Image restoration using constrained least square filter

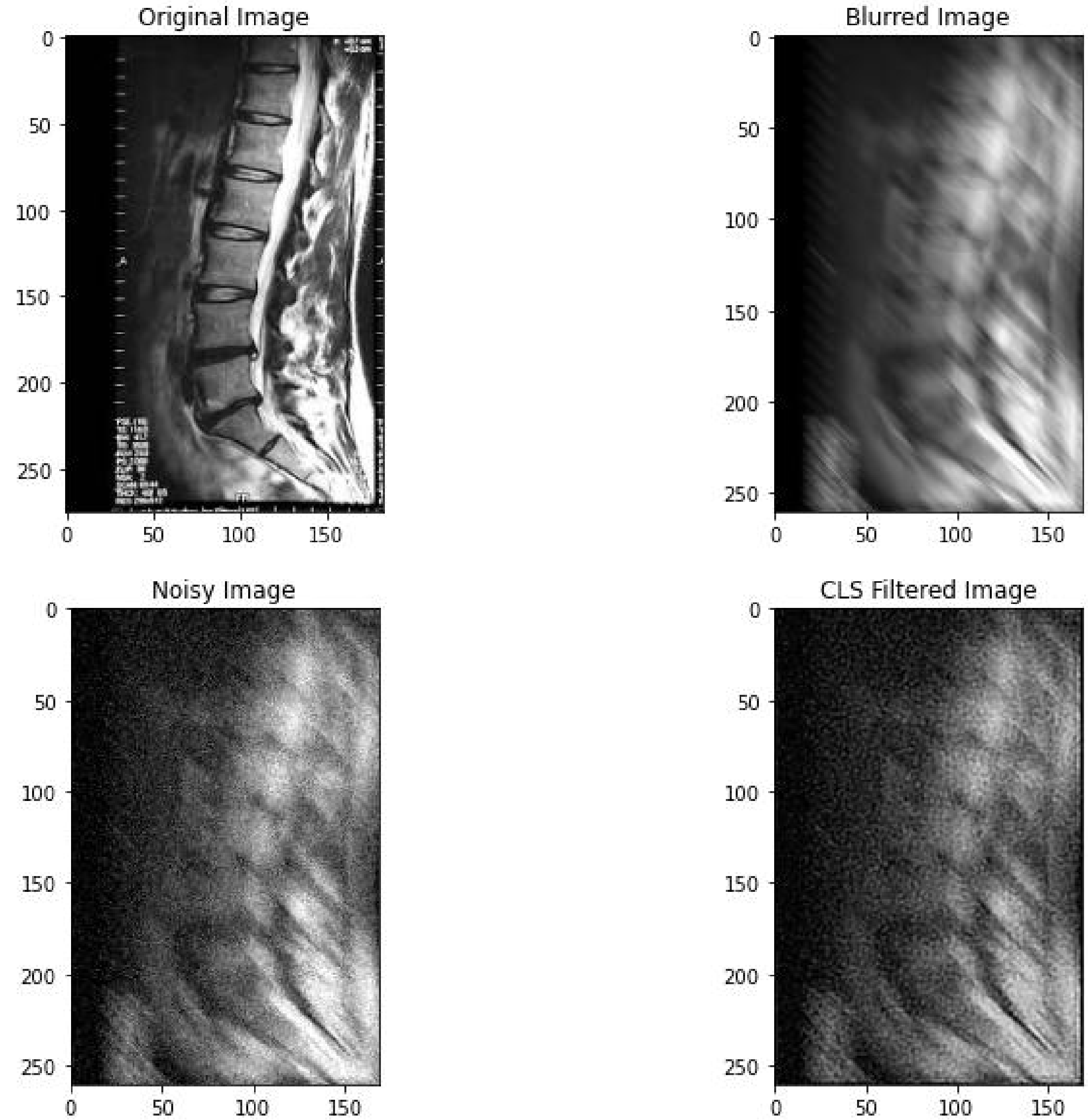
GANDEPALLI DHNAUSH M24EE002

CODE :-

```
import os
import cv2
import numpy as np
from numpy.fft import fft2, ifft2, ifftshift
from scipy.signal import gaussian, convolve2d
import matplotlib.pyplot as plt

filename = r"C:\Users\KIIT\Desktop\images.jpeg"
img = cv2.imread(filename)
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

def blur(img, kernel_size=3):
   dummy = np.copy(img)
   h = np.eye(kernel_size) / kernel_size
   dummy = convolve2d(dummy, h, mode='valid')
   return dummy

def add_gaussian_noise(img, sigma):
   gauss = np.random.normal(0, sigma, np.shape(img))
   noisy_img = img + gauss
   noisy_img[noisy_img < 0] = 0
   noisy_img[noisy_img > 255] = 255
   return noisy_img

def laplacian_operator(shape):
   laplacian = np.zeros(shape)
   laplacian[shape[0]//2-1:shape[0]//2+2,           shape[1]//2-1:shape[1]//2+2]           =
np.array([[0, -1, 0], [-1, 4, -1], [0, -1, 0]])
   return fft2(laplacian, s=shape)

def cls_filter(img, kernel, alpha):
   # Fourier transform of the input image
   img_fft = fft2(img)

   # Fourier transform of the kernel, padded to the image size
   kernel_fft = fft2(kernel, s=img.shape)

   # Laplacian operator in the frequency domain (used as regularization)
   laplacian_fft = laplacian_operator(img.shape)
```

```
    # CLS filter formula:
    # F_hat = (H*F) / (|H|^2 + alpha*|P|^2), where H is the kernel, P is the Laplacian operator
    denominator = np.abs(kernel_fft) ** 2 + alpha * np.abs(laplacian_fft) ** 2
    cls_result_fft = np.conj(kernel_fft) * img_fft / denominator

    # Inverse Fourier transform to get the filtered image
    cls_result = np.abs(ifft2(cls_result_fft))
    return cls_result

def gaussian_kernel(kernel_size=3):
    h = gaussian(kernel_size, kernel_size / 3).reshape(kernel_size, 1)
    h = np.dot(h, h.transpose())
    h /= np.sum(h)
    return h

if __name__ == '__main__':

    # Apply blur
    blurred_img = blur(img, kernel_size=15)

    # Add Gaussian noise
    noisy_img = add_gaussian_noise(blurred_img, sigma=20)

    # Apply Constrained Least Squares (CLS) Filter to restore image
    kernel = gaussian_kernel(3)
    alpha = 0.01  # Regularization parameter for controlling the balance between fidelity and smoothing
    restored_img = cls_filter(noisy_img, kernel, alpha)

    # Prepare images for display
    display = [img, blurred_img, noisy_img, restored_img]

    # Ensure we have a label for each image
    lable = ['Original Image', 'Blurred Image', 'Noisy Image', 'CLS Filtered Image']

    # Plot the images
    fig = plt.figure(figsize=(12, 10))

    for i in range(len(display)):
        fig.add_subplot(2, 2, i+1)
        plt.imshow(display[i], cmap='gray')
        plt.title(lable[i])

    plt.show()
```

Image :-


Original Image
Blurred Image
Noisy Image
CLS Filtered Image
0
50
100
150
200
250