CAP5415
Computer Vision

Yogesh S Rawat
yogesh@ucf.edu
HEC-241
Filtering
Part-II

Lecture 3
Derivative

\[
\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x
\]

\[
y = x^2 + x^4
\]
\[
\frac{dy}{dx} = 2x + 4x^3
\]

\[
y = \sin x + e^{-x}
\]
\[
\frac{dy}{dx} = \cos x + (-1)e^{-x}
\]
Discrete Derivative

\[
\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)
\]

\[
\frac{df}{dx} = \frac{f(x) - f(x - 1)}{1} = f'(x)
\]

\[
\frac{df}{dx} = f(x) - f(x - 1) = f'(x)
\]
Discrete Derivative / Finite Difference

\[
\frac{df}{dx} = f(x) - f(x-1) = f'(x) \quad \text{Backward difference}
\]

\[
\frac{df}{dx} = f(x) - f(x+1) = f'(x) \quad \text{Forward difference}
\]

\[
\frac{df}{dx} = f(x+1) - f(x-1) = f'(x) \quad \text{Central difference}
\]
Example: Finite Difference

\[ f(x) = 10 \ 15 \ 10 \ 10 \ 25 \ 20 \ 20 \ 20 \]
\[ f'(x) = 0 \ 5 \ -5 \ 0 \ 15 \ -5 \ 0 \ 0 \]
\[ f''(x) = 0 \ 5 \ 10 \ 5 \ 15 \ -20 \ 5 \ 0 \]

Derivative Masks

- Backward difference: \([-1 \ 1]\)
- Forward difference: \([1 \ -1]\)
- Central difference: \([-1 \ 0 \ 1]\)
Derivative in 2-D

Given function:
\[ f(x, y) \]

Gradient vector:
\[ \nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix} \]

Gradient magnitude:
\[ |\nabla f(x, y)| = \sqrt{f_x^2 + f_y^2} \]

Gradient direction:
\[ \theta = \tan^{-1} \left( \frac{f_x}{f_y} \right) \]
Derivative of Images

Derivative masks

\[
f_x \Rightarrow \frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \quad \quad \quad f_y \Rightarrow \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}
\]

\[
I = \begin{bmatrix}
10 & 10 & 20 & 20 \\
10 & 10 & 20 & 20 \\
10 & 10 & 20 & 20 \\
10 & 10 & 20 & 20 \\
\end{bmatrix}
\]

\[
I_x = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 10 & 10 & 0 & 0 \\
0 & 10 & 10 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

9/8/2021
Derivative of Images

Derivative masks

\[ f_x \Rightarrow \frac{1}{3} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \]

\[ f_y \Rightarrow \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \]

\[
I = \begin{bmatrix}
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
10 & 10 & 20 & 20 & 20 \\
\end{bmatrix}
\]

\[
I_y = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]
Averages

• Mean

\[ I = \frac{I_1 + I_2 + \ldots + I_n}{n} = \frac{\sum_{i=1}^{n} I_i}{n} \]

• Weighted mean

\[ I = \frac{w_1 I_1 + w_2 I_2 + \ldots + w_n I_n}{n} = \frac{\sum_{i=1}^{n} w_i I_i}{n} \]
Example

a. Original image
b. Laplacian operator
c. Horizontal derivative
d. Vertical derivative
Correlation (linear relationship)

\[ f \otimes h = \sum_{k} \sum_{l} f(k, l)h(k, l) \]

\[ f = \text{Image} \]
\[ h = \text{Kernel} \]

\[ f \]
\[ \begin{array}{ccc}
  f_1 & f_2 & f_3 \\
  f_4 & f_5 & f_6 \\
  f_7 & f_8 & f_9
\end{array} \]

\[ h \]
\[ \begin{array}{ccc}
  h_1 & h_2 & h_3 \\
  h_4 & h_5 & h_6 \\
  h_7 & h_8 & h_9
\end{array} \]

\[ f \otimes h = f_1 h_1 + f_2 h_2 + f_3 h_3 + f_4 h_4 + f_5 h_5 + f_6 h_6 + f_7 h_7 + f_8 h_8 + f_9 h_9 \]
Convolution

\[ f * h = \sum_{k} \sum_{l} f(k, l)h(-k, -l) \]

\[ f = \text{Image} \]

\[ h = \text{Kernel} \]

\[ X - \text{flip} \]

\[ Y - \text{flip} \]

\[ f * h = f_1 h_9 + f_2 h_8 + f_3 h_7 \]

\[ + f_4 h_6 + f_5 h_5 + f_6 h_4 \]

\[ + f_7 h_3 + f_8 h_2 + f_9 h_1 \]
Convolution

• Convolution is associative

\[ F \ast (G \ast I) = (F \ast G) \ast I \]
Correlation and Convolution

• **Convolution** is a filtering operation
  • expresses the amount of overlap of one function as it is shifted over another function

• **Correlation** compares the similarity of two sets of data
  • relatedness of the signals!
Gaussian filter
Gaussian filter

\[ g(x) = e^{-\frac{x^2}{2\sigma^2}} \]
Gaussian filter

\[ g(x) = e^{-\frac{x^2}{2\sigma^2}} \]
Gaussian filter

\[ g(x) = e^{-x^2} \]
Gaussian filter

\[ g(x) = e^{\frac{-x^2}{2\sigma^2}} \]

\[ g(x, y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}} \]
Gaussian filter

\[ g(x) = e^{-\frac{x^2}{2\sigma^2}} \]

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

\[ g(x) = [0.011, 0.13, 0.6, 1, 0.6, 0.13, 0.011] \]
Gaussian filter - properties

- Most common natural model
Gaussian filter - properties

• Most common natural model
• Smooth function, it has infinite number of derivatives
• It is Symmetric
• Fourier Transform of Gaussian is Gaussian.
• Convolution of a Gaussian with itself is a Gaussian.
• Gaussian is separable; 2D convolution can be performed by two 1-D convolutions
• There are cells in eye that perform Gaussian filtering.
Filtering Examples - 1

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

* =

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Filtering Examples - 2

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

= 

\[ \begin{array}{ccc}
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
\end{array} \]
Filtering Examples - 3

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

* \frac{1}{9} 

= 

\[
\begin{array}{ccc}
\end{array}
\]
Filtering Examples - 4

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
\frac{1}{25}
\]

\[
= 
\]

9/8/2021
Filtering Examples - 5

Gaussian Smoothing
Filtering Examples - 6

Gaussian Smoothing                      Smoothing by Averaging
Filtering Examples - 7

After additive Gaussian Noise

After Averaging

After Gaussian Smoothing
Example: box filter

What does it do?
- Replaces each pixel with an average of its neighborhood
Image filtering

\[ f[\ldots] \quad g[\cdot, \cdot] \quad h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l] \]
Image filtering

\[ f[\ldots] \]

\[ g[\cdot, \cdot] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l] \]
Image filtering

\[ f[\ldots] \]

\[ g[\cdot, \cdot] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]
Image filtering

\[ g[\cdot, \cdot] \]

\[ f[\ldots] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] \cdot f[m+k,n+l] \]
Image filtering

\[ g[\cdot, \cdot] \]

\[ f[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

\[ h[m, n] = \sum_{k, l} g[k, l] f[m + k, n + l] \]
Image filtering

\[ g[\cdot, \cdot] \]

\[ f[\cdot, \cdot] \]

\[ h[\cdot, \cdot] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l] \]
Image filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]

Credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]
Example: box filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)
Smoothing with Box filter
Practice with kernels

Original
Practice with kernels

Original

Filtered (no change)
Practice with kernels

Original
Practice with kernels

Original

Shifted left
By 1 pixel
Practice with kernels

Original

(Note that filter sums to 1)
Practice with kernels

Original

Sharpening filter
- Accentuates differences with local average
Sharpening Filter

before

after
Sobel Filtering

Sobel

Vertical Edge (absolute value)
Sobel Filtering

Sobel

Horizontal Edge (absolute value)
Key properties of linear filters

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

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

Any linear, shift-invariant operator can be represented as a **convolution**
More properties

• Commutative: $a * b = b * a$
  – Conceptually no difference between filter and signal
  – particular filtering implementations might break this equality

• Associative: $a * (b * c) = (a * b) * c$
  – Often apply several filters one after another: $(((a * b_1) * b_2) * b_3)$
  – This is equivalent to applying one filter: $a * (b_1 * b_2 * b_3)$

• Distributes over addition: $a * (b + c) = (a * b) + (a * c)$

• Scalars factor out: $ka * b = a * kb = k (a * b)$

• Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a * e = a$
Median Filter

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

• Advantage?

• Is it same as convolution?
Image filtering - mean

\[ f[\ldots] \]

\[ g[\cdot, \cdot] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k,n+l] \]
Image filtering - mean

\[ h[m,n] = \sum_{k,l} g[k,l] f[m+k, n+l] \]
Image filtering - median

\[ f[\ldots] \]

\[ h[\ldots] \]

\[
h[m,n] = \sum_{k,l} g[k,l] \cdot f[m+k,n+l]\]
Image filtering - median

\[ f[\ldots] \]

\[ h[\ldots] \]

\[ h[m,n] = \sum_{k,l} g[k,l] \ f[m+k,n+l] \]
Median Filter

3x3

4x4

7x7

Mean
Gaussian
Median
How big should the filter be?

- Values at edges should be near zero
- Gaussians have infinite extent...
- Rule of thumb for Gaussian: set filter half-width to about $3 \sigma$
Practical matters

What about near the edge?

• The filter window falls off the edge of the image
• Need to extrapolate
• methods:
  • clip filter (black)
  • wrap around
  • copy edge
  • reflect across edge

Source: S. Marschner
Questions?

Sources for this lecture include materials from works by Mubarak Shah, S. Seitz, James Tompkin and Ulas Bagci