Image Processing

Unit 2: Image Enhancement and Filtering in the Spatial Domain

Kiran Bagale

November, 2025

Unit 2 Overview

1. Introduction to Image Enhancement
2. Basic Gray Level Transformations
3. Histogram Processing
4. Spatial Filtering
5. Image Magnification

What is Image Enhancement?

- **Definition:** Process of improving the visual appearance of an image or converting it to a form better suited for analysis
- **Goal:** Improve interpretability or perception of information in images
- **Applications:**
  - Medical imaging
  - Satellite image processing
  - Photography and artistic effects
  - Security and surveillance
- **Note:** Enhancement is subjective and application-dependent

Spatial Domain vs Frequency Domain

Spatial Domain

- Direct manipulation of pixels
- Operations on image plane
- Form: $g(x, y) = T[f(x, y)]$
- Where:
  - $f(x, y)$ = input image
  - $g(x, y)$ = output image
  - T = transformation

This unit focuses on Spatial Domain techniques

Frequency Domain

- Fourier transform based
- Operations on transform coefficients
- More computational overhead
- Better for certain applications

Point Operations

- **Definition:** Transformation applied independently to each pixel
- General form: $s = T(r)$
  - r = input intensity level
  - s = output intensity level
  - T = transformation function
- **Characteristics:**
  - Output depends only on input pixel value
  - No neighborhood information used
  - Computationally efficient
  - Memory efficient
- **Types:** Contrast stretching, thresholding, negative, log transforms, power-law transforms

Contrast Stretching

Purpose:

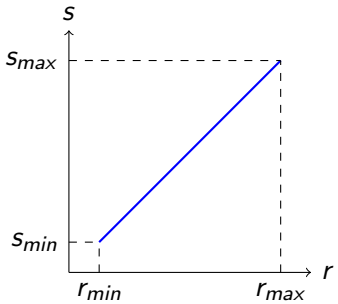
- Increase dynamic range of gray levels
- Improve contrast in low-contrast images

Linear Contrast Stretching:

$$s = \frac{s_{max} - s_{min}}{r_{max} - r_{min}}(r - r_{min}) + s_{min}$$

Parameters:

- (r_{min}, r_{max}) = input range
- (s_{min}, s_{max}) = output range



Effect:

- Expands input range to full output range
- Improves image contrast

Clipping and Thresholding

Clipping:

- Limits intensity values to specific range

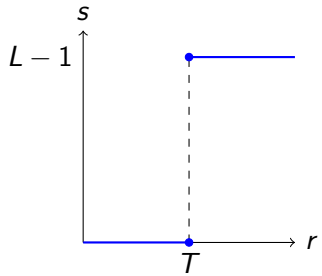
$$s = \begin{cases} a & \text{if } r < a \\ r & \text{if } a \leq r \leq b \\ b & \text{if } r > b \end{cases}$$

Thresholding:

- Binary segmentation

$$s = \begin{cases} 0 & \text{if } r < T \\ L-1 & \text{if } r \geq T \end{cases}$$

Threshold Function



Applications:

- Object segmentation
- Binary image creation
- Feature extraction

Digital Negative

Transformation:

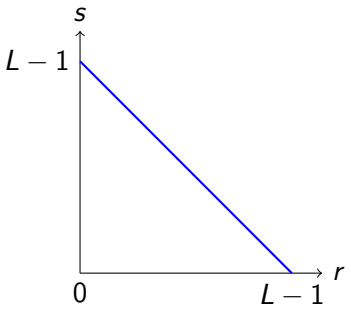
$$s = L - 1 - r$$

Where:

- L = number of gray levels (typically 256)
- r = input intensity
- s = output intensity

Effect:

- Dark areas become bright
- Bright areas become dark
- Reverses intensity scale



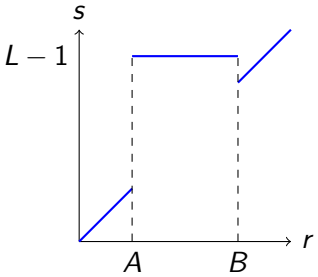
Applications:

- Medical imaging (enhancing white tissue in mammograms)
- Photography effects
- Better perception of dark regions

Intensity Level Slicing

Type 1: With Background

$$s = \begin{cases} L-1 & \text{if } A \leq r \leq B \\ r & \text{otherwise} \end{cases}$$



Type 2: Without Background

$$s = \begin{cases} L-1 & \text{if } A \leq r \leq B \\ 0 & \text{otherwise} \end{cases}$$

Applications:

- Highlighting specific intensity range
- Medical imaging (highlighting tumors, lesions)
- Satellite imagery (water bodies, vegetation)

Log Transformation

Formula:

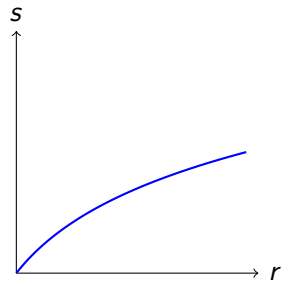
$$s = c \cdot \log(1 + r)$$

Where:

- c = scaling constant
- r = input intensity ≥ 0
- s = output intensity

Characteristics:

- Expands dark pixel values
- Compresses bright pixel values
- Maps narrow range to wider range



Log curve

Applications:

- Fourier spectrum visualization
- Dynamic range compression
- Enhancing details in dark regions

Power-Law (Gamma) Transformation

Formula:

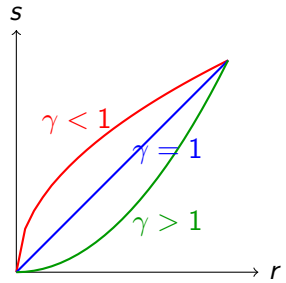
$$s = c \cdot r^{\gamma}$$

Where:

- c = scaling constant
- r = normalized input [0,1]
- γ = gamma parameter

Effect of γ:

- $\gamma < 1$: Brightens image (expands dark values)
- $\gamma = 1$: Identity (no change)
- $\gamma > 1$: Darkens image (compresses dark values)



Applications:

- Gamma correction for display devices
- CRT, LCD calibration
- Image enhancement based on perception

Bit Plane Slicing

- **Concept:** Decompose image into binary images based on bit positions
- 8-bit image has 8 bit planes (0-7)
- **Bit Plane k:** Extract k-th bit from each pixel

Properties:

- Higher-order bits (MSB): Contain most visual information
- Lower-order bits (LSB): Contain subtle details, appear random
- Bit plane 7 (MSB) most significant
- Bit plane 0 (LSB) least significant

Applications:

- Image compression
- Data hiding (steganography)
- Understanding image structure
- Analyzing contribution of each bit

Example:

- Pixel value: $182 = 10110110_2$
- Bit 7=1, Bit 6=0, Bit 5=1, ...

What is a Histogram?

- **Definition:** Graphical representation of pixel intensity distribution
- Shows frequency of each intensity level in image

Unnormalized Histogram:

$$h(r_k) = n_k$$

where n_k = number of pixels with intensity r_k

Normalized Histogram:

$$p(r_k) = \frac{n_k}{MN}$$

where:

- MN = total number of pixels in image
- $p(r_k)$ = probability of occurrence of intensity r_k
- $\sum_{k=0}^{L-1} p(r_k) = 1$

Histogram Characteristics

Dark Image:

- Histogram concentrated on left
- Low intensity values dominant

Bright Image:

- Histogram concentrated on right
- High intensity values dominant

Low Contrast:

- Narrow histogram
- Limited intensity range

High Contrast:

- Wide histogram
- Full intensity range utilized

Well-Balanced Image:

- Histogram spread across range
- Good distribution of intensities

Note: Different images can have same histogram (histogram is not unique)

Histogram Equalization - Concept

- **Goal:** Transform image to have uniform histogram
- Redistributes intensity values across entire range
- Enhances contrast automatically
- Based on cumulative distribution function (CDF)

Principle:

- Map input intensities to output using CDF
- Frequently occurring intensities spread out
- Infrequent intensities grouped together
- Results in approximately uniform output histogram

Advantages:

- Automatic process (no parameters needed)
- Effective for many images
- Simple to implement

Histogram Equalization - Mathematics

Transformation Function:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = (L-1)\sum_{j=0}^{k} \frac{n_j}{MN}$$

Where:

- s_k = output intensity level
- r_k = input intensity level
- L = number of gray levels (256 for 8-bit)
- $p_r(r_j)$ = normalized histogram (probability)
- MN = total pixels

Properties:

- $T(r)$ is monotonically increasing
- $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$
- Output histogram is approximately uniform

Histogram Equalization - Algorithm

1: Compute histogram $h(r_k)$ of input image
2: Compute normalized histogram: $p(r_k) = \frac{h(r_k)}{MN}$
3: Compute cumulative distribution function (CDF):

$$\text{CDF}(k) = \sum_{j=0}^{k} p(r_j)$$

4: Compute transformation: $s_k = \text{round}[(L-1) \times \text{CDF}(k)]$
5: Map each pixel: For each pixel with intensity r_k, replace with s_k

Example: For 8-bit image, $L = 256$, so multiply CDF by 255

Use of Histogram Statistics

Statistical Measures:

- **Mean (Average Intensity):**

$$m = \sum_{k=0}^{L-1} r_k \cdot p(r_k)$$

- **Variance (Contrast Measure):**

$$\sigma^2 = \sum_{k=0}^{L-1} (r_k - m)^2 \cdot p(r_k)$$

- **Standard Deviation:**

$$\sigma = \sqrt{\sigma^2}$$

Applications:

- Image quality assessment
- Adaptive thresholding
- Contrast evaluation

Basics of Spatial Filtering

- **Definition:** Process that operates on neighborhoods of pixels
- Uses a small matrix called *kernel*, *mask*, or *filter*
- Kernel moves across image (convolution/correlation)

General Form:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) \cdot f(x+s, y+t)$$

Where:

- $f(x,y)$ = input image
- $g(x,y)$ = output image
- $w(s,t)$ = filter kernel of size $(2a+1) \times (2b+1)$

Common kernel sizes: 3×3, 5×5, 7×7

Spatial Filtering Process

Convolution Steps:

1. Place kernel on image at position (x, y)
2. Multiply each kernel coefficient with corresponding pixel
3. Sum all products
4. Assign result to output pixel at (x, y)
5. Move kernel to next position
6. Repeat for all pixels

Boundary Handling:

- **Zero padding:** Assume zeros outside boundary
- **Replication:** Replicate edge pixels
- **Reflection:** Reflect image at boundaries
- **Wrap around:** Treat image as periodic

Linear vs Non-Linear Filters

Linear Filters:

- Output is weighted sum of inputs
- Satisfy superposition principle
- Examples:
  - Averaging filters
  - Gaussian filters
  - Laplacian filters
- Can be implemented using convolution

Non-Linear Filters:

- Output is non-linear function of inputs
- Don't satisfy superposition
- Examples:
  - Median filter
  - Max/Min filters
  - Bilateral filter
- Cannot use convolution

Superposition Principle:

$$T[a \cdot f_1 + b \cdot f_2] = a \cdot T[f_1] + b \cdot T[f_2]$$

Spatial Low-Pass Filters - Purpose

- **Purpose:** Smoothing, noise reduction, blurring
- Attenuate high-frequency components (edges, noise)
- Preserve low-frequency components (smooth regions)

Effects:

- Reduces noise and random variations
- Blurs sharp transitions
- Reduces fine details
- Creates smoother appearance

Applications:

- Noise reduction
- Pre-processing for edge detection
- Image blurring for artistic effects
- Reducing aliasing artifacts

Averaging Filter

Box/Mean Filter:
All coefficients equal:

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

Properties:

- Simple implementation
- Equal weight to all neighbors
- Effective noise reduction
- Causes blur

General Form:

$$g(x,y) = \frac{1}{mn}\sum_{s=-a}^{a}\sum_{t=-b}^{b} f(x+s, y+t)$$

For $m \times n$ kernel

Larger kernels:

- More smoothing
- More blur
- Higher computational cost

Trade-off:

- Noise reduction vs detail loss

Weighted Averaging Filter

Concept: Assign different weights to different positions
Gaussian Filter (most common):

$$w(s,t) = \frac{1}{2\pi\sigma^2} e^{-\frac{s^2+t^2}{2\sigma^2}}$$

Example 3×3 Gaussian-like kernel:

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

Advantages over simple averaging:

- More weight to center pixel
- Less weight to distant pixels
- Better edge preservation
- More natural blur
- Less block artifacts

Median Filter

Algorithm:

1. Place kernel on pixel
2. Sort all pixel values in neighborhood
3. Select middle value (median)
4. Assign to output

Properties:

- Non-linear filter
- Excellent for salt-and-pepper noise
- Preserves edges better than averaging
- Removes impulse noise

Example:

3×3 Neighborhood:

10	15	20
12	255	18
14	16	22

Sorted: 10, 12, 14, 15, 16, 18, 20, 22, 255

Median = 16

(Removes noise value 255)

Use case: Best for removing outliers while preserving edges

Maximum and Minimum Filters

Maximum Filter:

$$g(x,y) = \max_{(s,t)\in S_{xy}} f(s,t)$$

Effects:

- Enlarges bright regions
- Shrinks dark regions
- Removes pepper noise (dark spots)
- Erosion in morphology

Minimum Filter:

$$g(x,y) = \min_{(s,t)\in S_{xy}} f(s,t)$$

Effects:

- Enlarges dark regions
- Shrinks bright regions
- Removes salt noise (bright spots)
- Dilation in morphology

Applications:

- Noise removal (salt or pepper noise)
- Morphological operations
- Feature enhancement

High-Pass Sharpening Filters - Purpose

- **Purpose:** Enhance edges and fine details
- Emphasize high-frequency components
- Attenuate low-frequency components

Effects:

- Sharpens edges
- Enhances details
- Increases contrast at boundaries
- Makes transitions more pronounced

Basic Principle:

$$\text{High-pass} = \text{Original} - \text{Low-pass}$$

Applications:

- Edge detection
- Detail enhancement
- Medical imaging
- Quality inspection

High Boost Filter

Concept: Combine original image with sharpened version
Formula: $g(x, y) = A \cdot f(x, y) - f_{lowpass}(x, y)$
Or equivalently: $g(x, y) = (A - 1) \cdot f(x, y) + f_{highpass}(x, y)$
Where:

- $A \geq 1$ is amplification factor
- $A = 1$: Standard high-pass filter
- $A > 1$: High boost filter

Effect:

- Preserves overall appearance
- Enhances edges and details
- Controlled sharpening
- Less noise amplification than pure high-pass

High Frequency Emphasis Filter

Concept: Emphasize high frequencies while retaining low frequencies
Transfer Function (Frequency Domain): $H(u, v) = a + b \cdot H_{HP}(u, v)$
Where:

- H_{HP} = high-pass filter
- a = offset (typically 0.5 to 2)
- b = multiplier for high frequencies

Spatial Domain Equivalent: $g(x, y) = a \cdot f(x, y) + b \cdot f_{sharpened}(x, y)$
Advantages:

- Better control over enhancement
- Reduces over-sharpening
- Maintains image brightness
- Suitable for subtle detail enhancement

Gradient-Based Filters - Introduction

Gradient: Vector pointing in direction of maximum rate of change

Mathematical Definition: $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$

Gradient Magnitude: $|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$

Approximation: $|\nabla f| \approx |G_x| + |G_y|$

Where G_x and G_y are partial derivatives in x and y directions

Use: Edge detection and enhancement

Roberts Cross Gradient Operators

Kernels:

$$G_x = \begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix}$$

Properties:

- 2×2 kernels
- Simplest gradient operators
- Diagonal difference
- Fast computation

Gradient Magnitude: $|\nabla f| = \sqrt{G_x^2 + G_y^2}$

Or approximation: $|\nabla f| \approx |G_x| + |G_y|$

Characteristics:

- Very sensitive to noise
- Produces thin edges
- Limited use in modern applications
- Historical significance

Prewitt Filters

Kernels:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}$$

Applications:

- Edge detection
- Feature extraction
- Image segmentation

Properties:

- 3×3 kernels
- Simple to implement
- Averages along edge direction
- Better noise suppression than Roberts

Edge Direction: $\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$

Sobel Filters

Kernels:

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Properties:

- 3×3 kernels
- Weighted averaging
- More weight to center pixels
- Best among simple operators

Advantages over Prewitt:

- Better noise suppression
- Sharper edges
- More accurate gradient estimation
- Most widely used

Note: Sobel is the most popular first-order derivative operator

Comparison: Roberts, Prewitt, Sobel

Property	Roberts	Prewitt	Sobel
Kernel Size	2×2	3×3	3×3
Noise Sensitivity	High	Medium	Low
Edge Accuracy	Low	Medium	High
Computation	Fastest	Medium	Medium
Smoothing Effect	None	Moderate	Better
Edge Thickness	Thin	Medium	Medium
Common Usage	Rare	Moderate	Very Common

Recommendation:

- Use **Sobel** for most applications
- Use **Prewitt** if computational simplicity needed
- Avoid **Roberts** for noisy images

Second Derivative Filters - Introduction

First Derivative:

- Produces thick edges
- Responds to ramps
- Gradient-based

Second Derivative:

- Produces thin edges (zero-crossings)
- Stronger response to fine details
- More sensitive to noise

Properties of Second Derivative:

1. Zero in constant intensity regions
2. Non-zero at onset of intensity step or ramp
3. Zero along ramps of constant slope

Laplacian: Most common second derivative operator

Laplacian Filter - Theory

Mathematical Definition: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Discrete Approximation:

$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$

Properties:

- Isotropic (rotation invariant)
- Linear operator
- Scalar (no direction)
- Sensitive to noise
- Produces double edges

Laplacian Kernels

Basic Laplacian (4-neighbors):

$$\nabla^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Extended Laplacian (8-neighbors):

$$\nabla^2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Negative Laplacian: $-\nabla^2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

With Diagonal Terms:

$$-\nabla^2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Note: All kernels sum to zero (high-pass characteristic)

Laplacian Image Enhancement

Enhancement Formula: $g(x, y) = f(x, y) + c \cdot \nabla^2 f(x, y)$
Where:

- $c = -1$ if Laplacian kernel has negative center
- $c = +1$ if Laplacian kernel has positive center

Combined Kernel (for $c = -1$): $w = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$

Effect:

- Sharpens edges
- Enhances fine details
- Amplifies noise
- Single-pass operation

Laplacian of Gaussian (LoG)

Problem: Laplacian is very sensitive to noise
Solution: Smooth image first with Gaussian, then apply Laplacian
Formula: $\text{LoG} = \nabla^2[G(x,y) * f(x,y)] = [\nabla^2 G(x,y)] * f(x,y)$

LoG Function: $\nabla^2 G(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$

Characteristics:

- Also called "Mexican Hat" filter
- Better noise handling
- Wider kernel required
- Used in blob detection

Image Magnification - Introduction

Problem: How to increase image size (spatial resolution)?

Challenges:

- New pixel positions between existing pixels
- Need to estimate missing pixel values
- Balance between quality and computation

Common Methods:

- Replication (Nearest Neighbor)
- Bilinear Interpolation
- Bicubic Interpolation

Applications:

- Image zooming
- Display resolution matching
- Medical imaging
- Satellite imagery

Magnification by Replication

Method: Replicate pixel values (Nearest Neighbor Interpolation)
Algorithm:

1. For each position (x', y') in output image
2. Find nearest pixel in input image
3. Copy that pixel value to output

Formula: $g(x', y') = f\left(\text{round}\left(\frac{x'}{s_x}\right), \text{round}\left(\frac{y'}{s_y}\right)\right)$
Where s_x, s_y are scaling factors
Example: $2\times$ magnification

- Each pixel becomes 2×2 block
- Original: $100 \times 100 \rightarrow$ Output: 200×200

Replication - Properties

Advantages:

- Simplest method
- Fastest computation
- Easy to implement
- No arithmetic operations
- Preserves original pixel values

Disadvantages:

- Blocky appearance
- Jagged edges
- Poor visual quality
- Visible artifacts
- Not smooth

Visual Effect:

- Creates visible "stairs" on edges
- Square blocks visible
- Unnatural appearance

When to Use:

- Real-time applications
- Limited computational resources
- Binary images
- Quick preview

Better Alternatives:

- Bilinear interpolation
- Bicubic interpolation

Replication Example

Original 2×2 Image:

100	150
200	250

After $2\times$ Magnification by Replication:

100	100	150	150
100	100	150	150
200	200	250	250
200	200	250	250

Observation: Each pixel is replicated into a 2×2 block

Summary of Spatial Filtering

Filter Type	Purpose	Effect
Averaging	Smoothing	Blur, noise reduction
Median	Noise removal	Preserve edges
Max/Min	Morphological	Enlarge bright/dark
Sobel	Edge detection	Find edges
Laplacian	Sharpening	Enhance details
High Boost	Enhancement	Controlled sharpening

Key Principles:

- Low-pass filters smooth (blur) images
- High-pass filters sharpen images
- Linear filters use convolution
- Non-linear filters preserve edges better
- Trade-off: smoothing vs detail preservation

Comparison: Point vs Spatial Operations

Aspect	**Point Operations**	**Spatial Operations**
Input	Single pixel	Neighborhood of pixels
Complexity	Low	Higher
Speed	Fast	Slower
Examples	Negative, log, gamma	Smoothing, sharpening
Edge Effect	No	Yes (boundary handling)
Applications	Contrast, brightness	Noise, blur, edges

Key Formulas - Quick Reference

Point Operations:

- Negative: $s = L - 1 - r$
- Log: $s = c \log(1 + r)$
- Power-law: $s = cr^{\gamma}$

Histogram:

- Equalization: $s_k = (L - 1) \sum_{j=0}^{k} p_r(r_j)$

Spatial Filtering:

- Convolution: $g(x, y) = \sum_s \sum_t w(s, t) f(x + s, y + t)$
- Gradient: $|\nabla f| = \sqrt{G_x^2 + G_y^2}$
- Laplacian: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Applications Summary

- **Medical Imaging:**
  - Contrast enhancement (gamma correction)
  - Noise reduction (median filter)
  - Edge detection (Sobel, Laplacian)
- **Satellite Imagery:**
  - Histogram equalization
  - High-pass filtering for details
  - Intensity slicing for classification
- **Photography:**
  - Sharpening (high boost filter)
  - Artistic blur (Gaussian filter)
  - Dynamic range (log transform)
- **Industrial Inspection:**
  - Edge detection (gradient operators)
  - Defect highlighting (thresholding)
  - Feature extraction (bit-plane slicing)

Important Considerations

Filter Selection:

- Consider image characteristics (noise level, contrast)
- Application requirements (speed vs quality)
- Computational resources available

Parameter Tuning:

- Kernel size: larger = more smoothing/blur
- Gamma value: < 1 brightens, > 1 darkens
- Threshold selection: depends on histogram

Common Pitfalls:

- Over-sharpening causes noise amplification
- Over-smoothing loses important details
- Improper boundary handling creates artifacts
- Sequential operations may compound errors

Practice Problems

Problem 1: Apply histogram equalization to a 4×4 image with 8 gray levels
Problem 2: Design a 3×3 high-pass filter kernel
Problem 3: Compare Sobel and Prewitt operators on a noisy image
Problem 4: Perform $3\times$ magnification by replication on a 2×2 image
Problem 5: Apply median filter to remove salt-and-pepper noise
Problem 6: Compute Laplacian for edge enhancement
Problem 7: Design a transformation function for contrast stretching from range [50, 200] to [0, 255]

Further Reading

Recommended Topics:

- Adaptive histogram equalization (CLAHE)
- Bilateral filtering
- Anisotropic diffusion
- Canny edge detector
- Unsharp masking
- Morphological operations

Advanced Techniques:

- Multi-scale filtering
- Wavelet-based enhancement
- Machine learning for enhancement
- HDR imaging

Conclusion

Key Takeaways:

1. Point operations modify individual pixels independently
2. Spatial operations use neighborhood information
3. Histogram equalization automatically enhances contrast
4. Low-pass filters smooth, high-pass filters sharpen
5. Choice of filter depends on application and image characteristics
6. Gradient operators detect edges and boundaries
7. Second derivatives enhance fine details but amplify noise

Thank You!

Questions?