Filtering

Computer Vision I
CSE252A
Lecture 11

What is image filtering?

• Modify the pixels in an image based on some function of a local neighborhood of the pixels.

<table>
<thead>
<tr>
<th>Local image data</th>
<th>Modified image data =</th>
</tr>
</thead>
</table>

(From Bill Freeman)

Noise

• Simplest noise model
  - independent stationary additive Gaussian noise
  - the noise value at each pixel is given by an independent draw from the same normal probability distribution

• Issues
  - this model allows noise values that could be greater than maximum camera output or less than zero
  - for small standard deviations, this isn’t too much of a problem - it’s a fairly good model
  - independence may not be justified (e.g. damage to lens)
  - may not be stationary (e.g. thermal gradients in the ccd)

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 neighbourhood

• Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood

• Example: finding a derivative
  - form a weighted average of pixels in a neighbourhood

Linear functions

• Simplest: linear filtering.
  - Replace each pixel by a linear combination of its neighbors.

• The prescription for the linear combination is called the "convolution kernel".

<table>
<thead>
<tr>
<th>Local image data</th>
<th>Kernel</th>
<th>Modified image data =</th>
</tr>
</thead>
</table>

(Freeman)
Convolution

Image (I) * Kernel (K)

Note: Typically Kernel is relatively small in vision applications.

Convolution: \[ R(i, j) = \sum_{h=\text{m}\pm1}^{\text{m}} \sum_{k=\text{m}\pm1}^{\text{m}} K(h, k) I(i-h, j-k) \]

Kernel size is \( m+1 \) by \( m+1 \)
Convolution: \( R = K \ast I \)

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

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

original

Blurred (filter applied in both dimensions).

Blur examples

impulse

coefficient

Pixel offset

filtered

Linear filtering (warm-up slide)

original

??

Linear filtering (no change)

original

Filtered (no change)

Linear filtering

original

??
Smoothing by Averaging

Kernel:

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

Properties of convolution
Let \( f, g, h \) be images and \( * \) denote convolution

\[
(f \ast g)(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x-u)g(y-v) \, du \, dv
\]

- Commutative: \( f \ast g = g \ast f \)
- Associative: \( f \ast (g \ast h) = (f \ast g) \ast h \)
- Linear: for scalars \( a \) & \( b \) and images \( f, g, h \)
  \( (af + bg) \ast h = a(f \ast h) + b(g \ast h) \)
- Differentiation rule
  \[
  \frac{\partial}{\partial x}(f \ast g) = \frac{\partial f}{\partial x} \ast g = f \ast \frac{\partial g}{\partial x}
  \]
Filtering to reduce noise

- Noise is what we’re not interested in.
  - We’ll discuss simple, low-level noise today:
    Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous objects.
- A pixel’s neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

Additive noise

- I = S + N. Noise doesn’t depend on signal.
- We’ll consider:
  \[ I_j = s_j + n_j \]
  \[ s_j \text{ deterministic. } n_j \text{ a random var.} \]
  \[ n_i, n_j \text{ independent for } i \neq j \]
  \[ n_i, n_j \text{ identically distributed} \]

Average 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.

\[
F = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]

\[
E(\tilde{N}(i,j)) = 0
\]
\[
E(\tilde{N}^2(i,j)) = \frac{1}{m^2} \cdot m \cdot \sigma^2 = \frac{\sigma^2}{m} \Rightarrow \tilde{N}(i,j) \sim N(0, \frac{\sigma^2}{m})
\]

(Camps)
Smoothing with a Gaussian

- Notice “ringing”
  - apparently, a grid is superimposed
- Smoothing with an average actually doesn’t compare at all well with a defocussed lens
  - what does a point of light produce?
- A Gaussian gives a good model of a fuzzy blob

An Isotropic Gaussian

- The picture shows a smoothing kernel proportional to
  \[ \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

  (which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing by Averaging

Kernel:

Efficient Implementation

- Both, the BOX filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.
Fourier Transform

- 1-D transform (signal processing)
- 2-D transform (image processing)
- Consider 1-D
  
  - Time domain ↔ Frequency Domain
  - Real ↔ 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 \( u \).

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

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

Inverse DFT

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

\( x,y \): spatial domain
\( u,v \): frequency domain

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

Fourier basis element

- \( e^{-\frac{2\pi i}{N} (ux + vy)} \)
- 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 magnitude transform of the zebra pic

This is the phase transform of the cheetah pic

This is the phase transform of the zebra pic
The Fourier Transform and Convolution

- If \( H \) and \( G \) are images, and \( F(.) \) represents Fourier transform, then
  \[
  F(H \ast G) = F(H)F(G)
  \]
  
- 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 FT of a Gaussian is a Gaussian.
    
    compare to box function transform

Other Types of Noise

- Impulsive noise
  - randomly pick a pixel and randomly set ot 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 neighbourhood intensities
2. take middle value

- non-linear filter
- no new grey levels emerge...

Median filters: Example for window size of 3

1,1,1,7,1,1,1,1
↓
?,1,1,1.1,1,1,?

advantage of this type of filter is that it Eliminates spikes (salt & pepper noise).

Median filters: Example

filters have width 5:

Input

Median

Mean

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

DON'T become all too optimistic

Median filters: 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. ear-ring)