Linear filtering
Motivation: Image denoising

- How can we reduce noise in a photograph?
Moving average

• Let’s replace each pixel with a weighted average of its neighborhood
• The weights are called the filter kernel
• What are the weights for the average of a 3x3 neighborhood?

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

“box filter”

Source: D. Lowe
Defining convolution

• Let \( f \) be the image and \( g \) be the kernel. The output of convolving \( f \) with \( g \) is denoted \( f \ast g \).

\[
(f \ast g)[m,n] = \sum_{k,l} f[m-k, n-l] g[k, l]
\]

Convention: kernel is “flipped”

• MATLAB functions: \texttt{conv2}, \texttt{filter2}, \texttt{imfilter}

Source: F. Durand
Key properties

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

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

• Theoretical result: any linear shift-invariant operator can be represented as a convolution
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 \( e = […, 0, 0, 1, 0, 0, …] \), \( a * e = a \)
Annoying details

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`
Annoying details

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
Annoying details

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’)}

Source: S. Marschner
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

```
0 0 0
0 1 0
0 0 0
```

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{array}{ccc}
\frac{1}{9} & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

? 

Source: D. Lowe
Practice with linear filters

Original

Blur (with a box filter)

Source: D. Lowe
Practice with linear filters

Original

(Note that filter sums to 1)
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Sharpening

What does blurring take away?

Let’s add it back:
Smoothing with box filter revisited

- What’s wrong with this picture?
- What’s the solution?

Source: D. Forsyth
Smoothing with box filter revisited

- What’s wrong with this picture?
- What’s the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center.

“fuzzy blob”
Gaussian Kernel

\[ G_\sigma = \frac{1}{2\pi \sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

Source: C. Rasmussen
Gaussian Kernel

\[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

- Standard deviation \( \sigma \): determines extent of smoothing

Source: K. Grauman
Choosing kernel width

• The Gaussian function has infinite support, but discrete filters use finite kernels
Choosing kernel width

- Rule of thumb: set filter half-width to about $3\sigma$
Gaussian vs. box filtering
Gaussian filters

• Remove “high-frequency” components from the image (low-pass filter)

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

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

Source: K. Grauman
Separability of the Gaussian filter

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

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

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 convolution
(center location only)

The filter factors
into a product of 1D filters:

Perform convolution
along rows:

Followed by convolution
along the remaining column:

Source: 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?
  • $O(n^2 m)$
Noise

- **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

\[ f(x, y) = \hat{f}(x, y) + \eta(x, y) \]

Gaussian i.i.d. ("white") noise:
\[ \eta(x, y) \sim N(\mu, \sigma) \]

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.

![Median filtering diagram](image)

- Is median filtering linear?

Source: K. Grauman
Median filter

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

Source: K. Grauman
Median filter

MATLAB: medfilt2(image, [h w])
### Gaussian vs. Median Filtering

<table>
<thead>
<tr>
<th></th>
<th>3x3</th>
<th>5x5</th>
<th>7x7</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Gaussian</strong></td>
<td><img src="image" alt="Gaussian 3x3" /></td>
<td><img src="image" alt="Gaussian 5x5" /></td>
<td><img src="image" alt="Gaussian 7x7" /></td>
</tr>
<tr>
<td><strong>Median</strong></td>
<td><img src="image" alt="Median 3x3" /></td>
<td><img src="image" alt="Median 5x5" /></td>
<td><img src="image" alt="Median 7x7" /></td>
</tr>
</tbody>
</table>
Sharpening revisited

before

after

Source: D. Lowe
Sharpening revisited

What does blurring take away?

Let’s add it back:
Unsharp mask filter

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

- Image
- Blurred image
- Unit impulse (identity)
- Unit impulse
- Gaussian
- Laplacian of Gaussian
Application: Hybrid Images

Changing expression

Sad  Surprised
Application: Hybrid Images

Gaussian Filter

Laplacian Filter