What does Mean Filter do to Pixel Values?

Step 1 — What happens in smoothing?

- **Smoothing = Reducing sharp changes (reducing edges/noise).**
- It works by **averaging nearby pixels**.
- So if one pixel is **very bright (high value)** and its neighbors are **darker (low values)**, the mean filter will **reduce that pixel's value** to bring it **closer to the average**.
- Similarly, if one pixel is **very dark (low value)** and its neighbors are brighter, the filter will **increase that pixel's value**.

□ Mean Filter = Balance between High and Low Pixels

Pixel Type	Effect after Mean Filter
Very Bright Pixel (near dark ones)	Brightness **decreases**
Very Dark Pixel (near bright ones)	Brightness **increases**
Flat/Uniform Area	Very little change (values already close)

□ What does this mean for Edges?

- Edges are where **pixel values change sharply** (dark-to-bright or bright-to-dark).
- Mean filter **blurs** these edges by reducing the difference between edge pixels and their neighbors.
- This is why smoothing with a mean filter can **reduce edge sharpness** (edges become softer).

Filter

In **digital image processing**, a **filter** is used to modify or enhance an image by processing pixel values based on a specific rule or pattern.

Example:

A **blur filter** smooths an image by averaging the pixel values with its neighboring pixels.

Original:

```
[10, 20, 30]
[40, 50, 60]
[70, 80, 90]
```

Applying a 3x3 blur filter:

- New center value = (10 + 20 + 30 + 40 + 50 + 60 + 70 + 80 + 90) ÷ 9 = **50**

Result:

```
[10, 20, 30]
[40, 50, 60]
[70, 80, 90]
```

General Classification of Filters:

1. Linear Filter

A **linear filter** processes the image by applying a linear mathematical operation (like addition or multiplication) to the neighboring pixel values.

- The output is a weighted sum of the input pixel values.
- Examples: **Mean filter**, **Gaussian filter**, **Laplacian filter**

2. Nonlinear Filter

A **nonlinear filter** processes the image using a nonlinear operation (like finding the maximum, minimum, or median) on neighboring pixel values.

- Examples: **Median filter**, **Bilateral filter**

In the **spatial domain** → Convolution with a kernel (e.g., mean, Gaussian)

In the **frequency domain** → Multiplying the Fourier transform of the image with a filter (e.g., low-pass, high-pass)

Example:

- **Gaussian Filter** (spatial): Blurs the image by averaging neighboring pixels.
- **Low-pass Filter** (frequency): Removes high-frequency noise in the Fourier domain.

1. Blurring (Smoothing)

- Goal: Reduce noise and smooth the image.
- Method:
  - **Spatial domain** → Averaging or Gaussian filter (reduces pixel value variation).
  - **Frequency domain** → Low-pass filter (removes high-frequency components).

Effect: Loss of detail, reduced sharpness.

---

2. Sharpening

- Goal: Enhance edges and fine details.
- Method:
  - **Spatial domain** → Laplacian or Sobel filter (increases edge contrast).
  - **Frequency domain** → High-pass filter (retains high-frequency components).

Effect: Increased detail, better edge visibility.

Purpose of Sharpening Spatial Filters:

- Opposite of smoothing filters (which blur the image).
- Used to remove blurring and improve the visibility of edges and textures.
- Based on **first-order** and **second-order derivatives** (which measure how fast the pixel values change).

---

1. First-Order Derivative (Gradient-Based)

Measures the **rate of change** in pixel intensity — detects the presence of an edge.

Formula:

$f' = f(x+1) - f(x)$

where:

- $f(x)$ = pixel value at position x
- $f(x+1)$ = pixel value at next position

Properties:

 Zero in flat areas (no change in pixel intensity).
 Non-zero at the start of an edge (step change).
 Non-zero along a slope (gradual change).

First-Order Derivative in 2D (Edge Detection):

Common filters based on first-order derivatives:

- **Sobel filter**
- **Prewitt filter**
- **Roberts filter**

These filters calculate gradients in **both x and y directions** to detect edges.

2. Second-Order Derivative (Laplacian-Based)

Measures the **rate of change of the gradient** — highlights the exact location of edges.

Formula:

$f'' = f(x+1) + f(x-1) - 2f(x)$

Properties:

 Zero in flat areas (constant pixel intensity).
 Non-zero at the start **and end** of an edge (step change).
 Zero along gradual changes (ramps).

Second-Order Derivative in 2D (Edge Detection):

Common filters based on second-order derivatives:

- **Laplacian filter**
- **LoG (Laplacian of Gaussian)**

Summary :

1. First-Order Derivative → Edge Detection

- First-order filters (like **Sobel** or **Prewitt**) detect where the edges are by finding areas with a **sharp change** in pixel values.
- Output: Highlights edges but may not be sharp enough.

Example Workflow:

1. Apply a **Sobel filter** → Detect edges.
2. Apply a **Laplacian filter** → Sharpen the edges.
3. Combine the results → Enhanced, clear, and sharp edges.

2. Second-Order Derivative → Edge Sharpening

- Second-order filters (like **Laplacian**) enhance those detected edges by making them **sharper** and more defined.
- Output: Improves clarity and makes edges more prominent.

1. Flat Area (No Change in Brightness)

Example:

[10,10,10,10,10]

- All pixel values are the same → No change
- No edge or slope → Nothing to detect

□ First-order:

- 10−10=0 → No change → **Zero**

□ Second-order:

- 10+10−2(10)=20−20=0 → No change → **Zero**

✅Both first and second order = **0** (because there's no change).

□ 2. Step (Sudden Change in Brightness)

Example:

[10,10,50,50,50]

- Jump from **10 to 50** → Sharp edge
- First-order will **detect** the edge

- Second-order will **enhance** the edge

□ First-order:

- 50−10=40 → Detects edge ✅

□ Second-order:

- 50+10−2(10)=40 → Enhances edge ✅

□ 3. Ramp (Gradual Change in Brightness)

Example:

[10,20,30,40,50]

- Brightness increases gradually → Slope
- First-order detects the slope
- Second-order ignores it (because it's smooth)

□ First-order:

- 20−10=10, 30−20=10, 40−30=10 → Constant slope ✅

□ Second-order:

- 30+10−2(20)=40−40=0→ No sudden change → **Zero**

Flat Area → No enhancement

- No change → Both first and second derivatives are zero → No enhancement possible

Ramp → No enhancement

- Gradual change → First-order detects it, but second-order is zero → No enhancement

Step → Enhancement happens!

- Sudden change → First-order detects the edge → Second-order enhances it by increasing contrast → **Enhancement happens**

Example

First-Order Derivative → Edge Detection

- When you use a tool to **detect edges** in a photo (like "Find Edges" in Photoshop), it applies a first-order filter to identify the boundaries of objects.
  Example: Highlighting the outline of a face or object.

Second-Order Derivative → Edge Enhancement

- When you use a tool to **sharpen** a photo, it applies a second-order filter to enhance the edges by increasing contrast.
  Example: Making facial features sharper or improving text clarity in an image.