Announcement

• Just for today, my office hours will be slightly shifted, 4:30-6pm
Plan for next two lectures

- Filters: math and properties
- Types of filters
  - Linear
    - Smoothing
    - Other
  - Non-linear
    - Median
- Texture representation with filters
- Anti-aliasing for image subsampling
Images in Matlab

• Color images represented as a matrix with multiple channels (=1 if grayscale)
  • Suppose we have a NxM RGB image called “im”
    – \( \text{im}(1,1,1) = \) top-left pixel value in R-channel
    – \( \text{im}(y, x, b) = \) y pixels down (rows), x pixels to right (cols) in \( b^{\text{th}} \) channel
    – \( \text{im}(N, M, 3) = \) bottom-right pixel in B-channel
• \text{imread(filename)} returns a uint8 image (values 0 to 255)
Enter: Noise

• We talked about how the same object will look very different across images
• Even multiple images of the same static scene will not be identical
• How could we reduce the noise, i.e., give an estimate of the true intensities?
• What if there’s only one image?
Common types of noise

– **Salt and pepper noise:** random occurrences of black and white pixels

– **Impulse noise:** random occurrences of white pixels

– **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz
Gaussian noise

\[
\begin{align*}
    f(x, y) &= \hat{f}(x, y) + \eta(x, y) \\
    \eta(x, y) &\sim N(\mu, \sigma)
\end{align*}
\]

\[
\text{>> noise} = \text{randn(size(im))).*sigma;}
\text{>> output} = \text{im + noise;}
\]

What is impact of the sigma?
First attempt at a solution

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

• Assumptions:
  – Expect pixels to be like their neighbors
  – Expect noise processes to be independent from pixel to pixel
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* \([1, 1, 1, 1, 1] / 5\)
Weighted Moving Average

• Non-uniform weights \([1, 4, 6, 4, 1]/16\)

Central pixel =
\[10*1 + 14*4 + 12*6 + 13*4 + 20*1\]

Adapted from S. Marschner
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
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]
Moving Average In 2D

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

Source: S. Seitz
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)
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 to each pixel 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.
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.
Convolution

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

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

\[ G = H \star F \]

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?

Kristen Grauman
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**

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

\[ G = H \otimes F \]

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

**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**

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

\[ G = H \otimes F \]

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

**Convolution**

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

\[ G = H \ast F \]

\[ (i, j) \]

\[ (0, 0) \]
**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

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

\[
G = H \otimes F
\]

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

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

\[
G = H \otimes F
\]

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

#### Convolution

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

\[
G = H * F
\]
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

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

Kristen Grauman
Gaussian filter

- 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:

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

- Removes high-frequency components from the image ("low-pass filter").

Source: S. Seitz
Smoothing with a Gaussian

Vs box filter
Gaussian filters

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

\[ \sigma = 5 \text{ with 10 x 10 kernel} \]

\[ \sigma = 5 \text{ with 30 x 30 kernel} \]
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
Gaussian filter in Matlab

```matlab
>> 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.

for $\sigma=1:3:10$
    $h = \text{fspecial('gaussian', fsize, sigma)}$;
    $\text{out} = \text{imfilter}(\text{im}, h)$;
    $\text{imshow}(\text{out})$;
    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
Predict the outputs using correlation filtering

\[ \begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \ast \begin{bmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{bmatrix} = ? \]

\[ \begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} \ast \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix} - \frac{1}{9} = ? \]
Practice with linear filters

Original

![Filter Matrix]

0 0 0
0 1 0
0 0 0

Source: D. Lowe
Practice with linear filters

Original

Filtered (no change)

Source: D. Lowe
Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

Shifted left by 1 pixel with correlation

Source: D. Lowe
Practice with linear filters

Original

$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Source: D. Lowe
Practice with linear filters

Original

Blur (with a box filter)

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array}
- \frac{1}{9} \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Practice with linear filters

Original

Sharpening filter:
accentuates differences with local average

Source: D. Lowe
Sharpening

before

after

Kristen Grauman
Filters for computing gradients

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

intensity image

* =

Slide credit: Derek Hoiem
Median filter

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

Kristen Grauman
Median filter

- Median filter is edge preserving
Median filter

Salt and pepper noise

Median filtered

Plots of a row of the image

Matlab: output_im = medfilt2(im, [h w]);

Source: M. Hebert
What is the size of the output?

MATLAB: output size options

- `shape = 'full'`: output size is larger than the size of $f$
- `shape = 'same'`: output size is same as $f$
- `shape = 'valid'`: output size is difference of sizes of $f$ and $h$ [discontinued]

Adapted from S. Lazebnik
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
Properties of convolution

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

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

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

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

- Identity:
  unit impulse \( e = [..., 0, 0, 1, 0, 0, ...] \). \( f \ast e = f \)
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:
Application: Hybrid Images

What you see...

I see an angry guy

From Far Away

Up Close

It's a woman!
Application: Hybrid Images


Gaussian Filter

Laplacian Filter (sharpening)

unit impulse

Gaussian

Laplacian of Gaussian
Application: Hybrid Images
Application: Hybrid Images

Changing expression

Sad  Surprised

Kristen Grauman

Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006
Plan for next two lectures

- Filters: math and properties
- Types of filters
  - Linear
    - Smoothing
    - Other
  - Non-linear
    - Median
- Texture representation with filters
- Anti-aliasing for image subsampling
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
    • E.g. mean, standard deviation
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

Statistics to summarize patterns in small windows:

<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></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Texture representation: example

original image

derivative filter responses, squared

<table>
<thead>
<tr>
<th>Win.</th>
<th>mean d/dx value</th>
<th>mean d/dy value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>#2</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>#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 \( \frac{d}{dx} \) value) vs. Dimension 2 (mean \( \frac{d}{dy} \) value)

<table>
<thead>
<tr>
<th>Window</th>
<th>( \frac{d}{dx} ) Value</th>
<th>( \frac{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

Windows with primarily horizontal edges

Windows with small gradient in both directions

Windows with primarily vertical edges

Both

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>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>

Far: dissimilar textures
Close: similar textures
Computing distances using texture

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

\[ = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \]

Euclidean distance \((L_2)\)

Kristen Grauman
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.
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
Filter bank
Multivariate Gaussian

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

\[
\Sigma = \begin{bmatrix} 9 & 0 \\ 0 & 9 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 16 & 0 \\ 0 & 9 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 10 & 5 \\ 5 & 5 \end{bmatrix}
\]
Showing magnitude of responses

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

Mean abs responses

Derek Hoiem
Representing texture by mean abs response
Classifying materials, “stuff”

Figure by Varma & Zisserman
Plan for next two lectures

• Filters: math and properties
• Types of filters
  – Linear
    • Smoothing
    • Other
  – Non-linear
    • Median
• Texture representation with filters
• Anti-aliasing for image subsampling
Why does a lower resolution image still make sense to us? What do we lose?
Throw away every other row and column to create a 1/2 size image.
Aliasing problem

• 1D example (sinewave):

Source: S. Marschner
Aliasing problem

• 1D example (sinewave):
Aliasing problem

• Sub-sampling may be dangerous....

• Characteristic errors may appear:
  – “Wagon wheels rolling the wrong way in movies”
  – “Striped shirts look funny on color television”
Sampling and aliasing

256x256  128x128  64x64  32x32  16x16
Nyquist-Shannon Sampling Theorem

• When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{\text{max}}$
• $f_{\text{max}} = \text{max frequency of the input signal}$
• This will allow to reconstruct the original perfectly from the sampled version
Anti-aliasing

Solutions:

• Sample more often

• Get rid of all frequencies that are greater than half the new sampling frequency
  – Will lose information
  – But it’s better than aliasing
  – Apply a smoothing filter
Algorithm for downsampling by factor of 2

1. Start with image(h, w)
2. Apply low-pass filter
   \[
   \text{im\_blur} = \text{imfilter}(\text{image}, \text{fspecial(‘gaussian’, 7, 1)})
   \]
3. Sample every other pixel
   \[
   \text{im\_small} = \text{im\_blur}(1:2:end, 1:2:end);
   \]
Anti-aliasing

Forsyth and Ponce 2002
Subsampling without pre-filtering

1/2

1/4 (2x zoom)

1/8 (4x zoom)
Subsampling with Gaussian pre-filtering

Gaussian 1/2

G 1/4

G 1/8

Slide by Steve Seitz
Subsampling away...

Why would we want to do this?
Can we reconstruct the original from the Laplacian pyramid?
Gaussian pyramid

| 512 | 256 | 128 | 64 | 32 | 16 | 8 |

Source: Forsyth
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

Can use filtering to reduce the effects of subsampling