Experiment 8

Shashwat Shah
6000 4220126
BE comps C22

Aim: Justify: Average filter is used to remove salt and noise

Theory: Salt and Pepper noise also known as impulse noise is characterized by randomly occurring white and black pixels. An averaging filter, also known as the mean filter, is commonly employed in image processing to reduce noise. It works by replacing pixel values. This has a smoothening effect on the image, reducing variations between adjacent pixels and thus softening noise.

Justification

The averaging filter can smooth out some of the effects of salt and pepper noise by averaging the black and white noise pixels with surrounding pixels. However, it has significant drawbacks.

1) Blurring - The averaging filter tends to blur edges and fine details in the image, as it replaces each pixel with the average of its neighbours, even if there is no noise

2) Ineffectiveness with salt and pepper - Since salt and pepper involves extreme values (0 to 255) averaging with surrounding non-noise pixels often result in a loss of information. Instead of removing the noise entirely, it creates new pixel values that don't necessarily correspond to the original noise.

Conclusion: The use of an averaging filter to remove sa
& pepper noise is generally not ideal. While it
can reduce the visibility of the noise, it also causes
a significant loss of image detail due to the blurring
effect.

NAME: Shashwat Shah SAP ID: 60004220126
DIV/BATCH:

DIGITAL SIGNAL PROCESSING (DSP)
EXPERIMENT 08

AIM: To implement image smoothing / image sharping using low pass and high pass filter.

CODE:

IMAGE SMOOTHING USING LOW PASS FILTER

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

# Function to add Gaussian noise
def add_gaussian_noise(image, mean=0, sigma=25):
    noisy_image = image + np.random.normal(mean, sigma,
image.shape).astype(np.uint8)
    return np.clip(noisy_image, 0, 255)

# Function to apply low-pass averaging filter
def apply_averaging_filter(image, kernel_size=5):
    return cv2.blur(image, (kernel_size, kernel_size))

# Load an image
image = cv2.imread('chess.jpg', cv2.IMREAD_GRAYSCALE)

# Check if image was loaded
if image is None:
    print("Error: Image not found.")
else:
    # Add Gaussian noise to the image
    noisy_image = add_gaussian_noise(image)

    # Apply low-pass averaging filter to remove noise
    filtered_image = apply_averaging_filter(noisy_image)

    # Plot the images
    plt.figure(figsize=(12, 8))

    plt.subplot(1, 3, 1)
    plt.title('Original Image')
    plt.imshow(image, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 3, 2)
```

```
plt.title('Noisy Image')
plt.imshow(filtered_image, cmap='gray')
plt.axis('off')

plt.subplot(1, 3, 3)
plt.title('Filtered Image')
plt.imshow(noisy_image, cmap='gray')
plt.axis('off')

plt.tight_layout()
plt.show()
```

OUTPUT:

IMAGE SMOOTHING USING HIGH PASS FILTER

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

image_path = 'image3.jpg'
original_image = cv2.imread(image_path)
original_image = cv2.cvtColor(original_image, cv2.COLOR_BGR2RGB)

# Apply Gaussian Blur to create a low-pass filtered image
kernel_size = (5, 5)  # Size of the Gaussian kernel
low_pass = cv2.GaussianBlur(original_image, kernel_size, sigmaX=0)

# Create the high-pass filtered image
high_pass = cv2.subtract(original_image, low_pass)

# Sharpen the original image by adding the high-pass image
sharpened_image = cv2.add(original_image, high_pass)
plt.figure(figsize=(15, 5))

# Original image
plt.subplot(1, 3, 1)
plt.imshow(original_image)
plt.title('Original Image')
plt.axis('off')
```