Fourier Transform (FFT), applying a Gaussian low-pass filter mask in the frequency domain, and then converting back to the spatial domain using Inverse FFT. The implementation used numpy.fft functions to transform the image, and a Gaussian filter with a cutoff parameter to control the degree of smoothing. The cutoff value was set to 15, providing a medium blurring effect. The frequency domain approach produces smoother results than the spatial domain filter, with more natural-looking transitions between regions.

 # 2.6 Compute the histogram of the low-pass filtered checkerboard image. [3]

I computed and plotted the histogram of the frequency domain low-pass filtered checkerboard image. Unlike the original image's histogram that showed two distinct peaks, the filtered image's histogram reveals a more distributed range of intensity values. This occurs because the low-pass filter blends the sharp transitions between black and white squares, creating gradient areas with intermediate intensity values. The histogram shows higher frequencies at the extreme ends (near black and white) with a continuous distribution of values in between, reflecting the smoothing effect of the low-pass filter.



 # 2.7 Using cumulative distribution functions, convert your low-pass filtered checkerboard image to have a uniform probability of intensity levels. [7]

I implemented histogram equalization on the low-pass filtered checkerboard image to achieve a more uniform distribution of intensity values. The process involved computing the histogram, calculating the cumulative distribution function (CDF), and then using the CDF as a mapping function to redistribute pixel intensities. The equalized image shows enhanced contrast between different regions and more uniform utilization of the available intensity range. The histogram of the equalized image displays a more even distribution across the intensity spectrum compared to the pre-equalization histogram. This transformation makes subtle details in the gradient areas more apparent while preserving the overall structure of the filtered checkerboard.