Image Filtering

Computer Vision I
CSE 252A
Lecture 5
Announcements

• Assignment 1 is due Oct 20, 11:59 PM

• Reading:
  – Szeliski
    • Sections 3.2, 3.3.1, and 3.4
Image Filtering Example

Input

Output

Filter
What is image filtering?

Producing a new image where the value at a pixel in the output image is a function of a neighborhood of the pixel location in the input image.
Example: Smoothing by Averaging
Image Filtering

- Most common filters are linear filters and the process of applying a linear filter is called convolution
- Why filter
  - Enhance images
    - Denoise, resize, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching
  - The “convolution” in Convolutional Neural Networks
Linear Filters

• General process:
  – Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.

• Properties
  – Output is a linear function of the input
  – Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)

• Example: smoothing by averaging
  – form the average of pixels in a neighborhood

• Example: smoothing with a Gaussian
  – form a weighted average of pixels in a neighborhood

• Example: finding a derivative
  – form a difference of pixels in a neighborhood
Convolution

Image (I)

Kernel (K)

Note: Typically Kernel is relatively small in vision applications.
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i - h, j - k)
\]

\( m = 2 \)
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i - h, j - k)$$
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k)I(i-h, j-k)$$
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i - h, j - k)
\]
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i - h, j - k)$$
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} K(h, k) I(i-h, j-k)
\]
Convolution: $R = K \ast I$

Kernel size is $m+1$ by $m+1$

$$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k) I(i-h, j-k)$$
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{m}{2}}^{\frac{m}{2}} K(h, k) I(i-h, j-k)
\]
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h, k)I(i-h, j-k)
\]
Linear filtering (warm-up slide)
Linear filtering (warm-up slide)

original

Pixel offset

Filtered (no change)
Linear filtering

original

Pixel offset

0 0 0
0 0 1
0 0 0

?
Shift

original

Pixel offset

0 0 0
0 0 1
0 0 0

Shifted one Pixel to the left
Linear filtering
Blurring

original

Blurred (filter applied in both dimensions).
Blur Examples
1-Dimensional

impulse

original

coefficient

Pixel offset

filtered

edge

original

coefficient

Pixel offset

filtered
Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
- \frac{1}{9}
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Source: D. Lowe
Practice with linear filters

Sharpening filter
- Accentuates differences with local average

Original

Source: D. Lowe

Computer Vision I
Sharpening

before

after
Sharpening example

original * coefficient = Sharpened

(differences are accentuated; constant areas are left untouched).
Convolution and Correlation

• 2d convolution
  – Kernel is flipped over both axes
  $$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k)I(i-h, j-k)$$
  $$= \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(-h,-k)I(i+h, j+k)$$

• 2d correlation
  – Kernel is not flipped
  $$R(i, j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k)I(i+h, j+k)$$

• When kernel is symmetric, convolution and correlation give the same result
Filters are templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products

Insight
- Filters look like the effects they are intended to find
- Filters find effects they look like
Key properties of linear filters

Let $\text{imfilter}(I,k)$ be a function that applies kernel $k$ to image $I$

Linearity:
$$\text{imfilter}(I, k_1 + k_2) = \text{imfilter}(I,k_1) + \text{imfilter}(I,k_2)$$

Shift invariance:
Same behavior regardless of pixel location
$$\text{imfilter}(I, \text{shift}(k)) = \text{shift}(\text{imfilter}(I,k))$$

Any linear, shift-invariant operator can be represented as a convolution.

Source: S. Lazebnik
Convolution properties

Commutative: $k * I = I * k$

- Conceptually no difference between filter and image

Associative: $a * (b * c) = (a * b) * c$

- Often apply several filters one after another:
  $(((I * k_1) * k_2) * k_3)$
- This is equivalent to applying one filter:
  $I * (k_1 * k_2 * k_3)$

Correlation is NOT associative

Source: S. Lazebnik
Convolution properties (cont.)

• Distributes over addition:
  \[ I \ast (k_1 + k_2) = (I \ast k_1) + (I \ast k_2) \]

• Scalars factor out:
  – For scalar s
    \[ s (f \ast I) = (sf) \ast I = f \ast (sI) \]

• Identity:
  unit impulse e = [0, 0, 1, 0, 0]
  \[ I \ast e = I \]

Source: S. Lazebnik
Properties of Continuous Convolution

Let $f, g, h$ be images and $*$ denote convolution

$$f * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x-u, y-v)g(u, v)\,dudv$$

- Commutative: $f*g=g*f$
- Associative: $f*(g*h) = (f*g)*h$
- Linear: for scalars $a$ & $b$ and images $f, g, h$
  $$(af + bg)*h = a(f*h) + b(g*h)$$
- Differentiation rule
  $$\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g = f * \frac{\partial g}{\partial x}$$
Convolutional Neural Networks

• Core operation in CNN is, not surprisingly, convolution.

• During training of a CNN, the weights of the convolution kernels are learned.

• Can be extended to 3D – e.g.,
  – Image and R,G,B as channels (N x N x 3)
  – Volumetric data such as MRI, CT
Filtering to reduce noise

• Noise is what we are not interested in
  – Usually think of simple, low-level noise:
    • Light fluctuations; Sensor noise; Quantization effects; Finite precision
  – Complex noise: shadows; extraneous objects

• A pixel’s neighborhood contains information about its color and intensity

• Averaging noise reduces its effect
Additive noise

- $I = S + N$. Noise doesn’t depend on signal.
- We’ll consider:

$$I_i = s_i + n_i \text{ with } E(n_i) = 0$$

$s_i$ deterministic. $n_i$ a random var.

$n_i, n_j$ independent for $i \neq j$

$n_i, n_j$ identically distributed

- Gaussian noise, $n_i$ drawn from Gaussian.
Gaussian noise

Image is constant with $I_i = 0.5$

Gaussian noise, $n_i$ drawn from Gaussian distribution with zero mean and standard deviation $\sigma$
Gaussian Noise: 
\[ \sigma = 1 \]

Gaussian Noise: 
\[ \sigma = 16 \]
Averaging Filter

- Mask with positive entries, that sum 1
- Replaces each pixel with an average of its neighborhood
- If all weights are equal, it is called a Box filter

(Camps)
Smoothing by Averaging

Kernel: □
An Isotropic Gaussian Kernel

- Circularly symmetric Gaussian with variance $\sigma^2$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$
Smoothing with a Gaussian

Kernel:
Smoothing

Box filter

Gaussian filter
The effects of smoothing

Increased Noise $\rightarrow$

An image with constant intensity + noise:

Each row shows smoothing with Gaussians of different width; each column shows different realizations of an image of Gaussian noise.
Smoothing with Gaussian kernel

Volume under surface greater than $3\sigma$ is negligible
Smoothing with Gaussian kernel

$\sigma = 7$
43x43

$\sigma = 7$
85x85

Difference
Smoothing with Gaussian kernel

Input image

\( \sigma = 3.5 \)
21x21

\( \sigma = 7 \)
43x43
Border padding

Zero padding
when $v = 0$

Constant padding

Replicate padding

Mirror padding
Border padding

Zero padding

Mirror padding

Replicate padding
Gaussian Smoothing

original

$\sigma = 2$

$\sigma = 2.8$

$\sigma = 4$
Gaussian Smoothing

by Charles Allen Gillbert

by Harmon & Julesz

http://www.michaelbach.de/ot/cog_blureffects/index.html
Gaussian Smoothing

http://www.michaelbach.de/ot/cog_blureffects/index.html
Efficient Implementation

• Both, the Box filter and the Gaussian filter are separable:
  – First convolve each row with a 1D horizontal kernel
  – Then convolve each column with a 1D vertical kernel
Overview: Image processing in the frequency domain

Image in spatial domain $f(x,y)$

$\xrightarrow{\text{Fourier transform}}$

Image in frequency domain $F(u,v)$

Frequency domain processing

$\xleftarrow{\text{Frequency domain}}$

Image in frequency domain $G(u,v)$

Image in spatial domain $g(x,y)$
Fourier Transform

- 1-D transform (signal processing)
- 2-D transform (image processing)
- Consider 1-D
  
  Time domain $\leftrightarrow$ Frequency Domain
  
  Real $\leftrightarrow$ Complex

- Consider time domain signal to be expressed as weighted sum of sinusoid. A sinusoid $\cos(ut+\phi)$ is characterized by its phase $\phi$ and its frequency $u$

- The Fourier transform of the signal is a function giving the weights (and phase) as a function of frequency $u$. 
Fourier Transform

• 1D example
  – Sawtooth wave
    • Combination of harmonics
Fourier Transform

Discrete Fourier Transform (DFT) of $I[x,y]$

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

Inverse DFT

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

$x,y$: spatial domain
$u,v$: frequency domain

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

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

Transform is sum of orthogonal basis functions

Vector \((u,v)\)
- Magnitude gives frequency
- Direction gives orientation.
Here $u$ and $v$ are larger than in the previous slide.
And larger still...
Using Fourier Representations

Dominant Orientation

Limitations: not useful for local segmentation
Phase and Magnitude

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

- Fourier transform of a real function is complex
  - difficult to plot, visualize
  - instead, we can think of the phase and magnitude of the transform
- Phase is the phase of the complex transform
- Magnitude is the magnitude of the complex transform
- 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?
This is the magnitude transform of the cheetah pic
This is the phase transform of the cheetah pic.
This is the magnitude transform of the zebra pic
This is the phase transform of the zebra pic
Reconstruction with zebra phase, cheetah magnitude
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
Various Fourier Transform Pairs

• Important facts
  
  – scale function down $\Leftrightarrow$ scale transform up
    i.e. high frequency = small details
  
  – The Fourier transform of a Gaussian is a Gaussian.

compare to box function transform
Other Types of Noise

• Impulsive noise
  – randomly pick a pixel and randomly set to a value
  – saturated version is called salt and pepper noise

• Quantization effects
  – Often called noise although it is not statistical

• Unanticipated image structures
  – Also often called noise although it is a real repeatable signal
Some other useful filtering techniques

- Median filter
- Anisotropic diffusion
Median filters : principle

Method :

1. rank-order neighborhood intensities
2. take middle value

• non-linear filter
• no new gray levels emerge...
Median filters: Example for window size of 3

1,1,1,7,1,1,1,1

↓

?,1,1,1,1,1,1,?

The advantage of this type of filter is that it eliminates spikes (salt & pepper noise).
Median filters: 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 Signal" /></td>
<td><img src="image" alt="Median Filter Output" /></td>
<td><img src="image" alt="Mean Filter Output" /></td>
</tr>
</tbody>
</table>
Median filters: analysis

Median completely discards the spike, linear filter always responds to all aspects

Median filter preserves discontinuities, linear filter produces rounding-off effects

Median filters can destroy detail
Median filter : images

3 x 3 median filter : 

sharpens edges, destroys edge cusps and protrusions
Median filters: Gauss revisited

Comparison with Gaussian:

e.g. upper lip smoother, eye better preserved
Example of median

10 times 3 X 3 median

patchy effect
important details lost (e.g. earring)
Next Lecture

- Edge detection and corner detection
- Reading:
  - Szeliski
    - Sections 7.2 and 7.1.1