Image Filtering in Spatial domain

Computer Vision
Jia-Bin Huang, Virginia Tech
Administrative stuffs

• Lecture schedule changes

• Office hours - Jia-Bin (440 Whittemore Hall)
  • Friday at 11:00 AM – 12:00 PM

• Office hours - Yuliang (Whittemore Hall)
  • Wed at 2:00 – 3:00 PM

• MATLAB tutorial session by Yuliang
  • Bring your laptop

• Computational resources (CPU/GPU/MATLAB)
  • Added all students to ARC. Allocation: cv_fall_2017
Previous class: Light and Color

- Reflection models
  - diffuse/specular reflectance, albedo
  - Surface orientation and light intensity

- Color vision
  - physics of light, trichromacy, color consistency, color spaces (RGB, HSV, Lab)

- Object cast light and shadows to each other

- Interpret images from local differences
Reflection models

- Albedo: fraction of light that is reflected
  - Determines color (amount reflected at each wavelength)

Slide credit: Derek Hoiem
Reflection models

• Specular reflection: mirror-like
  • Light reflects at incident angle
  • Reflection color = incoming light color
Reflection models

• Diffuse reflection
  • Light scatters in all directions (proportional to cosine with surface normal)
  • Observed intensity is independent of viewing direction
  • Reflection color depends on light color and albedo
Surface orientation and light intensity

- Amount of light that hits surface from distant point source depends on angle between surface normal and source

\[ I(x) = \rho(x)(S \cdot N(x)) \]

prop to cosine of relative angle

Slide credit: Derek Hoiem
Application: Photometric Stereo

• Assume:
  • A Lambertian object
  • A *local shading model* (each point on a surface receives light only from sources visible at that point)
  • A set of *known* light source directions
  • A set of pictures of an object, obtained in exactly the same camera/object configuration but using different sources
  • Orthographic projection

• Goal: reconstruct object shape and albedo

Slide credit: S. Lazebnik

F&P 2nd ed., sec. 2.2.4
Example

Input

Recovered Albedo

Recovered normal field

 Recovered surface model

Slide credit: S. Lazebnik
Photometric Stereo

Can write this as a matrix equation:

\[
\begin{bmatrix}
I_1 \\
I_2 \\
I_3
\end{bmatrix}
= k_d \begin{bmatrix}
L_1^T \\
L_2^T \\
L_3^T
\end{bmatrix} N
\]
Solving the equations

\[
\begin{bmatrix}
I_1 \\
I_2 \\
I_3 \\
\end{bmatrix} = \begin{bmatrix}
L_1^T \\
L_2^T \\
L_3^T \\
\end{bmatrix} k_d N
\]

Intensity (Known)  Light direction (Known)  Albedo x Surface normal (Unknown)

\[
G = L^{-1}I
\]

How do we get the albedo and surface normal from G?

\[
k_d = \| G \| \quad N = \frac{1}{k_d} G
\]

Slide credit: N. Snavely
More than three lights

• Get better results by using more lights

\[
\begin{bmatrix}
I_1 \\
\vdots \\
I_n
\end{bmatrix}
= 
\begin{bmatrix}
L_1 \\
\vdots \\
L_n
\end{bmatrix}
k_dN
\]

Least squares solution:

\[
I = LG
\]

\[
L^TI = L^T L G
\]

\[
G = (L^T L)^{-1}(L^TI)
\]

In MATLAB

\[
G = L \backslash I;
\]

Solve for N, k_d as before

What’s the size of \( L^T L \)?
Next three classes: three views of filtering

• Image filters in spatial domain
  • Filter is a mathematical operation on values of each patch
  • Smoothing, sharpening, measuring texture

• Image filters in the frequency domain
  • Filtering is a way to modify the frequencies of images
  • Denoising, sampling, image compression

• Templates and Image Pyramids
  • Filtering is a way to match a template to the image
  • Detection, coarse-to-fine registration

Slide credit: Derek Hoiem
Today’s class

• Pixels and image filtering

• Application: Representing textures

• Application: Denoising and non-linear image filtering

• Goals:
  • Understanding image filter operations and the practical considerations
  • Understand how to use filtering for texture description and denoising
Why should we care?
Why should we care?

Image Pyramid

Image interpolation/resampling

Source: D Forsyth

Source: N Snavely
Why should we care?

Representing textures with filter banks

LM filter bank. Code [here](#)
The raster image (pixel matrix)
Image filtering

- Image filtering: for each pixel, compute function of local neighborhood and output a new value
  - Same function applied at each position
  - Output and input image are typically the same size

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

Local image data

Some function

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
</tr>
</tbody>
</table>

Modified image data

Slide Credit: L. Zhang
Image filtering

• Linear filtering
  • function is a weighted sum/difference of pixel values

• Really important!
  • Enhance images
    • Denoise, smooth, increase contrast, etc.
  • Extract information from images
    • Texture, edges, distinctive points, etc.
  • Detect patterns
    • Template matching
Question: Noise reduction

• Given a camera and a still scene, how can you reduce noise?

Take lots of images and average them!
What’s the next best thing?

Source: S. Seitz
First attempt at a solution

• Let’s replace each pixel with an average of all the values in its neighborhood

• Assumptions:
  • Expect pixels to be like their neighbors
  • Expect noise processes to be independent from pixel to pixel
Example: box filter

\[ \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \]
Image filtering

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

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

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

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

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

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

Slide credit: S. Seitz
Image filtering

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

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

Slide credit: S. Seitz
Image filtering

\[ f[.,.] \]

\[ g[\cdot, \cdot] = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \]

\[ h[.,.] \]

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

Slide credit: S. Seitz
Image filtering

\[ f[\ldots] \]

\[ g[\cdot,\cdot] = \frac{1}{9} \]

\[ h[\ldots] \]

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

Slide credit: S. Seitz
Image filtering

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

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

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

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

\[ f[.,.] \]

\[ h[.,.] \]

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

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

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

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

What does it do?

• Replaces each pixel with an average of its neighborhood

• Achieve smoothing effect (remove sharp features)

\[
g[\cdot, \cdot] = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]

Slide credit: David Lowe
Smoothing with box filter
Properties of smoothing filters

- **Smoothing**
  - Values positive
  - Sum to 1 $\Rightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
Correlation filtering

Say the averaging window size is $2k+1 \times 2k+1$:

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

Loop over all pixels in neighborhood around image pixel $F[i,j]$.

Attribute uniform weight to each pixel.

Now generalize to allow different weights depending on neighboring pixel’s relative position:

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

Non-uniform weights.
Correlation filtering

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

This is called cross-correlation, denoted \( G = H \otimes F \)

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter “kernel” or “mask” \( H[u,v] \) is the prescription for the weights in the linear combination.

Slide credit: Kristen Grauman
Filtering an impulse signal

What is the result of filtering the impulse signal (image) $F$ with the arbitrary kernel $H$?

$F[x, y] \ast H[u, v] = G[x, y]$
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

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

\[
G = H * F
\]

Notation for convolution operator

Slide credit: Kristen Grauman
Convolution vs. correlation

**Convolution**

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

\[ G = H \ast F \]

**Cross-correlation**

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

\[ G = H \otimes F \]

G=\text{conv2}(H,F); \quad \text{or} \quad G=\text{filter2}(H,F); \quad \text{or} \quad G=\text{imfilter}(F,H);\]

For a Gaussian or box filter, how will the outputs differ?
If the input is an impulse signal, how will the outputs differ?
Practice with linear filters

Original

Slide credit: David Lowe (UBC)
Practice with linear filters

Original

Filtered (no change)

Slide credit: David Lowe (UBC)
Practice with linear filters

Original

![Original Image](image.png)

Slide credit: David Lowe (UBC)
Practice with linear filters

Original

Shifted left
By 1 pixel
Practice with linear filters

Original

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

(Note that filter sums to 1)

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

?
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Slide credit: David Lowe (UBC)
Sharpening

before

after

Slide credit: David Lowe (UBC)
Other filters

Sobel

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

Vertical Edge (absolute value)
Other filters

Sobel

Horizontal Edge (absolute value)
Basic gradient filters

Horizontal Gradient

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Vertical Gradient

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

or

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Questions

Write as filtering operations, plus some pointwise operations: +, -, .*, >

1. Sum of four adjacent neighbors plus 1
   \[ out(m,n) = 1 + \sum_{k,l \in \{-1,1\}} in(m+k,n+l) \]

2. Sum of squared values of 3x3 windows around each pixel:
   \[ out(m,n) = \sum_{k,l \in \{-1,0,1\}} in(m+k,n+l)^2 \]

3. Center pixel value is larger than the average of the pixel values to the left and right:
   \[ out(m,n) = 1 \quad \text{if} \quad in(m,n) > (in(m,n-1) + in(m,n+1))/2 \]
   \[ out(m,n) = 0 \quad \text{if} \quad in(m,n) \leq (in(m,n-1) + in(m,n+1))/2 \]
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

Source: S. Lazebnik
More properties

• Commutative: $a \ast b = b \ast a$
  – Conceptually no difference between filter and signal

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

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

• Scalars factor out: $k a \ast b = a \ast k b = k (a \ast b)$

• Identity: unit impulse $e = [0, 0, 1, 0, 0]$, $a \ast e = a$
Important filter: Gaussian

• Spatially-weighted average

\[ G_\sigma = \frac{1}{2\pi \sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]
Smoothing with Gaussian filter
Smoothing with box filter
Gaussian filters

• Remove “high-frequency” components from the image (low-pass filter)
  – Images become more smooth

• Convolution with self is another Gaussian
  • So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have
  • Convolving two times with Gaussian kernel of width $\sigma$ is same as convolving once with kernel of width $\sigma\sqrt{2}$

• Separable kernel
  • Factors into product of two 1D Gaussians

Slide credit: Kristen Grauman
Gaussian filters

• What parameters matter here?
  • **Size** of kernel or mask
    • Note, Gaussian function has infinite support, but discrete filters use finite kernels

Slide credit: Kristen Grauman

\[ \sigma = 5 \text{ with } 10 \times 10 \text{ kernel} \]

\[ \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]
Gaussian filters

• What parameters matter here?
• Variance of Gaussian: determines extent of smoothing

\[ \sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \]

\[ \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]
Separability of the Gaussian filter

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

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

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

Source: D. Lowe
Separability example

2D filtering (center location only)

The filter factors into a product of 1D filters:

Perform filtering along rows:

Followed by filtering along the remaining column:

Source: K. Grauman
Separability

• Why is separability useful in practice?
• Separability means that a 2D convolution can be reduced to two 1D convolutions (one among rows and one among columns)
• What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?
  • $O(n^2 m^2)$
• What if the kernel is separable?
  • $O(n^2 m)$
Some practical matters
Practical matters

How big should the filter be?

- Values at edges should be near zero $\leftarrow$ important!
- 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
Practical matters

• methods (MATLAB):
  • clip filter (black): \texttt{imfilter(f, g, 0)}
  • wrap around: \texttt{imfilter(f, g, 'circular')}
  • copy edge: \texttt{imfilter(f, g, 'replicate')}
  • reflect across edge: \texttt{imfilter(f, g, 'symmetric')}

Source: S. Marschner
Practical matters

• What is the size of the output?
• MATLAB: `filter2(g, f, shape)`
  - `shape = 'full'`: output size is sum of sizes of f and g
  - `shape = 'same'`: output size is same as f
  - `shape = 'valid'`: output size is difference of sizes of f and g

Source: S. Lazebnik
2-mins break
Application: Representing Texture

Source: Forsyth
Texture and Material

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
Texture and Orientation

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/
Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data(texture_database/samples/)
What is texture?

Regular or stochastic patterns caused by bumps, grooves, and/or markings
How can we represent texture?

• Compute responses of blobs and edges at various orientations and scales
Overcomplete representation: filter banks

Code for filter banks: www.robots.ox.ac.uk/~vgg/research/texclass/filters.html
Filter banks

• Process image with each filter and keep responses (or squared/abs responses)
How can we represent texture?

• Measure responses of blobs and edges at various orientations and scales

• Idea 1: Record simple statistics (e.g., mean, std.) of absolute filter responses
Can you match the texture to the response?

Filters

1

2

3

Mean abs responses

A

B

C
Representing texture by mean abs response
Representing texture

- Idea 2: take vectors of filter responses at each pixel and cluster them, then take histograms (more on this in coming weeks)
Denoising and Nonlinear Image Filtering

- **Salt and pepper noise**: contains random occurrences of black and white pixels

- **Impulse noise**: contains random occurrences of white pixels

- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz
Gaussian noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise

Source: M. Hebert
Reducing Gaussian noise

Smoothing with larger standard deviations suppresses noise, but also blurs the image.
Reducing salt-and-pepper noise

- What’s wrong with the results?
Alternative idea: Median filtering

• A **median filter** operates over a window by selecting the median intensity in the window

![Diagram showing a 3x3 window with values 10 15 20, 23 90 27, 33 31 30. The 10 15 20 row is selected, sorted to 10 15 20, and then replaced outside the window.]

• Is median filtering linear?

Source: K. Grauman
Median filter

- Is median filtering linear?
- Let’s try filtering

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 2 \\
2 & 2 & 2 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 \\
+ & 0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Median filter

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

Salt-and-pepper noise  Median filtered

• MATLAB: medfilt2(image, [h w])

Source: M. Hebert
Gaussian vs. median filtering

Gaussian

Median
Other non-linear filters

• Weighted median (pixels further from center count less)

• Clipped mean (average, ignoring few brightest and darkest pixels)

• Bilateral filtering (weight by spatial distance and intensity difference)
Bilateral Filters

- Edge preserving: weights similar pixels more

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

with

\[ W_p^b = \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(\|I_p - I_q\|) \]

Guided Image Filters

Bilateral filters

\[ B = \text{imguidedfilter}(A, G); \]

Guided filters

Kaiming He, Jian Sun, Xiaou Tang, Guided Image Filtering. PAMI 2013
Things to remember

• Linear filtering is sum of dot product at each position
  • Can smooth, sharpen, translate (among many other uses)

• Gaussian filters
  • Low pass filters, separability, variance

• Attend to details:
  • filter size, extrapolation, cropping

• Application: representing textures

• Noise models and nonlinear image filters
Thank you

• Next class: Image Filters in Frequency Domain