Filtering
grayscale image

Replace each local 3x3 window with average
Weighted averages

Perhaps center pixel should get more weight

A Gaussian kernel gives less weight to pixels further from the center of the window.
Weighted averages

Perhaps center pixel should get more weight

\[
\begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
\]

\[H[i,j]\]

\[F[i,j]\]

Can we write an equation for output \[G[i,j]\]?

[on board]
Convolution vs correlation

Convolution:

\[ G[i, j] = F \ast H = \sum_{u} \sum_{v} H[u, v] F[i - u, j - v] \]

Correlation:

\[ G[i, j] = F \otimes H = \sum_{u} \sum_{v} H[u, v] F[i + u, j + v] \]

Matlab >> conv2(H,F)

How can one use convolution to perform correlation?
Cross-correlation

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

Matlab >> conv2(H,F)

How would be use convolution routine to perform cross correlation?
Examples

Original  \[ \frac{1}{9} \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} \] = Blur (with a mean filter)
Examples

Original

\[
\begin{array}{cccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Examples

Original

Filtered (no change)

Source: D. Lowe
Examples

<table>
<thead>
<tr>
<th>Original</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Original Image]</td>
<td>![Filtered Image]</td>
</tr>
</tbody>
</table>

0 0 0
0 0 1
0 0 0
Examples

What would this look like for convolution?
Examples

What would this look like for convolution?
Practice with linear filters

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Original

Original

Source: D. Lowe
Original

\[
\begin{pmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{pmatrix}
- \begin{pmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{pmatrix} \times \frac{1}{16} + \begin{pmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{pmatrix}
= ?
\]
Practice with linear filters

Original

Unsharp filter (HW 2)
Derivative filters

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

Vertical Edge (absolute value)
Examples

Horizontal Edge (absolute value)

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

\[ G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \]

\[ G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]
Gaussian

- Lots of special properties
- Statistical: central limit theorem
- Fourier transform of Gaussian is a Gaussian (next week)
- An infinite chain of repeated filter convolutions converges to a Gaussian
Gaussian as infinite sequence of repeated filters (nearly converges after 3 iterations of box filtering)
Gaussian filter

\[ G_\sigma = \frac{1}{2\pi \sigma^2} e^{-\frac{(x^2 + y^2)}{2\sigma^2}} \]

\[ \sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \]

\[ \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]

Standard deviation \( \sigma \): determines extent of smoothing

Matlab: \[ \text{>> } G = \text{FSPECIAL(}'gaussian'\text{,HSIZE,SIGMA)} \]
Gaussian filters

$\sigma = 1$ pixel
$\sigma = 5$ pixels
$\sigma = 10$ pixels
$\sigma = 30$ pixels
Finite-support filters

What should HSIZE be?
Rule-of-thumb

Set radius of filter to be 3 sigma
Denoising

Additive Gaussian Noise

Gaussian Filter
Smoothing with larger standard deviations suppresses noise, but also blurs the image.

Reducing Gaussian noise

Source: S. Lazebnik
Reducing salt-and-pepper noise by Gaussian smoothing

3x3  5x5  7x7
Alternative idea: Median filtering

A **median filter** operates over a window by selecting the median intensity in the window.
Median filter

What advantage does median filtering have over Gaussian filtering?

- Robustness to outliers

Source: K. Grauman
Median filter

Salt-and-pepper noise  Median filtered

MATLAB: medfilt2(image, [h w])

Is this a linear shift-invariant operator?

Source: M. Hebert