Practical Work 11

Using Gaussian Filter to Reduce Noise in Digital Images

1. Objectives of this Laboratory Work

- Understand the concept and mathematical basis of the Gaussian filter.
- Learn how Gaussian smoothing reduces noise in digital images.
- Compare Gaussian filtering with other smoothing techniques.
- Implement Gaussian filtering using Python and OpenCV.
- Evaluate the effect of different kernel sizes and standard deviations (σ) on image quality.

2. Theoretical Background

Noise is an unavoidable component in digital images and can arise from multiple sources such as sensor imperfections, transmission errors, or environmental conditions. Image denoising is a fundamental process in image enhancement to improve the quality and readability of images for further analysis.

The **Gaussian filter** is one of the most widely used techniques for image smoothing and noise reduction. It is a type of **low-pass filter** that suppresses high-frequency noise while retaining low-frequency information such as general image shapes and smooth transitions.

Gaussian Function

The Gaussian function in two dimensions is defined as:

$G(x,y)=12\pi\sigma 2e-x2+y22\sigma 2$

Where:

- x,y: spatial coordinates from the center pixel
- σ\sigmaσ: standard deviation controlling the spread (or blur strength)

The function produces a bell-shaped curve, giving higher weights to pixels near the center and smaller weights to pixels farther away. This weighting ensures smooth transitions without introducing harsh edges.

Working Principle

Gaussian filtering operates by **convolving** the image with a Gaussian kernel. Each output pixel is calculated as the weighted average of its neighborhood, where weights follow the Gaussian distribution. The filter smooths variations and reduces random noise while maintaining structural information.

Key effects of Gaussian filtering:

- Reduces high-frequency (noise) components.
- Slightly blurs edges but less than a mean filter.
- Controlled by kernel size and standard deviation (σ).
    - Larger σ → stronger blur
    - Smaller σ → minor smoothing

Comparison with Other Filters

Filter Type	Method	Advantage	Limitation	Use Case
Mean Filter	Average of neighborhood	Simple and fast	Blurs edges heavily	Basic denoising
Median Filter	Nonlinear median of values	Removes salt & pepper noise	Slower	Impulse noise removal
Gaussian Filter	Weighted average with Gaussian kernel	Smooths while preserving edges	Parameter tuning needed	General noise reduction

Applications of Gaussian Filtering

- Preprocessing for edge detection (Canny, Laplacian, Sobel)
- Image denoising in digital photography
- Medical image enhancement
- Feature extraction in computer vision

- Smoothing textures in face and object recognition

3. Practical Section: Implementing Gaussian Filtering in Python

Task 1: Applying Gaussian Filter to a Grayscale Image

Objective:

Reduce random noise from a grayscale image using Gaussian blur.

Instructions:

1. Load a grayscale image.
2. Apply Gaussian filtering with different kernel sizes.
3. Display and compare the results.

Code Example:

```
import cv2
import matplotlib.pyplot as plt

# Load grayscale image
img = cv2.imread('/content/sample_data/lena.png', cv2.IMREAD_GRAYSCALE)

# Apply Gaussian blur with different kernel sizes
blur3 = cv2.GaussianBlur(img, (3,3), 0)
blur7 = cv2.GaussianBlur(img, (7,7), 0)
blur11 = cv2.GaussianBlur(img, (11,11), 0)

# Display results
plt.figure(figsize=(12,4))
plt.subplot(1,4,1), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.subplot(1,4,2), plt.imshow(blur3, cmap='gray'), plt.title('3x3 Kernel')
plt.subplot(1,4,3), plt.imshow(blur7, cmap='gray'), plt.title('7x7 Kernel')
```

```
plt.subplot(1,4,4), plt.imshow(blur11, cmap='gray'), plt.title('11x11 Kernel')
plt.show()
```

Expected Outcome:

As the kernel size increases, the image appears smoother and noise decreases, but fine details may blur slightly.

Task 2: Effect of Standard Deviation (σ) on Filtering

Objective:

Observe how changing σ affects the smoothness of the image.

Instructions:

1. Use the same kernel size but vary σ values.
2. Compare results visually.

Code Example:

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('/content/sample_data/lena.png', cv2.IMREAD_GRAYSCALE)

# Apply Gaussian blur with different sigma values
blur_sigma1 = cv2.GaussianBlur(img, (9,9), 1)
blur_sigma3 = cv2.GaussianBlur(img, (9,9), 3)
blur_sigma5 = cv2.GaussianBlur(img, (9,9), 5)

plt.figure(figsize=(12,4))
plt.subplot(1,4,1), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.subplot(1,4,2), plt.imshow(blur_sigma1, cmap='gray'), plt.title('σ = 1')
plt.subplot(1,4,3), plt.imshow(blur_sigma3, cmap='gray'), plt.title('σ = 3')
plt.subplot(1,4,4), plt.imshow(blur_sigma5, cmap='gray'), plt.title('σ = 5')
plt.show()
```

Expected Outcome:

Larger σ values produce stronger blurring. Small σ values reduce mild noise without losing much detail.

Task 3: Removing Gaussian Noise

Objective:

Add artificial Gaussian noise to the image and then remove it using a Gaussian filter.

Code Example:

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

# Load image
img = cv2.imread('/content/sample_data/lena.png', cv2.IMREAD_GRAYSCALE)

# Add Gaussian noise
mean = 0
sigma = 25
gaussian_noise = np.random.normal(mean, sigma, img.shape)
noisy_img = cv2.add(img, gaussian_noise.astype('uint8'))

# Apply Gaussian filter
denoised = cv2.GaussianBlur(noisy_img, (5,5), 1)

# Display results
plt.figure(figsize=(10,4))
plt.subplot(1,3,1), plt.imshow(img, cmap='gray'), plt.title('Original')
plt.subplot(1,3,2), plt.imshow(noisy_img, cmap='gray'), plt.title('Noisy Image')
plt.subplot(1,3,3), plt.imshow(denoised, cmap='gray'), plt.title('Filtered (σ=1)')
```

```
plt.show()
```

Expected Outcome:

The Gaussian filter effectively reduces random noise while maintaining smooth transitions in brightness.

Task 4: Comparing Gaussian and Mean Filters

Objective:

Compare Gaussian filtering with the simple mean filter.

Code Example:

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('/content/sample_data/lena.png', cv2.IMREAD_GRAYSCALE)
noisy = cv2.GaussianBlur(img, (1,1), 10)  # Artificial blur to simulate noise

mean_filtered = cv2.blur(noisy, (5,5))
gaussian_filtered = cv2.GaussianBlur(noisy, (5,5), 1)

plt.figure(figsize=(10,4))
plt.subplot(1,3,1), plt.imshow(noisy, cmap='gray'), plt.title('Noisy')
plt.subplot(1,3,2), plt.imshow(mean_filtered, cmap='gray'), plt.title('Mean Filter')
plt.subplot(1,3,3), plt.imshow(gaussian_filtered, cmap='gray'), plt.title('Gaussian Filter')
plt.show()
```

Expected Outcome:

The Gaussian filter produces smoother, more natural results with fewer edge distortions than the mean filter.

Assignments for Students

Assignment Task 1 — Gaussian Smoothing with Different Parameters

Apply Gaussian filtering with different kernel sizes (3×3, 5×5, 7×7) and σ values (1, 2, 3). Compare results visually and discuss the effect of each parameter.

Assignment Task 2 — Noise Reduction Experiment

Add Gaussian noise to an image and apply Gaussian, median, and mean filters. Compare which filter performs best for noise removal and edge preservation.

Assignment Task 3 — Edge Detection Preprocessing

Use a Gaussian filter as a preprocessing step before applying Sobel edge detection. Observe how noise reduction affects edge detection accuracy.

Assignment Task 4 — Quantitative Evaluation

Compute Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) before and after Gaussian filtering to evaluate noise reduction performance.

Control Questions

1. What is the main purpose of the Gaussian filter in image processing?
2. Explain the mathematical form of the Gaussian function.
3. What role does the standard deviation (σ) play in the Gaussian filter?
4. How does increasing kernel size affect smoothing performance?
5. Why is Gaussian filtering considered a low-pass operation?
6. Compare Gaussian and mean filters in terms of edge preservation.
7. What happens when σ is set too high?
8. Describe how Gaussian noise differs from salt-and-pepper noise.
9. Why is Gaussian filtering preferred before edge detection?
10. How does convolution implement the Gaussian smoothing process?
11. What is the effect of using different kernel sizes on image sharpness?
12. Can Gaussian filtering remove impulsive noise effectively? Why or why not?
13. How do you select appropriate σ and kernel size values in practice?
14. Give two real-world applications where Gaussian filtering is essential.