Linear Filters
Tues, Jan 23, 2018

Announcements

- Piazza for assignment questions
- A0 due today. Submit on Canvas.
- Office hours posted on class website

Course homepage

Plan for today

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

Images as matrices

Result of averaging 100 similar snapshots

From: 100 Special Moments, by Jason Salavon (2004)
http://salavon.com/SpecialMoments/SpecialMoments.shtml
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:
    \[ f: [a, b] \times [c, d] \rightarrow [0, 255] \]

- 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 camera

A digital camera replaces film with a sensor array
- Each cell in the array is a 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 b-th 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
Main idea: 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
Gaussian noise

\[
\text{New Image} = \text{Old Image} + \text{Gaussian noise with mean 0 and standard deviation } \sigma
\]

What is impact of the sigma?

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

\[
\text{New Image} = \text{Old Image} + \text{Gaussian noise with mean 0 and standard deviation } \sigma
\]

\[
\text{New Image} = \text{Old Image} + \text{Gaussian noise with mean 0 and standard deviation } \sigma
\]
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:

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.
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

\[ F[x, y] \]

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

$F[x, y]$  

$G[x, y]$  

Source: S. Seitz
Moving Average In 2D

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

Source: S. Seitz

---

Moving Average In 2D

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

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
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?

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

\textbf{"box filter"}

\[ G = H \otimes F \]

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)
  - wrap around
  - copy edge
  - reflect across edge

Source: S. Lazebnik

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')

Source: S. Marschner
Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?
- Removes high-frequency components from the image ("low-pass filter").

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

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

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

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

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

![Gaussian filters diagram](image)

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

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

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 = 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

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?

Predict the outputs using correlation filtering
Practice with linear filters

Original

?  

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

Blur (with a box filter)

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\cdot
\frac{1}{9}
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
= ?
\]

Source: D. Lowe

Practice with linear filters

Original

Sharpening filter: accentuates differences with local average

Filtering examples: sharpening

Before

After
Convolution

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

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

\[
G = H \ast F
\]

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:
  - \( f \ast g = g \ast f \)

- 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 \( e = \ldots, 0, 1, 0, 0, \ldots \). \( f \ast e = f \)
Separability

- In some cases, filter is separable, and we can factor into two steps:
  - Convolve all rows with a 1D filter
  - Convolve all columns with a 1D filter

Separability

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

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

Effect of smoothing filters

- 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

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

Median filter

- Median filter is edge preserving
Filtering application: Hybrid Images

Application: Hybrid Images

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

- **Thursday:**
  - Filtering part 2: filtering for features (edges, gradients, seam carving application)
  - See reading assignment on webpage

- **Today:**
  - Assignment 0 is due on Canvas 11:59 PM