Linear filtering
Motivation: 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
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?

```
1 1 1
1 1 1
1 1 1
```

"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 $f * g$.

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

- Convention: kernel is “flipped”
- MATLAB: conv2 vs. filter2 (also 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 \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \), \( a * e = a \)
Annoying details

What is the size of the output?

• MATLAB: filter2(g, f, \textit{shape})
  • \textit{shape} = ‘full’: output size is sum of sizes of f and g
  • \textit{shape} = ‘same’: output size is same as f
  • \textit{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

Filtering matrix:

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

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

\[ \frac{1}{9} \]

?  

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)

Source: D. Lowe
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Source: D. Lowe
Smoothing with box filter revisited

- Smoothing with an average actually doesn’t compare at all well with a defocused lens
- Most obvious difference is that a single point of light viewed in a defocused lens looks like a fuzzy blob; but the averaging process would give a little square

Source: D. Forsyth
Smoothing with box filter revisited

- Smoothing with an average actually doesn’t compare at all well with a defocused lens.
- Most obvious difference is that a single point of light viewed in a defocused lens looks like a fuzzy blob; but the averaging process would give a little square.
- Better idea: to eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center, like so:

  "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, as we should re-normalize weights to sum to 1 in any case)

Source: C. Rasmussen
Choosing kernel width

- Gaussian filters have infinite support, but discrete filters use finite kernels

Source: K. Grauman
Choosing kernel width

- Rule of thumb: set filter half-width to about $3 \sigma$
Example: Smoothing with a Gaussian
Mean vs. Gaussian filtering
Gaussian filters

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

Source: K. Grauman
Separability of the Gaussian filter

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

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

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
Separability

• Why is separability useful in practice?
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

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

\[ \eta(x, y) \sim N(\mu, \sigma) \]

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.

```
+-----+-----+-----+
| 10  | 15  | 20  |
| 23  | 90  | 27  |
| 33  | 31  | 30  |
+-----+-----+-----+
```

**Median value**

```
+-----+-----+-----+
| 10  | 15  | 20  |
| 23  | 27  | 27  |
| 33  | 31  | 30  |
+-----+-----+-----+
```

- 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

Salt-and-pepper noise

Median filtered

MATLAB: medfilt2(image, [h w])

Source: M. Hebert
Median vs. Gaussian filtering

Gaussian

Median

3x3  5x5  7x7
Sharpening revisited

What does blurring take away?

Let's add it back:
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
Application: Hybrid Images

Applicatioin: Hybrid Images