Lecture 2: Images and image filtering
Last time

• Natural images are *not* arbitrary 2D arrays
• They have properties resulting from physics / math of image formation
• Solving computer vision requires using these properties
Last time: Some primitives

- Edge detection: identifying where pixels change color
  - Cue to object boundary
  - Cue to shape
  - More resilient to lighting than pixel color

- Zooming into or out of images
  - Searching for both nearby and far-off objects

- Matching patches from two different images
  - First step in identifying 3D location
Last time: Related problems

- Image Restoration
  - denoising
  - deblurring
- Image Compression
  - JPEG, JPEG2000, MPEG..

- Again, use the same ``priors``
Image denoising
Why would images have noise?

- Sensor noise
  - Sensors count photons: noise in count
- Dead pixels
- Old photographs
- ...


What is an image?

• A grid (matrix) of intensity values: 1 color or 3 colors

(common to use one byte per value: 0 = black, 255 = white)
An assumption about noise

• Let us assume noise at a pixel is
  – independent of other pixels
  – distributed according to a Gaussian distribution
    • i.e., low noise values are more likely than high noise values
    • “grainy images”
Noise reduction

• Nearby pixels are likely to belong to same object
  – thus likely to have similar color
• Replace each pixel by *average of neighbors*
Mean filtering

\[(0 + 0 + 0 + 10 + 40 + 0 + 10 + 0 + 0 + 0)/9 = 6.66\]
Mean filtering

\[
\frac{(0 + 0 + 0 + 0 + 10 + 0 + 0 + 0 + 0 + 0 + 20 + 10 + 40 + 0 + 0 + 20 + 10 + 0 + 0 + 0 + 0 + 0 + 30 + 20 + 10 + 0 + 0)}{25} = 6.8
\]
Mean filtering

\[
\frac{0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 10}{9} = 1.11
\]
Mean filtering

\[
\frac{(0 + 0 + 0 + 0 + 0 + 10 + 0 + 10 + 20)}{9} = 4.44
\]
Mean filtering

\[(0 + 0 + 0 + 0 + 10 + 10 + 10 + 20 + 20)/9 = 7.77\]
Mean filtering

<p>| | | | | | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Noise reduction using mean filtering
Mean filtering

• Replace pixel by mean of neighborhood

\[
S[f](m, n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(m + i, n + j) / 9
\]
A more general version

Local image data

\[
S[f](m, n) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} w(i, j) f(m + i, n + j)
\]
A more general version

Local image data

Kernel size = \(2k+1\)

\[
S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)
\]
Convolution and cross-correlation

- Cross correlation

\[ S[f] = w \otimes f \]
\[ S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j) \]

- Convolution

\[ S[f] = w * f \]
\[ S[f](m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m - i, n - j) \]
Cross-correlation

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

$1 \times 1 + 2 \times 2 + 3 \times 3 + 4 \times 4 + 5 \times 5 + 6 \times 6 + 7 \times 7 + 8 \times 8 + 9 \times 9$
Convolution

\[
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array} 
\quad \quad \quad 
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\end{array}
\]

\[1 \times 9 + 2 \times 8 + 3 \times 7 + 4 \times 6 + 5 \times 5 + 6 \times 4 + 7 \times 3 + 8 \times 2 + 9 \times 1\]
Convolution

Adapted from F. Durand
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f'(m, n) = af(m, n)\]

\[(w \otimes f')(m, n) = a(w \otimes f)(m, n)\]
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f' = af\]

\[(w \otimes f') = a(w \otimes f)\]
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f' = af + bg\]

\[w \otimes f' = a(w \otimes f) + b(w \otimes g)\]
Properties: Linearity

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[w' = aw + bv\]

\[w' \otimes f = a(w \otimes f) + b(v \otimes f)\]
Properties: Shift invariance

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f'(m, n) = f(m - m_0, n - n_0)\]
Shift invariance

\[(w \otimes f)(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i, n + j)\]

\[f'(m, n) = f(m - m_0, n - n_0)\]

\[(w \otimes f')(m, n) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f'(m + i, n + j)\]

\[= \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(m + i - m_0, n + j - n_0)\]

\[= (w \otimes f)(m - m_0, n - n_0)\]
Shift invariance

\[ f'(m, n) = f(m - m_0, n - n_0) \]

\[(w \otimes f')(m, n) = (w \otimes f)(m - m_0, n - n_0)\]

- Shift, then convolve = convolve, then shift
- Output of convolution does not depend on where the pixel is
Filters: examples

Original (f) * \frac{1}{9} \begin{array}{ccc} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{array} = \text{Blur (with a mean filter) (g)}

Source: D. Lowe
Filters: examples

Original (f) * Kernel (k) = Identical image (g)

Source: D. Lowe
Sharpening

before
after

Source: D. Lowe
Sharpening

• What does blurring take away?

Let’s add it back:

\[ \text{original} - \text{detail} + \alpha = \text{original} \]
Sharpening
Sharpening

\[
\begin{align*}
    f_{\text{sharp}} &= f + \alpha(f - f_{\text{blur}}) \\
    &= (1 + \alpha) f - \alpha f_{\text{blur}} \\
    &= (1 + \alpha)(w \ast f) - \alpha(v \ast f) \\
    &= ((1 + \alpha)w - \alpha v) \ast f
\end{align*}
\]
Sharpening filter

Original

* \[ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} - \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \]

Sharpening filter
(accentuates edges)

Source: D. Lowe
Another example

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
</tr>
</tbody>
</table>
Another example
Cross-correlation and dot products

\[ \sum_{i} w_i f_i = \vec{w} \cdot \vec{f} \]

\[ \vec{w} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{bmatrix} \]

\[ \vec{f} = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & f_8 & f_9 \end{bmatrix} \]
Cross-correlation and dot products
Dot products

\[ \vec{w} \cdot \vec{f} = ||\vec{w}|| ||\vec{f}|| \cos \theta \]
Dot products

\[ \vec{w} \cdot \vec{f} = \|\vec{w}\| \|\vec{f}\| \cos \theta \]

- \(\cos \theta\) indicates similarity
- Can measure how much \(f\) “matches” \(w\)
  - Central component of “template matching”
  - But might need to divide by magnitude
  - Cosine distance
- Cross-correlation \(\approx\) template matching
Dot products

\[ \vec{w} \cdot \vec{f} = ||\vec{w}|| ||\vec{f}|| \cos \theta \]

- Usefulness *really* depends on space
- E.g., pixel intensities are often a *bad* space.
- Why?
Cross correlation as template matching

\[(a - b) \otimes\]
Convolution is everywhere.
Why is convolution important?

• Shift invariance is a crucial property
Why is convolution important?

• We *like* linearity
  – Linear functions behave predictably when input changes
  – Lots of theory just easier with linear functions

• *All linear shift-invariant systems can be expressed as a convolution*
Non-linear filters: Thresholding

\[ g(m, n) = \begin{cases} 
255, & f(m, n) > A \\
0, & \text{otherwise} 
\end{cases} \]
Non-linear filters: Rectification

- $g(m,n) = \max(f(m,n), 0)$

- Crucial component of modern convolutional networks
Non-linear filters

- Sometimes mean filtering does not work
Non-linear filters

- Sometimes mean filtering does not work
Non-linear filters

• Mean is sensitive to outliers
• Median filter: Replace pixel by median of neighbors
Non-linear filters
Takeaway

• Two general recipes:
  – convolution
  – cross-correlation

• Properties
  – Shift-invariant: a sensible thing to require
  – Linearity: convenient

• Can be used for smoothing, sharpening

• Also main component of CNNs
Next up

• Back to linear filters
• Signal processing view of filtering
• Filtering for detecting edges etc
Images as functions

• An image contains discrete numbers of pixels

• Pixel value
  – grayscale/intensity
    • [0,255]
  – Color
    • RGB [R, G, B], where [0,255] per channel
Images as functions

• Can think of image as a function, \( f \), from \( \mathbb{R}^2 \) to \( \mathbb{R} \) or \( \mathbb{R}^M \):
  – Grayscale: \( f(x,y) \) gives intensity at position \((x,y)\)
    • \( f: [a,b] \times [c,d] \rightarrow [0,255] \)
  – Color: \( f(x,y) = [r(x,y), g(x,y), b(x,y)] \)

• Most adjacent pixels are correlated \( \Rightarrow \) function is continuous
What is an image?

A digital image is a discrete (sampled, quantized) version of this function.