DIP Assignment 6

CS20B1012
Muhammad Fazil K

Problem statement:

1. Download Lena color image, convert it to grayscale image and add salt and pepper noise with noise quantity 0.1,0.2 up to 1 and generate 10 noisy images.
2. Correlate each noisy image with Gaussian filters of varying size. Filter size can be 3 x 3, 5 x 5 and 7 x 7.

Code :

```
#CS20B1012
#Muhammad Fazil K

import cv2
import numpy as np
import random
import matplotlib.pyplot as plt


def correlation(img, kernel):
    # Get the dimensions of the input image and the kernel
    img_height, img_width = img.shape
    kernel_height, kernel_width = kernel.shape

    # Compute the number of padding pixels
    pad_height = kernel_height // 2
```

```
    pad_width = kernel_width // 2

    # Create a new image to hold the correlation result
    result = np.zeros((img_height, img_width), dtype=np.float32)

    # Pad the input image with zeros
    padded_img = np.pad(img, ((pad_height, pad_height),
                        (pad_width, pad_width)), 'constant')

    # Perform the correlation operation
    for x in range(pad_height, img_height + pad_height):
        for y in range(pad_width, img_width + pad_width):
            region = padded_img[x - pad_height:x +
                                pad_height + 1, y - pad_width:y +
pad_width + 1]
            result[x - pad_height, y - pad_width] = np.sum(region *
kernel)

    return result


# Load the color image
img = cv2.imread('Lena.png')

# Convert the color image to grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Generate 10 noisy images with increasing noise quantity
for i in range(1, 11):
    # Calculate the noise probability based on the current iteration
    noise_prob = i / 10.0

    # Create a copy of the grayscale image to add noise to
    noisy = gray.copy()

    # Generate salt and pepper noise
    for x in range(noisy.shape[0]):
        for y in range(noisy.shape[1]):
            if random.random() < noise_prob:
                noisy[x, y] = random.choice([0, 255])
```

```
    # Apply Gaussian filters of varying sizes to the noisy image
    ifiltered3 = cv2.GaussianBlur(noisy, (3, 3), 1)
    ifiltered5 = cv2.GaussianBlur(noisy, (5, 5), 1)
    ifiltered7 = cv2.GaussianBlur(noisy, (7, 7), 1)

    # Create a Gaussian kernel with sigma=1 and size 3x3
    kernel3 = cv2.getGaussianKernel(3, 1)
    kernel3 = np.outer(kernel3, kernel3)

    # Create a Gaussian kernel with sigma=1 and size 5x5
    kernel5 = cv2.getGaussianKernel(5, 1)
    kernel5 = np.outer(kernel5, kernel5)

    # Create a Gaussian kernel with sigma=1 and size 7x7
    kernel7 = cv2.getGaussianKernel(7, 1)
    kernel7 = np.outer(kernel7, kernel7)

    # Apply the kernels to the noisy image using the correlation function
    filtered3 = correlation(noisy, kernel3)
    filtered5 = correlation(noisy, kernel5)
    filtered7 = correlation(noisy, kernel7)

    # Display the grayscale, noisy, and filtered images using
matplotlib.pyplot
    fig, axs = plt.subplots(nrows=2, ncols=5, figsize=(15, 5))
    axs[0, 0].imshow(gray, cmap='gray')
    axs[0, 0].set_title('Grayscale')
    axs[0, 1].imshow(noisy, cmap='gray')
    axs[0, 1].set_title('Noisy')
    axs[0, 2].imshow(ifiltered3, cmap='gray')
    axs[0, 2].set_title('Filtered (3x3)')
    axs[0, 3].imshow(ifiltered5, cmap='gray')
    axs[0, 3].set_title('Filtered (5x5)')
    axs[0, 4].imshow(ifiltered7, cmap='gray')
    axs[0, 4].set_title('Filtered (7x7)')
    axs[1, 0].imshow(gray, cmap='gray')
    axs[1, 0].set_title('Grayscale')
    axs[1, 1].imshow(noisy, cmap='gray')
    axs[1, 1].set_title('Noisy')
```

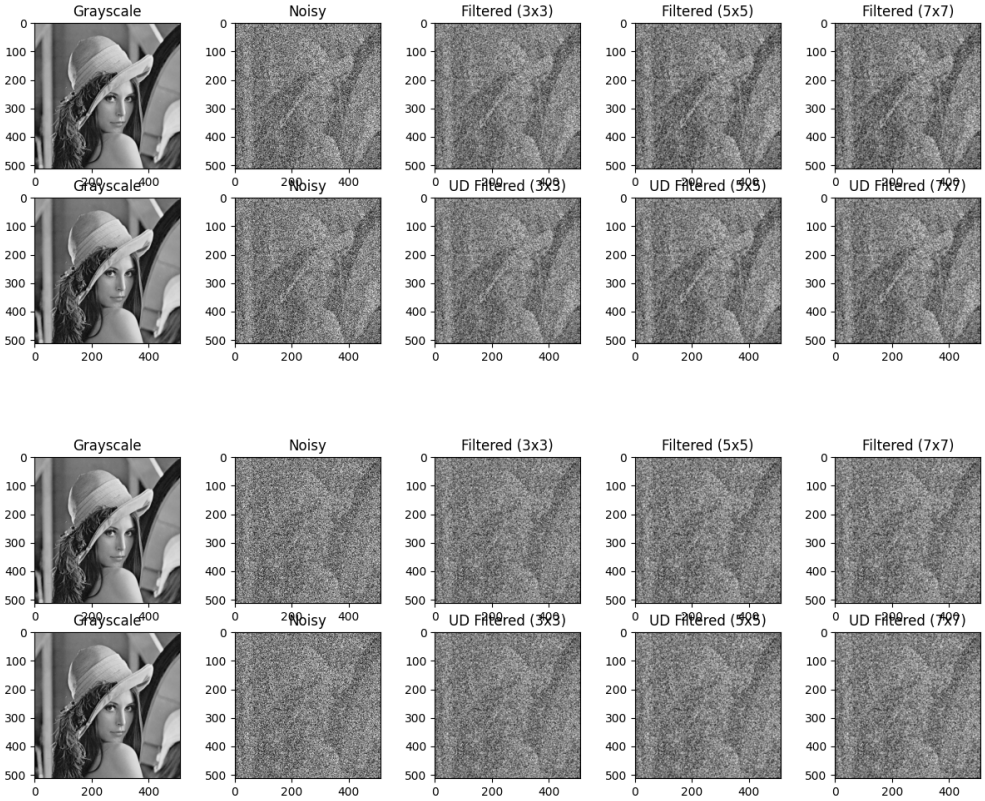
```
    axs[1, 2].imshow(filtered3, cmap='gray')
    axs[1, 2].set_title('UD Filtered (3x3)')
    axs[1, 3].imshow(filtered5, cmap='gray')
    axs[1, 3].set_title('UD Filtered (5x5)')
    axs[1, 4].imshow(filtered7, cmap='gray')
    axs[1, 4].set_title('UD Filtered (7x7)')
    plt.show()

    # Pause the display for 1 second (1000 milliseconds)
    plt.pause(1)
    plt.close()
```

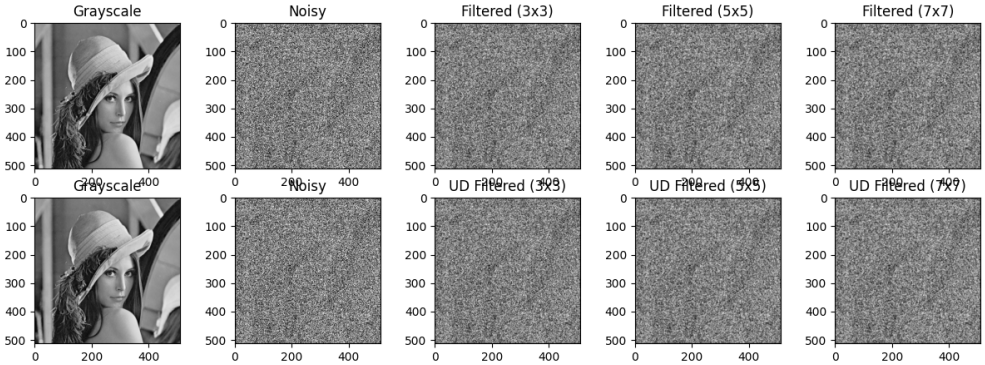
Output :

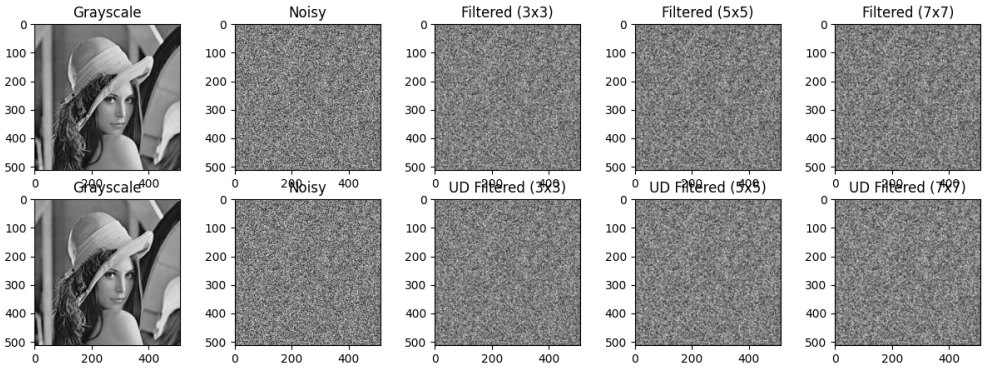

Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)
Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)
Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)


Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)
Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)
Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)


Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)

Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)


Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)


Grayscale
Noisy
Filtered (3x3)
Filtered (5x5)
Filtered (7x7)
Grayscale
Noisy
UD Filtered (3x3)
UD Filtered (5x5)
UD Filtered (7x7)