IPCV Practical No : 2

10:51

Aim

To study and implement **Smoothing (Low Pass)** and **Sharpening (High Pass)** filters in spatial domain using Python OpenCV.

Requirements

Python (OpenCV) installed
Sample digital images (grayscale or color)
Understanding of image noise types and spatial filtering

Theory

1. Introduction: Noise in Digital Images

Digital images often contain **noise**, i.e., unwanted variations in brightness or color.
Common types of noise:

- **Salt & Pepper Noise:** Appears as black and white dots.
- **Gaussian Noise:** Random variations following Gaussian distribution.
- **Speckle Noise:** Grainy noise in radar/medical images.

Purpose:

- To **remove noise → apply Smoothing filters**
- To **enhance details → apply Sharpening filters**

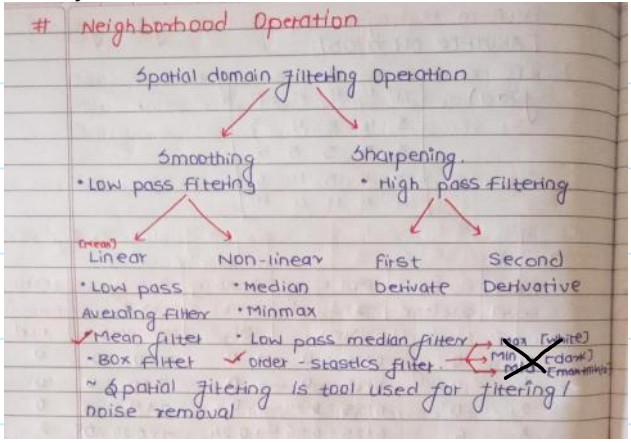
2. Spatial Filtering

Spatial Filtering = Modifying an image using a **kernel (mask/filter)** that slides over the image.
Key points:

- Output pixel depends on **neighborhood pixels**
- Common mask sizes: **3×3** or **5×5**

Operations:

- **Correlation:** Mask applied directly
- **Convolution:** Mask is flipped before application
- For symmetric masks, correlation ≈ convolution



3. Smoothing Filters (Low Pass / Noise Reduction)

Definition: Reduce noise by averaging/replacing a pixel with neighbors.
Preserves slow intensity variations but blurs sharp transitions.
Types:
Averaging Filter

- **Concept:** Each pixel is replaced by the average of its neighborhood.
- **Example 3×3 Kernel (all equal weights):**

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

- **Use:** Simple and effective for random noise.
- **Limitation:** Blurs image and edges.

Gaussian Blur

- **Concept:** Pixels are weighted using a Gaussian distribution (center pixel has higher weight).
- **Example 3×3 Kernel:**

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$$

- **Use:** Produces natural blurring, used in pre-processing for edge detection.

Median Filter

- **Concept:** Pixel is replaced by the **median value** of the neighborhood.
- **Example:**
  Neighborhood values: [100, 102, 250, 99, 101, 255, 98, 100, 102]
  Sorted: [98, 99, 100, 100, 101, 102, 102, 250, 255]
  Median = 101 → replaces center pixel.
- **Use:** Best for **salt-and-pepper noise**, preserves edges better than averaging.

Bilateral Filter (Edge-Preserving)

- **Concept:** Combines **spatial closeness** and **intensity similarity** to reduce noise but keep edges.
- **No fixed kernel** like Gaussian; weights are dynamically computed.
- **Example:** If two pixels are close in space but intensity differs a lot (like at an edge), weight is reduced → edge preserved.

Order-Statistic Filters

- **Concept:** Based on sorting neighborhood values.
- **Examples:**
  - **Median Filter:** (already explained)
  - **Max Filter:** Selects the **maximum value** in neighborhood → highlights bright regions.
  - **Min Filter:** Selects the **minimum value** in neighborhood → highlights dark regions.

4. Sharpening Filters (High Pass / Edge Enhancement)

Sharpening filters are used to enhance fine details and edges in an image. They work by emphasizing **high-frequency components** (edges, sudden changes in intensity) and suppressing slow intensity variations (smooth areas).

1. High Pass Filter (HPF)

- A **linear filter** that detects and enhances edges.
- It uses a mask where center pixel has a positive weight and surrounding pixels have negative weights.
- This emphasizes changes in intensity.

Working:

- Smooth regions (low frequency) → suppressed.
- Rapid intensity changes (high frequency like edges) → emphasized.

Applications:

Edge detection.
Medical and satellite image enhancement.

2. High Boost Filtering

A generalization of the high-pass filter.

- Instead of using only the high-frequency part, it **adds back some of the original image** to retain details.

Applications:

- Fingerprint analysis.
- Forensic image enhancement.
- Sharpening blurred photographs.

3. Laplacian Filter (Second Derivative Method)

The Laplacian filter is a **second-order derivative filter** used for **edge detection and image sharpening**. Unlike first-order derivatives (like Sobel/Prewitt), which highlight gradient directions, Laplacian highlights regions of **rapid intensity change in all directions**.

Mathematical Form:

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Mask Examples (Discrete Implementation):

- 4-neighbor:

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

- 8-neighbor:

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

Properties:

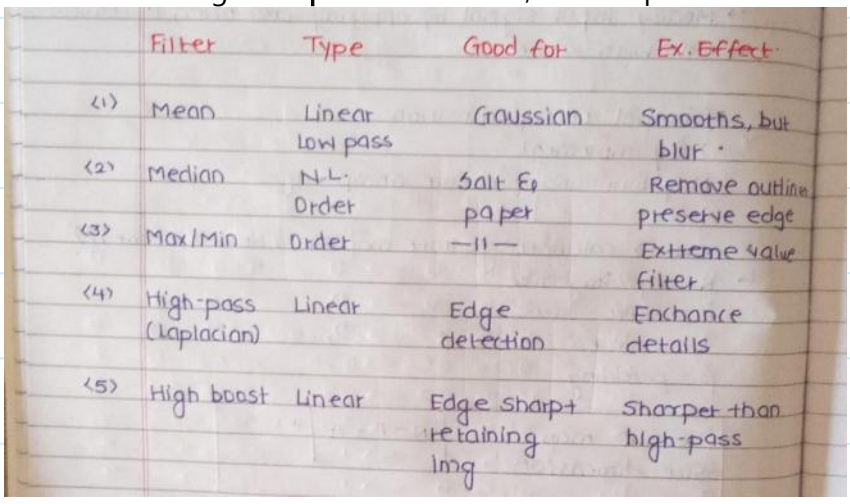
- Detects **edges and fine details**.
- Sensitive to noise (because noise also has rapid intensity changes).
- Often combined with smoothing filters (Gaussian + Laplacian → LoG filter).

Applications:

- Edge detection in medical images (tumor detection).
- Satellite imaging for terrain analysis.
- Handwriting and fingerprint recognition.

4. Gradient-based Filters (First Derivative)

- **Sobel Operator:** Uses weighted derivative in x and y directions.
- **Prewitt Operator:** Similar to Sobel, but simpler.
- These detect edges in **specific directions**, while Laplacian detects in **all directions**.



	Filter	Type	Good for	Ex. Effect
(1)	Mean	Linear Low pass	Gaussian	Smooths, but blur.
(2)	Median	N.L. Order	Salt & paper	Remove outline preserve edge
(3)	Max/Min	Order	—11—	Extreme value filter.
(4)	High-pass (Laplacian)	Linear	Edge detection	Enchance details
(5)	High boost	Linear	Edge sharpt retaining Img	Sharper than high-pass

5 .Difference Between Smoothing & Sharpening Filters

Aspect	Smoothing (Low Pass)	Sharpening (High Pass)
Purpose	Reduce noise, blur image	Enhance edges, fine details
Effect	Removes high-frequency details	Highlights high-frequency details
Example	Mean, Gaussian, Median, Bilateral	Laplacian, HPF, High-Boost
Applications	Pre-processing, denoising	Edge detection, feature extraction

Steps to Perform

1. Start
2. Import libraries (**OpenCV, NumPy, Matplotlib**)
3. Read a grayscale image
4. Apply **Smoothing Filters**: Averaging, Gaussian, Median, Bilateral
5. Apply **Sharpening Filters**: High Pass, High Boost, Laplacian
6. Display results
7. Stop

Output

Attach screenshots of Smoothing and sharpening filter

Conclusion

Thus, the program for implementing **Smoothing and Sharpening Filters** using Python and OpenCV was successfully executed, and the effects of noise reduction and edge enhancement were

verified.