

# Image Filtering - Overview

## Key Concepts to Master

### 1. Linear Filtering (Convolution)

**CRITICAL: Must be able to do by hand!**

**Process:**

1. Place filter mask over image region
2. Multiply corresponding elements
3. Sum all products
4. Result is output pixel value

**Example: 3x3 Sobel filter**

```
Filter Gx:           Image patch:  
[-1  0  1]           [10  20  30]  
[-2  0  2]   *       [15  25  35]  
[-1  0  1]           [20  30  40]  
  
Result = (-1×10)+(0×20)+(1×30)+(-2×15)+(0×25)+(2×35)+(-1×20)+(0×30)+(1×40)  
= -10 + 0 + 30 - 30 + 0 + 70 - 20 + 0 + 40  
= 80
```

**Boundary handling:**

- Zero padding
- Replicate edges
- Symmetric/mirror

### 2. Median Filtering

**Non-linear filter - replaces with median value**

**Process:**

1. Extract neighborhood (e.g., 3x3)
2. Sort pixel values
3. Take middle value

**Example:**

```
Neighborhood: [10, 15, 20, 25, 30, 100, 35, 40, 45]  
Sorted:       [10, 15, 20, 25, 30, 35, 40, 45, 100]  
Median:      30 (removes outlier 100!)
```

### 3. Noise Types

Noise Type	Characteristics	Probability Distribution
<b>Gaussian</b>	Additive, normally distributed	Bell curve, mean $\mu$ , std $\sigma$
<b>Salt &amp; Pepper</b>	Random black/white pixels	Impulse noise at extremes

<b>Uniform</b>	Equal probability across range	Flat distribution
<b>Speckle</b>	Multiplicative noise	Common in ultrasound/radar

#### 4. Filter Type Comparison

Filter	Construction	Best For	Edge Preservation	Speed
<b>Mean</b>	All weights = 1/N	Gaussian noise	Poor - blurs edges	Fast
<b>Gaussian</b>	Weights from Gaussian function	Gaussian noise	Better than mean	Fast
<b>Median</b>	Non-linear, sort values	Salt & pepper	Excellent	Slower

#### 5. Mean Filter

**Box filter - all weights equal**

3×3 Mean filter:

```
[1 1 1]
[1 1 1] × (1/9)
[1 1 1]
```

**Properties:**

- Simple averaging
- Reduces Gaussian noise
- Blurs edges significantly

#### 6. Gaussian Filter

**Weights based on 2D Gaussian function:**

$$G(x,y) = (1 / (2\pi\sigma^2)) \times \exp(-(x^2 + y^2) / (2\sigma^2))$$

**Example 3×3 ( $\sigma=1$ , approximated):**

```
[1 2 1]
[2 4 2] × (1/16)
[1 2 1]
```

**Properties:**

- Weighted smoothing (center pixel weighted more)
- $\sigma$  controls amount of smoothing
- Separable: 2D filter = 1D horizontal × 1D vertical
- Better edge preservation than box filter

#### 7. Filter Effectiveness

**Gaussian Noise:**

- Mean filter: Good
- Gaussian filter: Better (weighted)

- Median filter: Poor (doesn't assume distribution)

#### Salt & Pepper Noise:

- Mean filter: Poor (spreads noise)
- Gaussian filter: Poor (spreads noise)
- Median filter: Excellent (removes outliers)

## MATLAB Quick Reference

```
% Linear filtering (convolution)
h = ones(3,3) / 9; % Mean filter
filtered = imfilter(img, h);

% Median filtering
filtered = medfilt2(img, [3 3]);

% Create Gaussian filter
h = fspecial('gaussian', [5 5], sigma);
filtered = imfilter(img, h);

% Add noise
noisy = imnoise(img, 'gaussian', 0, 0.01);
noisy = imnoise(img, 'salt & pepper', 0.05);

% Correlation (similar to convolution)
result = imfilter(img, h, 'corr');
```

## Hand Calculation Practice

### Problem 1: Mean filtering

```
Image patch:      3x3 Mean filter:
[10 20 30]      [1 1 1]
[40 50 60]  *  [1 1 1]  × (1/9)
[70 80 90]      [1 1 1]

Result = (10+20+30+40+50+60+70+80+90) / 9 = 50
```

### Problem 2: Gaussian approximation

```
Image patch:      Gaussian filter:
[10 20 30]      [1 2 1]
[40 50 60]  *  [2 4 2]  × (1/16)
[70 80 90]      [1 2 1]

Result = (1×10+2×20+1×30+2×40+4×50+2×60+1×70+2×80+1×90) / 16
        = (10+40+30+80+200+120+70+160+90) / 16
        = 800 / 16 = 50
```

## Study Checklist

- Can perform  $3 \times 3$  convolution by hand
- Can perform median filtering by hand
- Know construction of mean filter
- Know construction of Gaussian filter
- Can match filter type to noise type
- Understand why median preserves edges
- Know properties of each filter

## Common Exam Questions

1. "Apply this filter to this image patch" → Show work!
2. "Which filter for salt & pepper noise?" → Median
3. "Compare mean vs Gaussian filter" → Weighted vs uniform
4. "Why does median preserve edges?" → Non-linear, removes outliers

## Answers to Common Exam Questions

**1. "Apply this filter to this image patch"** Multiply each filter weight by the corresponding image pixel, sum all products. For a  $3 \times 3$  mean filter on patch [10 20 30; 40 50 60; 70 80 90]: sum all = 450, divide by 9 = **50**. Show every multiply-and-add step.

**2. "Which filter for salt & pepper noise?"** **Median filter.** Salt & pepper creates isolated extreme-value pixels. Median sorts the neighborhood and picks the middle value, so the outlier pixels are ignored entirely. Mean/Gaussian filters would average the outlier in, spreading the noise.

### 3. "Compare mean vs Gaussian filter"

- Mean: all weights equal ( $1/N$ ). Treats every neighbor the same. Blurs edges heavily.
- Gaussian: center-weighted (weights fall off with distance). Better edge preservation because distant pixels contribute less. Both reduce Gaussian noise; Gaussian filter is generally preferred.

**4. "Why does median preserve edges?"** It is non-linear -- it selects an existing pixel value rather than averaging. At an edge, the majority of pixels in the window are on one side, so the median picks a value from that side. The edge stays sharp. Linear filters blur by averaging across the edge boundary.

## Related Topics

- Lecture 3: Image Filtering
- Project 2: Image Filtering
- Edge Detection (uses filtering)