Linear Filters and Image Processing

EECS 598-08 Fall 2014  
Foundations of Computer Vision  
Instructor: Jason Corso (jjcorso)  
web.eecs.umich.edu/~jjcorso/t/598F14

Readings:  FP 4, 6.1, 6.4; SZ 3  
Date:  9/24/14

Materials on these slides have come from many sources in addition to myself; I am infinitely grateful to these, especially Greg Hager, Silvio Savarese, and Steve Seitz.
Topics

• Linear filters
• Scale-space and image pyramids
• Image denoising
• Representing texture by filters
De-noising

Original
Salt and pepper noise

Super-resolution

In-painting

Image Inpainting, M. Bertalmio et al.
http://www.iua.upf.es/~mbertalmio/restoration.html

Image Inpainting, M. Bertalmio et al.
http://www.iua.upf.es/~mbertalmio/restoration.html

Source: Savarese Slides
Images as functions

- We can think of an image as a function, \( f \), from \( \mathbb{R}^2 \rightarrow \mathbb{R} \):
  - \( f(x, y) \) gives the intensity at position \((x, y)\)
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    \[
    f : [a, b] \times [c, d] \rightarrow [0, 1]
    \]

- A color image is just three functions pasted together. We can write this as a "vector-valued" function:
  \[
  f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}
  \]
Images as functions

Source: Seitz and Szeliski Slides
What is a digital image?

- We usually work with **digital (discrete)** images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)
- If our samples are $\Delta$ apart, we can write this as:
  \[ f[i, j] = \text{Quantize}\{f(i\Delta, j\Delta)\} \]
- The image can now be represented as a matrix of integer values.

Source: Seitz and Szeliski Slides
Filtering noise

• How can we “smooth” away noise in an image?

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>130</td>
<td>110</td>
<td>120</td>
<td>110</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>110</td>
<td>90</td>
<td>100</td>
<td>90</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
<td>100</td>
<td>90</td>
<td>130</td>
<td>110</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>120</td>
<td>100</td>
<td>130</td>
<td>110</td>
<td>120</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>110</td>
<td>80</td>
<td>120</td>
<td>100</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Source: Seitz and Szeliski Slides
Mean filtering

$$F[x, y]$$

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

$$G[x, y]$$

Source: Seitz and Szeliski Slides
Mean filtering

\[ F[x, y] \]

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ G[x, y] \]

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>60</td>
<td>60</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>60</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>50</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>50</td>
<td>80</td>
<td>80</td>
<td>90</td>
<td>60</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>30</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td>40</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Seitz and Szeliski Slides
Cross-correlation filtering

- As an equation: Assume the window is \((2k+1)\times(2k+1)\):

\[
G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]
\]

- We can generalize this idea by allowing different weights for different neighboring pixels:

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

- This is called a **cross-correlation** operation and written:

\[
G = H \otimes F
\]

- \(H\) is called the **filter**, **kernel**, or **mask**.

Source: Seitz and Szeliski Slides
Mean kernel

• What’s the kernel for a 3x3 mean filter?

Source: Seitz and Szeliski Slides
Gaussian filtering

- A Gaussian kernel gives less weight to pixels further from the center of the window

\[
\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

\[\frac{1}{16}\]

\[H[u, v]\]

- This kernel is an approximation of a Gaussian function:

\[
H[u, v] = \frac{1}{2\pi\sigma^2} \exp \left( -\frac{u^2 + v^2}{2\sigma^2} \right)
\]

- What happens if you increase \(\sigma\) ?

Source: Seitz and Szeliski Slides
Separability of the Gaussian filter

- The Gaussian function (2D) can be expressed as the product of two one-dimensional functions in each coordinate axis.
  - They are identical functions in this case.

\[
H[u, v] = \frac{1}{2\pi \sigma^2} \exp \left( - \frac{u^2 + v^2}{2\sigma^2} \right)
= \left( \frac{1}{\sqrt{2\pi} \sigma} \exp \left( - \frac{u^2}{2\sigma^2} \right) \right) \left( \frac{1}{\sqrt{2\pi} \sigma} \exp \left( - \frac{v^2}{2\sigma^2} \right) \right)
\]

- What are the implications for filtering?

Source: D. Lowe
Cameras are not perfect sensors *and*
Scenes never quite match our expectations
Noise Models

- Noise is commonly modeled using the notion of “additive white noise.”
  - Images: $I(u,v,t) = I^*(u,v,t) + n(u,v,t)$
  - Note that $n(u,v,t)$ is independent of $n(u',v',t')$ unless $u' = u, u' = u, t' = t$.
  - Typically we assume that $n$ (noise) is independent of image location as well --- that is, it is i.i.d
  - Typically we assume the $n$ is zero mean, that is $E[n(u,v,t)] = 0$

- A typical noise model is the Gaussian (or normal) distribution parametrized by $\pi$ and $\sigma$
  \[
p(x) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{(x - \mu)^2}{2\sigma^2} \right),\]

- This implies that no two images of the same scene are ever identical

Source: G Hager Slides
Gaussian Noise: 
sigma=1

Source: G Hager Slides
Gaussian Noise: sigma=16

Source: G Hager Slides
Mean vs. Gaussian filtering

Source: Seitz and Szeliski Slides
Smoothing by Averaging

Kernel: □

Source: G Hager, D. Kriegman Slides
Smoothing with a Gaussian

Kernel:
The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Source: G Hager, D. Kriegman Slides
Properties of Noise Processes

- Properties of temporal image noise:

\[
\text{Mean } \mu(i,j) = \frac{\sum I(u,v,t)}{n}
\]

\[
\text{Standard Deviation } \sigma_{i,j} = \sqrt{\frac{\sum (\mu(i,\varphi) - I(u,v,t))^2}{n}}
\]

\[
\text{Signal-to-noise Ratio } \frac{\mu(i,j)}{\sigma_{i,j}}
\]
**Image Noise**

- An experiment: take several images of a static scene and look at the pixel values

\[
\text{mean} = 38.6 \\
\text{std} = 2.99 \\
\text{Snr} = 38.6/2.99 \approx 13 \\
\text{max snr} = 255/3 \approx 85
\]

Source: G Hager Slides
PROPERTIES OF TEMPORAL IMAGE NOISE
(i.e., successive images)

- If standard deviation of grey values at a pixel is $s$ for a pixel for a single image, then the laws of statistics states that for independent sampling of grey values, for a temporal average of $n$ images, the standard deviation is:

\[
\frac{\sigma}{\text{Sqrt}(n)}
\]

- For example, if we want to double the signal to noise ratio, we could average 4 images.

Source: G Hager Slides
Temporal vs. Spatial Noise

• It is common to assume that:
  – spatial noise in an image is consistent with the temporal image noise
  – the spatial noise is independent and identically distributed

• Thus, we can think of a neighborhood of the image itself as approximated by an additive noise process

• Averaging is a common way to reduce noise
  – instead of temporal averaging, how about spatial?
• For example, for a pixel in image I at i,j

\[ I'(i, j) = \frac{1}{9} \sum_{i'=i-1}^{i+1} \sum_{j'=j-1}^{j+1} I(i', j') \]
Correlation and Convolution

- **Correlation:** \( G = H \otimes F \)

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

- **Convolution:** \( G = H \ast F \)

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

Source: https://www.youtube.com/watch?v=Ma0YONjMZLI
Convolution: Shift Invariant Linear Systems

- **Commutative:** \( F \ast H = H \ast F \)
  - Conceptually no difference between filter and signal

- **Associative:** \( F \ast (H \ast L) = (F \ast H) \ast L \)
  - Often apply several filters in sequence: \( (((F \ast H_1) \ast H_2) \ast H_3) \)
  - This is equivalent to applying one filter: \( F \ast (H_1 \ast H_2 \ast H_3) \)

- **Linearity / Distributes over addition:** \( F \ast (H_1 + H_2) = (F \ast H_1) + (F \ast H_2) \)

- **Scalars factor out:** \( kF \ast H = F \ast kH = k(F \ast H) \)

- **Shift-Invariance:** \( H \ast \text{Shift}(F) = \text{Shift}(H \ast F) \)

- **Identity:** unit impulse \( F \ast e = F \)

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

Source: Savarese Slides
Convolution: Properties

• **Linearity**: \( \text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2) \)

• **Shift invariance**: \( \text{filter} (\text{shift} (f)) = \text{shift} (\text{filter} (f)) \)
  (same behavior regardless of pixel location)

• Theoretany linear shift-invariant operator can be represented as a convolutionical result:

Source: Savarese Slides
Linear Filtering: Status Check!

original

Pixel offset

coefficient

1.0

?
Linear filtering (warm-up slide)
Linear filtering

original
shift

original

shifted
Linear filtering
Blurring

original

Blurred (filter applied in both dimensions).

Source: B. Freeman Slides
Blur examples

impulse

original

coefficient

Pixel offset

filtered

8

0.3

2.4
Blur examples

impulse

original

edge

original

coefficient

0.3

Pixel offset

0

filtered

2.4
Linear filtering (warm-up slide)

original

Source: B. Freeman Slides
Linear filtering (no change)
Linear filtering

original
(remember blurring)

original

Blurred (filter applied in both dimensions).
Sharpening

original

Sharpened original

Source: B. Freeman Slides
Sharpening

before

after

Source: B. Freeman Slides
What does blurring take away?

- original
- smoothed (5x5)
- detail

• Let’s add it back:

+ a

= sharpened

Source: Savarese Slides
Image gradient

• How can we differentiate a digital image $F(x, y)$?
  – Option 1: reconstruct a continuous image, $f$, then take gradient
  – Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx F[x + 1, y] - F[x, y]$$

How would you implement this as a cross-correlation?
Image gradient

\[ \nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}] \]

It points in the direction of most rapid change in intensity

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

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

The gradient direction is given by:

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

• how does this relate to the direction of the edge?

The edge strength is given by the gradient magnitude

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

Source: Seitz and Szeliski Slides
Physical causes of edges

1. Object boundaries
2. Surface normal discontinuities
3. Reflectance (albedo) discontinuities
4. Lighting discontinuities

Source: G Hager Slides
Object Boundaries

Source: G Hager Slides
Surface normal discontinuities
Boundaries of material properties
Boundaries of lighting
Edge Types

Step

Ridge

Roof

Which of these do you suppose a derivative filter detects best?
Some Other Interesting Kernels

The Roberts Operator
\[
\begin{bmatrix}
1 & 0 \\
0 & -1 \\
\end{bmatrix}
\begin{bmatrix}
0 & 1 \\
-1 & 0 \\
\end{bmatrix}
\]

The Prewitt Operator
\[
\begin{bmatrix}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1 \\
\end{bmatrix}
\begin{bmatrix}
0 & 1 & 1 \\
-1 & 0 & 1 \\
-1 & -1 & 0 \\
\end{bmatrix}
\]

Source: G Hager Slides
Some Other Interesting Kernels

The Sobel Operator

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

The Laplacian Operator

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0 \\
\end{bmatrix}
\text{or}
\begin{bmatrix}
1 & 1 & 1 \\
1 & -8 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

A good exercise: derive the Laplacian from 1-D derivative filters.

Note the Laplacian is rotationally symmetric!

Source: G Hager Slides
Edge is Where Change Occurs 1D

- Change is measured by derivative in 1D
  - Biggest change, derivative has maximum magnitude
  - Or 2\textsuperscript{nd} derivative is zero.

Source: G Hager Slides
Noisy Step Edge

- Derivative is high everywhere.
- Must smooth before taking gradient.

Source: G Hager Slides
Smoothing Plus Derivatives

• One problem with differences is that they by definition reduce the signal to noise ratio.

• Recall smoothing operators (the Gaussian!) reduce noise.

• Hence, an obvious way of getting clean images with derivatives is to combine derivative filtering and smoothing: e.g.

\[(F \ast G) \ast D_x = F \ast (G \ast D_x)\]
The Fourier Spectrum of DOG

PS of central slice

Derivative of a Gaussian

Source: G Hager Slides
The DoG: Derivative of a Gaussian
Properties of the DoG operator

- Now, going back to the directional derivative:
  \[ D_u(f(x,y)) = f_x(x,y)u_1 + f_y(x,y)u_2 \]

- Now, including a Gaussian convolution, we see
  \[ D_u[G*I] = D_u[G]*I = [u_1G_x + u_2G_y]*I = u_1G_y*I + u_2G_x*I \]

- The two components \( I*G_x \) and \( I*G_y \) are the image gradient

- Note the directional derivative is maximized in the direction of the gradient

- (note some authors use DoG as “Difference of Gaussian” which we’ll run into soon ....)

Source: G Hager Slides
Algorithm: Simple Edge Detection

1. Compute $I_x = I_g * (G(\sigma) * G(\sigma)' * [1,-1;1,-1])$
2. Compute $I_y = I_g * (G(\sigma) * G(\sigma)' * [1,-1;1,-1]' )$
3. Compute $I_{mag} = sqrt(I_x.* I_x + I_y .* I_y)$
4. Threshold: $I_{res} = I_{mag} > \tau$

It is interesting to note that if we wanted an edge detector for a specific direction of edges, we can simply choose the appropriate projection (weighting) of the component derivatives.

Source: G Hager Slides
Example

\[ \sigma = 1 \quad \sigma = 2 \]

\[ \sigma = 5 \]

Source: G Hager Slides
Limitations of Linear Operators on Impulsive Noise
Nonlinear Filtering: The Median Filter

Suppose I look at the local statistics and replace each pixel with the *median* of its neighbors:
Median Filtering Example

filters have width 5:

<table>
<thead>
<tr>
<th>INPUT</th>
<th>MEDIAN</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Input Diagram" /></td>
<td><img src="image" alt="Median Diagram" /></td>
<td><img src="image" alt="Mean Diagram" /></td>
</tr>
</tbody>
</table>
Median Filtering: Example

Original  Salt and Pepper  Gaussian Filter

Median Filter

Source: G Hager Slides
Non-local Means for Image Denoising

\[ S(i) = \sum_j w(i, j) v(j) \]

**Similarity Between Two Locations**

Typically, the Euclidean distance in a Gaussian kernel.

NL Means Weight Distribution

(a)  (b)  (c)

(d)  (e)  (f)

NL Means Example Result

Noisy Input  Gaussian Filtering  Anisotropic Filtering

Total Variation  Neighborhood Filtering  Non-Local Means

Filter Pyramids

• Recall we can always filter with $G(\sigma)$ for any $\sigma$

• As a result, we can think of a continuum of filtered images as $\sigma$ grows.
  – This is referred to as the “scale space” of the images. We will see this show up several times.

• As a related note, suppose I want to subsample images
  – Subsampling reduces the highest frequencies
  – Averaging reduces noise
  – Pyramids are a way of doing both
Gaussian Pyramid

- Algorithm:
  - 1. Filter with $\mathcal{G}(\sigma = 1)$
  - 2. Resample at every other pixel
  - 3. Repeat

Source: G Hager Slides
Laplacian Pyramid Algorithm

- Create a Gaussian pyramid by successive smoothing with a Gaussian and down sampling

- Set the coarsest layer of the Laplacian pyramid to be the coarsest layer of the Gaussian pyramid

- For each subsequent layer $n+1$, compute

$$L(n + 1) = G(n + 1) = \text{Upsample}(G(n))$$

Source: G Hager Slides
Laplacian of Gaussian Pyramid

Source: G Hager Slides
Laplacian of Gaussian Pyramid

Source: G Hager Slides
Understanding Convolution

• Another way to think about convolution is in terms of how it changes the frequency distribution in the image.

• Recall the Fourier representation of a function

\[ F(u) = \int f(x) e^{-2\pi i u x} \, dx \]

– recall that \( e^{-2\pi i u x} = \cos(2\pi u x) - i \sin(2\pi u x) \)

– Also we have \( f(x) = \int F(u) e^{2\pi i u x} \, du \)

\[ F(u) = |F(u)| e^{i \Phi(u)} \]

• a decomposition into magnitude (\( |F(u)| \)) and phase \( \Phi(u) \)

• If \( F(u) = a + i b \) then

• \( |F(u)| = (a^2 + b^2)^{1/2} \) and \( \Phi(u) = \text{atan2}(a,b) \)

\[ |F(u)|^2 \] is the power spectrum

• Questions: what function takes many many many many terms in the Fourier expansion?

Source: G Hager, D. Kriegman Slides
Understanding Convolution

Discrete Fourier Transform (DFT)

\[ F[u, v] = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I[x, y] e^{-\frac{2\pi i}{N} (xu + yv)} \]

Inverse DFT

\[ I[x, y] = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F[u, v] e^{\frac{2\pi i}{N} (ux + vy)} \]

Implemented via the “Fast Fourier Transform” algorithm (FFT)

Source: G Hager, D. Kriegman Slides
Fourier basis element

$$e^{-i2\pi(ux+vy)}$$

Transform is sum of orthogonal basis functions

Vector \((u,v)\)
- Magnitude gives frequency
- Direction gives orientation.

Source: G Hager, D. Kriegman Slides
Here $u$ and $v$ are larger than in the previous slide.
And larger still...
The Fourier “Hammer”

“Power Spectrum”

Linear Combination:

Basis vectors
Frequency Decomposition

All Basis Vectors

Example

intensity ~ that frequency’s coefficient

Source: G Hager, D. Kriegman Slides
Using Fourier Representations

Data Reduction: only use *some* of the existing frequencies

Source: G Hager, D. Kriegman Slides
Using Fourier Representations

Dominant Orientation

Limitations: not useful for local segmentation

Source: G Hager, D. Kriegman Slides
Phase and Magnitude

\[ e^{it} = \cos t + i \sin t \]

- Fourier transform of a real function is complex with real (R) and imaginary (I) components
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
  - \( p(u) = \text{atan}(I(u)/R(u)) \)
- Magnitude is the magnitude of the complex transform
  - \( m(u) = \sqrt{R^2(u) + I^2(u)} \)
- Curious fact
  - all natural images have about the same magnitude transform
  - hence, phase seems to matter, but magnitude largely doesn’t
- Demonstration
  - Take two pictures, swap the phase transforms, compute the inverse - what does the result look like?

Source: G Hager, D. Kriegman Slides
This is the magnitude transform of the cheetah pic

Source: G Hager, D. Kriegman Slides
This is the phase transform of the cheetah pic

Source: G Hager, D. Kriegman Slides
This is the magnitude transform of the zebra pic.
This is the phase transform of the zebra pic.

Source: G Hager, D. Kriegman Slides
Reconstruction with zebra phase, cheetah magnitude

Source: G Hager, D. Kriegman Slides
Reconstruction with cheetah phase, zebra magnitude
The Fourier Transform and Convolution

• If $H$ and $G$ are images, and $F(.)$ represents Fourier transform, then

$$F(H*G) = F(H)F(G)$$

• Thus, one way of thinking about the properties of a convolution is by thinking of how it modifies the frequencies of the image to which it is applied.

• In particular, if we look at the power spectrum, then we see that convolving image $H$ by $G$ attenuates frequencies where $G$ has low power, and amplifies those which have high power.

• This is referred to as the **Convolution Theorem**
The Properties of the Box Filter

\[ F(\text{mean filter}) = \]

Thus, the mean filter enhances low frequencies but also has “side lobes” that admit higher frequencies.

Source: G Hager, D. Kriegman Slides
The Gaussian Filter: A Better Noise Reducer

• Ideally, we would like an averaging filter that removes (or at least attenuates) high frequencies beyond a given range.

• It is not hard to show that the FT of a Gaussian is again a Gaussian.
  – What does this imply? \[ \text{FT}(e^{-\alpha x^2}) = \sqrt{\frac{\pi}{\alpha}} \cdot e^{-\frac{(\pi \xi)^2}{\alpha}} \]

• Note that in general, we truncate --- a good general rule is that the width (w) of the filter is at least such that \( w > 5 \sigma \). Alternatively we can just stipulate that the width of the filter determines \( \sigma \) (or vice-versa).

• Note that in the discrete domain, we truncate the Gaussian, thus we are still subject to ringing like the box filter.

Source: G Hager, D. Kriegman Slides
Smoothing by Averaging

Kernel: □

Source: G Hager, D. Kriegman Slides
Smoothing with a Gaussian

Kernel:
Why Not a Frequency Domain Filter?
Gabor Filters

- Fourier decompositions are a way of measuring “texture” properties of an image, but they are global.

- Gabor filters are a “local” way of getting image frequency content.

\[
g(x,y) = s(x,y) w(x,y) = a \sin \text{ and a “weight”}
\]

\[
s(x,y) = \exp(-i (2 \pi (x u + y v)))
\]
\[
w(x,y) = \exp(-1/2 (x^2 + y^2)/ \sigma^2)
\]

Now, we have several choices to make:
1. \(u\) and \(v\) defines frequency and orientation
2. \(\sigma\) defines scale (or locality)

Thus, Gabor filters for texture can be computationally expensive as we often must compute many scales, orientations, and frequencies.

Source: G Hager, D. Kriegman Slides
Filtering for Texture

• The Leung-Malik (LM Filter): set of edge and bar filters plus Gaussian and Laplacian of Gaussian

Source: G Hager, D. Kriegman Slides
Next Lecture: Local Image Features

• Readings: FP 5; SZ 4.2, 4.3