Plan for today

- Image noise
- Linear filters
  - Smoothing filters
- Convolution / correlation

Images as functions

- We can think of an image as a function, $f$, from $\mathbb{R}^2$ to $\mathbb{R}$:
  - $f(x,y)$ gives the intensity at position $(x,y)$
  - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    - $f: [a,b] \times [c,d] \rightarrow [0, 1]$.
  - A color image is just three functions pasted together. We can write this as a "vector-valued" function:
    $$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Digital images

- In computer vision we operate on digital (discrete) 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.

Images as matrices

- Intensity: $[0,255]$
Images as matrices
Result of averaging 100 similar snapshots

From: *100 Special Moments*, by Jason Salavon (2004)
http://salavon.com/SpecialMoments/SpecialMoments.shtml

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

Effect of sigma on Gaussian noise:
Image shows the noise values themselves.
Effect of sigma on Gaussian noise:
Image shows the noise values themselves.

Effect of sigma on Gaussian noise:
This shows the noise values added to the raw intensities of an image.

Effect of sigma on Gaussian noise
This shows the noise values added to the raw intensities of an image.

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

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

Source: S. Seitz

Moving Average In 2D

Source: S. Seitz

Moving Average In 2D

Source: S. Seitz

Moving Average In 2D

Source: S. Seitz
Correlation filtering

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

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

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

Attribute uniform weight to each pixel.

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

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
Gaussian filter

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

This kernel is an approximation of a Gaussian function:

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

Gaussian filters

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

\[ \sigma = 5 \]

Gaussian filters

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

\[ \sigma = 2 \]

Matlab

```matlab
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);

>> mesh(h);
>> imagesc(h);
>> outim = imfilter(im, h);
>> imshow(outim);
```
Boundary issues

What is the size of the output?
- MATLAB: `filter2(g, f, shape)`
  - `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`

![Diagram showing output sizes 'full', 'same', and 'valid'](source: S. Lazebnik)

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')`

![Diagram showing boundary methods](source: S. Marschner)

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): `imfilter(f, g, 0)`
  - wrap around: `imfilter(f, g, 'circular')`
  - copy edge: `imfilter(f, g, 'replicate')`
  - reflect across edge: `imfilter(f, g, 'symmetric')`

![Diagram showing boundary methods](source: S. Marschner)

Filtering an impulse signal

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

![Diagram showing filtering an impulse signal](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

![Diagram showing convolution](source: S. Marschner)

Convolution vs. correlation

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

\[ G = H \ast F \]

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?
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 $\rightarrow$ constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove “high-frequency” components; “low-pass” filter

Predict the filtered outputs

Practice with linear filters
Practice with linear filters

Original

0 0 0
0 0 1
0 0 0

Shifted left by 1 pixel with correlation

Source: D. Lowe

Original

1 1 1
1 1 1
1 1 1

Original

1 9

Blur (with a box filter)

Source: D. Lowe

Original

0 0 0
0 2 0
0 0 0

Blur (with a box filter)

Source: D. Lowe

Original

0 0 0
1 1 1
1 1 1

Blur (with a box filter)

Source: D. Lowe

Original

0 0 0
0 2 0
0 0 0

Blur (with a box filter)

Source: D. Lowe

Filtering examples: sharpening

Original

0 0 0
0 2 0
0 0 0

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe

before

after
Shift invariant linear system

- **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.
- **Linear:**
  - Superposition: \( h \ast (f_1 + f_2) = (h \ast f_1) + (h \ast f_2) \)
  - Scaling: \( h \ast (k \cdot f) = k \cdot (h \ast f) \)

Properties of convolution

- Linear & shift invariant
- Commutative: \( f \ast g = g \ast f \)
- Associative
  \[ (f \ast g) \ast h = f \ast (g \ast h) \]
- Identity:
  \[ \text{unit impulse } e = [..., 0, 1, 0, 0, ...]. f \ast e = f \]
- Differentiation:
  \[ \frac{d}{dx} (f \ast g) = \frac{d}{dx} \cdot g \]

Separability

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

Separability

- In some cases, filter is separable, and we can factor into two steps: e.g.,

\[
\begin{array}{c|c|c|c}
& 1 & 2 & 1 \\
\hline
2 & 3 & 2 \\
3 & 5 & 3 \\
4 & 6 & 4 \\
\end{array}
\]

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

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
- Linear?
**Summary**

- Various models for image “noise”
- Linear filters and convolution useful for
  - Image 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**

- **Tomorrow (Wed):** my office hours cancelled
- **TA’s available as usual**
- **Thursday:**
  - Matlab tutorial, with guest lecture by Yong Jae
  - Bring questions about Pset 0
- **Monday:**
  - Pset 0 is due, 11:59 PM
- **Tuesday:**
  - Lecture: Linear filters, part 2
  - See course page for reading
  - Pset 1 out