Image restoration using inverse filter

Gandepalli dhanush M24EE002

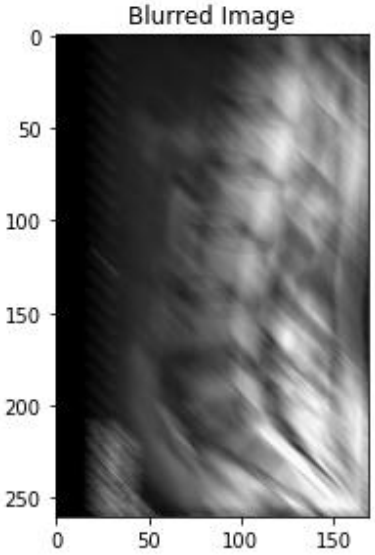
CODE :-

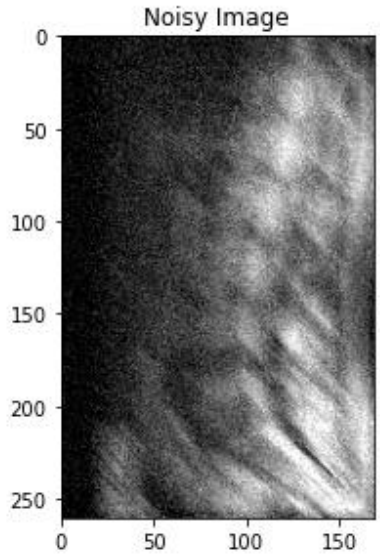
```
import os
import cv2
import numpy as np
from numpy.fft import fft2, ifft2, ifftshift
from scipy.signal import gaussian, convolve2d
import matplotlib.pyplot as plt

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 inverse_filter(img, kernel):
    # Take FFT of the image
    img_fft = fft2(img)

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

    # Avoid division by zero
    kernel_fft = np.where(kernel_fft == 0, 1e-8, kernel_fft)

    # Perform inverse filtering (dividing the FFTs)
    result_fft = img_fft / kernel_fft

    # Get the result by performing the inverse FFT
    result = np.abs(ifft2(result_fft))
    return result
```

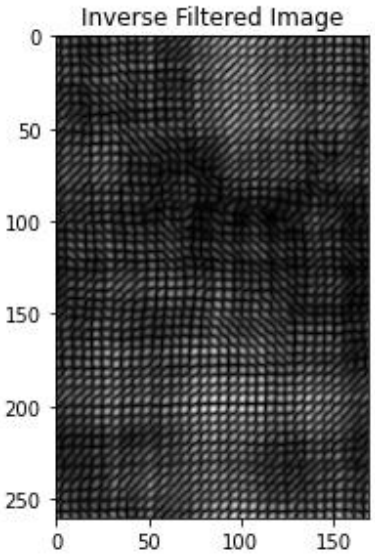
```
def gaussian_kernel(kernel_size=3):
    h = gaussian(kernel_size, kernel_size / 3).reshape(kernel_size, 1)
    h = np.dot(h, h.transpose())
    h /= np.sum(h)
    return h

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 inverse filter to restore image
    kernel = gaussian_kernel(3)
    restored_img = inverse_filter(noisy_img, kernel)

    # 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', 'Inverse 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
0
50
100
150
200
250
0
50
100
150


Blurred Image
0
50
100
150
200
250
0
50
100
150


Noisy Image
0
50
100
150
200
250
0
50
100
150


Inverse Filtered Image
0
50
100
150
200
250
0
50
100
150