# Aim****:****

**EX.NO:4 DATE: 06.02.2025**

**IMAGE**

To apply various image processing techniques, including histogram operations, image negation, blurring, edge detection, frequency domain transformation, thresholding, and contrast adjustment, to enhance or analyze images.

# Algorithm****:****

1. **Histogram Shifting & Equalization:**
   * Convert the image to grayscale.
   * Apply histogram shifting by adding or subtracting a fixed intensity value.
   * Apply histogram equalization to enhance contrast.
   * Display the original, shifted, and equalized images along with their histograms.
2. **Negative Image Transformation:**
   * Compute the negative of the image by subtracting each pixel value from the maximum intensity (255 for 8-bit images).
   * Display the original and negative images with their histograms.
3. **Image Blurring & Edge Detection:**
   * Apply different linear filters (Blur, Sharpening, Gaussian, Sobel X & Y).
   * Use convolution with predefined kernels to filter the image.
   * Display the original, blurred, sharpened, and edge-detected images along with their histograms.
4. **DFT (Discrete Fourier Transform):**
   * Convert the image to grayscale.
   * Perform the DFT to transform the image into the frequency domain.
   * Shift the zero-frequency component to the center.
   * Compute the magnitude spectrum and display the result.
5. **Thresholding:**
   * Convert the image to grayscale.
   * Apply different threshold values to convert the image into a binary form.
   * Display the original and thresholded images with histograms.
6. **Contrast Enhancement:**
   * Normalize pixel values by stretching the intensity range between the minimum and maximum values.
   * Display the original and contrast-enhanced images with their histograms.

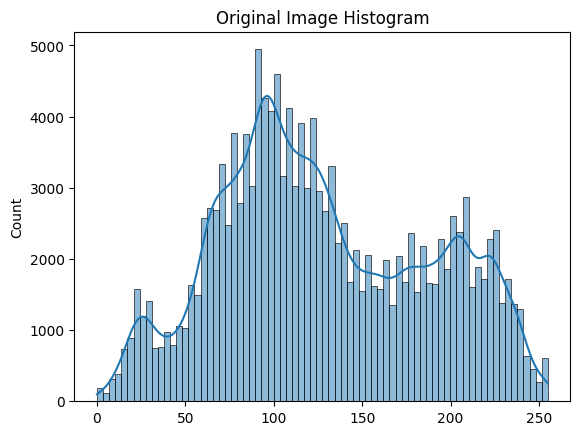
# Code:

import cv2  
from PIL import Image  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns

image = cv2.imread('lena.jpeg')  
image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

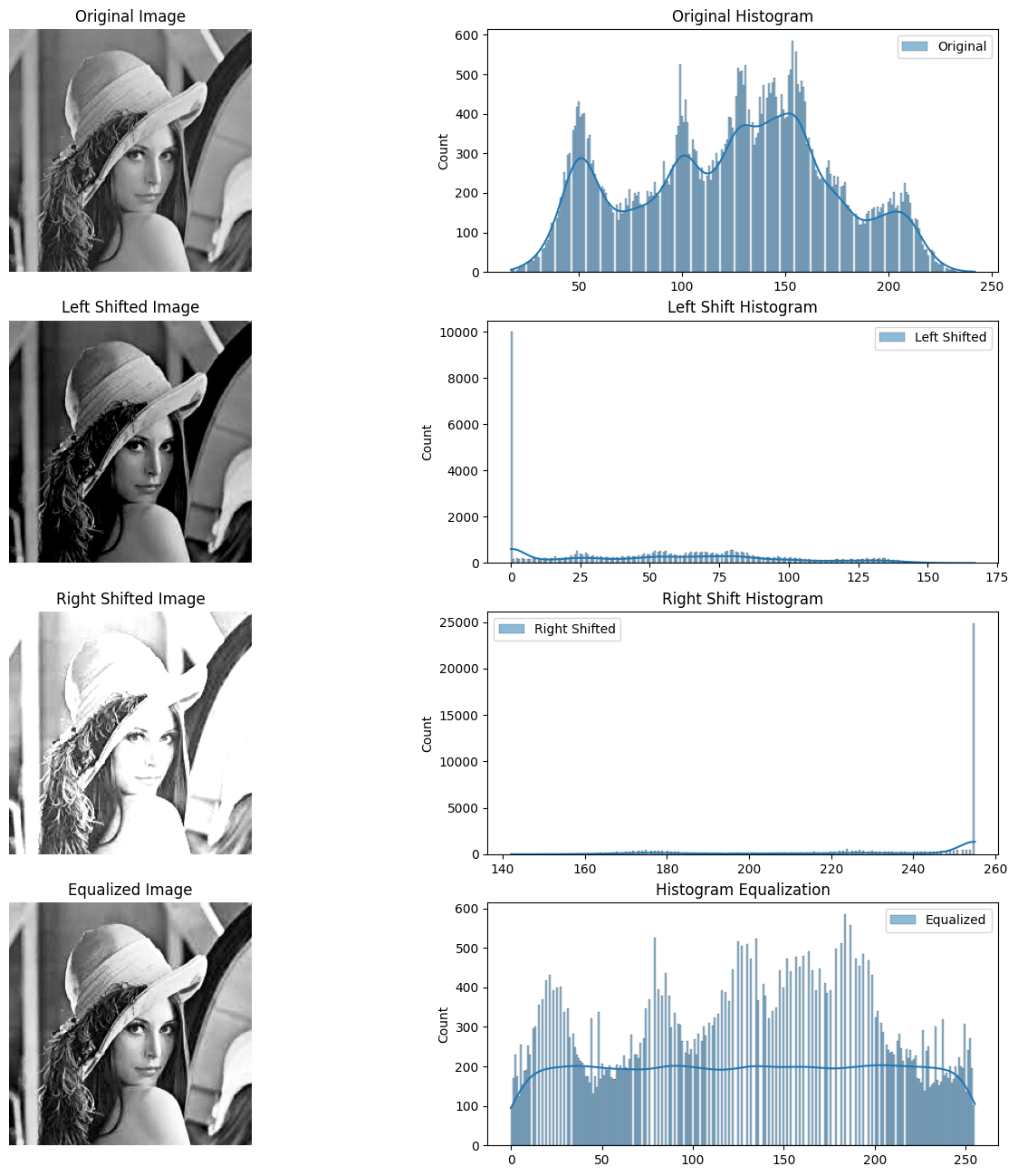
sns.histplot(image.flatten(),kde =True)  
plt.title('Original Image Histogram')

Text(0.5, 1.0, 'Original Image Histogram')



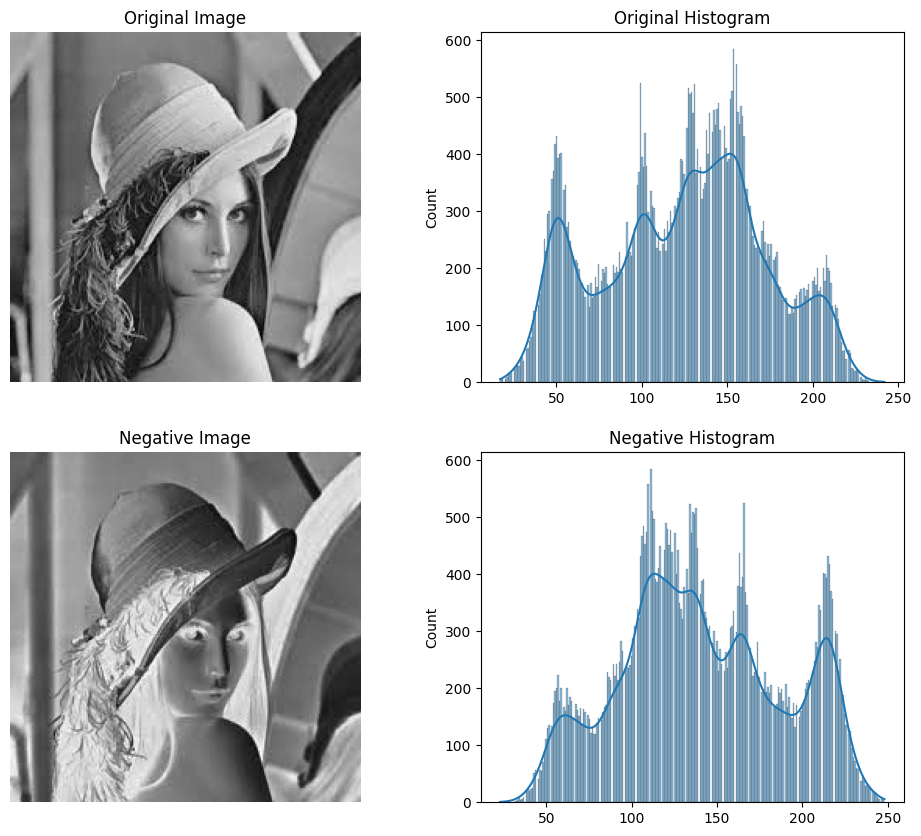
# Histogram Shifting and Equalization

def process\_histogram(image: np.ndarray, method='shift', shift\_value=50):  
   
 if method == 'shift':  
 if shift\_value >= 0:  
 processed\_image = cv2.add(image, shift\_value)  
 else:  
 processed\_image = cv2.subtract(image, abs(shift\_value))  
 elif method == 'equalize':  
 processed\_image = cv2.equalizeHist(image)  
 else:  
 raise ValueError("Invalid method. Use 'shift' or 'equalize'.")  
   
 return processed\_image  
  
image = cv2.imread("lena.jpeg")  
image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)  
image = cv2.cvtColor(image, cv2.COLOR\_RGB2GRAY)  
right\_shifted\_img = process\_histogram(image, method='shift', shift\_value=125)  
   
left\_shifted\_img = process\_histogram(image, method='shift', shift\_value=-75)  
  
equalized\_img = process\_histogram(image, method='equalize')  
  
plt.figure(figsize=(16, 16))  
  
plt.subplot(4, 2, 1)  
plt.imshow(image, cmap='gray')  
plt.title('Original Image')  
plt.axis('Off')  
plt.subplot(4,2,2)  
sns.histplot(image.ravel(), bins=256, alpha=0.5, label="Original", kde=True)  
plt.legend()  
plt.title("Original Histogram")  
  
plt.subplot(4, 2, 3)  
plt.imshow(left\_shifted\_img, cmap='gray')  
plt.title('Left Shifted Image')  
plt.axis('Off')  
plt.subplot(4,2,4)  
sns.histplot(left\_shifted\_img.ravel(), bins=256, alpha=0.5, label="Left Shifted", kde=True)  
plt.legend()  
plt.title("Left Shift Histogram")  
  
plt.subplot(4, 2, 5)  
plt.imshow(right\_shifted\_img, cmap='gray')  
plt.title('Right Shifted Image')  
plt.axis('Off')  
plt.subplot(4,2,6)  
sns.histplot(right\_shifted\_img.ravel(), bins=256, alpha=0.5, label="Right Shifted", kde=True)  
plt.legend()  
plt.title("Right Shift Histogram")  
  
plt.subplot(4, 2, 7)  
plt.imshow(equalized\_img, cmap='gray')  
plt.title('Equalized Image')  
plt.axis('Off')  
plt.subplot(4,2,8)  
sns.histplot(equalized\_img.ravel(), bins=256, alpha=0.5, label="Equalized", kde=True)  
plt.legend()  
plt.title("Histogram Equalization")  
plt.show()



# Negative image

L = 2 ^ 8  
S = (L-1) -image  
plt.figure(figsize=(12, 10))  
plt.subplot(2, 2, 1)  
plt.imshow(image, cmap='gray')  
plt.title('Original Image')  
plt.axis('Off')  
  
plt.subplot(2,2,2)  
sns.histplot(image.ravel(), bins=256, alpha=0.5, kde=True)  
plt.title("Original Histogram")  
  
plt.subplot(2, 2, 3)  
plt.imshow(S, cmap='gray')  
plt.title('Negative Image')  
plt.axis('Off')  
  
plt.subplot(2,2,4)  
sns.histplot(S.ravel(), bins=256, alpha=0.5, kde=True)  
plt.title("Negative Histogram")  
plt.show()

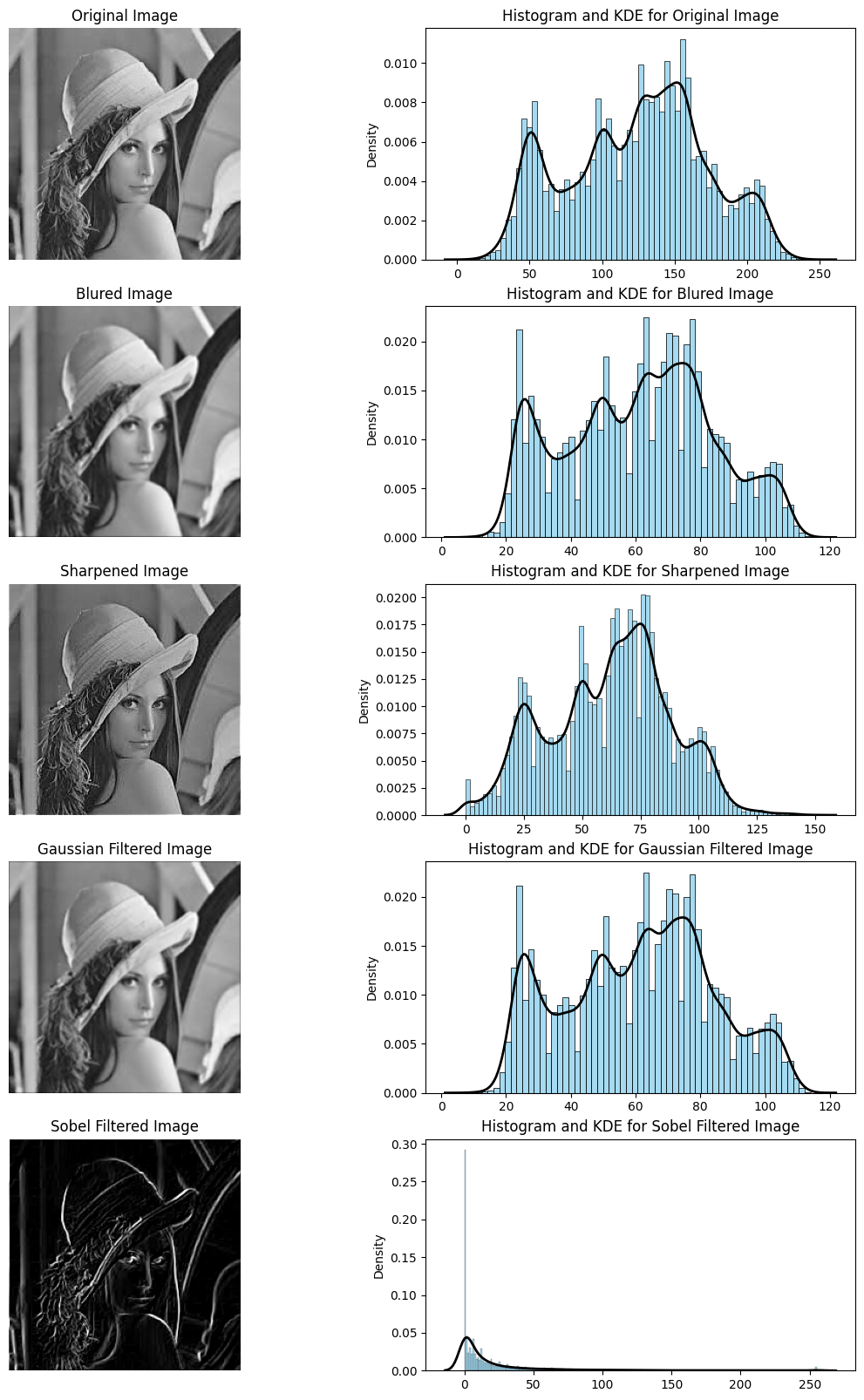


# Image Blurring

def linear\_filters(img , Filter):  
   
 h,w=img.shape  
 bordered\_img = np.zeros((h+2,w+2) , dtype=np.uint8)  
 bordered\_img[ 1:h+1 , 1:w+1 ] = img  
  
 kernel = 0  
 match Filter:  
 case 'blur':  
 Kernel=np.ones((3,3))/9   
 case 'sharp':  
 Kernel=np.array([[0,0,0],[0,2,0],[0,0,0]]) -( np.ones((3,3))/9)  
 case 'Gx':  
 Kernel=np.array([[-1,0,1],[-2,0,2],[-1,0,1]])   
 case 'Gy' :   
 Kernel=np.array([[-1,-2,-1],[0,0,0],[1,2,1]])  
 case 'Gaussian':  
 Kernel=np.array([[0.075, 0.125, 0.075],[0.125, 0.200, 0.125],[0.075, 0.125, 0.075]])  
 case \_:  
 raise ValueError("The value of argument Filter is wrong")  
  
 filtered\_img = np.zeros\_like(img)  
 for i in range(1,h+1):  
 for j in range(1,w+1):  
 window = bordered\_img[ i-1:i+2 , j-1:j+2 ]  
 matrix = Kernel \* window  
 avg = np.sum(matrix)//2  
 filtered\_img[i-1,j-1] = max(0 , min(255 , avg))  
 return filtered\_img

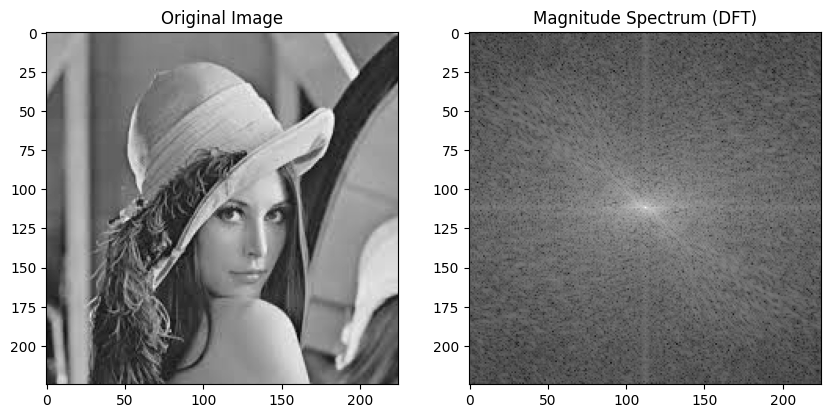
def Sobel\_filter(img):  
 Gx = linear\_filters(img , 'Gx').astype(np.int64)  
 Gy = linear\_filters(img , 'Gy').astype(np.int64)  
 G = np.clip(np.sqrt(Gx\*\*2 + Gy\*\*2) , 0 ,255)  
 return G.astype(np.uint8)

img = cv2.imread("lena.jpeg",0)  
fig , axes= plt.subplots(5,2 , figsize = (14,20))  
  
img\_lst = [img ,linear\_filters(img,'blur') ,linear\_filters(img,'sharp') , linear\_filters(img,'Gaussian') , Sobel\_filter(img)]  
  
title = ['Original' , 'Blured' , 'Sharpened' , 'Gaussian Filtered' , 'Sobel Filtered']  
  
for i in range(5):  
 axes[i,0].imshow(img\_lst[i] , cmap='gray')  
 axes[i,0].axis('off')  
 axes[i,0].set\_title(f'{title[i]} Image')  
 sns.histplot(img\_lst[i].ravel() , kde = False , ax=axes[i,1] , color='skyblue' , stat='density')  
 sns.kdeplot(img\_lst[i].ravel() , linewidth=2 , color='black' , ax=axes[i,1] )  
 axes[i,1].set\_title(f'Histogram and KDE for {title[i]} Image')



# DFT

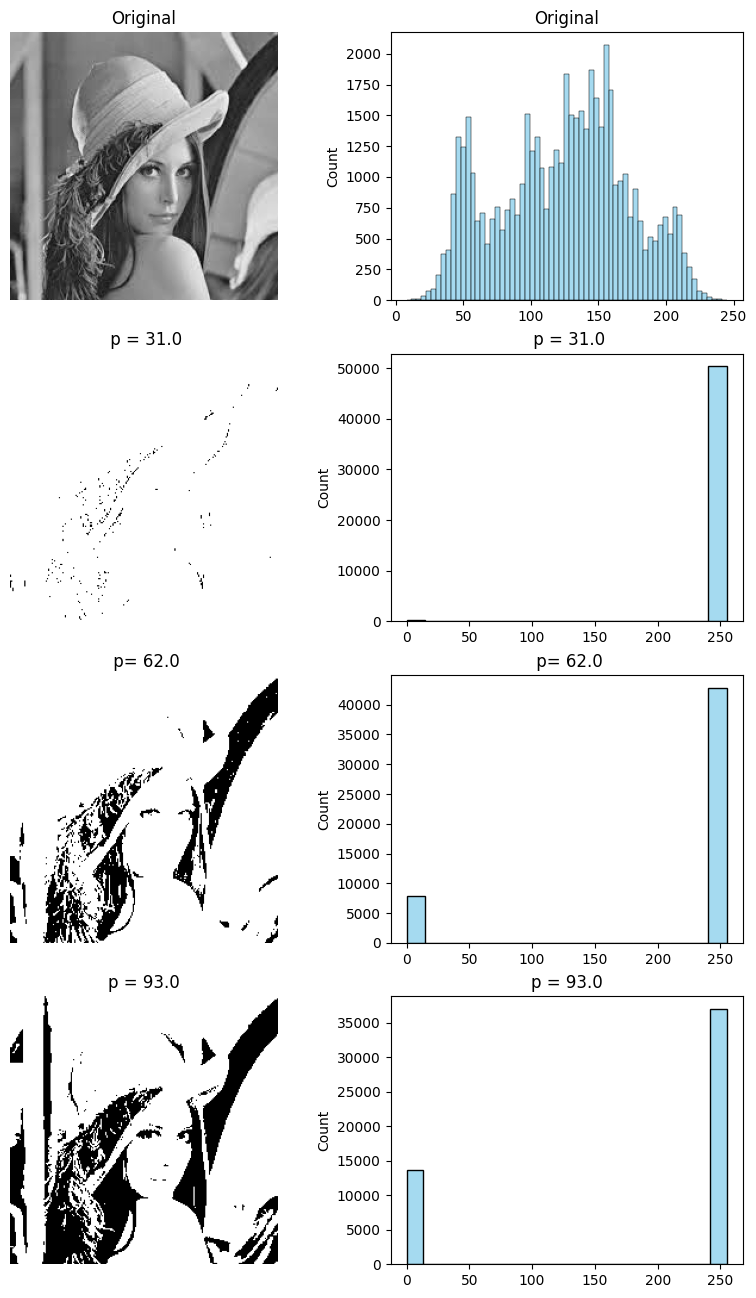
def dft\_image(image\_path):  
   
 img = cv2.imread(image\_path, 0)  
   
 dft = np.fft.fft2(img)  
 dft\_shifted = np.fft.fftshift(dft)  
   
 magnitude\_spectrum = np.log(np.abs(dft\_shifted) + 1)  
   
 plt.figure(figsize=(10, 5))  
 plt.subplot(1, 2, 1)  
 plt.imshow(img, cmap='gray')  
 plt.title('Original Image')  
   
 plt.subplot(1, 2, 2)  
 plt.imshow(magnitude\_spectrum, cmap='gray')  
 plt.title('Magnitude Spectrum (DFT)')  
   
 plt.show()  
  
dft\_image('Lena.jpeg')



# Thresholding

def bw\_transform(img , val):  
 bw\_img = np.zeros\_like(img)  
 bw\_img[img > val] = 255  
 return bw\_img

fig , axes = plt.subplots(4,2 , figsize = (10,16))  
  
mean =np.round( np.mean(img))  
  
lst = [img , bw\_transform(img , mean/4) , bw\_transform(img , mean/2) , bw\_transform(img , 3\*mean/4)]  
  
title = ['Original',f' p = {mean//4}',f' p= {mean//2}',f'p = {3 \*mean//4}']  
  
for i in range(4):  
 axes[i,0].imshow( lst[i] , cmap='gray')  
 axes[i,0].set\_title(title[i])  
 axes[i,0].axis('off')  
 sns.histplot(lst[i].ravel() , kde = False , ax=axes[i,1] , color='skyblue')  
 axes[i,1].set\_title(title[i])



# Contrast

fig , axes= plt.subplots(2,2 , figsize = (14,10))  
  
title= ['Original','Contract']  
  
Min,Max = np.min(img),np.max(img)  
  
contrast\_img=img.copy().astype(np.int64)  
contrast\_img=(contrast\_img-Min)\*255/(Max-Min)  
contrast\_img=contrast\_img.astype(np.uint8)  
  
  
lst = [img , contrast\_img]  
  
for i in [0,1 ]:  
 axes[i,0].imshow(lst[i] , cmap='gray')  
 axes[i,0].set\_title(f' ({title[i]}) Image')  
 axes[i,0].axis('off')  
 sns.kdeplot(lst[i].ravel() , linewidth=2 , color='black' , ax=axes[i,1] )  
 sns.histplot(lst[i].ravel() , kde = False , ax=axes[i,1] ,stat='density' , color='skyblue')  
 axes[i,1].set\_title(f'Histogram and KDE of ({title[i]}) Image')

