Linear Filters
April 7th, 2020

Yong Jae Lee
UC Davis

Announcements

- PS0 due 4/10 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
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 images

CMOS sensor

Semiconductor that records light electronically
Each sensor cell records amount of light coming in

Digital color images

Bayer filter

Color Sensing: Bayer Grid

Estimate RGB at each cell from neighboring values

http://en.wikipedia.org/wiki/Bayer_filter
Digital color images

Images in Matlab

- Images represented as a matrix
- Suppose we have an NxM RGB image called “im”
  - \(\text{im}(1,1)\) = top-left pixel value in R-channel
  - \(\text{im}(y, x, b)\) = \(y\) pixels down, \(x\) pixels to right in the \(b\)th channel
  - \(\text{im}(N, M, 3)\) = 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}

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

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

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](image1.png)
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

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

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

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

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

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

Attribute uniform weight to each pixel. 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:

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

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?

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

\[
g = h \otimes f
\]

Smoothing by averaging

- 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')}

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 (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')}

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 \( g \)

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:

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

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:

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

\[
\sigma = 5 \text{ with } 10 \times 10 \text{ kernel} \quad \text{and} \quad \sigma = 5 \text{ with } 30 \times 30 \text{ kernel}
\]
Gaussian filters

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

$\sigma = 2$ with 30 x 30 kernel
$\sigma = 5$ with 30 x 30 kernel

Matlab

```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.

```matlab
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 \( h \)?

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

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Convolution

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

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

\[
g = h \ast f
\]

Notation for convolution operator

![Convolution Diagram](attachment://convolution_diagram.png)
Convolution vs. correlation

Convolution

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

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
Practice with linear filters

Original

Filtered (no change)

Practice with linear filters

Original

Practice with linear filters

Original

Shifted left by 1 pixel with correlation
Practice with linear filters

Original

 Blur (with a box filter)

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></td>
<td>0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 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:
  \[ \text{unit impulse } e = [\ldots, 0, 0, 1, 0, 0, \ldots]. \quad f * e = f \]

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.,

\[ g = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 5 & 3 \\ 4 & 4 & 6 \end{bmatrix} \quad h \]

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

- 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

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

- Median filter is edge preserving

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

---

Filtering application: Hybrid Images

Application: Hybrid Images

A. Oliva, A. Torralba, P.G. Schyns,
"Hybrid Images," SIGGRAPH 2006
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

- Thursday:
  - Filtering part 2: filtering for features

Questions?

See you Thursday!