**Outline**

- What is smoothing
- Linear filters:
  - Mean filter;
  - Triangle filter;
  - Gaussian filter
- Non-linear filters:
  - Median filter
  - Kuwahara filter

---

**Smoothing**

- Objectives:
  - Reduce noise in the image;
  - Prepare images for further processing

---

**Mean Filter**

- Also called uniform filter or box filter
- Use the mean of its neighboring pixels’ intensities
- Equaling to convoluting the image with a kernel filled with same value

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

**Mean Filter Example**

1 4 3 8 0 ... 
0 8 1 7 4 ... [1 1 1] - 3 5 4 ... 
2 3 5 6 2 ... [1 1 1] = - 4 6 5 ... 
1 9 7 8 5 ... [1 1 1] - ... 
[... ...]

- \((1 + 4 + 3 + 0 + 8 + 1 + 2 + 3 + 5) / 9 = 3\)
- \((4 + 3 + 8 + 8 + 1 + 7 + 3 + 5 + 6) / 9 = 5\)
- \((2 + 8 + 0 + 1 + 7 + 0 + 5 + 6 + 2) / 9 = 4\)
- \((0 + 8 + 1 + 2 + 3 + 5 + 1 + 9 + 7) / 9 = 4\)
- \((8 + 1 + 7 + 3 + 5 + 6 + 9 + 7 + 8) / 9 = 6\)
- \((1 + 7 + 0 + 5 + 6 + 2 + 7 + 8 + 5) / 9 = 5\)

**Properties of Mean Filter**

- Both separable and incremental

$$S_{mean} = \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$S_{mean} - S_{mean}^5 = \begin{bmatrix} -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 & 1 \end{bmatrix}$$
Naïve vs. Efficient Implementations

• for ( int w = 0 ; w < width ; w++ )
  • for ( int p = w ; p < width-w ; p++ ) {
    • int sum = 0;
    • for ( int v = -w ; v <= w ; v++ )
      • for ( int u = -w ; u <= w ; u++ )
        • sum += F[q+v][p+u];
    • G[q][p] = sum / ((2*w+1) * (2*w+1));
  }
• for ( int q = 0 ; q < height ; q++ )
  • for ( int p = w+1 ; p < width-w ; p++ )
    • sum += F[q][p+w] – F[q][p-w-1],
  • T[q][p] = sum / (2*w+1);
• for ( int p = w ; p < width-w ; p++ )
  • for ( int q = w+1 ; q < height-w ; q++ )
    • sum += T[q+w][p] – T[q-w-1][p],
  • G[q][p] = sum / (2*w+1);

Triangle Filter

• Use the weighted average of its neighboring pixels’ intensities
  • Center pixel has the highest weight
  • Boundary pixels have the lowest weight

Properties of Triangle Filter

• Separable but not incremental
  • Complexity is O(M×N×W)

Properties of Triangle Filter (Cont’d)

• Can be considered as the convolution of two identical mean filters
  • Complexity is reduced to O(M×N)

Gaussian Function

• The Gaussian kernel is a quantized 2D Gaussian function
  • The size of the kernel depends on parameter σ
  • When σ=1:
    • G(0) = 0.399
    • G(1) = 0.242
    • G(2) = 0.054
    • G(3) = 0.004

Gaussian Filter

• Use true values of 2D Gaussian function:
  • Involves floating point calculation
  • The kernel is separable but not incremental
• Use quantized values:
  • Involves integer calculation only
  • May not be separable
• Use approximate values:
  • Efficient but inaccurate
### Approximate Gaussian Filter

- Repetitive convolution using a mean filter:

\[
S_{\text{conv}} = S_{\text{mean}} \otimes S_{\text{mean}} \otimes S_{\text{mean}} = \frac{1}{729}
\]

### Median Filter

- Use the median of neighboring pixels’ intensities:
  - Sort nearby pixels based on their intensities;
  - Use the intensity of the pixel in the middle;
  - No roundoff error
- Complexity:
  - \(O(M \times N \times W^2 \times \log W)\)
  - Faster approach exist

### Kuwahara Filter

- One of the edge-preserving filters:
  - Smooth image without disturbing the sharpness and position of edges
- Basic idea:
  - Defines four square regions in the window
  - Calculate the mean and variance of each region
  - Use the mean of the region that has the smallest variance

### Result Comparison I

<table>
<thead>
<tr>
<th>Original</th>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
</table>

### Result Comparison II

<table>
<thead>
<tr>
<th>Original</th>
<th>Mean</th>
<th>Gaussian</th>
<th>Median</th>
</tr>
</thead>
</table>