Lecture 2: Filtering
Recall: blurring and edge detection

Denoising

Edge detection
Recall: neighborhood filtering
Recall: neighborhood filtering

Input

Output

Filter kernel
Recall: neighborhood filtering

Filter kernel

Input

Output

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

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

\[
\begin{align*}
\frac{1}{9} \times 0 + \frac{1}{9} \times 0 + \frac{1}{9} \times 0 \\
\frac{1}{9} \times 0 + \frac{1}{9} \times 0 + \frac{1}{9} \times 0 \\
\frac{1}{9} \times 0 + \frac{1}{9} \times 90 + \frac{1}{9} \times 90 \\
\frac{1}{9} \times 0 + \frac{1}{9} \times 90 + \frac{1}{9} \times 90 \\
\end{align*}
\]
Recall: neighborhood filtering

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 90 & 90 & 90 \\
0 & 90 & 90 & 90 \\
0 & 90 & 90 & 90 \\
0 & 90 & 0 & 90 \\
0 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

Input

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 90 & 90 & 90 \\
0 & 90 & 90 & 90 \\
0 & 90 & 90 & 90 \\
0 & 90 & 90 & 90 \\
0 & 90 & 0 & 90 \\
0 & 90 & 90 & 90 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

Output

Filter kernel

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

\[\frac{1}{9}\]
Recall: neighborhood filtering

Filter kernel

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

\[
\begin{align*}
\frac{1}{9} \times 0 + \frac{1}{9} \times 0 + \frac{1}{9} \times 0 \\
\frac{1}{9} \times 90 + \frac{1}{9} \times 90 + \frac{1}{9} \times 90 \\
\frac{1}{9} \times 90 + \frac{1}{9} \times 90 + \frac{1}{9} \times 90 \\
\end{align*}
\]
Recall: neighborhood filtering

Filter kernel

Input

Output
Today

• Linear filtering
• More neighborhood filters
• Nonlinear filters
Filtering

Remove unwanted sources of variation, keeping only information that’s relevant for a task.

\[ f[n, m] \xrightarrow{H} g[n, m] \]
Linear filtering

- Linear transformation:
  \[ f[n, m] = \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} h[n, m, k, l] g[k, l] \]

- Equivalent to multiplication with a matrix \( H \):
  \[ f = Hg \]

- A very general transformation. Allows for many types of image changes.

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

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

Often want translation invariance for our linear filters.
Cross-correlation

Let $f$ be the image and $h$ be the kernel. The output of cross-correlating $f$ with $h$ is:

$$g[m, n] = f \circ h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m + k, n + l]h[k, l]$$
Cross-correlation

Let $f$ be the image and $h$ be the kernel. The output of cross-correlating $f$ with $h$ is:

$$g[m, n] = f \ast h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m + k, n + l]h[k, l]$$
Cross-correlation

Let $f$ be the image and $h$ be the kernel. The output of cross-correlating $f$ with $h$ is:

$$g[m, n] = f \circ h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m + k, n + l] h[k, l]$$

Source: F. Durand
Cross-correlation

Let $f$ be the image and $h$ be the kernel. The output of cross-correlating $f$ with $h$ is:

\[ g[m, n] = f \circ h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m + k, n + l] h[k, l] \]
Cross-correlation

Is this equivalent to what we saw last class? Yes.

- Given a tiny filter, pad it to make it $N \times N$.
- Convention: out of bounds indexes wrap around.

\[ g[m, n] = f \cdot h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m + k, n + l]h[k, l] \]
Let $g$ be the image and $h$ be the kernel. The output of convolving $f$ with $h$ is:

$$g[m, n] = f \ast h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m - k, n - l] h[k, l]$$
Why flip the kernel?

\[ g[m, n] = f \cdot h = \sum_{k=1}^{N} \sum_{l=1}^{N} f[m - k, n - l]h[k, l] \]

Indexes go backward!

- Gives it nice mathematical properties.
- When filter is symmetric, equivalent to cross-correlation.
Properties of the convolution

Commutative (no distinction between filter and image):
\[ h[n] \circledast g[n] = g[n] \circledast h[n] \]

Associative (doesn’t matter what order you do 2 convolutions):
\[ h[n] \circledast g[n] \circledast f[n] = h[n] \circledast (g[n] \circledast f[n]) = (h[n] \circledast g[n]) \circledast f[n] \]

Distributive with respect to the sum:
\[ h[n] \circledast (f[n] + g[n]) = h[n] \circledast f[n] + h[n] \circledast g[n] \]
Today

- Linear filtering
- More neighborhood filters
- Nonlinear filters
Rectangular filters

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

Source: Torralba, Freeman, Isola
Rectangular filters

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

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

[Fergus et al., 2007]

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

Blur
Blurring revisited

What’s wrong with this result?

Box filter
**Blurring revisited**

**Idea:** weight contribution of neighborhood pixels according to their closeness to the center
Gaussian kernel

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

- Constant factor makes kernel sum to 1
  - In practice: create filter without the factor, then normalize it.
- Make sure to center it: \((x, y)\) is the center of the kernel, not a corner
Gaussian vs. box filtering

Source: S. Lazebnik
Gaussian standard deviation

\[ \sigma = 2 \quad \sigma = 4 \quad \sigma = 8 \]

Source: Torralba, Freeman, Isola
Useful properties of Gaussian filters

Convolve a Gaussian filter with itself? Get another Gaussian.
Useful properties of Gaussian filters

Smooth with small $\sigma$ repeatedly. Equivalent to smoothing with large $\sigma$!

Exploits associativity:

$$I \circ (h \circ h \circ \ldots \circ h) = ((I \circ h) \circ h) \circ \ldots \circ h$$
Useful properties of Gaussian filters

Gaussian kernels are *separable*.

- Blur with 1D Gaussian in one direction, then the other. Produces same result!
- These are \((n \times 1)\) and \((1 \times n)\) rectangular filters.
- Fast! For an \(n \times n\) kernel, \(\mathcal{O}(n)\) instead of \(\mathcal{O}(n^2)\).
Recall: derivatives

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

Another option:

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

Adapted from: Torralba, Freeman, Isola
Problems with simple derivative filters

- [1 -1]

Sensitive to “tiny” edges
Where’s the edge?

Noisy input image

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]
Solution: smooth first

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

Source: S. Seitz
Derivative of Gaussian filter

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

\[ \frac{\partial}{\partial x} h_\sigma(u, v) \]
Derivative of Gaussian filter

x-direction

y-direction

Source: N. Snavely
Gaussian Scale

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$

Source: Torralba, Freeman, Isola
Derivative of Gaussian scales

Source: Torralba, Freeman, Isola
Derivatives in other directions

Define a filter?

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

Image $f$

desired direction

$\nabla_{\vec{u}} f = ?$

$\nabla_{\vec{u}} f$

What if we need lots of angles? This could get expensive.

Adapted from N. Snavely
Steerable filters

Given \( \frac{\partial f}{\partial x} \) and \( \frac{\partial f}{\partial y} \), steer to direction \( \vec{u} = [u_x \ u_y] \).

Use a multivariable calculus fact:
\[
\nabla_{\vec{u}} f(x) = \nabla f(x) \cdot \vec{u}
\]

Directional derivative = linear combination of partial derivatives.

Source: N. Snavely
Also works for derivative of Gaussian filter

\[
\cos(\theta) + \sin(\theta) = \text{Source: N. Snavely}
\]

Filter approximations

Binomial filter $\approx$ Gaussian

\[
\begin{array}{ccc}
\frac{1}{4} & 1 & 2 & 1 \\
\frac{1}{8} & 1 & 3 & 3 & 1 \\
\frac{1}{16} & 1 & 4 & 6 & 4 & 1 \\
\end{array}
\]

Sobel filter $\approx$

Derivative of Gaussian

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

\[
\begin{pmatrix}
1 & -2 & 1 \\
1 & -1 & 1 \\
\end{pmatrix}
\approx
\begin{pmatrix}
1 & -1 \\
\partial f / \partial x \\
\end{pmatrix}
\odot
\begin{pmatrix}
1 & -1 \\
\partial f / \partial x \\
\end{pmatrix}
\]

\[
\frac{\partial f^2}{\partial x^2}
\]
Laplacian filter

\[ \frac{\partial I^2}{\partial x^2} + \frac{\partial I^2}{\partial y^2} \approx I \circ \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \]

\[ = I \circ [1 \quad -2 \quad 1] + I \circ [1 \quad -2 \quad 1]^T \]
Laplacian filter

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

Detects edges and blobs, and not sensitive to orientation. But sensitive to noise.

Source: Torralba, Freeman, and Isola
Laplacian of Gaussian

\[
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\circ
= \quad
\begin{bmatrix}
0 & 1 & 0 \\
1 & -4 & 1 \\
0 & 1 & 0
\end{bmatrix}
\]

Combine with Gaussian. Picks up “blob-like” structures.
“Blob” detection with Laplacian

Source: S. Lazebnik
Decomposing an image

Gaussian filter
(a.k.a. “low pass”)

Approx. Laplacian
(a.k.a. “high pass”)

Source: Torralba, Freeman, Isola
Application: Hybrid Images

[A. Oliva, A. Torralba, P.G. Schyns, Hybrid Images, SIGGRAPH 2006]
Hybrid Images

Oliva & Schyns
Hybrid Images
Today

- Linear filtering
- More neighborhood filters
- Nonlinear filters
Denoising revisited: salt and pepper noise

Gaussian filter doesn’t work as well!

Source: S. Lazebnik
Blurring

Gaussian filter

Image filters aren’t “aware” of edges

Median filtering

\[ MB[p] = \text{median}_{q \in N} I[q] \]

- **Median filtered result**
- **Other pixels in a window**
- **Intensity of pixel**
Median filtering window sizes

Original
w = 5
w = 13
w = 21
Another approach

Let’s start by rewriting the Gaussian filter:

\[ GB[p] = \sum_{q \in \mathcal{N}} G_{\sigma}(||p - q||) I[q] \]

Gaussian density:

\[ G_{\sigma}(d) = \frac{1}{\sigma \sqrt{2\pi}} \exp \left( -\frac{d^2}{2\sigma} \right) \]
Bilateral filtering

What if we weight by appearance?
Bilateral filtering

Gaussian filter:

$$GB[p] = \sum_{q \in N} G_\sigma(||p - q||)I[q]$$

Bilateral filter:

$$BB[p] = \frac{1}{W_p} \sum_{q \in N} G_\sigma_s(||p - q||)G_\sigma_r(||I[p] - I[q]||)I[q]$$

Normalization constant (to make weights sum to 1)
Bilateral filter


input

spatial weight

range weight

multiplication of range and spatial weights

result
### Bilateral filter

<table>
<thead>
<tr>
<th>$\sigma_s / \sigma_r$</th>
<th>0.05</th>
<th>0.2</th>
<th>0.8</th>
<th>GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>8</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>16</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Denoising with Gaussian

Noisy image

Gaussian blur \( \sigma = 3 \)

Gaussian blur \( \sigma = 5 \)

Source: [Paris et al. 2008]
Denoising with Bilateral Filter

Original

Noisy image

Bilateral filtering

Source: [Paris et al. 2008]
Iteratively applying the filter

1 iteration  
2 iteration  
4 iteration

Source: [Paris et al. 2008]
Iteratively applying the filter

“Cartoonifying” a video

[Winnemoller et al., 2006]
Image patches as filters

• Goal: find in an image

• What’s a good similarity/distance measure between two patches?

Source: D. Hoiem
Matching with filters

Method 1: Filter the image with

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

What went wrong?

Source: D. Hoiem
Matching with filters

Method 2: Filter the image with zero-mean eye.

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

Source: D. Hoiem
Matching with filters

**Method 3:** Normalized cross-correlation.
Divide by standard deviation of both patches, so they are unit vectors.

\[
h[m, n] = \frac{\sum_{k,l}(g[k, l] - \bar{g})(f[m + k, n + l] - \bar{f}_{m,n})}{\sqrt{\sum_{k,l}(g[k, l] - \bar{g})^2 \sum_{k,l}(f[m + k, n + l] - \bar{f}_{m,n})^2}}
\]
Matching with filters

Method 3: Normalized cross-correlation.

Source: D. Hoiem
Recognizing objects: is it really so hard?

Find the chair in this image

Output of normalized correlation

This is a chair

Source: A. Torralba
Recognizing objects: is it really so hard?

Find the chair in this image

Not so great!

Source: A. Torralba
What makes an image “natural”?

Natural image

“Fake” image

Source: Torralba, Freeman, Isola
Is it the distribution of pixel intensities?

No real structure here…
What about gradients?

\[ g[m,n] \times [-1, 1] \rightarrow h[m,n] = f[m,n] \]

Source: Torralba, Freeman, Isola
What about gradients?

\[ g[m,n] \times [-1, 1]^T = h[m,n] = f[m,n] \]
Filter response distribution is pretty consistent!
Image

Intensity histogram

$[1\,-1]$ filter output

$[1\,-1]$ output histogram

Red – true pdf
Black – best Gaussian fit

Source: Torralba, Freeman, Isola
Applications of image statistics

Compression  Image restoration  Learning
(later in course)
Taking a picture...

What the camera give us...  How do we correct this?
Deblurring
Deblurring
Image formation process

Blurry image
Input to algorithm

\[ \cong \]

Sharp image
Desired output

Convolution operator

\[ \times \]
Blur kernel

Source: R. Fergus
Multiple possible solutions

Blurry image = Sharp image = Blur kernel

Source: R. Fergus
Natural image statistics

Characteristic distribution with heavy tails

Histogram of image gradients

Source: R. Fergus
Blurry images have different statistics

Histogram of image gradients

Source: R. Fergus
Removing motion blur

Solve for an image with a distribution of edge gradients that “matches” a normal image.

Source: R. Fergus
Next class: image pyramids