Mask Processing Techniques

- Image enhancement using filters.
- Filters are used to modify pixel values based on their neighboring pixel values.
- **Types of Filters Discussed:**
    - **Linear Smoothing Filter:**
        - Blurs an image.
        - Reduces noise.
    - **Median Filter (Nonlinear):**
        - Replaces each pixel value with the median value of its neighboring pixels.
        - Effective for removing certain types of noise while preserving edges.
    - **Sharpening Filter:**
        - Enhances edges and fine details in an image.

Neighborhood Operations

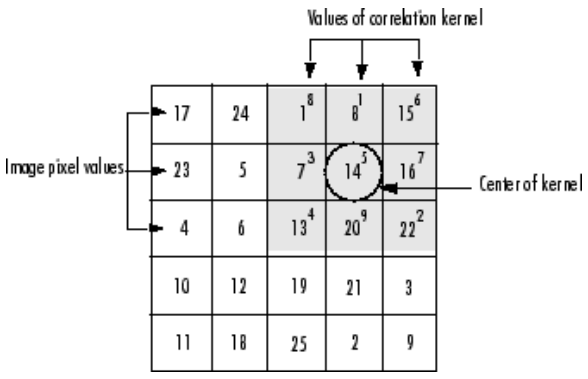
- **Core Idea:** Pixel values are changed based on the gray levels of the pixels around them (neighbors).
- **Mask Operations:** Also called template, window, or filter operations.

Mask Processing Details

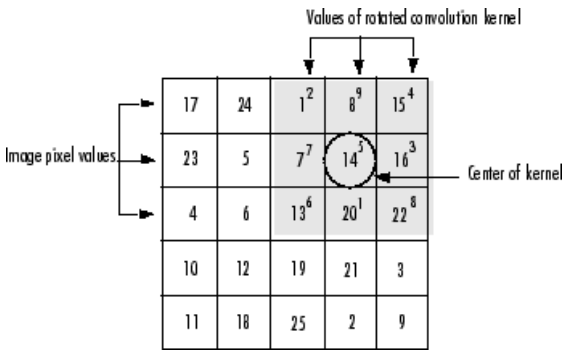
- **Terminology:**
    - A sub-image (called a filter, mask, kernel, template, or window) is used for processing
    - The values within the mask are called coefficients.
- **Boundary Issues:**
    - When processing pixels near the edges of an image, special handling is needed because some neighboring pixels might be outside the image boundary.
- **Mask Math:**
    - The new pixel value is calculated as a weighted sum of the neighboring pixel values and the filter coefficients.

Convolution and Correlation

- **Correlation:** The output pixel value is calculated as a weighted sum of neighboring pixel values.



- **Convolution:** A convolution kernel is a correlation kernel rotated by 180 degrees.



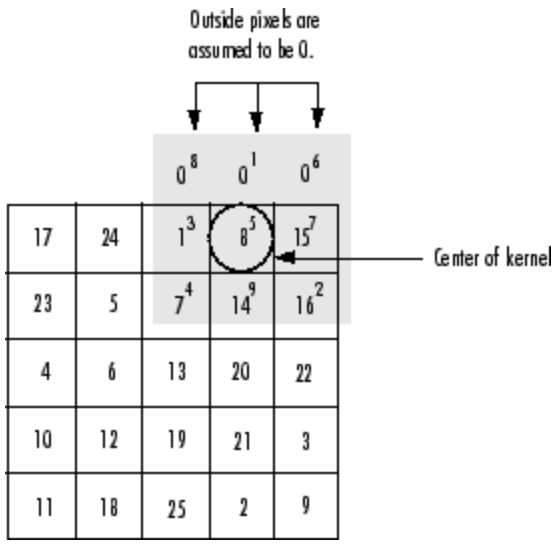
Spatial Filtering

- **Definition:** Filtering an image using spatial masks.
- **Types of Filters:**
    - **Linear Filters:**
        - **Lowpass Filters:** Attenuate (reduce) high-frequency components (like edges). Result: Blurring effect.
        - **Highpass Filters:** Attenuate low-frequency components. Result: Sharpening effect.
        - **Bandpass Filters:** Attenuate a specific range of frequencies. Used in image restoration.

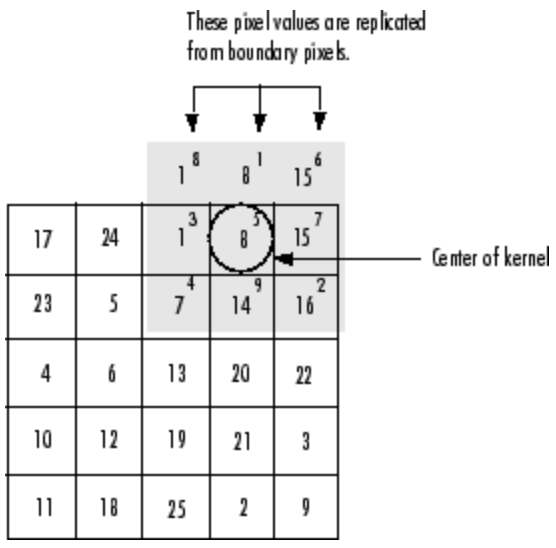
- **Nonlinear Filters:** Operate on neighborhoods based on relationships between pixel values, not on weighted sums.
  - **Median Filter:** A nonlinear filter for noise reduction.
  - **Max Filter:** Finds the brightest pixels in a neighborhood.
  - **Min Filter:** Finds the darkest pixels in a neighborhood.

Linear Filtering Details

- **Example Operations:** Blurring, de-noising, edge detection.
- **Boundary Handling:** When the Values of the Kernel Fall Outside the Image
- **Methods:**
  - **Zero Padding:** Outside pixels are assumed to be zeros.

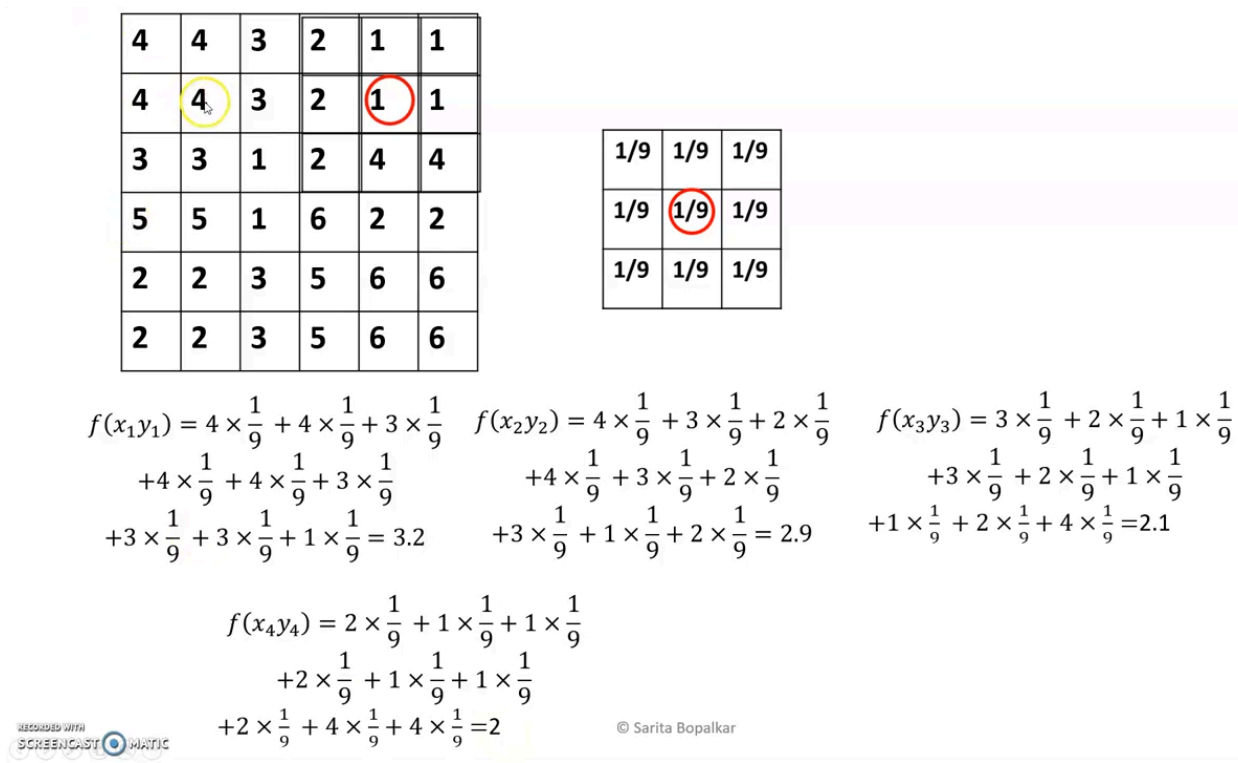


  - **Replicated Boundary Pixels:** Replicate by repeating the boundary pixels.



Lowpass (Averaging) Filters

- Reduce high-frequency components (like edges). Blur images.
- Implementation:
  - Typically uses positive coefficients.
  - A simple 3x3 filter averages the pixel values in the neighborhood.
  - The constant multiplier in front of each mask is equal to the sum of the values of its coefficients.
- **Example**: Image Low Pass Filtering in Spatial Domain



- **Lowpass Filter Applications**
    - **Noise Reduction:** Reduces sharp transitions in gray levels caused by random noise.
    - **Edge Blurring:** Blurs edges.
    - **Detail Reduction:** Reduces small, irrelevant details.
    - **Gap Bridging:** Can fill small gaps in lines or curves.
- **Averaging Masks Types:**
    - **Box Filter:** Uses equal coefficients for all pixels in the neighborhood.
    - **Weighted Average Filter:** Coefficients are weighted based on distance (e.g., center pixel has a higher weight).
    - A general formula for weighted averaging is provided.

Highpass Filters

- Reduce low-frequency components.
- Makes edges and fine details more prominent.
- Used to detect edges in images.

Nonlinear Filters

- **Operation:** Based on pixel values in the neighborhood, not on coefficients.
- **Order-Statistics Filters:** A type of nonlinear filter.

Order-Statistics Filters

- **Types:**
    - **Median Filter:** Reduces noise by replacing a pixel's value with the median value of its neighbors.
    - **Max Filter:** Finds the brightest pixel in a neighborhood.
    - **Min Filter:** Finds the dimmest pixel in a neighborhood.
    - **Midpoint Filter:** Computes the midpoint between the maximum and minimum values in the area encompassed by the filter.
    - **Alpha-trimmed mean filter:**

Median Filter

- Replaces a pixel's value with the median of the gray levels in its neighborhood.
- Excellent at removing "salt-and-pepper" noise (random black and white pixels). This type of noise often creates extreme values that stand out from their neighbors. Because the median filter relies on the middle value rather than the average, it's less affected by these outliers.
- **How it Works**
    a. **Define a Window:** A small grid that slides over the image. A common size is 3x3.
    b. **Select a Pixel:** Consider a pixel in the image you want to filter.
    c. **Gather Neighbors:** Place the center of the window over that pixel. Collect the pixel values within the window.
    d. **Sort the Values:** Sort the pixel values in ascending (or descending) order.
    e. **Find the Median:** The middle value. If there's an even number of values, it's the average of the two middle values.
    f. **Replace:** Replace the original pixel's value with the calculated median value.
    g. **Repeat:** Move the window to the next pixel and repeat steps 3-6 until you've processed all pixels in the image.
- **Example:** Let's say we have a small 3x3 section of an image:

    10 15 12
    50 **20** 13
    11 14 60

    We'll apply a 3x3 median filter to the center pixel (20).

    1. **Window:** 3x3 (as shown above)
    2. **Center Pixel:** 20
    3. **Neighbors:** 10, 15, 12, 50, 20, 13, 11, 14, 60
    4. **Sort:** 10, 11, 12, 13, **14**, 15, 20, 50, 60
    5. **Median:** 14 (the middle value)
    6. **Replace:** The center pixel's new value becomes 14.

Smoothing with Median Filters

- **Comparison:**
    - Averaging (lowpass) filters blur edges.
    - Median filters are better at preserving edge sharpness while reducing noise, especially impulse noise (salt-and-pepper noise).

Max and Min Filters

- **Max Filter:**
    - Replaces the center pixel of a window with the *maximum* pixel value within that window.
    - The max filter is effective at removing pepper noise (black pixels). If a black pixel is in the window, the max filter will likely replace it with a neighboring pixel's value, as the neighbors will have higher values.
- **Min Filter:**
    - Replaces the center pixel of a window with the *minimum* pixel value within that window.
    - The min filter is effective at removing salt noise (white pixels). If a white pixel is in the window, the min filter will likely replace it with a neighboring pixel's value, as the neighbors will have lower values.

Midpoint Filter

- Computes the midpoint between the maximum and minimum values in the filter's area.
- Effective for random noise like Gaussian or uniform noise.

Alpha-Trimmed Mean Filter

- Deletes the lowest and highest gray-level values in the filter's area.
- Averages the remaining pixels.
- Can act as an arithmetic mean filter or a median filter, depending on the parameters.
- Useful for mixed noise types.

Noise

- Real-world sensors are affected by noise (thermal, electrical, etc.).

Gaussian Noise

- Statistical noise that has a probability density function equal to the normal distribution (bell-shaped curve).
- In an image, Gaussian noise appears as random variations in brightness or color.
- It can make the image look grainy or fuzzy. The noise values are generally distributed around the true pixel values.
- It often arises from electronic noise in sensors or during transmission.

Salt and Pepper Noise

- Type of noise that introduces sparse white and black pixels into an image.
- The image will contain sudden, sharp disturbances.
- It will have black pixels in bright regions and white pixels in dark regions.
- The noisy pixels take either the maximum or minimum possible values.
- For an 8-bit image, the salt-and-pepper noise pixels will typically be 0 (black) or 255 (white).
- It can be caused by errors in data transmission, malfunctioning camera sensors, or timing errors during the digitization process.

Mean Filters

- Simple methods to reduce noise in the spatial domain.
- Types:
    - **Arithmetic Mean Filter:** Replaces each pixel's value with the *average* value of its neighbors. {mean=sum/count}
    - **Geometric Mean Filter:** Instead of averaging pixel values, it multiplies them and then takes the nth root, where n is the number of pixels in the filter window.
    - **Harmonic Mean Filter:** Divide the num of pixels in the window by the sum of the reciprocals (1/value) of all pixels in the window.
    - **Contraharmonic Mean Filter:**

Arithmetic Mean Filter

- Computes the average value of corrupted pixels in a defined area.
- Replaces each pixel's value with the *average* (mean) value of its neighboring pixels.
- Reduces noise by blurring.

Geometric Mean Filter

- **What it does:** The geometric mean filter is a non-linear filter used for noise reduction. It's particularly effective at dealing with multiplicative noise (where noise is multiplied with the image signal), unlike the arithmetic mean filter, which is better suited for additive noise. Instead of averaging pixel values, it multiplies them and then takes the nth root, where n is the number of pixels in the filter window.

- **How it works:**

    1. **Window:** Define a window (kernel), like 3x3.
    2. **Center Pixel:** Select the pixel to filter.
    3. **Gather Neighbors:** Place the window over the pixel and collect the pixel values within.
    4. **Calculate the Geometric Mean:**
        - Multiply all the pixel values within the window.
        - Take the nth root of the product, where n is the number of pixels in the window. (For a 3x3 window, you'd take the 9th root).
    5. **Replace:** Set the center pixel's new value to the calculated geometric mean.
    6. **Repeat:** Move the window and repeat.

- **Example:**

    Image Section:

    10 15 12
    50 20 13
    11 14 60

- **Steps:**
    - 3x3 Geometric Mean Filter on center pixel (20):
    - Values: 10, 15, 12, 50, 20, 13, 11, 14, 60
    - Product: 10 * 15 * 12 * 50 * 20 * 13 * 11 * 14 * 60 = 215,764,800,000
    - 9th Root: (215,764,800,000)^(1/9) ≈ 18.16
    - New center pixel value: 18 (rounded)

- **Result:**

    10 15 12
    50 18 13
    11 14 60

Key Points

- The geometric mean filter tends to preserve edges and fine details better than the arithmetic mean filter.
- It's effective at reducing multiplicative noise.

Harmonic Mean Filter

Simple Explanation

Imagine you have a group of friends who are running a race. You want to find their "average" speed. If they run at normal speeds, you could just take the regular average (arithmetic mean).

But what if one friend runs *really* slowly? That slow speed will drastically pull down the regular average.

The harmonic mean is a different kind of "average" that's useful when dealing with rates or values where outliers (especially low values) have a big impact. In our race example, the harmonic mean would give a better idea of the "typical" speed if there's a very slow runner.

In Images

In images, pixels are like the runners, and their brightness values are like their speeds. Sometimes, you get "noisy" pixels that are much darker than their neighbors (like that very slow runner). These are often "salt" pixels in salt-and-pepper noise.

The harmonic mean filter is good at dealing with these dark, noisy pixels. It tends to bring their values closer to their neighbors.

How it Works (Step-by-Step)

1. **Pick a Pixel:** Choose a pixel in the image.
2. **Look Around:** Create a small "window" around that pixel (usually 3x3, meaning you look at the pixel and its 8 neighbors).
3. **Flip the Values:** Take "1 divided by" each pixel value in the window.
4. **Add Them Up:** Add all those flipped values together.
5. **Divide:** Divide the *number* of pixels in the window by the sum you just calculated.
6. **Replace:** Change the original pixel's value to the result.

Example Exercise

Let's say we have a small 3x3 image section with some salt-and-pepper noise. We'll apply a 3x3 harmonic mean filter to the center pixel (20).

10 15 255
50 **20** 13
11 14 60

Steps:

1. **Window:** The 3x3 section shown above.
2. **Values:** 10, 15, 255, 50, 20, 13, 11, 14, 60
3. **Flip (Reciprocals):**
   - 1/10 = 0.1
   - 1/15 = 0.0667
   - 1/255 = 0.0039
   - 1/50 = 0.02
   - 1/20 = 0.05
   - 1/13 = 0.0769
   - 1/11 = 0.0909
   - 1/14 = 0.0714
   - 1/60 = 0.0167
4. **Add:** 0.1 + 0.0667 + 0.0039 + 0.02 + 0.05 + 0.0769 + 0.0909 + 0.0714 + 0.0167 = 0.4965
5. **Divide:** Number of pixels in the window = 9 [9 / 0.4965 = 18.13]
6. **Replace:** The new center pixel value is approximately 18.

Result: The filtered image section would be:

10 15 255
50 **18** 13
11 14 60

- "20" → "18". The harmonic mean helped to reduce the influence of the "255" (a salt noise pixel) in the calculation.

Contraharmonic Mean Filter

The Contraharmonic Mean Filter is a more advanced filter used to remove salt-and-pepper noise from images. Unlike the simple average, it uses a power value (Q) to give more weight to either bright or dark pixels in the neighborhood.

Why Q Matters:

- If Q is positive, the filter is good at removing "pepper" noise (dark pixels).
- If Q is negative, the filter is good at removing "salt" noise (bright pixels).

How it Works (Step-by-Step)

1. **Pick a Pixel:** Choose a pixel in the image.
2. **Look Around:** Create a small "window" around that pixel (usually 3x3).
3. **Choose a Q:** Decide on the value for Q (e.g., 1, -1, 2, -2). This choice will determine if you're targeting salt or pepper noise.
4. **Calculate Two Sums:**
   - **Sum 1:** Raise each pixel value in the window to the power of (Q + 1) and then add all those values together.
   - **Sum 2:** Raise each pixel value in the window to the power of Q and then add all those values together.
5. **Divide:** Divide Sum 1 by Sum 2.
6. **Replace:** Change the original pixel's value to the result.

Example Exercise

Let's use a 3x3 image section and apply the Contraharmonic Mean Filter with Q = 1 (to remove pepper noise). We'll apply the filter to the center pixel (20).

Image Section:

```
10  15  0
50  20  13
11  14  60
```

Steps:

1. **Window:** The 3x3 section shown above.
2. **Q:** 1
3. **Sum 1 (values raised to Q + 1 = 2):**
    - 10^2 + 15^2 + 0^2 + 50^2 + 20^2 + 13^2 + 11^2 + 14^2 + 60^2 = 100 + 225 + 0 + 2500 + 400 + 169 + 121 + 196 + 3600 = 7311
4. **Sum 2 (values raised to Q = 1):**
    - 10 + 15 + 0 + 50 + 20 + 13 + 11 + 14 + 60 = 193
5. **Divide:** 7311 / 193 = 37.88
6. **Replace:** The new center pixel value is approximately 38.

Result: The filtered image section would be:

```
10  15  0
50  38  13
11  14  60
```

In this example, the Contraharmonic Mean Filter with Q=1 has adjusted the center pixel value. It's important to remember that the choice of Q significantly impacts the filter's output.