Linear filtering
Overview: Linear filtering

• Linear filters
  • Definition and properties
  • Examples
  • Gaussian smoothing
  • Separability

• Applications
  • Denoising
    – Non-linear filters: median filter
  • Sharpening
  • Subsampling
Images as functions

Source: S. Seitz
Images as functions

- We can think of an image as a function, $f$, from $\mathbb{R}^2$ to $\mathbb{R}$:
  - $f(x, y)$ gives the intensity at position $(x, y)$
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - $f: [a, b] \times [c, d] \rightarrow [0, 1]$

- A color image is just three functions pasted together. We can write this as a “vector-valued” function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Source: S. Seitz
What is a digital image?

- In computer vision we usually operate on **digital (discrete)** images:
  - **Sample** the 2D space on a regular grid
  - **Quantize** each sample (round to nearest integer)

- The image can now be represented as a matrix of integer values

\[
\begin{array}{cccccccc}
    & & & & & & & \\
   |\  & 62 & 79 & 23 & 119 & 120 & 105 & 4 & 0 \\
   i | 10 & 10 & 9 & 62 & 12 & 78 & 34 & 0 \\
   | 10 & 58 & 197 & 46 & 46 & 0 & 0 & 48 \\
   | 176 & 135 & 5 & 188 & 191 & 68 & 0 & 49 \\
   | 2 & 1 & 1 & 29 & 26 & 37 & 0 & 77 \\
   | 0 & 89 & 144 & 147 & 187 & 102 & 62 & 208 \\
   | 255 & 252 & 0 & 166 & 123 & 62 & 0 & 31 \\
   | 166 & 63 & 127 & 17 & 1 & 0 & 99 & 30 \\
\end{array}
\]

Source: S. Seitz
Motivation 1: 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
Motivation 2: Image half-sizing

• This image is too big to fit on the screen. How can we reduce it by a factor of two?

• How about taking every second pixel?

Source: S. Seitz
First attempt at a solution

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

Source: 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:

Source: S. Marschner
Weighted Moving Average

Can add weights to our moving average

Weights \[\{1, 1, 1, 1, 1\} / 5\]

Source: S. Marschner
Weighted Moving Average

Non-uniform weights $[1, 4, 6, 4, 1] / 16$

Source: S. Marschner
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \quad \quad G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz
Generalization of 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 a 3x3 moving average?

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

“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 \( f * g \).

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

- 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: \( a \ast b = b \ast a \)
  • 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: \( ka \ast b = a \ast kb = k (a \ast b) \)

• Identity: unit impulse \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \), \( a \ast e = a \)
Yucky 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`
Yucky 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
Yucky 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

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

Source: D. Lowe
Practice with linear filters

Original

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

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

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

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

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

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 $\sqrt{2}\sigma$
- **Separable** kernel
  - Factors into product of two 1D Gaussians
  - Useful: can convolve all rows, then all columns
  - How does this change the computational complexity?

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)

\[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix} \ast \begin{bmatrix}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6
\end{bmatrix}
= \begin{bmatrix}
2 + 6 + 3 = 11 \\
6 + 20 + 10 = 36 \\
4 + 8 + 6 = 18
\end{bmatrix}
\]

The filter factors into a product of 1D filters:

\[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix} \ast \begin{bmatrix}
1 \\
2 \\
1
\end{bmatrix} = \begin{bmatrix}
11 \\
18 \\
18
\end{bmatrix}
\]

Perform convolution along rows:

\[
\begin{bmatrix}
1 & 2 & 1
\end{bmatrix} \ast \begin{bmatrix}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6
\end{bmatrix} = \begin{bmatrix}
11 \\
18 \\
18
\end{bmatrix}
\]

Followed by convolution along the remaining column:

\[
\begin{bmatrix}
1 \\
2 \\
1
\end{bmatrix} \ast \begin{bmatrix}
11 \\
18 \\
18
\end{bmatrix} = \begin{bmatrix}
65
\end{bmatrix}
\]

Source: K. Grauman
Review: Linear filtering

- What are the defining mathematical properties of a convolution?
- What is the difference between blurring with a box filter and blurring with a Gaussian?
- What happens when we convolve a Gaussian with another Gaussian?
- What is separability?
- How does separability affect computational complexity?
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:
\[ \eta(x, y) \sim \mathcal{N}(\mu, \sigma) \]

Source: K. Grauman
Reducing Gaussian noise

<table>
<thead>
<tr>
<th>$\sigma=0.05$</th>
<th>$\sigma=0.1$</th>
<th>$\sigma=0.2$</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
<td><img src="https://via.placeholder.com/150" alt="Image" /></td>
</tr>
</tbody>
</table>

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

3x3 5x5 7x7

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

• 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: K. Grauman
Median vs. Gaussian filtering

Gaussian

Median
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
Recall

**Sharpening filter**
- Accentuates differences with local average
Image half-sizing

• This image is too big to fit on the screen. How can we reduce it by a factor of two?

• How about taking every second pixel?

Source: S. Seitz
Image sub-sampling

Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*
Aliasing

• Not a good idea to shrink an image by taking every second pixel

• If we do, characteristic errors appear
  • Wagon wheels rolling the wrong way in movies
  • Checkerboards disintegrate in ray tracing
  • Striped shirts look funny on color television

Source: D. Forsyth
Example

Resample the checkerboard by taking one sample at each circle. Top left and top right: reasonable results. Bottom left is all black and bottom right has checks that are too big.

Source: D. Forsyth
Problem: Undersampling

• What if we “missed” things between the samples?
• Simple example: undersampling a sine wave
  • unsurprising result: information is lost

Source: S. Marschner
Problem: Undersampling

• What if we “missed” things between the samples?

• Simple example: undersampling a sine wave
  • unsurprising result: information is lost
  • surprising result: indistinguishable from lower frequency

Source: S. Marschner
Problem: Undersampling

- What if we “missed” things between the samples?
- Simple example: undersampling a sine wave
  - unsurprising result: information is lost
  - surprising result: indistinguishable from lower frequency
- **Aliasing**: signals “traveling in disguise” as other frequencies

Source: S. Marschner
Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what’s happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):

Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)
Aliasing in graphics

Disintegrating textures

Source: A. Efros
Sampling Theorem

• When sampling a signal at discrete intervals, the sampling frequency must be greater than twice the highest frequency of the input signal in order to be able to reconstruct the original perfectly from the sampled version (Shannon, Nyquist, Whittaker, Kotelnikov)

Source: F. Durand
Antialiasing

• What can we do about aliasing?
• Sample more often
  • But when we want to downsample a given digital image, we can’t increase its original sampling resolution
• Make the signal less “wiggly”
  • Get rid of all frequencies that are greater than half the new sampling frequency
  • Will lose information
  • But it’s better than aliasing

Source: A. Efros
Gaussian (low-pass) pre-filtering

Solution: filter the image, then subsample

- Filter size should double for each $\frac{1}{2}$ size reduction

Source: Steve Seitz
Subsampling with Gaussian pre-filtering

1/2

1/4 (2x zoom)

1/8 (4x zoom)

Source: Steve Seitz
Without prefiltering:

1/2

1/4 (2x zoom)

1/8 (4x zoom)

Source: Steve Seitz