Filtering Images in the Spatial Domain
Chapter 3b G&W
Ross Whitaker
(modified by Guido Gerig)
School of Computing
University of Utah
Overview

• Correlation and convolution
• Linear filtering
  – Smoothing, kernels, models
  – Detection
  – Derivatives
• Nonlinear filtering
  – Median filtering
  – Bilateral filtering
  – Neighborhood statistics and nonlocal filtering
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104 99 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>87 95 103 150 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>50 36 150 104 ...</td>
<td>0.1 0.6 0.1</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>20 47 205 77 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>. . . . . .</td>
<td>. . . . . .</td>
<td>. . . . . .</td>
</tr>
</tbody>
</table>
Cross Correlation

- Operation on image neighborhood and small ...  
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104</td>
<td>99 ...</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>87 95 103</td>
<td>150 ...</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>50 36 150</td>
<td>104 ...</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>20 47 205</td>
<td>77 ...</td>
<td>. . . . . .</td>
</tr>
<tr>
<td></td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td></td>
<td>0.1 0.6 0.1</td>
<td>. . . . . .</td>
</tr>
<tr>
<td></td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td></td>
<td>. . . . .</td>
<td>. . . . .</td>
</tr>
</tbody>
</table>
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>87 95 103</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>50 36 150</td>
<td>0.1 0.6 0.1</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>20 47 205</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
</tbody>
</table>
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104</td>
<td>99</td>
<td></td>
</tr>
<tr>
<td>87 95 103</td>
<td>150</td>
<td>0.0 0.1 0.0</td>
</tr>
<tr>
<td>50 36 150</td>
<td>104</td>
<td>0.1 0.6 0.1</td>
</tr>
<tr>
<td>20 47 205</td>
<td>77</td>
<td>0.0 0.1 0.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.0<em>87 + 0.1</em>95 + 0.0*103</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 0.1<em>50 + 0.6</em>36 + 0.1*150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>+ 0.0<em>20 + 0.1</em>47 + 0.0*205 = 55.8</td>
</tr>
</tbody>
</table>
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

\[
\begin{align*}
\text{Input image:} & \quad 100 & 130 & 104 & 99 & \ldots \\
& \quad 87 & 95 & 103 & 150 & \ldots \\
& \quad 50 & 36 & 150 & 104 & \ldots \\
& \quad 20 & 47 & 205 & 77 & \ldots \\
\end{align*}
\]

\[
\begin{align*}
\text{Filter:} & \quad \begin{bmatrix} 0.0 & 0.1 & 0.0 \\ 0.1 & 0.6 & 0.1 \\ 0.0 & 0.1 & 0.0 \end{bmatrix} \\
\text{Output image:} & \quad \begin{bmatrix} \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot \\
\end{bmatrix}
\end{align*}
\]

\[
0.0 \times 87 + 0.1 \times 95 + 0.0 \times 103 \\
+ 0.1 \times 50 + 0.6 \times 36 + 0.1 \times 150 \\
+ 0.0 \times 20 + 0.1 \times 47 + 0.0 \times 205 = 55.8
\]
Cross Correlation

- Operation on image neighborhood and small “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104 99 ...</td>
<td>0.0 0.1 0.0</td>
<td>0.0 0.1 0.0</td>
</tr>
<tr>
<td>87 95 103 150 ...</td>
<td>0.1 0.6 0.1</td>
<td>. . 55.8 .</td>
</tr>
<tr>
<td>50 36 150 104 ...</td>
<td>. . 0.0 0.1 0.0</td>
<td>. . .</td>
</tr>
<tr>
<td>20 47 205 77 ...</td>
<td>. . . .</td>
<td>. . .</td>
</tr>
</tbody>
</table>

\[
0.0 \times 87 + 0.1 \times 95 + 0.0 \times 103 + 0.1 \times 50 + 0.6 \times 36 + 0.1 \times 150 + 0.0 \times 20 + 0.1 \times 47 + 0.0 \times 205 = 55.8
\]
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104 99</td>
<td>0.0 0.1 0.0</td>
<td>. . . . .</td>
</tr>
<tr>
<td>87 95 103 150</td>
<td>0.1 0.6 0.1</td>
<td>. 55.8 .</td>
</tr>
<tr>
<td>50 36 150 104</td>
<td>0.0 0.1 0.0</td>
<td>. . . .</td>
</tr>
<tr>
<td>20 47 205 77</td>
<td>. . . . .</td>
<td>. . . .</td>
</tr>
<tr>
<td>. . . . .</td>
<td>. . . .</td>
<td>. . . .</td>
</tr>
</tbody>
</table>
# Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104 99 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>87 95 103 150 ...</td>
<td>0.1 0.6 0.1</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>50 36 150 104 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>20 47 205 77 ...</td>
<td></td>
<td>. . . . . .</td>
</tr>
<tr>
<td>. . . . . .</td>
<td></td>
<td>. . . . . .</td>
</tr>
</tbody>
</table>
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104 99 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>87 95 103 150 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>50 36 150 104 ...</td>
<td>0.1 0.6 0.1</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>20 47 205 77 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . . .</td>
</tr>
<tr>
<td>. . . . . .</td>
<td>. . . . . .</td>
<td>. . . . . .</td>
</tr>
</tbody>
</table>
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

<table>
<thead>
<tr>
<th>Input image</th>
<th>Filter</th>
<th>Output image</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 130 104 99 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . .</td>
</tr>
<tr>
<td>87 95 103 150 ...</td>
<td>0.1 0.6 0.1</td>
<td>. . . . .</td>
</tr>
<tr>
<td>50 36 150 104 ...</td>
<td>0.0 0.1 0.0</td>
<td>. . . . .</td>
</tr>
<tr>
<td>20 47 205 77 ...</td>
<td></td>
<td>. . . . .</td>
</tr>
</tbody>
</table>

\[
0.0\times95 + 0.1\times103 + 0.0\times150 + 0.1\times36 + 0.6\times150 + 0.1\times104 + 0.0\times47 + 0.1\times205 + 0.0\times77 = 134.8
\]
Cross Correlation

- Operation on image neighborhood and small …
  - “mask”, “filter”, “stencil”, “kernel”
- Linear operations within a moving window

\[
\begin{align*}
100 & \quad 130 & \quad 104 & \quad 99 & \quad \ldots \\
87 & \quad 95 & \quad 103 & \quad 150 & \quad \ldots \\
50 & \quad 36 & \quad 150 & \quad 104 & \quad \ldots \\
20 & \quad 47 & \quad 205 & \quad 77 & \quad \ldots \\
\end{align*}
\]

\[
\begin{align*}
\text{Filter} & : & 0.0 & 0.1 & 0.0 \\
& : & 0.1 & 0.6 & 0.1 \\
& : & 0.0 & 0.1 & 0.0 \\
\end{align*}
\]

\[
\begin{align*}
\text{Input image} & \rightarrow \text{Output image} \\
0.0 \times 95 & + 0.1 \times 103 & + 0.0 \times 150 \\
& + 0.1 \times 36 & + 0.6 \times 150 & + 0.1 \times 104 \\
& + 0.0 \times 47 & + 0.1 \times 205 & + 0.0 \times 77 & = 134.8
\end{align*}
\]
Cross Correlation

• **1D**
  \[ g(x) = \sum_{s=-a}^{a} w(s) f(x + s) \]

• **2D**
  \[ g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t) \]

\[
\begin{bmatrix}
  w(-a, -b) & \cdots & \cdots & w(a, -b) \\
  \vdots & \ddots & \ddots & \vdots \\
  \vdots & \cdots & w(0, 0) & \cdots \\
  w(-a, b) & \cdots & \cdots & w(a, b)
\end{bmatrix}
\]
Correlation: Technical Details

• How to filter boundary?
Correlation: Technical Details

- **Boundary conditions**
  - Boundary not filtered (keep it 0)
  - Pad image with amount \((a, b)\)
    - Constant value or repeat edge values
  - Cyclical boundary conditions
    - Wrap or mirroring
Correlation: Technical Details

• Boundaries
  – Can also modify kernel – no longer correlation

• For analysis
  – Image domains infinite
  – Data compact (goes to zero far away from origin)

\[ g(x, y) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s, t) f(x + s, y + t) \]
Correlation: Properties

• Shift invariant

\[ g = w \circ f \quad g(x, y) = w(x, y) \circ f(x, y) \]

\[ w(x, y) \circ f(x-x_0, y-y_0) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s, t) f(x-x_0+s, y-y_0+t) = g(x-x_0, y-y_0) \]
Correlation: Properties

• **Shift invariant**

\[ g = w \circ f \quad \text{and} \quad g(x, y) = w(x, y) \circ f(x, y) \]

\[ w(x, y) \circ f(x-x_0, y-y_0) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} w(s, t) f(x-x_0+s, y-y_0+t) = g(x-x_0, y-y_0) \]

• **Linear**

\[ w \circ (\alpha e + \beta f) = \alpha w \circ e + \beta w \circ f \]

**Compact notation**

\[ C_{wf} = w \circ f \]
Filters: Considerations

- **Normalize**
  - Sums to one
  - Sums to zero (some cases, see later)
- **Symmetry**
  - Left, right, up, down
  - Rotational
- **Special case: auto correlation**

\[ C_{ff} = f \circ f \]
Examples 1
Examples 1

```
0 0 0
0 1 0
0 0 0
0 0 0
```
Examples 1

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}
\]

\[
\frac{1}{9} * 
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\
\end{bmatrix}
\]
Examples 1

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1
\end{bmatrix}
\]
Examples 1
Examples 2

\[
\begin{array}{cccc}
1 & 1 & 1 & 1 \\
1/9 * & 1 & 1 & 1 \\
1 & 1 & 1 & 1 \\

1 & 1 & 1 & 1 \\
1/25 * & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
\]
Smoothing and Noise

Noisy image

5x5 box filter
Noise Analysis

• Consider an a simple image $I()$ with additive, uncorrelated, zero-mean noise of variance $s$
• What is the expected rms error of the corrupted image?
• If we process the image with a box filter of size $2a+1$ what is the expected error of the filtered image?

\[
\text{RMSE} = \left( \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} (\tilde{I}(x,y) - I(x,y))^2 \right)^{\frac{1}{2}}
\]
Other Filters

• **Disk**
  – Circularly symmetric, jagged in discrete case

• **Gaussians**
  – Circularly symmetric, smooth for large enough stdev
  – Must normalize in order to sum to one

• **Derivatives** – discrete/finite differences
  – Operators
Gaussian Kernel

\[ \sigma = 1; \quad \text{Plot} \left[ \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}, \{x, -4, 4\}, \text{ImageSize} \to \right] \]

Figure 2.1 The Gaussian kernel with unit standard deviation in 1D.
Gaussian Kernel

\[ G_{1D}(x; \sigma) = \frac{1}{\sqrt{2\pi} \sigma} e^{-\frac{x^2}{2\sigma^2}}, \quad G_{2D}(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad G_{ND}(x; \sigma) = \frac{1}{(\sqrt{2\pi} \sigma)^N} e^{-\frac{\|x\|^2}{2\sigma^2}} \]

Normalization to 1.0

Figure 3.2 The Gaussian function at scales \( \sigma = .3 \), \( \sigma = 1 \) and \( \sigma = 2 \). The kernel is normalized, so the total area under the curve is always unity.
Box versus Gaussian
Convolution

Java demo: http://www.jhu.edu/signals/convolve/
http://www.jhu.edu/signals/discreteconv2/index.html

- **Discrete**

\[ g(x, y) = w(x, y) * f(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t) \]

- **Continuous**

\[ g(x, y) = w(x, y) * f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} w(s, t) f(x - s, y - t) ds \, dt \]

- Same as cross correlation with kernel transposed around each axis

- The two operations (correlation and convolution) are the same if the kernel is symmetric about axes

\[ g = w \circ f = w^* * f \]

\[ w^* \text{ reflection of } w \]
Convolution: Properties

• Shift invariant, linear
• Commutative

\[ f \ast g = g \ast f \]

• Associative

\[ f \ast (g \ast h) = (f \ast g) \ast h \]

• Others (discussed later):
  – Derivatives, convolution theorem, spectrum…
Computing Convolution
Computing Convolution

• Compute time
  – MxM mask
  – NxN image
Computing Convolution

• Compute time
  – $M \times M$ mask
  – $N \times N$ image

\[
O(M^2N^2) \quad \text{“for” loops are nested 4 deep}
\]
Computing Convolution

• Compute time
  – MxM mask
  – NxN image

\[
O(M^2N^2) \quad \text{“for” loops are nested 4 deep}
\]

• Special case: separable

Two 1D kernels

\[
w = w_x * w_y
\]

\[
w * f = (w_x * w_y) * f = w_x * (w_y * f)
\]

O(M^2N^2)  O(MN^2)
Separable Kernels

• Examples
  – Box/rectangle
  – Bilinear interpolation
  – Combinations of partial derivatives
    • $\frac{d^2f}{dx\,dy}$
  – Gaussian
    • Only filter that is both circularly symmetric and separable

• Counter examples
  – Disk
  – Cone
  – Pyramid
Separability

\[ G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

\[ = \left( \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{x^2}{2\sigma^2} \right) \right) \left( \frac{1}{\sqrt{2\pi}\sigma} \exp \left( -\frac{y^2}{2\sigma^2} \right) \right) \]

The 2D Gaussian can be expressed as the product of two functions, one a function of \( x \) and the other a function of \( y \).

In this case, the two functions are the (identical) 1D Gaussian.

![2D Gaussian](image)

**Figure 3.7** A product of Gaussian functions gives a higher dimensional Gaussian function. This is a consequence of the separability.
Digital Images: Boundaries are “Lines” or “Discontinuities”

Example: Characterization of discontinuities?

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf
Digital Images: Boundaries are “Lines” or “Discontinuities”

Example: Characterization of discontinuities?

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf
Characterizing edges

An edge is a place of rapid change in the image intensity function.

- Image
- Intensity function (along horizontal scanline)
- First derivative
  
  Edges correspond to extrema of derivative.

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf
Differentiation and convolution

- Recall, for 2D function, \( f(x,y) \):

\[
\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x+\varepsilon,y) - f(x,y)}{\varepsilon} \right)
\]

- We could approximate this as

\[
\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1},y) - f(x_n,y)}{\Delta x}
\]

- This is linear and shift invariant, so must be the result of a convolution.

- (which is obviously a convolution)

\[
\begin{bmatrix}
-1 & 1
\end{bmatrix}
\]
Derivatives: Finite Differences

\[ \frac{\partial f}{\partial x} \approx \frac{1}{2h} (f(x + 1, y) - f(x - 1, y)) \]

\[ \frac{\partial f}{\partial x} \approx w_{dx} \circ f \quad w_{dx} = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \]

\[ \frac{\partial f}{\partial y} \approx w_{dy} \circ f \quad w_{dy} = \begin{bmatrix} -\frac{1}{2} \\ 0 \\ \frac{1}{2} \end{bmatrix} \]
Derivative Example

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
\end{bmatrix}
\]
Image gradient

The gradient of an image: \[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid change in intensity.

\[ \nabla f = \left[ \frac{\partial f}{\partial x}, 0 \right] \]

\[ \nabla f = \left[ 0, \frac{\partial f}{\partial y} \right] \]

The gradient direction is given by:

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]

- How does this relate to the direction of the edge? *perpendicular*

The edge strength is given by the gradient magnitude:

\[ ||\nabla f|| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Finite differences: example

Which one is the gradient in the x-direction (resp. y-direction)?
Finite difference filters

Other approximations of derivative filters exist:

Prewitt: \[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}; \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \]

Sobel: \[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}; \quad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

Roberts: \[ M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}; \quad M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]
Pattern Matching
Pattern Matching/Detection

• The optimal (highest) response from a filter is the autocorrelation evaluated at position zero

\[
\max_{\bar{x}} C_{f,f}(\bar{x}) = C_{f,f}(0) = \int f(\bar{s})f(\bar{s}) d\bar{s}
\]

• A filter responds best when it matches a pattern that looks itself

• Strategy
  – Detect objects in images by correlation with “matched” filter
Matched Filter Example
Matched Filter Example:
Correlation of template with image
Matched Filter Example:
Thresholding of correlation results
Matched Filter Example: High correlation → template found
Summary of 9/15

• Spatial Filtering
  – Consider neighborhood information
  – Special consideration at boundary
Summary of 9/15

- Common Filters

<table>
<thead>
<tr>
<th>Box</th>
<th>Gaussian</th>
<th>Derivative</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Box Filter" /></td>
<td><img src="image2" alt="Gaussian Filter" /></td>
<td><img src="image3" alt="Derivative Filter" /></td>
</tr>
<tr>
<td><img src="image4" alt="Box Filter Coefficients" /></td>
<td><img src="image5" alt="Gaussian Filter Coefficients" /></td>
<td><img src="image6" alt="Derivative Filter Coefficients" /></td>
</tr>
</tbody>
</table>
Box versus Gaussian
Gaussian Filtering
Gaussian Filtering

σ = 2.72
σ = 3.00
σ = 3.32
σ = 3.67
σ = 4.06
σ = 4.48
σ = 4.95

σ = 5.47
σ = 6.05
σ = 6.69
σ = 7.39
σ = 8.17
σ = 9.03
σ = 9.97

σ = 11.02
σ = 12.18
σ = 13.46
σ = 14.88
σ = 16.44
σ = 18.17
σ = 20.09
Finite differences: example

- Which one is the gradient in the x-direction (resp. y-direction)?
Cross-correlation and Convolution

• Cross-correlation

\[ g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x + s, y + t) \]

• Convolution

\[ g(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t) \]
Key Concepts

• Separability

\[ w = w_x \ast w_y \]

\[ w \ast f = (w_x \ast w_y) \ast f = w_x \ast (w_y \ast f) \]

\[ O(M^2N^2) \quad O(MN^2) \]

\[ G_\sigma(x, y) = \frac{1}{2\pi \sigma^2} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

\[ = \left( \frac{1}{\sqrt{2\pi} \sigma} \exp \left( -\frac{x^2}{2\sigma^2} \right) \right) \left( \frac{1}{\sqrt{2\pi} \sigma} \exp \left( -\frac{y^2}{2\sigma^2} \right) \right) \]

The 2D Gaussian can be expressed as the product of two functions, one a function of \( x \) and the other a function of \( y \).

In this case, the two functions are the (identical) 1D Gaussian
Nonlinear Methods For Filtering

- Median filtering
- Bilateral filtering
- Neighborhood statistics and nonlocal filtering
Median Filtering

- For each neighborhood in image
  - Sliding window
  - Usually odd size (symmetric) 5x5, 7x7,…
- Sort the greyscale values
- Set the center pixel to the median
- Important: use “Jacobi” updates
  - Separate input and output buffers
  - All statistics on the original image
Median vs Gaussian

Original + Gaussian Noise

3x3 Median

3x3 Box
Median Filter

• **Issues**
  – Boundaries
    • Compute on pixels that fall within window
  – **Computational efficiency**
    • What is the best algorithm?

• **Properties**
  – Removes outliers (replacement noise – salt and pepper)
  – Window size controls size of structures
  – Preserves straight edges, but rounds corners and features
Replacement Noise

- Also: “shot noise”, “salt&pepper”
- Replace certain % of pixels with samples from pdf
- Best filtering strategy: filter to avoid outliers
Smoothing of S&P Noise

- It’s not zero mean (locally)
- Averaging produces local biases
Smoothing of S&P Noise

- It’s not zero mean (locally)
- Averaging produces local biases
Median Filtering

Median 3x3

Median 5x5
Median Filtering
Median Filtering

- Iterate

Median 3x3

2x Median 3x3
Median Filtering

- **Image model:** piecewise constant (flat)
Median Filtering

• Image model: piecewise constant (flat)
Median Filtering

• Image model: piecewise constant (flat)
Median Filtering

- **Image model:** piecewise constant (flat)
Order Statistics

- Median is special case of order-statistics filters
- Instead of weights based on neighborhoods, weights are based on ordering of data

\[
\begin{align*}
\text{Neighborhood} & : X_1, X_2, \ldots, X_N \\
\text{Ordering} & : X(1) \leq X(2) \leq \ldots \leq X(N)
\end{align*}
\]

Filter
\[
F(X_1, X_2, \ldots, X_N) = \alpha_1 X(1) + \alpha_2 X(2) + \ldots + \alpha_N X(N)
\]

- Neighborhood average (box)
  \[
  \alpha_i = 1/N
  \]
- Median filter
  \[
  \alpha_i = \begin{cases} 
  1 & i = (N + 1)/2 \\
  0 & \text{otherwise}
  \end{cases}
  \]
- Trimmed average (outlier removal)
  \[
  \alpha_i = \begin{cases} 
  1/M & (N - M + 1)/2 \leq i \leq (N + M + 1)/2 \\
  0 & \text{otherwise}
  \end{cases}
  \]
Median filter

- What advantage does median filtering have over Gaussian filtering?
- Robustness to outliers

Source: K. Grauman

Source: http://web.media.mit.edu/~maov/classes/vision09/lect/09_Image_Filtering_Edge_Detection_09.pdf
Piecewise Flat Image Models

- Image piecewise flat -> average only within similar regions
- Problem: don’t know region boundaries
Piecewise-Flat Image Models

• Assign probabilities to other pixels in the image belonging to the same region

• Two considerations
  – **Distance**: far away pixels are less likely to be same region
  – **Intensity**: pixels with different intensities are less likely to be same region
Gaussian: Blur Comes from Averaging across Edges

Same Gaussian kernel everywhere.

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt
Bilateral Filter
No Averaging across Edges

The kernel shape depends on the image content.

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt
Main Idea

Distance (kernel/pdf)

$$G(x_i - x_j)$$

Prob pixel belongs to same region as $i$

position

Distance (pdf)

$$H(f_i - f_j)$$

Prob pixel belongs to same region as $i$

intensity
Bilateral Filter Definition: an Additional Edge Term

Same idea: **weighted average of pixels.**

\[
BF \left[ I \right]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\| p - q \|) G_{\sigma_r}(\| I_p - I_q \|) I_q
\]

- **Normalization factor**
- **Space weight**
- **Range weight**

Source: [http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt](http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt)
Illustration a 1D Image

- 1D image = line of pixels

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt
Gaussian Blur and Bilateral Filter

Gaussian blur

\[
GB[I]_p = \sum_{q \in S} G_{\sigma}(\|p - q\|) I_q
\]

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt
Gaussian Blur and Bilateral Filter

Gaussian blur

\[
GB[I]_p = \sum_{q \in S} G_{\sigma}(\|p - q\|) I_q
\]

Bilateral filter

\[
BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) I_q
\]

[Aurich 95, Smith 97, Tomasi 98]

Source: http://people.csail.mit.edu/sparis/bf_course/slides/03_definition_bf.ppt
Bilateral Filter

- Neighborhood – sliding window
- Weight contribution of neighbors according to:
  \[ f_i \leftarrow k_i^{-1} \sum_{j \in N} f_j G(x_i - x_j) H(f_i - f_j) \]
  \[ k_i = \sum_{j \in N} G(x_i - x_j) H(f_i - f_j) \]
  normalization: all weights add up to 1

- \( G \) is a Gaussian (or lowpass), as is \( H \), \( N \) is neighborhood,
  – Often use \( G(r_{ij}) \) where \( r_{ij} \) is distance between pixels
  – Update must be normalized for the samples used in this (particular) summation

- Spatial Gaussian with extra weighting for intensity
  – Weighted average in neighborhood with downgrading of intensity outliers

Tomasi, Manduchi: http://en.wikipedia.org/wiki/Bilateral_filter
When the bilateral filter is centered, say, on a pixel on the bright side of the boundary, the similarity function $s$ assumes values close to one for pixels on the same side, and values close to zero for pixels on the dark side. The similarity function is shown in figure 1(b) for a 23x23 filter support centered two pixels to the right of the step in figure 1(a).
Bilateral Filtering

Replaces the pixel value at $x$ with an average of similar and nearby pixel values.
Bilateral Filtering

Gaussian Blurring  Bilateral
Nonlocal Averaging

• Recent algorithm
  – NL-means, Baudes et al., 2005
  – UINTA, Awate & Whitaker, 2005

• Different model
  – No need for piecewise-flat
  – Images consist of some set of pixels with similar neighborhoods → average several of those
    • Scattered around
      – General area of a pixel
      – All around

• Idea
  – Average sets of pixels with similar neighborhoods
Nonparametric Markov Modeling

Neighborhoods in images lie on low-dimensional manifolds in high dimensional spaces (we call this a feature space) (i) several studies on natural image statistics; and (ii) several previous works on empirical Markov statistics.

High-dimensional feature space of image neighborhoods

Suyash P. Awate, Ross T. Whitaker
Unsupervised, Information-Theoretic, Adaptive Image Filtering with Applications to Image Restoration

http://www.cs.utah.edu/~suyash/pubs/uinta/
Nonlocal Averaging

• **Strategy:**
  – Average pixels to alleviate noise
  – Combine pixels with similar neighborhoods

• **Formulation**
  – $n_{i,j}$ – vector of pixels values, indexed by $j$, from neighborhood around pixel $i$
Nonlocal Averaging Formulation

• Distance between neighborhoods

\[ d_{i,k} = d(n_i, n_k) = ||n_i - n_k|| = \left( \sum_{j=1}^{N} (n_{i,j} - n_{k,j})^2 \right)^{\frac{1}{2}} \]

• Kernel weights based on distances

\[ w_{i,k} = K(d_{i,k}) = e^{-\frac{d_{i,k}^2}{2\sigma^2}} \]

• Pixel values of k neighborhoods: \( f_k \)
Averaging Pixels Based on Weights

- For each pixel, i, choose a set of pixel locations k:
  - $k = 1, \ldots, M$
  - Average them together based on neighborhood weights (prop. to intensity pattern difference)

\[
g_i \leftarrow \frac{1}{\sum_{k=1}^{M} w_{i,k}} \sum_{k=1}^{M} w_{i,k} f_k
\]
Nonlocal Averaging
Some Details

• Window sizes: good range is 5x5- >11x11

• How to choose samples:
  – Random samples from around the image
    • UINTA, Awate&Whitaker
  – Block around pixel (bigger than window, e.g. 51x51)
    • NL-means

• Iterate
  – UNITA: smaller updates and iterate
NL-Means Algorithm

• For each pixel, p
  – Loop over set of pixels nearby
  – Compare the neighborhoods of those pixels to the neighborhood of p and construct a set of weights
  – Replace the value of p with a weighted combination of values of other pixels

• Repeat… but 1 iteration is pretty good
Results

Noisy image (range 0.0-1.0)  Bilateral filter (3.0, 0.1)
Results

Bilateral filter (3.0, 0.1)  
NL means (7, 31, 1.0)
Results

Bilateral filter (3.0, 0.1)  NL means (7, 31, 1.0)
Less Noisy Example
Less Noisy Example
Results

Original

Noisy

Filtered
Checkerboard With Noise

Original

Noisy

Filtered
Quality of Denoising

- σ, joint entropy, and RMS-error vs. number of iterations
MRI Head
MRI Head
Fingerprint

![Fingerprint Image 1](image1)

![Fingerprint Image 2](image2)
Fingerprint
Results

Original  Noisy  Filtered
Results

Original

Noisy

Filtered
Results
Fractal

Original  Noisy  Filtered
Piecewise Constant

- Several 10s of Iterations
- Tends to obliterate rare events (e.g. corners)
Texture, Structure