Announcements

- **Office hours** Mon-Thurs 5-6 pm
  - Mon: Yong Jae, PAI 5.33
  - Tues/Thurs: Shalini, PAI 5.33
  - Wed: Me, ACES 3.446

- [cv-spring2011@cs.utexas.edu](mailto:cv-spring2011@cs.utexas.edu) for assignment questions outside of office hours

- **Pset 0** due Friday Jan 28. Drop box in PAI 5.38. Attach cover page with name and CS 376

---

Plan for today

- Image noise
- Linear filters
  - Examples: smoothing filters
- Convolution / correlation

---

Image Formation

- Digital camera

A digital camera replaces film with a sensor array
- Each cell in the array is light-sensitive diode that converts photons to electrons
Digital images

Sample the 2D space on a regular grid
Quantize each sample (round to nearest integer)
Image thus represented as a matrix of integer values.

Digital color images

Color images, RGB color space

Images in Matlab

- Images represented as a matrix
- Suppose we have a N x M RGB image called “im”
  - im(1, 1) = top-left pixel value in R-channel
  - im(y, x, b) = y pixels down, x pixels to right in the bth channel
- imread(filename) returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with im2double

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.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)
Motivation: noise reduction
• Even multiple images of the same static scene will not be identical.

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

Original
Salt and pepper noise
Impulse noise
Gaussian noise
Source: S. Seitz

Gaussian noise

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 \[ [1, 1, 1, 1, 1] / 5 \]

Source: S. Marschner

Weighted Moving Average

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

Source: S. Marschner

Moving Average In 2D

\[
\begin{array}{c|cccc}
F[x, y] & & & & \\
\hline
& & & & \\
G[x, y] & & & & \\
\end{array}
\]

Source: S. Seitz

Moving Average In 2D

\[
\begin{array}{c|cccc}
F[x, y] & & & & \\
\hline
& & & & \\
G[x, y] & & & & \\
\end{array}
\]

Source: S. Seitz
Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:
\[ G[i, j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v] \]

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

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

What if the filter size was 5 x 5 instead of 3 x 3?
Boundary issues
What is the size of the output?
• MATLAB: output size / “shape” options
  • shape = ‘full’: output size is sum of sizes of f and g
  • shape = ‘same’: output size is same as f
  • shape = ‘valid’: output size is difference of sizes of f and g

Boundary issues
What about near the edge?
• the filter window falls off the edge of the image
• need to extrapolate
• methods:
  – clip filter (black): imfilter(f, g, 0)
  – wrap around: imfilter(f, g, ‘circular’)
  – copy edge: imfilter(f, g, ‘replicate’)
  – reflect across edge: imfilter(f, g, ‘symmetric’)

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:
  \( h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}} \)
  \( F[x, y] \)
  \( H[u, v] \)
• Removes high-frequency components from the
  image (“low-pass filter”).

Gaussian filters
• What parameters matter here?
• Size of kernel or mask
  – Note, Gaussian function has infinite support, but discrete
  filters use finite kernels
  \( \sigma = 5 \) with
  10 x 10 kernel
  \( \sigma = 5 \) with
  30 x 30 kernel
Gaussian filters

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

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

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

Matlab

\[ >> \text{hsize} = 10; \]
\[ >> \text{sigma} = 5; \]
\[ >> \text{h} = \text{fspecial('gaussian', hsize, sigma);} \]

\[ >> \text{mesh}(\text{h}); \]
\[ >> \text{imagesc}(\text{h}); \]

\[ >> \text{outim} = \text{imfilter(\text{im}, \text{h});} \text{ % correlation} \]
\[ >> \text{imshow(outim);} \]

Smoothing with a Gaussian

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

\[ \text{for sigma} = 1:3:10 \]
\[ \quad \text{h} = \text{fspecial('gaussian', fsize, sigma);} \]
\[ \quad \text{out} = \text{imfilter(\text{im}, \text{h});} \]
\[ \quad \text{imshow(out);} \]
\[ \quad \text{pause;} \]
\[ \text{end} \]

Properties of smoothing filters

- **Smoothing**
  - Values positive
  - Sum to 1 \( \rightarrow \) constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter

Filtering an impulse signal

What is the result of filtering the impulse signal (image) \( F \) with the arbitrary kernel \( H \)?

\[ F[x, y] \]

\[ H[u, v] \]

\[ G[x, y] \]

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
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?  If the input is an impulse signal, how will the outputs differ?

#### Practice with linear filters

**Original**

0 0 0
0 1 0
0 0 0

**Predict the outputs using correlation filtering**

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

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

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

**Original**

0 0 0
0 1 0
0 0 0

**Filtered (no change)**

0 0 0
0 0 0
0 0 0

**Original**

0 0 0
0 0 1
0 0 0

**Shifted left by 1 pixel with correlation**

0 0 0
0 0 1
0 0 0

Source: D. Lowe
Practice with linear filters

Original

Blur (with a box filter)

Original

Blur (with a box filter)

Original

Blur (with a box filter)

Original

Sharpening filter: accentuates differences with local average

Filtering examples: sharpening

before

after

Properties of convolution

• **Shift invariant:**
  - Operator behaves the same everywhere, i.e. the value of the output depends on the pattern in the image neighborhood, not the position of the neighborhood.

• **Superposition:**
  - \( h \ast (f_1 + f_2) = (h \ast f_1) + (h \ast f_2) \)
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:
  unit impulse \( e = [..., 0, 0, 1, 0, 0, ...] \). \( f * e = f \)

Separability

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

Effect of smoothing filters

5x5

Additive Gaussian noise
Salt and pepper noise

Median filter

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

Median filter

Plots of a row of the image
Matlab: output \( im = medfilt2(im, [h w]) \):

Source: M. Hebert
Median filter

- Median filter is edge preserving

<table>
<thead>
<tr>
<th>INPUT</th>
<th>MEDIAN</th>
<th>MEAN</th>
</tr>
</thead>
</table>

Filtering application: Hybrid Images

Aude Oliva, Antonio Torralba, Philippe G. Schyns, SIGGRAPH 2006

Application: Hybrid Images

Gaussian Filter

Laplacian Filter


Summary

- Image “noise”
- Linear filters and convolution useful for
  - Enhancing images (smoothing, removing noise)
    - Box filter
    - Gaussian filter
    - Impact of scale / width of smoothing filter
  - Detecting features (next time)
- Separable filters more efficient
- Median filter: a non-linear filter, edge-preserving
Coming up

• **Wednesday:**
  – Filtering part 2: filtering for features

• **Friday:**
  – Pset 0 is due via turnin, 11:59 PM