Lecture 2: Image filtering
• PS1 due next Tuesday
• Updated office hours next week, due to holiday. New times will be on Piazza.
• Questions?
Recall last week…

Input image

Edges

Extra edges

Missing edges
In this lecture

What *other* transformations can we do?
Filtering

\[ g[n, m] \rightarrow H \rightarrow f[n, m] \]

Our goal: remove unwanted sources of variation, and keep the information relevant for whatever task we need to solve.
Linear filtering

Very general! For a filter, $H$, to be linear, it has to satisfy:

$$H(a[m, n] + b[m, n]) + H(a[m, n]) + H(b[m, n])$$

$$H(Ca[m, n]) = CH(a[m, n])$$

Source: Torralba, Freeman, Isola
A linear filter in its most general form can be written as (for a 1D signal of length N):

$$f[n] = \sum_{k=0}^{N-1} h[n,k] g[k]$$

In matrix form:

$$\begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M-1] \end{bmatrix} = \begin{bmatrix} h[0,0] & h[0,1] & \cdots & h[0,N-1] \\ h[1,0] & h[1,1] & \cdots & h[1,N-1] \\ \vdots & \vdots & \ddots & \vdots \\ h[M-1,0] & h[M-1,1] & \cdots & h[M-1,N-1] \end{bmatrix} \begin{bmatrix} g[0] \\ g[1] \\ \vdots \\ g[N-1] \end{bmatrix}$$

Source: Torralba, Freeman, Isola
Why handle each spatial position differently?

Want translation invariance!

\[
\begin{bmatrix}
  f[0] \\
  f[1] \\
  \vdots \\
  f[M-1]
\end{bmatrix}
= \begin{bmatrix}
  h[0,0] & h[0,1] & \ldots & h[0,N-1] \\
  h[1,0] & h[1,1] & \ldots & h[1,N-1] \\
  \vdots & \vdots & \ddots & \vdots \\
  h[M-1,0] & h[M-1,1] & \ldots & h[M-1,N-1]
\end{bmatrix}
\begin{bmatrix}
  g[0] \\
  g[1] \\
  \vdots \\
  g[N-1]
\end{bmatrix}
\]
Image denoising
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
Moving average

Input

Output

Filter kernel

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

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

Filter kernel

Input

Output

{0, 0, 0, 0, 0, 0, 0, 0, 0}
{0, 90, 90, 90, 90, 90, 90, 90, 90}
{0, 90, 90, 90, 90, 90, 90, 90, 90}
{0, 90, 90, 90, 90, 90, 90, 90, 90}
{0, 90, 0, 90, 90, 90, 90, 90, 90}
{0, 90, 90, 90, 90, 90, 90, 90, 90}
{0, 90, 90, 90, 90, 90, 90, 90, 90}
{0, 0, 0, 0, 0, 0, 0, 0, 0}

\[
\frac{1}{9} \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]
Moving average

Filter kernel

Input

Output
Moving average

Input

Output

Filter kernel

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

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

Filter kernel

Input

Output
Moving average

Input

Output

Filter kernel

1/9
Moving average

Input

Output

Filter kernel

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

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

Filter kernel

Input

Output
Handling boundaries

Source: Torralba, Freeman, Isola
Handling boundaries

Zero padding

Source: Torralba, Freeman, Isola
Handling boundaries

Source: Torralba, Freeman, Isola
## Moving average

**Filter kernel**

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

### Input

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>90</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Output

```
40 60 60 40 20
60 90 60 40 20
50 80 80 60 30
50 80 80 60 30
30 50 50 40 20
```
Convolution

• Let \( h \) be the image and \( g \) be the kernel. The output of convolving \( h \) with \( g \) is:

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

Convention:
kernel is “flipped”

Source: F. Durand
Properties of the convolution

Commutative

\[ h[n] \circ g[n] = g[n] \circ h[n] \]

Associative

\[ h[n] \circ g[n] \circ q[n] = h[n] \circ (g[n] \circ q[n]) = (h[n] \circ g[n]) \circ q[n] \]

Distributive with respect to the sum

\[ h[n] \circ (f[n] + g[n]) = h[n] \circ f[n] + h[n] \circ g[n] \]
Why flip the kernel?

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

Indexes go backward!
Cross correlation

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

No flipping!

- Sometimes called just **correlation**
- Neither associative nor commutative
- In the literature, people often just call both “convolution”
- Filters often symmetric, so won’t matter
Convolutional neural networks

- Neural network with specialized connectivity structure
- Mostly just convolutions!

(LeCun et al. 1989)

Source: Torralba, Freeman, Isola
Filtering examples
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

“Impulse”

Filtered
(no change)

Source: D. Lowe
Practice with linear filters

Original

“Translated Impulse”

Source: D. Lowe
Practice with linear filters

Original

Shifted left
By 1 pixel

Source: D. Lowe
Practice with linear filters

Original

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

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)

Source: D. Lowe
Practice with linear filters

Original

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe
Practice with linear filters

Original

Can you do this?
Rectangular filter

\[ g[m,n] \times h[m,n] = f[m,n] \]

Source: Torralba, Freeman, Isola
Rectangular filter

\[ g[m,n] \otimes h[m,n] = f[m,n] \]

Source: Torralba, Freeman, Isola
“Naturally” occurring filters

Input image

Motion blur

Source: Torralba, Freeman, Isola
“Naturally” occurring filters

Input image  Convolution weights  Convolution output

Source: Torralba, Freeman, Isola
Camera shake

(from Fergus et al, 2007)
Blur occurs in many natural situations

Source: Torralba, Freeman, Isola
Smoothing with box filter revisited

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

Source: S. Lazebnik
Gaussian kernel

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

- Constant factor in front makes kernel sum to 1 (can also omit it and just divide by sum of filter weights).

Source: K. Grauman
Gaussian vs. box filtering

Source: S. Lazebnik
Gaussian standard deviation

\[ \sigma = 2 \]

\[ \sigma = 4 \]

\[ \sigma = 8 \]

Source: Torralba, Freeman, Isola
Gaussian filters

- Convolution with self is another Gaussian
- Can smooth with small-\(\sigma\) kernel, repeat, get same result as larger-\(\sigma\)
- Convolving two times with Gaussian kernel with std. dev. \(\sigma\) is same as convolving once with kernel with std. dev. \(\sigma \sqrt{2}\)

Source: K. Grauman
Gaussian filters

- It’s a **separable kernel**
  - Blur with 1D Gaussian in one direction, then the other.
  - Faster to compute. $O(n)$ time for an $n \times n$ kernel instead of $O(n^2)$
  - Learn more about this in Problem Set 1!

\[ I \quad \text{blur}_x(I) \quad \text{blur}_y(\text{blur}_x(I)) \]
Edges: recall last lecture...

Image gradient:

$$\nabla I = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial I}{\partial x} \approx I(x, y) - I(x - 1, y)$$

Edge strength:

$$E(x, y) = |\nabla I(x, y)|$$

Edge orientation:

$$\theta(x, y) = \angle \nabla I = \arctan \frac{\partial I/\partial y}{\partial I/\partial x}$$

Edge normal:

$$n = \frac{\nabla I}{|\nabla I|}$$

Slide credit: Antonio Torralba
Discrete derivatives

\[ d_0 = [1, -1] \]
\[ f \circ d_0 = f[n] - f[n - 1] \]

\[ d_1 = [1, 0, -1] / 2 \]
\[ f \circ d_1 = \frac{f[n + 1] - f[n - 1]}{2} \]

Source: Torralba, Freeman, Isola
\[ \begin{bmatrix} -1 & 1 \end{bmatrix} \]

\[ g[m,n] \quad \circ \quad [-1, 1] \quad = \quad h[m,n] \quad \circ \quad f[m,n] \]

Source: Torralba, Freeman, Isola
$[-1 \ 1]^{T}$

$g[m,n] \circ [-1, 1]^{T} = h[m,n] = f[m,n]$
Can we recover the image?

Source: Torralba, Freeman, Isola
Reconstruction from 2D derivatives

In 2D, we have multiple derivatives (along \( n \) and \( m \))

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

\( \mathbf{c} \)

\( \mathbf{c} \)

\( \mathbf{c} \)

\( \mathbf{c} \)

\( \mathbf{c} \)

\( \mathbf{c} \)

\( \mathbf{c} \)

\( \mathbf{c} \)

and we compute the pseudo-inverse of the full matrix.

Source: Torralba, Freeman, Isola
Reconstruction from 2D derivatives

Source: Torralba, Freeman, Isola
Editing the edge image

Source: Torralba, Freeman, Isola
Thresholding edges

Source: Torralba, Freeman, Isola
Issues with derivative filters

- Sensitive to edges at small spatial scales
- Also sensitive to noise
- You'll see this in Problem Set 1
Why is this happening?

Noisy input image

Source: S. Seitz
Solution: smooth first

To find edges, look for peaks in $\frac{d}{dx}(f \ast h)$
Derivative of Gaussian filter

\[ h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \]

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]

Source: N. Snavely
Derivative of Gaussian filter

$x$-direction

$y$-direction

Source: N. Snavely
Derivatives of Gaussians: Scale

Source: Torralba, Freeman, Isola
Picks up larger-scale edges

\[ g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi \sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

\[ g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi \sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}} \]

Source: Torralba, Freeman, Isola
Computing a directional derivative

(From multivariable calculus)
\[ \nabla_{\vec{u}} f(\vec{x}) = \nabla f(\vec{x}) \cdot \vec{u} \]

Directional derivative is a linear combination of partial derivatives

Source: N. Snavely
Derivative of Gaussian filter

\[ \text{cos}(\theta) + \text{sin}(\theta) = \]

Source: N. Snavely
The Sobel operator

- Common approximation to derivative of Gaussian
- Where does this come from?

\[
\begin{array}{ccc}
\frac{1}{8} & -1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
\frac{1}{8} & 1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1 \\
\end{array}
\]

Source: N. Snavely
An approximation to the Gaussian

- Apply filter to itself repeatedly.
- Converges to Gaussian, due to Central Limit Theorem

\[
\begin{align*}
  b_1 &= [1 \ 1] \\
  b_2 &= [1 \ 1] \circ [1 \ 1] = [1 \ 2 \ 1] \\
  b_3 &= [1 \ 1] \circ [1 \ 1] \circ [1 \ 1] = [1 \ 3 \ 3 \ 1]
\end{align*}
\]

Source: Torralba, Freeman, Isola
Binomial filter

\[
\begin{array}{cccccc}
  b_1 & & & & & \\
  b_2 & 1 & 1 & 1 & & \\
  b_3 & 1 & 3 & 3 & 1 & \\
  b_4 & 1 & 4 & 6 & 4 & 1 \\
  b_5 & 1 & 5 & 10 & 10 & 5 & 1 \\
  b_6 & 1 & 6 & 15 & 20 & 15 & 6 & 1 \\
  b_7 & 1 & 7 & 21 & 35 & 35 & 21 & 7 & 1 \\
  b_8 & 1 & 8 & 28 & 56 & 70 & 56 & 28 & 8 & 1 \\
\end{array}
\]

\[
\begin{align*}
\sigma_1^2 &= 1/4 \\
\sigma_2^2 &= 1/2 \\
\sigma_3^2 &= 3/4 \\
\sigma_4^2 &= 1 \\
\sigma_5^2 &= 5/4 \\
\sigma_6^2 &= 3/2 \\
\sigma_7^2 &= 7/4 \\
\sigma_8^2 &= 2
\end{align*}
\]
Sobel operator: example

Source: N. Snavely / Wikipedia
Next class: frequency