**EXPERIMENT 5**

|  |  |
| --- | --- |
| **Name: Shivam Bhosle** | **Roll No: 2** |
| **Subject: Computer Vision (CV)** | **Class/Batch: B1** |
| **Date of Performance:** | **Date of Submission:** |

|  |
| --- |
| **AIM** |

# **To Perform Noise reduction , image sharpening /blur using Gaussian Median Filtering**

|  |
| --- |
| **Theory** |

**1. Noise Reduction**

Noise in images refers to unwanted random variations in brightness or color, often due to poor lighting, sensor limitations, or transmission errors. Reducing noise is crucial for enhancing image quality without losing important details.

* **Median Filtering**: This is a non-linear filtering technique commonly used for noise reduction. The filter replaces each pixel's value with the median of the intensities of surrounding pixels. Since it effectively removes outliers without blurring edges, it's especially useful for reducing "salt and pepper" noise, where pixels randomly become very bright or dark.
* **Gaussian Filtering**: Gaussian filters are widely used for smoothing an image. It’s a linear filter that uses a Gaussian function to weigh the surrounding pixel values, resulting in the averaging of pixel intensities. Unlike median filtering, Gaussian filtering tends to blur edges slightly but does well in reducing continuous types of noise like Gaussian noise, typically found in sensor noise.

**2. Image Sharpening**

Image sharpening enhances edges and fine details by increasing the contrast between neighboring pixels. It is used to highlight boundaries and make an image appear more distinct and focused.

* **Unsharp Masking (with Gaussian Blur)**: This popular technique for sharpening first blurs the image with a Gaussian filter, which produces a smoothed version of the image. The difference between the original image and the blurred version is then amplified and added back to the original image. This process emphasizes edges and creates a sharpening effect.
* **High-pass Filtering**: A variation of Gaussian filtering that enhances high-frequency content (edges and details). It is achieved by subtracting a blurred (low-pass filtered) version of the image from the original, highlighting rapid intensity changes (edges).

**3. Blurring**

Blurring smooths out details in an image, reducing noise or making the image look softer. It's also useful in reducing image artifacts or pixelation.

* **Gaussian Blurring**: This is one of the most common techniques for image blurring. It works by averaging pixels using weights from a Gaussian distribution. The result is a smoothly blurred image where the degree of blurring can be controlled by adjusting the filter size and variance of the Gaussian function. This method retains the overall structure of the image but smooths out fine details.
* **Median Filtering for Blurring**: While primarily used for noise reduction, median filtering can also introduce a mild blurring effect. Since it replaces each pixel's value with the median from its neighborhood, it softens the image while preserving sharp transitions (such as edges).

**Summary of Techniques:**

* **Noise Reduction**:
  + **Median Filtering** for removing salt-and-pepper noise while preserving edges.
  + **Gaussian Filtering** for smoothing continuous noise at the cost of slightly blurring edges.
* **Image Sharpening**:
  + **Unsharp Masking** combines Gaussian blurring with a sharpening step.
  + **High-pass Filtering** enhances details by emphasizing high-frequency content.
* **Blurring**:
  + **Gaussian Blurring** provides a smooth, natural blur by averaging pixel values with a Gaussian weight.
  + **Median Filtering** can also result in mild blurring while maintaining edge clarity.

**Procedure**

**Removal of Noise from image using filtering**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Read the image

img = cv2.imread(r"C:\\Users\\Shivam 007\\Downloads\\images (1).jpeg")

# Apply median blur for noise removal

median = cv2.medianBlur(img, 5)

# Concatenate original and filtered images for comparison

compare = np.concatenate((img, median), axis=1)

# Convert from BGR to RGB for displaying with plt (matplotlib uses RGB format)

compare\_rgb = cv2.cvtColor(compare, cv2.COLOR\_BGR2RGB)

# Display the image using matplotlib

plt.figure(figsize=(10,5))

plt.imshow(compare\_rgb)

plt.title('Original Image (Left) vs Denoised Image (Right)')

plt.axis('off') # Hides the axis

plt.show()

**Image blurring using filtering**

import cv2

import numpy as np

import matplotlib.pyplot as plt

# Read the image

image = cv2.imread(r"C:\\Users\\Shivam 007\\Downloads\\Sports Law.jpeg")

# Gaussian Blur

Gaussian = cv2.GaussianBlur(image, (7, 7), 0)

# Median Blur

median = cv2.medianBlur(image, 5)

# Convert from BGR to RGB for displaying with matplotlib

image\_rgb = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

Gaussian\_rgb = cv2.cvtColor(Gaussian, cv2.COLOR\_BGR2RGB)

median\_rgb = cv2.cvtColor(median, cv2.COLOR\_BGR2RGB)

# Plotting the images side by side

plt.figure(figsize=(15, 5))

plt.subplot(1, 3, 1)

plt.imshow(image\_rgb)

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 3, 2)

plt.imshow(Gaussian\_rgb)

plt.title('Gaussian Blurring')

plt.axis('off')

plt.subplot(1, 3, 3)

plt.imshow(median\_rgb)

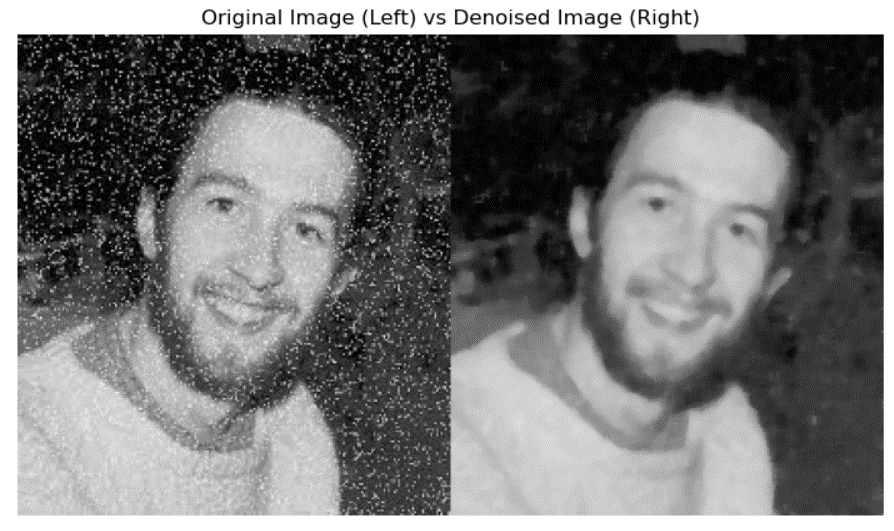
plt.title('Median Blurring')

plt.axis('off')

plt.show()

**Output**

**Removal of Noise from image using filtering**



**Image blurring using filtering**



|  |
| --- |
| **Conclusion** |

From this experiment we have successfully learned how to perform noise reduction , image sharpening /blur using Gaussian Median Filtering

|  |
| --- |
| **Assessment** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Timely Submission**  **(7)** | **Presentation**  **(06)** | **Understanding**  **(12)** | **Total**  **(25)** | **Sign** |
|  |  |  |  |  |