Image Filtering

Outline
- The definition of filtering
  - Linear filters vs. non-linear filters
- Linear filter:
  - Kernel
  - Convolution
- Implementations:
  - for general filters
  - for incremental filters
  - for separable filters

Filtering
- Definition:
  - The process of replacing a pixel's intensity with a value based on some functions of its nearby pixels' intensities
- Classify by effect:
  - Smoothing filters
  - Sharpening filters
- Classify by function used:
  - Linear filters
  - Non-linear filters

Example of Filtering
- Characteristic:
  - Each output pixel uses a different set of neighbors
  - Most of the neighbors used for the current pixel are also used for the next pixel
  - For some filters, it is possible to derive a more efficient implementation

Linear vs. Non-linear
- Linear filters:
  - Calculate the linear combination of the intensities of neighboring pixels
  - Can be expressed as convolution with a kernel
  - Can be analyzed in frequency (Fourier) domain
- Non-linear filters:
  - Involve more complex calculation
  - Cannot be expressed as convolution
  - Cannot be analyzed in frequency domain

Linear Convolution
- For each pixel in digital image:
  - Kernel is aligned with the pixel
  - Multiplicative sum is calculated and normalized
  - The result is stored in the corresponding pixel of the result image
  - Process is repeated for all pixels
Kernel

- Kernel keeps the coefficients used in linear combination
- Width & height are odd so that the kernel has a center
- Can be considered as an image of (2W+1)×(2W+1)
- W determines how many nearby pixels are involved
- Normalized so that the sum of all entries equals to 1

\[
K = \sum_{u,v} k_{u,v}
\]

\[
\begin{bmatrix}
  k_{-2,-2} & k_{-2,-1} & k_{-2,0} & k_{-2,1} & k_{-2,2} \\
  k_{-1,-2} & k_{-1,-1} & k_{-1,0} & k_{-1,1} & k_{-1,2} \\
  k_{0,-2} & k_{0,-1} & k_{0,0} & k_{0,1} & k_{0,2} \\
  k_{1,-2} & k_{1,-1} & k_{1,0} & k_{1,1} & k_{1,2} \\
  k_{2,-2} & k_{2,-1} & k_{2,0} & k_{2,1} & k_{2,2}
\end{bmatrix}
\]

- Properties:
  - Commutative: a ⊗ b = b ⊗ a
  - Associative: a ⊗ (b ⊗ c) = (a ⊗ b) ⊗ c
  - Distributive: a ⊗ (b + c) = (a ⊗ b) + (a ⊗ c)

Convolution

- Properties:
  - Commutative: a ⊗ b = b ⊗ a
  - Associative: a ⊗ (b ⊗ c) = (a ⊗ b) ⊗ c
  - Distributive: a ⊗ (b + c) = (a ⊗ b) + (a ⊗ c)

Handling the Boundaries

- Different approaches can be used for handling the boundaries:
  - Ignore the boundary — the output image will contain a white/black border
  - Assume that pixels that are outside of the input image have a fixed intensity, i.e., zero
  - Assume that a pixel that is outside of the input image has the same intensity as its closest neighbor that is in the image
- F[-u,q] = F[0,q], F[width+u,q] = F[width-1,q];
- F[p,-v] = F[p,0], F[p,height+v] = F[p,height-1];

Implementation for General Filters

- Calculations for each pixel:
  - (2W+1)×(2W+1) multiplications
  - (2W+1)×(2W+1) additions
  - Overall complexity:
    - O(M×N×W²)

Separable Property

- Some kernels can be represented as the production of two vectors
- The 2D convolution can be decomposed into two 1D convolutions

Implementation for Separable Filters

- Calculations for each pixel:
  - 2×(2W+1) multiplications
  - 2×(2W+1) additions
  - Overall complexity:
    - O(M×N×W)
**Incremental Property**

For some kernels, the above subtraction produces a sparse matrix.

The intensity of the next pixel in the scanline can be efficiently calculated through updating the intensity of the current pixel.

\[
\begin{bmatrix}
0 & k_{x_1} & \ldots & k_{x_n} \\
\vdots & \ddots & \ddots & \vdots \\
0 & k_{x_n} & \ldots & 0
\end{bmatrix}
- \begin{bmatrix}
0 & k_{x_1} & \ldots & k_{x_n} \\
\vdots & \ddots & \ddots & \vdots \\
0 & k_{x_n} & \ldots & 0
\end{bmatrix}
\]

**Implementation for Incremental Filters**

- Calculations for each pixel:
  - \( L \) multiplications & additions
  - \( L \) is the number of non-zero entries in the matrix after subtraction
  - Overall complexity: \( O(M \times N \times L) \)

\[
\text{for ( int w = \text{width}; w < \text{height}; w++) }
\text{for ( int p = \text{width}; p < \text{width} + \text{w} ; p++ ) }
\]

\[
\text{G \[ p \] \[ q \] = \text{sum} / \left( (2 * \text{w} + 1) \times (2 * \text{w} + 1) \right);}
\]

**Example Filters I**

- Original
- Emboss
- Glass Edge
- Foil Engrave

**Example Filters II**

- Original
- Emboss
- Glass Edge
- Foil Engrave