LAB Manual

PART A

(PART A : TO BE REFFERED BY STUDENTS)

**Experiment No.04**

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case the there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.: C026 | Name: Anirbaan Ghatak |
| Class : B | Batch : B1 |
| Date of Experiment: | Date of Submission |
| Grade : |  |

**B.1 Software Code written by student:**

# Aim: Write a program to enhance the quality of an image by noise removal

# Name: Anirbaan Ghatak

# Roll No.: C026

import cv2

import numpy as np

def show\_image(image, title='image'):

    cv2.imshow(title, image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

image\_path = 'IMG\_2458.jpg'

image = cv2.imread(image\_path)

image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

image = cv2.resize(image, (500, 500))

show\_image(image)

def salt\_pepper(image):

    noisy\_imagesp = np.copy(image)

    num\_pixels = int(0.02 \* image.size)

    # Add salt noise (white pixels)

    salt\_coords = [np.random.randint(0, i - 1, num\_pixels) for i in image.shape]

    noisy\_imagesp[salt\_coords[0], salt\_coords[1]] = 255

    # Add pepper noise (black pixels)

    pepper\_coords = [np.random.randint(0, i - 1, num\_pixels) for i in image.shape]

    noisy\_imagesp[pepper\_coords[0], pepper\_coords[1]] = 0

    return noisy\_imagesp

#adding noise to the image

def addnoise(image):

    mean = 0

    stddev = 180

    noise = np.zeros(image.shape, np.uint8)

    cv2.randn(noise, mean, stddev)

    noisy\_img = cv2.add(image, noise)

    return noisy\_img

def high\_pass\_filter(noisy\_img, kernel\_size = 3):

    kernel = np.ones((kernel\_size, kernel\_size), np.float32) / (kernel\_size\*\*2)

    blurred = cv2.filter2D(noisy\_img, -1, kernel)

    hpf = noisy\_img - blurred

    return hpf

def low\_pass\_filter(image, kernel\_size=3):

    kernel = np.ones((kernel\_size, kernel\_size), np.float32) / (kernel\_size\*\*2)

    lpf = cv2.filter2D(image, -1, kernel)

    return lpf

def median\_filter(noisy\_img, kernel\_size=3):

    median\_filter = cv2.medianBlur(noisy\_img, kernel\_size)

    return median\_filter

noisy\_image = addnoise(image)

show\_image(high\_pass\_filter(noisy\_image, 2), 'High Pass Filter')

show\_image(low\_pass\_filter(noisy\_image), 'Low Pass Filter')

show\_image(median\_filter(noisy\_image), 'Median Filter')

sp\_img = salt\_pepper(image)

show\_image(high\_pass\_filter(sp\_img, 3), 'Salt&Pepper High Pass Filter')

show\_image(low\_pass\_filter(sp\_img), 'Salt&Pepper Low Pass Filter')

show\_image(median\_filter(sp\_img), 'Salt&Pepper Median Filter')

**B.2 Input and Output:**

**Input Images: Noisy images as per prerequisite**

****

**Output Images:**

****

****

**Salt and Pepper Noise Image:**

****

****

****

**B.3 Observations and learning:**

***This experiment demonstrated noise removal techniques for enhancing image quality. It demonstrates the addition of both Gaussian and salt-and-pepper noise, followed by the application of high-pass, low-pass, and median filters. Through visual comparisons, the code effectively illustrates how these filters mitigate noise while preserving essential image features.***

**B.4 Conclusion:**

***The code's implementation of noise removal techniques and filtering shows their pivotal role in enhancing image clarity and quality. These methods are indispensable tools for optimizing image analysis and interpretation across diverse fields***

**B.5 Question of Curiosity**

***(To be answered by student based on the practical performed and learning/observations)***

Q1: Low pass filters are efficient and suitable to remove salt and pepper noise. Is this statement true or false? Justify.

Ans

False. Low-pass filters are not particularly efficient for removing salt-and-pepper noise. While they can help reduce certain types of noise, such as Gaussian noise, they are less effective at handling salt-and-pepper noise. Salt-and-pepper noise involves random high-intensity (salt) and low-intensity (pepper) pixels, and low-pass filters might not completely remove these extreme pixel values, potentially leaving some noise behind. Median filters are more suitable for effectively eliminating salt-and-pepper noise without significantly blurring the image.

Q2: Correlate sharpening, smoothing, blurring, background elimination with types of filters you have studied.

Ans:

Sharpening: Correlated with High-Pass Filters, which enhance edges and details by emphasizing high-frequency components.

Smoothing/Blurring: Associated with Low-Pass Filters, which reduce noise and details by averaging or smoothing high-frequency components.

Background Elimination: Achieved through specialized filters like Median Filters, which effectively remove noise like salt-and-pepper noise while preserving the main image features.

Q3. List out and explain at least one real life application where these filtering techniques are useful.

Ans:

Medical Image Enhancement

Filtering techniques play a crucial role in enhancing medical images like X-rays, MRIs, and CT scans. For instance, low-pass filters are used to reduce noise and create smoother images, aiding radiologists in identifying subtle structures. High-pass filters can emphasize critical features, helping detect anomalies like tumors. Median filters are employed to remove salt-and-pepper noise, enhancing the clarity of medical images for accurate diagnosis and treatment planning. These techniques ensure better visibility of details and contribute to more accurate medical assessments.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*