Linear Filters
Thurs Jan 19, 2017

Announcements

• Piazza for assignment questions

• A0 due Friday Jan 27. Submit on Canvas.
Course homepage


<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Textbook</th>
<th>Slides</th>
<th>Additional Reading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tues Jan 17</td>
<td>Course intro</td>
<td>Sec 1.1-1.3</td>
<td></td>
<td>UTCI account setup, Basic Matlab tutorial, Running Matlab at UT</td>
</tr>
<tr>
<td>Thurs Jan 19</td>
<td>Features and filters</td>
<td>Sec 3.1-1.2, 3.2</td>
<td>Linear filters</td>
<td></td>
</tr>
<tr>
<td>Tues Jan 24</td>
<td></td>
<td>Sec 3.2-3, 4.2</td>
<td>Gradients and</td>
<td>AoS due Friday Jan 27</td>
</tr>
</tbody>
</table>
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
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.

Adapted from S. Seitz
Digital color images

Bayer filter

Color images, RGB color space

R  G  B
Images in Matlab

- Images represented as a matrix
- Suppose we have a N×M RGB image called “im”
  - \text{im}(1, 1) = \text{top-left pixel value in R-channel}
  - \text{im}(y, x, b) = y \text{ pixels down, } x \text{ pixels to right in the } b^{th} \text{ channel}
  - \text{im}(N, M, 3) = \text{bottom-right pixel in B-channel}
- \text{imread(filename)} returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with \text{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

Source: S. Seitz
Gaussian noise

\[ f(x,y) = f(x,y) + \eta(x,y) \]

Gaussian i.i.d. ("white") noise:
\[ \eta(x,y) \sim N(\mu,\sigma) \]

\>
noise = randn(size(im)).*sigma;
\>
output = im + noise;

What is impact of the 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.

\[ \text{sigma} = 16 \]

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:

![Original and smoothed data](source: S. Marschner)

Weighted Moving Average

Can add weights to our moving average

*Weights* \([1, 1, 1, 1, 1] / 5\)

![Weighted moving average graph](source: S. Marschner)
Weighted Moving Average

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

Source: S. Marschner

Moving Average In 2D

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

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \]

\[ G[x, y] \]

Source: S. Seitz

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

\[
G[i, j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v] \\
\text{Attribute uniform weight to each pixel} \\
\text{Loop over all pixels in neighborhood around image pixel } F[i, j]
\]

Now generalize to allow different weights depending on neighboring pixel’s relative position:

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v] \\
\text{Non-uniform weights}
\]
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?

\[ \begin{array}{cccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

“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

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?

\[ H[u, v] \]

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

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

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

Source: S. Seitz
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

\[ \sigma = 5 \] with

- 10 x 10 kernel
- 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

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

>> mesh(h);
>> imagesc(h);

>> outim = imfilter(im, h); % correlation
>> imshow(outim);
```

outim
Smoothing with a Gaussian

Parameter σ 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
```

Keeping the two Gaussians in play straight...

More noise →

Wider smoothing kernel →
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

Predict the outputs using correlation filtering:

```
0 0 0
0 1 0
0 0 0
```

```
0 0 0
0 0 1
0 0 0
```

```
0 0 0
0 2 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```

```
0 0 0
0 1 0
0 0 0
```

```
1 1 1
1 1 1
1 1 1
```
Convolutions

- **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

**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:**
  - \( h \ast (f1 + f2) = (h \ast f1) + (h \ast f2) \)
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 with a 1D filter
  - Convolve all columns with a 1D filter

\[
\begin{bmatrix}
1 & 1 & 1 \\
9 & 9 & 9 \\
1 & 1 & 1 \\
9 & 9 & 9 \\
1 & 1 & 1 \\
9 & 9 & 9
\end{bmatrix}
\begin{bmatrix}
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0 \\
0 & 1 & 0
\end{bmatrix}
= 
\begin{bmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
3 & 3 & 3 \\
0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 \\
1 & 1 & 1 \\
3 & 3 & 3 \\
0 & 0 & 0
\end{bmatrix}
\]
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

- Median filter is edge preserving

Plots of a row of the image

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

Source: M. Hebert
Filtering application: Hybrid Images

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

Application: Hybrid Images

Changing expression

Sad  Surprised

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

• Tuesday:
  – Filtering part 2: filtering for features (edges, gradients, seam carving application)
  – See reading assignment on webpage

• Next Friday:
  – Assignment 0 is due on Canvas 11:59 PM