Reminder

- HW1P, HW2W due tonight, 11:59pm
Plan for today

- Filters: math and properties
- Types of filters
  - Linear
    - Smoothing
    - Other
  - Non-linear
    - Median
- Texture representation with filters
Image filtering

• Compute a function of the local neighborhood at each pixel in the image
  – Function specified by a “filter” or mask saying how to combine values from neighbors
  – Element-wise multiplication

• Uses of filtering:
  – Enhance an image (denoise, resize, etc)
  – Extract information (texture, edges, etc)
  – Detect patterns (template matching)
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

\[ F[x, y] \]

\[ G[x, y] \]
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

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

Say the averaging window size is $2k+1 \times 2k+1$:

$$G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]$$

Attribute uniform weight  
Loop over all pixels in neighborhood around image pixel $F[i,j]$

Now generalize to allow different weights depending on neighboring pixel’s relative position:

$$G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]$$

Non-uniform weights

Kristen Grauman
Correlation filtering

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

This is called **cross-correlation**, denoted \( G = H \otimes F \)

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter “**kernel**” or “**mask**” \( H[u, v] \) is the prescription for the weights in the linear combination.
Averaging filter

• What values belong in the kernel $H$ for the moving average example?

\[ G = H \otimes F \]
Smoothing by averaging

depicts box filter:
white = high value, black = low value

original
filtered

What if the filter size was 5 x 5 instead of 3 x 3?
What is the size of the output?

MATLAB: output size options

- \textit{shape} = ‘full’: output size is larger than the size of \( f \)
- \textit{shape} = ‘same’: output size is same as \( f \)
- \textit{shape} = ‘valid’: output size is difference of sizes of \( f \) and \( h \) [discontinued]

\( f = \text{image} \)

\( h = \text{filter} \)
Boundary issues

• What about near the edge?
  – the filter window might fall off the edge of the image (in ‘same’ or ‘full’)
  – need to extrapolate
  – methods:
    • clip filter (black)
    • wrap around
    • copy edge
    • reflect across edge

Source: S. Marschner
Boundary issues

• What about near the edge?
  – the filter window might fall off the edge of the image (in ‘same’ or ‘full’)
  – 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
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v]
\]

\[G = H \ast F\]

Notation for convolution operator

Kristen Grauman
Convolution vs. correlation

Convolution

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

\[ G = H \ast F \]

Cross-correlation

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

\[ G = H \otimes F \]

For a Gaussian or box filter, how will the outputs differ?
Convolution vs. correlation

**Cross-correlation**

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

\[ G = H \otimes F \]

**Convolution**

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v]
\]

\[ G = H * F \]

\[
F = \begin{pmatrix}
5 & 2 & 5 & 4 & 4 \\
5 & 200 & 3 & 200 & 4 \\
1 & 5 & 5 & 4 & 4 \\
5 & 5 & 1 & 1 & 2 \\
200 & 1 & 3 & 5 & 200 \\
1 & 200 & 200 & 200 & 1
\end{pmatrix}
\]

\[
H = \begin{pmatrix}
.06 & .12 & .06 \\
.12 & .25 & .12 \\
.06 & .12 & .06
\end{pmatrix}
\]
Convolution vs. correlation

**Cross-correlation**

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

\[ G = H \otimes F \]

\[ u = -1, \ v = -1 \]
\[ v = 0 \]

**ConVolution**

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

\[ G = H \ast F \]
Convolution vs. correlation

**Cross-correlation**

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v]
\]

\[
G = H \otimes F
\]

- \(u = -1, v = -1\)
- \(v = 0\)
- \(v = +1\)

**Convolution**

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v]
\]

\[
G = H \ast F
\]
Convolution vs. correlation

Cross-correlation

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

\[ G = H \otimes F \]

Convolution

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

\[ G = H \ast F \]
Convolution vs. correlation

**Cross-correlation**

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

\[ G = H \otimes F \]

**Convolution**

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

\[ G = H \ast F \]
Convolution vs. correlation

Cross-correlation

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \]

\[ G = H \otimes F \]

Convolution

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i - u, j - v] \]

\[ G = H \ast F \]
Convolution vs. correlation

Cross-correlation

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v]
\]

\[
G = H \otimes F
\]

Convolution

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v]
\]

\[
G = H \ast F
\]
Convolution vs. correlation

Cross-correlation

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i + u, j + v] \]

\[ G = H \otimes F \]

Convolution

\[ G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v]F[i - u, j - v] \]

\[ G = H \ast F \]
Properties of convolution

- Commutative:
  \[ f * g = g * f \]

- Associative
  \[ (f * g) * h = f * (g * h) \]

- Distributes over addition
  \[ f * (g + h) = (f * g) + (f * h) \]

- Scalars factor out
  \[ kf * g = f * kg = k(f * g) \]

- Identity:
  \[ \text{unit impulse } e = [\ldots, 0, 0, 1, 0, 0, \ldots]. \ f * e = f \]
Separability

• In some cases, filter is separable, and we can factor into two steps:
  – Convolve all rows
  – Convolve all columns
Separability example

2D filtering (center location only)

The filter factors into an outer product of 1D filters:

Perform filtering along rows:

Followed by filtering along the remaining column:
Plan for today

• Filters: math and properties

• Types of filters
  – Linear
    • Smoothing
    • Other
  – Non-linear
    • Median

• Texture representation with filters
• What if we want nearest neighboring pixels to have the most influence on the output?

This kernel is an approximation of a 2d Gaussian function:

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

Source: S. Seitz
Smoothing with a Gaussian
Gaussian filters

• What parameters matter here?
• **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels

\[
\begin{align*}
\sigma = 5 & \text{ with } 10 \times 10 \text{ kernel} \\
\sigma = 5 & \text{ with } 30 \times 30 \text{ kernel}
\end{align*}
\]
Gaussian filters

• What parameters matter here?
• **Variance** of Gaussian: determines extent of smoothing

\[ \sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \]

\[ \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]
Gaussian filters

How big should the filter be?

• Values at edges should be near zero ← important!
• Rule of thumb for Gaussian: set filter half-width to about 3 $\sigma$

Source: Derek Hoiem
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);

>> mesh(h);

>> imagesc(h);

>> outim = imfilter(im, h); % correlation
>> imshow(outim);
Smoothing with a Gaussian

Parameter $\sigma$ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

```matlab
for sigma=1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```
Properties of smoothing filters

- **Smoothing**
  - Values positive
  - Sum to 1 → overall intensity same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter
Plan for today

• Filters: math and properties
• Types of filters
  – Linear
    • Smoothing
    • Other
  – Non-linear
    • Median
• Texture representation with filters
Predict the outputs using correlation filtering

\[
\begin{array}{c}
\begin{array}{c}
\text{Image} \\
\times \\
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\end{array}
= ?
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{Image} \\
\times \\
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{array}
\end{array}
= ?
\end{array}
\end{array}
\begin{array}{c}
\begin{array}{c}
\text{Image} \\
\times \\
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array}
\end{array}
- \frac{1}{9}
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
= ?
\end{array}
\end{array}
\]
Sharpening with linear filters

Original

Sharpening filter:
accentuates differences with local average

Source: D. Lowe
Sharpening with linear filters

before

after
Application: Hybrid Images

What you see...

I see an angry guy

From Far Away

It's a woman!

Up Close
Application: Hybrid Images

Gaussian Filter

Laplacian Filter (sharpening)


Kristen Grauman
Application: Hybrid Images
Application: Hybrid Images

Changing expression

Sad ← Surprised
Filters for Computing Gradients

\[
\begin{array}{ccc}
1 & 0 & -1 \\
2 & 0 & -2 \\
1 & 0 & -1 \\
\end{array}
\]
Plan for today

• Filters: math and properties
• Types of filters
  – Linear
    • Smoothing
    • Other
  – Non-linear
    • Median
• Texture representation with filters
Median filter

- No new pixel values introduced
- Removes spikes: good for impulse, salt & pepper noise
- Non-linear filter
Median filter

- Median filter is edge preserving

<table>
<thead>
<tr>
<th>INPUT</th>
<th>MEDIAN</th>
<th>MEAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
<tr>
<td>......</td>
<td>......</td>
<td>......</td>
</tr>
</tbody>
</table>
Median filter

Salt and pepper noise

Plots of a row of the image

Matlab: output \texttt{im = medfilt2(im, [h w]);}

Source: M. Hebert
1-minute break
Plan for today

• Filters: math and properties
• Types of filters
  – Linear
    • Smoothing
    • Other
  – Non-linear
    • Median
• Texture representation with filters
Texture

What defines a texture?
Includes: more regular patterns
Includes: more random patterns
Why analyze texture?

• Important for how we perceive objects
• Often indicative of a material’s properties
• Can be important appearance cue, especially if shape is similar across objects
• To represent objects, we want a feature one step above “building blocks” of filters, edges

Adapted from Kristen Grauman
Texture representation

- Textures are made up of repeated local patterns, so:
  - Find the patterns
    - Use filters that look like patterns (spots, bars, raw patches...)
    - Consider magnitude of response
  - Describe their statistics within each local window
    - Mean, standard deviation
    - Histogram
Texture representation: example

original image

derivative filter responses, squared

<table>
<thead>
<tr>
<th></th>
<th>mean ( d/dx ) value</th>
<th>mean ( d/dy ) value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Texture representation: example

original image

derivative filter responses, squared

mean $d/dx$ value  mean $d/dy$ value
Win. #1  4  10
Win. #2  18  7

statistics to summarize patterns in small windows

Kristen Grauman
Texture representation: example

original image

derivative filter responses, squared

<table>
<thead>
<tr>
<th>Window</th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

statistics to summarize patterns in small windows

Kristen Grauman
Texture representation: example

Dimension 1 (mean d/dx value)

Dimension 2 (mean d/dy value)

<table>
<thead>
<tr>
<th>Window</th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

statistics to summarize patterns in small windows
Texture representation: example

Windows with primarily horizontal edges

Windows with small gradient in both directions

Both

Windows with primarily vertical edges

Dimension 1 (mean $d/dx$ value)

Dimension 2 (mean $d/dy$ value)

<table>
<thead>
<tr>
<th></th>
<th>mean $d/dx$ value</th>
<th>mean $d/dy$ value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

statistics to summarize patterns in small windows

Kristen Grauman
Texture representation: example

original image

derivative filter responses, squared

visualization of the assignment to texture “types”
Texture representation: example

Statistics to summarize patterns in small windows

<table>
<thead>
<tr>
<th></th>
<th>Dimension 1 (mean $d/dx$ value)</th>
<th>Dimension 2 (mean $d/dy$ value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win. #1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Win. #2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Win. #9</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Kristen Grauman
Computing distances using texture

\[ D(a, b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2} \]
Texture representation: example

Distance reveals how dissimilar texture from window a is from texture in window b.
Filter banks

• Our previous example used two filters, and resulted in a 2-dimensional feature vector to describe texture in a window.
  – x and y derivatives revealed something about local structure.

• We can generalize to apply a collection of multiple (d) filters: a “filter bank”

• Then our feature vectors will be d-dimensional.
  – still can think of nearness, farness in feature space

Adapted from Kristen Grauman
Computing distances with $d$-dimensional features

$$D(a, b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

Euclidean distance ($L_2$)

Kristen Grauman
Filter banks

- What filters to put in the bank?
  - Typically we want a combination of scales and orientations, different types of patterns.

Matlab code available for these examples:
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html

Kristen Grauman
Filter bank
Multivariate Gaussian

\[ p(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right). \]

\[ \Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix} \]
Kristen Grauman
We can form a “feature vector” from the list of responses at each pixel; gives us a representation of the pixel, image.
You try: Can you match the texture to the response?

Filters

A

B

C

1

2

3

Mean abs responses

Derek Hoiem
Representing texture by mean abs response

Filters

Mean abs responses

Derek Hoiem
Classifying materials, “stuff”

Figure by Varma & Zisserman

Kristen Grauman
Summary

• Filters useful for
  – Enhancing images (smoothing, removing noise), e.g.
    • Box filter (linear)
    • Gaussian filter (linear)
    • Median filter
  – Detecting patterns (e.g. gradients)

• Texture is a useful property that is often indicative of materials, appearance cues
  – Texture representations attempt to summarize repeating patterns of local structure
  – Filter banks useful to measure redundant variety of structures in local neighborhood