Image Filtering
January 24, 2007

Reading Today’s Lecture


Reading for Next Time


• Posted on the reading web page
  • (not WebCT)

Some Questions from Last Lecture

What would be the result?
Which one is correct?

Let's verify this using matlab

Histogram Modification

- Scaling
- Equalization
- Normalization

Histogram Scaling

Histogram Equalization

Histogram Scaling (Contrast Stretching)

The pixels in the range \([a, b]\) are expanded to fill the range \([z_1, z_2]\)

\[
z' = \frac{z_k - z_1}{b - a} (z - a) + z_1 = \frac{z_k - z_1}{b - a} z + \frac{z_1b - z_k a}{b - a}.
\]
Histogram Equalization

The number of pixels at level $z_i$ in the old histogram

$$\sum_{k=1}^{k_1} p_i \leq q_i < \sum_{k=1}^{k_2} p_i$$

The pixels at levels $z_1, z_2, \ldots, z_{k_1}$ map to level $z_i$ in the new image.

Histogram Normalization

Linear Systems

Input $\delta(x, y)$ → Linear system → Output $g(x, y)$
A system whose response remains the same irrespective of the position of the input pulse is called a space invariant system.

For such a system the output $h(x,y)$ is the convolution of $f(x,y)$ with the impulse response $g(x,y)$

$$h(x,y) = f(x,y) * g(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x',y') g(x-x',y-y') \, dx' \, dy'.$$

The output image corresponding to $f_i$ is

$$h_i(x,y) = a_i \cdot h_1(x,y) + b_i \cdot h_2(x,y).$$

### Convolution example

Example of 3x3 convolution mask

$$h[i,j] = \sum_{k=1}^{m} \sum_{l=1}^{m} f[k,l] g[i-k,j-l].$$

$$h[i,j] = A P_i + B P_2 + C P_3 + D P_4 + E P_5 + F P_6 + G P_7 + H P_8 + I P_9.$$
Example of 3x3 convolution mask

In plain words

Convolution is essentially equivalent to computing a weighted sum of image pixels.

Convolution is a linear operation

Types of Image Noise

- Salt and Pepper Noise
  - random occurrences of black and white pixels

- Impulse noise
  - Random occurrences of white pixels only

- Gaussian noise
  - Variations of intensity that are drawn from a Gaussian or normal distribution

Mean Filter

- Arbitrary neighborhood

Mean Filter

- For a 3x3 neighborhood
3x3 Mean Filter

$\frac{1}{9} \times \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}$

3x3 Linear Smoothing Filter

In general, it is a good idea to have only a single peak in your smoothing filter:

$\begin{bmatrix}
\frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\
\frac{1}{8} & \frac{1}{16} & \frac{1}{8} \\
\frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\
\end{bmatrix}$

Median Filter

- Sort the pixels into ascending order by their gray level values
- Select the value of the middle pixel as the new value for pixel $[i, j]$

3x3 Median Filter

Matlab Demo

Gaussian Smoothing
The Gaussian Function

- Zero mean 1D Gaussian
  \[ g(x) = e^{-\frac{x^2}{2\sigma^2}} \]
- Zero mean 2D Gaussian for image processing applications
  \[ g[i, j] = e^{-\frac{i^2 + j^2}{2\sigma^2}} \]

Gaussian Properties

- Rotationally symmetric in 2D
- Has a single peak
- The width of the filter and the degree of smoothing are determined by sigma
- Large Gaussian filters can be implemented very efficiently using small Gaussian filters

Rotational Symmetry

- Original formula
  \[ g[i, j] = e^{-\frac{i^2 + j^2}{2\sigma^2}} \]
- Switch to polar coordinates
  \[ r^2 = i^2 + j^2 \]
- Result (does not depend on \( \theta \))
  \[ g(r, \theta) = e^{-\frac{r^2}{2\sigma^2}} \]

Gaussian Separability

The convolution of the input image \( f[i, j] \) with a vertical 1D Gaussian function

Cascading Gaussians

[Image of cascading Gaussians]
The convolution of a Gaussian with itself yields a scaled Gaussian with larger sigma

\[
g(x) * g(x) = \int_{-\infty}^{\infty} e^{-\frac{x^2}{2\sigma^2}} e^{-\frac{(x-\xi)^2}{2\sigma^2}} d\xi
\]

= \int_{-\infty}^{\infty} e^{-\frac{(x^2 + \xi^2)}{2\sigma^2}} e^{-\frac{2\xi x}{2\sigma^2}} d\xi

= \int_{-\infty}^{\infty} e^{-\frac{\xi^2}{2\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \frac{2\xi}{2\sigma^2} d\xi

= \sqrt{2\pi \sigma^2} e^{-\frac{x^2}{2\sigma^2}}

Properties

The product of the convolution of two Gaussian functions with a spread \( \sigma \) is a Gaussian function with a spread \( \sqrt{2\sigma} \) scaled by the area of the Gaussian filter.

Properties

The product of the convolution of two Gaussian functions with a spread \( \sigma \) is a Gaussian function with a spread \( \sqrt{2\sigma} \) scaled by the area of the Gaussian filter.

Designing Gaussian Filters

Pascal's Triangle

(Binomial Expansion)

\[(1 + x)^n = \binom{n}{0} + \binom{n}{1} x + \binom{n}{2} x^2 + \cdots + \binom{n}{n} x^n.
\]

Binomial Coefficients

\[
(x+1)^0 = 1
\]

\[
(x+1)^1 = 1 + x
\]

\[
(x+1)^2 = 1 + 2x + x^2
\]

\[
(x+1)^3 = 1 + 3x + 3x^2 + x^3
\]

\[
(x+1)^4 = 1 + 4x + 6x^2 + 4x^3 + x^4
\]

\[
(x+1)^5 = 1 + 5x + 10x^2 + 10x^3 + 5x^4 + x^5
\]

Example: 6 choose 3

\[
6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1
\]

For example, \[6;3\] = \[3;2\] \[2;1\] = 20

\[
3 \cdot 2 \cdot 1 \cdot 3 \cdot 2 \cdot 1
\]

\[
(x+1)^5 = 1 + 5x + 10x^2 + 10x^3 + 5x^4 + x^5
\]
Another Way: Compute the Weights

- Start with a discrete Gaussian

\[ g[i, j] = e^{-\frac{(i^2 + j^2)}{2\sigma^2}} \]

- Normalize the weights

\[ g[i, j] = \frac{e^{-\frac{(i^2 + j^2)}{2\sigma^2}}}{c} \]

Example: \(\sigma^2=2\), \(n=7\)

<table>
<thead>
<tr>
<th>(i, j)</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>0.011</td>
<td>0.039</td>
<td>0.082</td>
<td>0.105</td>
<td>0.082</td>
<td>0.039</td>
<td>0.011</td>
</tr>
<tr>
<td>-2</td>
<td>0.039</td>
<td>0.135</td>
<td>0.287</td>
<td>0.368</td>
<td>0.287</td>
<td>0.135</td>
<td>0.039</td>
</tr>
<tr>
<td>-1</td>
<td>0.082</td>
<td>0.287</td>
<td>0.606</td>
<td>0.779</td>
<td>0.606</td>
<td>0.287</td>
<td>0.082</td>
</tr>
<tr>
<td>0</td>
<td>0.105</td>
<td>0.368</td>
<td>0.779</td>
<td>1.000</td>
<td>0.779</td>
<td>0.368</td>
<td>0.105</td>
</tr>
<tr>
<td>1</td>
<td>0.082</td>
<td>0.287</td>
<td>0.606</td>
<td>0.779</td>
<td>0.606</td>
<td>0.287</td>
<td>0.082</td>
</tr>
<tr>
<td>2</td>
<td>0.039</td>
<td>0.135</td>
<td>0.287</td>
<td>0.368</td>
<td>0.287</td>
<td>0.135</td>
<td>0.039</td>
</tr>
<tr>
<td>3</td>
<td>0.011</td>
<td>0.039</td>
<td>0.082</td>
<td>0.105</td>
<td>0.082</td>
<td>0.039</td>
<td>0.011</td>
</tr>
</tbody>
</table>

To keep them all integers

\[ \frac{g[3,3]}{k} = e^{-\frac{(3^2 + 3^2)}{2\sigma^2}} = 0.011 \implies k = \frac{g[3,3]}{k} = \frac{1.0}{0.011} = 91. \]

Integer Weights

<table>
<thead>
<tr>
<th>(i, j)</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>4</td>
<td>12</td>
<td>26</td>
<td>33</td>
<td>26</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>-1</td>
<td>7</td>
<td>26</td>
<td>55</td>
<td>71</td>
<td>55</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>33</td>
<td>71</td>
<td>91</td>
<td>71</td>
<td>33</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>26</td>
<td>55</td>
<td>71</td>
<td>55</td>
<td>26</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>12</td>
<td>26</td>
<td>33</td>
<td>26</td>
<td>12</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
Normalization constant

\[ \sum_{i=-3}^{3} \sum_{j=-3}^{3} g[i, j] = 1115. \]

\[ h[i, j] = \frac{1}{1115} (f[i, j] \ast g[i, j]) \]

Discrete Gaussian Filters

7x7 Gaussian Mask

3D Plot of the 7x7 Gaussian

15 x 15 Gaussian Mask

Properties of Discrete Gaussian Filters

- Step 1: smooth with n x n discrete Gaussian Filter
- Step 2: smooth the intermediary result from Step 1 with m x m discrete Gaussian Filter
- Step 1 + Step 2 are equivalent to smoothing the original with (n+m-1)x(n+m-1) discrete Gaussian Filter
THE END