Announcements

• PS0 out today; due 4/14 Friday at 11:59 pm

• Carefully read course website

• Sign-up for piazza

Plan for today

• Image formation
• Image noise
• Linear filters
  – Examples: smoothing filters
• Convolution / correlation
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 images

CMOS sensor

Digital color images

Bayer filter

Color Sensing: Bayer Grid

Estimate RGB at each cell from neighboring values

http://en.wikipedia.org/wiki/Bayer_filter

Slide by Steve Seitz
4/6/2017

Digital color images

Color images, RGB color space

Images in Matlab

• Images represented as a matrix
• Suppose we have an NxM RGB image called “im”
  – im(1,1) = top-left pixel value in R-channel
  – im(i, j, b) = i pixels down, j pixels to right in the bth channel
  – im(N, M, 3) = bottom-right pixel in B-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, increase contrast, etc)
  – Extract information (texture, edges, interest points, 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

Gaussian noise

$\text{output} = \text{im} + \text{noise}$

What is impact of the sigma?
Motivation: noise reduction

• 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?

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

Moving average in 1D:
Weighted Moving Average
Can add weights to our moving average
Weights \([1, 1, 1, 1, 1] / 5\)

Weighted Moving Average
Non-uniform weights \([1, 4, 6, 4, 1] / 16\)

Moving Average In 2D
\[ f[x, y] \quad g[x, y] \]
Moving Average In 2D

\[ f[x, y] \quad g[x, y] \]

Slide credit: Steve Seitz
Correlation filtering

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

$$g(i,j) = \frac{1}{(2k+1)^2} \sum_{k} \sum_{b} f(i+k,j+b)$$

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:

$$g(i,j) = \sum_{k} \sum_{b} h(u,v) f(i+k,j+b)$$

Non-uniform weights

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?

$$f[x, y] \otimes h[u, v] = g[x, y]$$

$g = h \otimes f$

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?

Boundary issues

What about near the edge?

- The filter window falls off the edge of the image
- Need to extrapolate
- Methods:
  - Clip filter (black)
  - Wrap around
  - Copy edge
  - Reflect across edge
Boundary issues
What about near the edge?
• the filter window falls off the edge of the image
• 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')}

Boundary issues
What is the size of the output?
• MATLAB: output size / "shape" options
  • \texttt{shape = 'full'}: output size is sum of sizes of \(f\) and \(g\)
  • \texttt{shape = 'same'}: output size is same as \(f\)
  • \texttt{shape = 'valid'}: output size is difference of sizes of \(f\) and \(g\)

Gaussian filter
• What if we want nearest neighboring pixels to have the most influence on the output?

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

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

\[ f[x, y] \]

• Removes high-frequency components from the image ("low-pass filter").
Smoothing with a Gaussian

Smoothing with a 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

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

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

\[ \sigma = 2 \text{ with } 30 \times 30 \text{ kernel} \]
\[ \sigma = 5 \text{ with } 30 \times 30 \text{ kernel} \]

---

Matlab

```
>> hsize = 30;
>> 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 = fspecial('gaussian', hsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```
Properties of smoothing filters

- **Smoothing**
  - Values positive
  - Sum to 1 → 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 \( h \)?

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\otimes
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\Rightarrow
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

Convolution

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

\[
g(i,j) = \sum_{u=-b}^{b} \sum_{v=-b}^{b} h(u,v) f(i-u, j-v)
\]

\[
g = h \ast f
\]
Convolution vs. correlation

**Convolution**

\[ g(i, j) = \sum_{v=-k}^{k} \sum_{w=-k}^{k} h(v, w) f(i - w, j - v) \]

\[ g = h \ast f \]

**Cross-correlation**

\[ g(i, j) = \sum_{v=-k}^{k} \sum_{w=-k}^{k} h(v, w) f(i + w, j + v) \]

\[ g = h \circ f \]

For a Gaussian or box filter, how will the outputs differ?

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

---

**Predict the outputs using correlation filtering**

**Original**

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

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

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

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

**Practice with linear filters**

**Original**

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

\[
\begin{array}{ccc}
? & ? & ? \\
? & ? & ? \\
? & ? & ? \\
\end{array}
\]
Practice with linear filters

Original

Filtered (no change)

Original

Shifted left by 1 pixel with correlation

Slide credit: David Lowe
Practice with linear filters

Original

1 1 1
1 1 1
1 1 1

? 

Blur (with a box filter)

Original

1 1 1
1 1 1
1 1 1

Practice with linear filters

Original

0 0 0
0 2 0
0 0 0

1 1 1
1 1 1
1 1 1

? 

Practice with linear filters

Original

0 0 0
0 2 0
0 0 0

1 1 1
1 1 1
1 1 1

? 

Slide credit: David Lowe
Practice with linear filters

<table>
<thead>
<tr>
<th>Original</th>
<th>Sharpening filter: accentuates differences with local average</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>0 2 0</td>
<td>1 1 1</td>
</tr>
<tr>
<td>0 0 0</td>
<td>1 1 1</td>
</tr>
</tbody>
</table>

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 \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \). \( f * e = f \)

Separability

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

What is the computational complexity advantage for a separable filter of size \( k \times k \), in terms of number of operations per output pixel?
Effect of smoothing filters

Gaussian noise
Salt and pepper noise

3x3
5x5
7x7

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

Matlab: output im = medfilt2(im, [h w]);
Median filter

- Median filter is edge preserving

Filtering application: Hybrid Images

Application: Hybrid Images
Summary

• Image formation
• 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

• Next Friday (4/14):
  – PS0 is due via Canvas, 11:59 PM

• Tuesday:
  – Filtering part 2: filtering for features

Questions?

See you Tuesday!