CS 335
Graphics and Multimedia

Neighborhood Operations
Neighborhood Operators

groups of pixels in proximity influence the operator

- Convolution
  - components (kernel, target/destination images)
  - applications and computational issues

- Correlation
  - components (templates, target/destination images)
  - applications
Neighborhood Operations

- **Input**
  - input image
  - kernel

- **Output**
  - output image
Neighborhood Operations

Input image

Output image
Neighborhood Operations

Input image

Output image
Neighborhood Operations

Input image

Output image
Neighborhood Operations

- Kernel size can change
- Must handle “boundary” cases
- Avoid cascading computations

- 3x3
- 5x5
Convolution

\[ g(x, y) = h(-1, -1) \cdot f(x+1, y+1) + h(0, -1) \cdot f(x, y+1) + h(1, -1) \cdot f(x-1, y+1) + h(-1, 0) \cdot f(x+1, y) + h(0, 0) \cdot f(x, y) + h(1, 0) \cdot f(x-1, y) + h(-1, 1) \cdot f(x, y-1) + h(0, 1) \cdot f(x, y-1) + h(1, 1) \cdot f(x-1, y-1) \]
Convolution

- Convolution “flips” the kernel when it applies it.
- The flip has no effect if the kernel is symmetric.

\[
g(x, y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j, k) f(x - j, y - k)
\]

\[
g(x, y) = h * f(x, y)
\]
Convolution: Issues

- Boundary conditions
  - ignore border
  - copy input pixel on unprocessed border pixels
  - image/kernel truncation
  - reflection

- Destination pixel values
  - can be negative
  - range can scale beyond input domain
Correlation

- Similar to convolution: do not “flip” kernel
- Useful for measuring similarity between kernel (“template”) and image region of interest

\[
g(x, y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j, k)f(x + j, y + k)
\]

\[
g(x, y) = \sum_{k=-n_2}^{n_2} \sum_{j=-m_2}^{m_2} h(j, k)f(x - j, y - k)
\] (convolution)
Correlation for Matching: Limitations

- Scale dependence
- Pixel resolution limits
- Rotational dependence
- Dynamic range of template/image
Filtering

removes something from an image

- Filter names: sometimes related to what “passes through”:
  - high pass filter
  - low pass filter

- Mechanisms for filtering:
  - linear and non-linear neighborhood operations: spatial (e.g., pixel) domain
  - operations in the frequency domain: working with the Fourier transform
The Low-Pass Filter

- Allow low frequency to pass (to destination image)
- Remove high frequency
- Control frequency cutoff
- Low-pass filter causes blurring: high-frequency gives fine detail, or sharpness
The Low-Pass Filter

- Any convolution kernel with all positive values will act as a low-pass filter
- Normalized convolution kernels keep the range of the filter within the same interval as the domain
- Kernel with all values equal, normalized by the number of entries in the kernel, is called a mean filter

\[
\begin{pmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{pmatrix}
\]
Gaussian Low-Pass Filter

- Kernel contains all positive values
- Shape of kernel (kernel coefficients) is gaussian
- Symmetric

\[ h(x, y) = \exp \left[ \frac{- (x^2 + y^2)}{2\sigma^2} \right] \]

\[ h(x, y) = \frac{1}{mn} \]
Low-Pass Filters

- **Purpose:** remove noise
- **Cost:** loss of high-frequency resolution
- **Control:** filter more high frequency via kernel size
High-Pass Filters

- Kernel shape: positive near center pixel, negative on boundaries
- Symmetry: responds to high-frequency variation in all directions
- Coefficient sum of 0:
  - What’s the output for a uniformly colored area??
High-Pass Filtering

- **Goal:** enhance edges, or rapid transitions in the image intensities

- **Unsharp masking:** subtract a blurred image from itself (same as high-pass filtering)
High-Boost Filtering

- Add a high-pass filtered image to itself
- Boost parts of image where high-frequency transitions occur

When $k$ is 8, this is a high-pass filter.
When $k>8$, this filters and “boosts”

<table>
<thead>
<tr>
<th>-1</th>
<th>-1</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>k</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Edge Detection

- Edge → sharp changes in intensity
- Basic Detection Principle
  - FIND the gradient (difference)
    \[
    dx = f(x+1, y) - f(x-1, y) \\
    dy = f(x, y+1) - f(x, y-1)
    \]
  - Average over a small neighbor (fight noise)

<table>
<thead>
<tr>
<th>-1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

-1 -1 -1
0 0 0
1 1 1

Prewitt Kernel
Samples

- Input and Gradient Images

- After Thresholding (different thresholds)
Other Edge Detectors

- **Sobel**
  -1 0 1  
-2 0 2  
-1 0 1

- **Laplacian**
  -1 -2 -1  
  0 0 0  
  1 2 1

- **Canny**
Non-linear Filtering

- Rank Filtering
  - Median (not mean)
  - Min
  - Max
Take Home Exercise

- Ex 7, 9: 3, 6, 7, pp 187, Efford