Spatial Filtering
Filtering

- The name “filter” is borrowed from frequency domain processing (next week’s topic)
- Accept or reject certain frequency components
- **Fourier (1807):** Periodic functions could be represented as a weighted sum of sines and cosines

Image courtesy of Technology Review
Signals

• A signal is composed of low and high frequency components

low frequency components: smooth / piecewise smooth
- Neighboring pixels have similar brightness values
- You’re within a region

high frequency components: oscillatory
- Neighboring pixels have different brightness values
- You’re either at the edges or noise points
Signals – Examples
Motivation: noise reduction

• Assume image is degraded with an additive model.
• Then,

\[
\text{Observation} = \text{True signal} + \text{noise}
\]

\[
\text{Observed image} = \text{Actual image} + \text{noise}
\]

low-pass filters

\[
\downarrow
\]

high-pass filters

smooth the image
Common types of noise

- **Salt and pepper noise:** random occurrences of black and white pixels
- **Impulse noise:** random occurrences of white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution
What is the impact of the sigma?
Motivation: noise reduction

- Make multiple observations of the same **static** scene
- Take the average
- Even multiple images of the same static scene will not be identical.
Motivation: noise reduction

- Make multiple observations of the same static scene
- Take the average
- Even multiple images of the same static scene will not be identical.

Adapted from: K. Grauman
Motivation: noise reduction

• Make multiple observations of the same static scene
• Take the average
• Even multiple images of the same static scene will not be identical.

Adapted from: K. Grauman
Motivation: noise reduction

• Make multiple observations of the same static scene
• Take the average
• Even multiple images of the same static scene will not be identical.
• What if we can’t make multiple observations?

What if there’s only one image?

Adapted from: K. Grauman
Image Filtering

• **Idea:** Use the information coming from the neighboring pixels for processing

• Design a transformation function of the local neighborhood at each pixel in the image
  – Function specified by a “filter” or mask saying how to combine values from neighbors.

• Various uses of filtering:
  – Enhance an image (denoise, resize, etc)
  – Extract information (texture, edges, etc)
  – Detect patterns (template matching)

Adapted from: K. Grauman
Filtering

• Processing done on a function
  – can be executed in continuous form (e.g. analog circuit)
  – but can also be executed using sampled representation

• Simple example: smoothing by averaging
Linear filtering

- Filtered value is the linear combination of neighboring pixel values.

- Key properties
  - linearity: $\text{filter}(f + g) = \text{filter}(f) + \text{filter}(g)$
  - shift invariance: behavior invariant to shifting the input
    - delaying an audio signal
    - sliding an image around

- Can be modeled mathematically by convolution

Adapted from: S. Marschner
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 (spatial regularity in images)
  – Expect noise processes to be independent from pixel to pixel
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

Slide credit: S. Marschner
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

Slide credit: S. Marschner
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

Slide credit: S. Marschner
First attempt at a solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

Slide credit: S. Marschner
Convolution warm-up

- Same moving average operation, expressed mathematically:

\[ b_{\text{smooth}}[i] = \frac{1}{2r + 1} \sum_{j=i-r}^{i+r} b[j] \]
Discrete convolution

• Simple averaging:

\[ b_{\text{smooth}}[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} b[j] \]

– every sample gets the same weight

• Convolution: same idea but with *weighted* average

\[ (a \ast b)[i] = \sum_j a[j]b[i-j] \]

– each sample gets its own weight (normally zero far away)

• This is all convolution is: it is a **moving weighted average**
Filters

• Sequence of weights $a[j]$ is called a filter
• Filter is nonzero over its region of support
  – usually centered on zero: support radius $r$
• Filter is normalized so that it sums to 1.0
  – this makes for a weighted average, not just any old weighted sum
• Most filters are symmetric about 0
  – since for images we usually want to treat left and right the same

Slide credit: S. Marschner

[Diagram of a box filter]
Convolution and filtering

- Can express sliding average as convolution with a *box filter*
- $a_{box} = [\ldots, 0, 1, 1, 1, 1, 1, 0, \ldots]$
Example: box and step
Example: box and step

\[ b(i) \times a[j] = a[j] b[i-j] \]

Slide credit: S. Marschner
Example: box and step
Example: box and step

\[
\begin{align*}
    b(i) & \quad \times \\
    a(j) & \quad \sum_{j} a[j] \\
    a[j] b[i-j] & \quad \sum \\
    (a \ast b)[i] & \quad 0.6
\end{align*}
\]
Example: box and step
Convolution and filtering

- Convolution applies with any sequence of weights
- Example: bell curve (gaussian-like) \([..., 1, 4, 6, 4, 1, ...]/16\)
Convolution and filtering

• Convolution applies with any sequence of weights
• Example: bell curve (gaussian-like) \([\ldots, 1, 4, 6, 4, 1, \ldots]/16\)

Slide credit: S. Marschner
And in pseudocode...

```pseudocode
function convolve(sequence a, sequence b, int r, int i )
    s = 0
    for j = -r to r
        s = s + a[j]b[i - j]
    return s
```
Key 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, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.

- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Slide credit: S. Lazebnik
Properties in more detail

- Commutative: $a * b = b * a$
  - Conceptually no difference between filter and signal

- 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 $e = [..., 0, 0, 1, 0, 0, ...]$, $a * e = a$
A gallery of filters

• Box filter
  – Simple and cheap

• Tent filter
  – Linear interpolation

• Gaussian filter
  – Very smooth antialiasing filter
Box filter

\[ a_{\text{box}, r}[i] = \begin{cases} \frac{1}{2r+1} & |i| \leq r, \\ 0 & \text{otherwise}. \end{cases} \]

\[ f_{\text{box}, r}(x) = \begin{cases} \frac{1}{2r} & -r \leq x < r, \\ 0 & \text{otherwise}. \end{cases} \]

Slide credit: S. Marschner
Tent filter

\[ f_{\text{tent}}(x) = \begin{cases} 
1 - |x| & |x| < 1, \\
0 & \text{otherwise}; 
\end{cases} \]

\[ f_{\text{tent}, r}(x) = \frac{f_{\text{tent}}(x/r)}{r}. \]
Gaussian filter

\[ f_g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2} \]
Discrete filtering in 2D

• Same equation, one more index

\[(a \ast b)[i, j] = \sum_{i', j'} a[i', j'] b[i - i', j - j']\]

– now the filter is a rectangle you slide around over a grid of numbers

• Usefulness of associativity
  – 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)

Slide credit: S. Marschner
And in pseudocode...

```plaintext
function convolve2d(filter2d a, filter2d b, int i, int j)
    s = 0
    r = a.radius
    for i' = -r to r do
        for j' = -r to r do
            s = s + a[i'][j']b[i - i'][j - j']
    return s
```
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
# Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Slide credit: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Slide credit: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Slide credit: S. Seitz
Correlation filtering

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

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

Attribute uniform weight to each pixel
Loop over all pixels in neighborhood around image pixel $F[i,j]$  

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  

Slide credit: K. Grauman
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.
Correlation filtering

Scene

Template (mask)
Correlation filtering

Detected template

Correlation map
Cross correlation example

Left

Right

Scanline

Norm. corr

Slide credit: Fei-Fei Li
Averaging filter

• What values belong in the kernel $H$ for the moving average example?

\[ G = H \otimes F \]
Averaging filter

- What values belong in the kernel $H$ for the moving average example?

\[ G = H \otimes F \]
Smoothing by averaging

original

filtered

depicts box filter:
white = high value, black = low value

Slide credit: K. Grauman
Smoothing by averaging

depicts box filter:
white = high value, black = low value

original

What if the filter size was 5 x 5 instead of 3 x 3?

filtered

Slide credit: K. Grauman
Boundary issues

• What is the size of the output?

• MATLAB: output size / “shape” options
  – \textit{shape} = ‘full’: output size is sum of sizes of \textit{f} and \textit{g}
  – \textit{shape} = ‘same’: output size is same as \textit{f}
  – \textit{shape} = ‘valid’: output size is difference of sizes of \textit{f} and \textit{g}
Boundary issues

• 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
Boundary issues

- 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
Boundary issues

- 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
Boundary issues

• 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
Boundary issues

• 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
Boundary issues

• 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
Boundary issues

• 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

Slide credit: S. Marschner
Boundary issues

• 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
Boundary issues

• 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

Slide credit: S. Marschner
Boundary issues

• 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
Boundary issues

• 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
Boundary issues

• 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
Boundary issues

- 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
Boundary issues

• What about near the edge?
  – the filter window falls off the edge of the image
  – need to extrapolate
  – 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')}
Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

This kernel is an approximation of a 2d Gaussian function:

\[ h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}} \]
Smoothing with a Gaussian
Gaussian filters

• What parameters matter here?

• **Size** of kernel or mask
  – Note, Gaussian function has infinite support, but discrete filters use finite kernels

\[ \sigma = 5 \text{ with } 10 \times 10 \text{ kernel} \quad \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]
Gaussian filters

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

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

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

Slide credit: K. Grauman
Choosing kernel width

- Rule of thumb: set filter half-width to about $3\sigma$
Matlab

```matlab
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian' hsize, sigma);

>> mesh(h);

>> imagesc(h);

>> outim = imfilter(im, h); % correlation
>> imshow(outim);
```

Slide credit: K. Grauman
Smoothing with a Gaussian

Parameter $\sigma$ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

for sigma=1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end

Slide credit: K. Grauman
Separability

• In some cases, filter is separable, and we can factor into two steps:
  – Convolve all rows
  – Convolve all columns
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.
Separability example

2D convolution (center location only)

The filter factors into a product of 1D filters:

Perform convolution along rows:

Followed by convolution along the remaining column:

Slide credit: K. Grauman
Why is separability useful?

• 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?
  – \( O(n^2 \, m) \)
Properties of smoothing filters

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

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

\[
G = H \ast F
\]

Notation for convolution operator

Slide credit: K. Grauman
Convolution vs. Correlation

- A **convolution** is an integral that expresses the amount of overlap of one function as it is shifted over another function.
  - convolution is a filtering operation

- **Correlation** compares the *similarity of two sets of data*. Correlation computes a measure of similarity of two input signals as they are shifted by one another. The correlation result reaches a maximum at the time when the two signals match best.
  - correlation is a measure of relatedness of two signals

Slide credit: Fei-Fei Li
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

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

\[ G = H \otimes F \]

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

Slide credit: K. Grauman
Predict the outputs using correlation filtering

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

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

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

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

Slide credit: K. Grauman
Practice with linear filters

Original

?
Practice with linear filters

Original

Filtered
(no change)

Slide credit: D. Lowe
Practice with linear filters

Original

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

?
Practice with linear filters

Original

Shifted left by 1 pixel with correlation

Slide credit: D. Lowe
Practice with linear filters

Original

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\]
Practice with linear filters

Original

Blur (with a box filter)

Slide credit: D. Lowe
Practice with linear filters

Original

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

?
Practice with linear filters

Sharpening filter:
accentuates differences with local average

Slide credit: D. Lowe
Filtering examples: sharpening
Sharpening

- What does blurring take away?

Let's add it back:

Slide credit: S. Lazebnik
Unsharp mask filter

\[ f + \alpha (f - f \ast g) = (1 + \alpha) f - \alpha f \ast g = f \ast ((1 + \alpha) e - g) \]

- Image
- Blurred image
- Unit impulse (identity)

Unit impulse

Gaussian

Laplacian of Gaussian

Slide credit: S. Lazebnik
Other filters

Sobel

Vertical Edge
(absolute value)

Slide credit: J. Hays
Other filters

Sobel

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

Horizontal Edge (absolute value)

Slide credit: J. Hays
Median filters

- A **Median Filter** operates over a window by selecting the median intensity in the window.
- What advantage does a median filter have over a mean filter?
- Is a median filter a kind of convolution?

adapted from: S. Seitz
Median filter

- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter

Slide credit: K. Grauman
Median filter

Salt and pepper noise

Median filtered

Plots of a row of the image

Matlab: output im = medfilt2(im, [h w]);

Slide credit: M. Hebert
Median filter

• What advantage does median filtering have over Gaussian filtering?
  – Robustness to outliers
  – Median filter is edge preserving