Image filtering, image operations

Jana Kosecka

- photometric aspects of image formation
- gray level images
- point-wise operations
- linear filtering

Image

Brightness values

$I(x,y)$
Images

- Images contain noise - sources sensor quality, light fluctuations, quantization effects

Filtering and Image Features

Given a noisy image

How do we reduce noise?
How do we find useful features?

Today:
- Filtering
- Point-wise operation
- Edge detection
**Motivation: Image denoising**

- How can we reduce noise in a photograph?

**Moving average**

- Let’s replace each pixel with a weighted average of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for the average of a 3x3 neighborhood?

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

“box filter”

*Source: D. Lowe*
Defining convolution

- Let \( f \) be the image and \( g \) be the kernel. The output of convolving \( f \) with \( g \) is denoted \( f \ast g \).

\[
(f \ast g)[m,n] = \sum_{k,l} f[m-k, n-l] g[k,l]
\]

Convention:
kernel is “flipped”

\[ f \]

MATLAB functions: \texttt{conv2, filter2, imfilter}

Annoying details

- What is the size of the output?
- MATLAB: \texttt{filter2(g, f, shape)}
  - \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 \( g \)
Key properties

- **Linearity:** \( \text{filter}(f_1 + f_2) = \text{filter}(f_1) + \text{filter}(f_2) \)
- **Shift invariance:** same behavior regardless of pixel location: \( \text{filter}(\text{shift}(f)) = \text{shift}(\text{filter}(f)) \)
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Properties in more detail

- **Commutative:** \( a \ast b = b \ast a \)
  - Conceptually no difference between filter and signal
- **Associative:** \( a \ast (b \ast c) = (a \ast b) \ast c \)
  - Often apply several filters one after another: \( (((a \ast b_1) \ast b_2) \ast b_3) \)
    - This is equivalent to applying one filter: \( a \ast (b_1 \ast b_2 \ast b_3) \)
- Distributes over addition: \( a \ast (b + c) = (a \ast b) + (a \ast c) \)
- Scalars factor out: \( ka \ast b = a \ast kb = k(a \ast b) \)
- **Identity:** unit impulse \( e = [\ldots, 0, 0, 1, 0, 0, \ldots] \), \( a \ast e = a \)
Averaging filter

Original image

Smoothed image

Convolution in 2D

\[ g[x, y] = \sum_{k=-w/2}^{w/2} \sum_{l=-w/2}^{w/2} f[k, l] h[x-k, y-l] \]

\[
\begin{array}{cccc}
10 & 11 & 10 & 0 \\
9 & 10 & 11 & 1 \\
10 & 9 & 10 & 0 \\
11 & 10 & 9 & 10 \\
9 & 10 & 11 & 9 \\
10 & 9 & 11 & 10 \\
\end{array}
\]

\[
\begin{array}{cccc}
10 & 10 & 0 \\
10 & 9 & 10 \\
10 & 9 & 11 \\
9 & 9 & 99 \\
10 & 9 & 11 \\
\end{array}
\]

\[
\frac{1}{9}(10 \times 1 + 11 \times 1 + 10 \times 1 + 9 \times 1 + 10 \times 1 + 11 \times 1 + 10 \times 1 + 9 \times 1 + 10 \times 1) = 10
\]
Example:

$$\frac{1}{9}.(10 \times 1 + 0 \times 1 + 0 \times 1 + 11 \times 1 + 1 \times 1 + 0 \times 1 + 10 \times 1 + 0 \times 1 + 2 \times 1) = \frac{1}{9}.(34) = 3.7778$$

Example:

$$\frac{1}{9}.(10 \times 1 + 9 \times 1 + 11 \times 1 + 9 \times 1 + 99 \times 1 + 11 \times 1 + 11 \times 1 + 10 \times 1 + 10 \times 1) = \frac{1}{9}.(180) = 20$$
Example:

\[
\frac{1}{9} \left( 10x1 + 0x1 + 2x1 + 9x1 + 10x1 + 9x1 + 11x1 + 9x1 + 99x1 \right) = \frac{1}{9} \cdot 159 = 17.6667
\]

How big should the mask be?

- The bigger the mask,
  - more neighbors contribute.
  - smaller noise variance of the output.
  - bigger noise spread.
  - more blurring.
  - more expensive to compute.
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

Source: D. Lowe

Practice with linear filters

Original

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Blur (with a box filter)

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Source: D. Lowe

Practice with linear filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

(Note that filter sums to 1)

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\]

Source: D. Lowe
Practice with linear filters

Sharpening filter
- Accentuates differences with local average

Source: D. Lowe

Sharpening

before

after

Source: D. Lowe
Example: Smoothing by Averaging

What is wrong with the picture?

Box filter

Smoothing with box filter revisited

- What’s wrong with this picture?
- What’s the solution?
  - To eliminate edge effects, weight contribution of neighborhood pixels according to their closeness to the center

“fuzzy blob”
Gaussian Filter

- A particular case of averaging
  - The coefficients are samples of a 1D Gaussian.
  - Gives more weight at the central pixel and less weights to the neighbors.
  - The further away the neighbors, the smaller the weight.

\[ g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}. \]

Sample from the continuous Gaussian

How big should the mask be?

- The std. dev of the Gaussian \( \sigma \) determines the amount of smoothing.
- The samples should adequately represent a Gaussian
- For a 98.76% of the area, we need
  \[ m = 5\sigma \text{ or } 3\sigma \]
  \[ 5.(1/\sigma) \leq 2\pi \Rightarrow \sigma \geq 0.796, \ m \geq 5 \]

5-tap filter

\[ g[x] = [0.136, 0.6065, 1.00, 0.606, 0.136] \]
Gaussian Filter

\[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

- Constant factor at front makes volume sum to 1 (can be ignored when computing the filter values, as we should renormalize weights to sum to 1 in any case)

Source: C. Rasmussen

Gaussian filter

\[ G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \]

- Standard deviation \( \sigma \): determines extent of smoothing

Source: K. Grauman
Choosing filter width

- The Gaussian function has infinite support, but discrete filters use finite kernels

Gaussian vs. box filtering
**Gaussian filters**

- Remove “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian
  - So can smooth with small-$\sigma$ kernel, repeat, and get same result as larger-$\sigma$ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. $\sigma$
    is same as convolving once with kernel with std. dev. $\sqrt{2}\sigma$
- **Separable** kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

---

**Separability of the Gaussian filter**

\[ G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} \exp\left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]

\[ = \left( \frac{1}{\sqrt{2\pi}\sigma} \exp\left( -\frac{x^2}{2\sigma^2} \right) \right) \left( \frac{1}{\sqrt{2\pi}\sigma} \exp\left( -\frac{y^2}{2\sigma^2} \right) \right) \]

The 2D Gaussian can be expressed as the product of two functions, one a function of $x$ and the other a function of $y$

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe
Separability example

2D convolution (center location only)

\[
\begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
\] \[\times\] \[
\begin{array}{ccc}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6 \\
\end{array}
\]

The filter factors into a product of 1D filters:

\[
\begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
\] \[=\] \[
\begin{array}{c}
1 \\
2 \\
1 \\
\end{array}
\] \[\times\] \[
\begin{array}{c}
1 \\
2 \\
1 \\
\end{array}
\]

Perform convolution along rows:

\[
\begin{array}{ccc}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{array}
\] \[\ast\] \[
\begin{array}{ccc}
2 & 3 & 3 \\
3 & 5 & 5 \\
4 & 4 & 6 \\
\end{array}
\] \[=\] \[
\begin{array}{c}
11 \\
18 \\
18 \\
\end{array}
\]

Followed by convolution along the remaining column:

Source: K. Grauman

Why is separability useful?

• What is the complexity of filtering an \(n \times n\) image with an \(m \times m\) kernel?
  - \(O(n^2 m^2)\)
• What if the kernel is separable?
  - \(O(n^2 m)\)
Image Smoothing

- Convolution with a 2D Gaussian filter
  \[ \tilde{I}(x, y) = I(x, y) * g(x, y) = I(x, y) * g(x) * g(y) \]
- Gaussian filter is separable, convolution can be accomplished as two 1-D convolutions
  \[ \tilde{I}[x, y] = I[x, y] * g[x, y] = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}} \sum_{l=-\frac{N}{2}}^{\frac{N}{2}} I[k, l]g[x - k]g[y - l] \]

Non-linear Filtering

- Replace each pixel with the MEDIAN value of all the pixels in the neighborhood.
- Non-linear
- Does not spread the noise
- Can remove spike noise
- Expensive to run
Noise

- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise**: contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Source: S. Seitz

Reducing salt-and-pepper noise

- What’s wrong with the results?
Example:

\[
\begin{array}{cccccccc}
10 & 11 & 10 & 0 & 0 & 1 & 9 & 10 \\
9 & 10 & 11 & 1 & 0 & 1 & & \\
10 & 9 & 10 & 0 & 2 & 1 & & \\
11 & 10 & 9 & 10 & 9 & 11 & & \\
9 & 10 & 11 & 9 & 99 & 11 & & \\
10 & 9 & 9 & 11 & 10 & 10 & & \\
\end{array}
\]

I

\[
\begin{array}{cccccccc}
11 & 10 & 9 & 10,11,9,10 & 9 & 10,9,10,10,10 & 9 & 10,9,10,10,10 \\
10 & 11 & 10 & 0 & 0 & 1 & 9 & 10 \\
9 & 10 & 11 & 1 & 0 & 1 & & \\
10 & 9 & 10 & 0 & 2 & 1 & & \\
11 & 10 & 9 & 10 & 9 & 11 & & \\
9 & 10 & 11 & 9 & 99 & 11 & & \\
10 & 9 & 9 & 11 & 10 & 10 & & \\
\end{array}
\]

O

\[
\text{median}
\]

sort

10,11,10,9,10,11,9,10,10,11,9,10,10,11

Example:

\[
\begin{array}{cccccccc}
10 & 11 & 10 & 0 & 0 & 1 & 9 & 10 \\
9 & 10 & 11 & 1 & 0 & 1 & & \\
10 & 9 & 10 & 0 & 2 & 1 & & \\
11 & 10 & 9 & 10 & 9 & 11 & & \\
9 & 10 & 11 & 9 & 99 & 11 & & \\
10 & 9 & 9 & 11 & 10 & 10 & & \\
\end{array}
\]

I

\[
\begin{array}{cccccccc}
11,10,0,10,11,1,9,10,0 & 1 & 0 & 9 & 10,10,10,11,11 & 0,0,1,9,10,10,10,11,11 \\
10 & 11 & 10 & 0 & 0 & 1 & 9 & 10 \\
9 & 10 & 11 & 1 & 0 & 1 & & \\
10 & 9 & 10 & 0 & 2 & 1 & & \\
11 & 10 & 9 & 10 & 9 & 11 & & \\
9 & 10 & 11 & 9 & 99 & 11 & & \\
10 & 9 & 9 & 11 & 10 & 10 & & \\
\end{array}
\]

O

\[
\text{median}
\]

sort

11,10,0,10,11,1,9,10,0,10,10,11,11

sort
Example:

\[
\begin{array}{cccccc}
10 & 11 & 10 & 0 & 0 & 1 \\
9 & 10 & 11 & 1 & 0 & 1 \\
10 & 9 & 10 & 0 & 2 & 1 \\
11 & 10 & 9 & 10 & 9 & 11 \\
9 & 10 & 11 & 9 & 99 & 11 \\
10 & 9 & 9 & 11 & 10 & 10 \\
\end{array}
\]

\[
\begin{array}{ccccccc}
\times & \times & \times & \times & \times & \times \\
\times & 10 & & & & \times \\
\times & & & & & \times \\
\times & & & & 9 & \times \\
\times & & & 9 & \times & \times \\
\times & \times & \times & \times & \times & \times \\
\end{array}
\]

\[
10,9,11,99,11,11,10,10 \\
\rightarrow \\
9,9,10,10,10,11,11,11,99
\]

Gaussian vs. median filtering

<table>
<thead>
<tr>
<th>3x3</th>
<th>5x5</th>
<th>7x7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gaussian</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Median</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Image Smoothing With Gaussian (MATLAB)

figure(3);
sigma = 3;
width = 3 * sigma;
support = -width : width;
gauss2D = exp(- (support / sigma).^2 / 2);
gauss2D = gauss2D / sum(gauss2D);
smooth = conv2(conv2(bw, gauss2D, 'same'), gauss2D', 'same');
image(smooth);
colormap(gray(255));

gauss3D = gauss2D' * gauss2D;
tic;
smooth = conv2(bw, gauss3D, 'same');
toc

Demonstrates separability

Example of Blurring

Image Smoothing With Gaussian (MATLAB)
Sharpening - Blurring revisited

- What does blurring take away?

\[
\text{original} - \text{smoothed (5x5)} = \text{detail}
\]

Let's add it back:

\[
\text{original} + \alpha \text{detail} = \text{sharpened}
\]

Edge detection

- **Goal**: Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

- **Ideal**: artist’s line drawing (but artist is also using object-level knowledge)

Source: D. Lowe
Origin of edges

- Edges are caused by a variety of factors:
  - depth discontinuity
  - surface color discontinuity
  - illumination discontinuity
  - surface normal discontinuity

Characterizing edges

- An edge is a place of rapid change in the image intensity function
  - First derivative edges correspond to extrema of derivative

Source: Steve Seitz
**Edge detection (1D)**

Edge = sharp variation

Large first derivative

**Digital Approximation of 1st derivatives**

\[
\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}
\]

\[
\frac{df(x)}{dx} \approx \frac{f(x + 1) - f(x - 1)}{2}
\]

Convolve with:  
-1   0   1
Edge Detection (2D)

Vertical Edges:
Convolve with: \[-1 \quad 0 \quad 1\]

Horizontal Edges:
Convolve with: \[-1\]
\[\begin{array}{c}
0 \\
1
\end{array}\]

Partial derivatives of an image

\[\frac{\partial f(x,y)}{\partial x}\] \[\frac{\partial f(x,y)}{\partial y}\]

Which shows changes with respect to x?
Noise cleaning and Edge Detection

\[ I(x,y) \xrightarrow{\text{Noise Filter}} E(x,y) \]

We need to also deal with noise
Combine Linear Filters

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

\[ f(x) \]

\[ \frac{d}{dx} f(x) \]

Where is the edge? 

Source: S. Seitz
Noise Smoothing & Edge Detection

Convolve with:

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

Vertical Edge Detection

This mask is called the (vertical) Prewitt Edge Detector

Outer product of box filter \([1 1 1]^T\) and \([-1 0 1]\)

Noise Smoothing & Edge Detection

Convolve with:

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

Horizontal Edge Detection

This mask is called the (horizontal) Prewitt Edge Detector
Sobel Edge Detector

Convolve with:

\[
\begin{array}{ccc}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1 \\
\end{array}
\]

and

\[
\begin{array}{ccc}
-1 & -2 & -1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{array}
\]

Gives more weight to the 4-neighbors

Example

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 50 & 50 & 50 \\
0 & 50 & 50 & 50 \\
0 & 50 & 50 & 50 \\
\end{array}
\]

\[
\begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 50 & 100 & 150 \\
0 & 50 & 100 & 150 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{cccc}
-1 & -1 & -1 \\
0 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\[
I_x = \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 50 & 50 & 0 \\
0 & 100 & 100 & 0 \\
0 & 150 & 150 & 0 \\
0 & 150 & 150 & 0 \\
\end{array}
\]

\[
I_y = \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 50 & 50 & 0 \\
0 & 100 & 100 & 0 \\
0 & 150 & 150 & 0 \\
0 & 150 & 150 & 0 \\
\end{array}
\]
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative.
- This saves us one operation:

\[ f \ast \frac{d}{dx} g \]

Source: S. Seitz

Image Derivatives

We know better alternative to smoothing
Smooth using Gaussian filter

\[ g(x) \text{ is a 1-D gaussian kernel, } g(x,y) \text{ - 2-D gaussian kernel} \]

\[ I[x, y] = I[x, y] \ast g[x, y] = \sum_{k=-\frac{\sigma}{2}}^{\frac{\sigma}{2}} \sum_{l=-\frac{\sigma}{2}}^{\frac{\sigma}{2}} I[k, l]g[x-k]g[y-l] \]

Taking a derivative - linear operation (take the derivative of the filter)

\[ I_x[x, y] = I[x, y] \ast g'[x] \ast g[y] = \sum_{k=-\frac{\sigma}{2}}^{\frac{\sigma}{2}} \sum_{l=-\frac{\sigma}{2}}^{\frac{\sigma}{2}} I[k, l]g'[x-k]g[y-l], \]

\[ I_y[x, y] = I[x, y] \ast g[x] \ast g'[y] = \sum_{k=-\frac{\sigma}{2}}^{\frac{\sigma}{2}} \sum_{l=-\frac{\sigma}{2}}^{\frac{\sigma}{2}} I[k, l]g[x-k]g'[y-l]. \]
Gaussian and its derivative

\[ g(x) = \frac{1}{\sqrt{2\pi\sigma} e^{-\frac{x^2}{2\sigma^2}}}, \quad g'(x) = -\frac{x}{\sigma^2\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}. \]

Derivative of Gaussian filter

- Are these filters separable?
Derivative of Gaussian filter

- Which one finds horizontal/vertical edges?

Vertical edges $I_x(x, y) = \frac{\partial I}{\partial x}$

Horizontal edges $I_y(x, y) = \frac{\partial I}{\partial y}$
Edge Detection With Smoothed Images

\[ \nabla I = \left[ \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right] \]

- **Image Gradient**
- **Gradient Magnitude**
  \[ m = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2} \]
- **Gradient Orientation**
  \[ \theta = \tan^{-1}\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right) \]

\[
\begin{align*}
(dx, dy) &= \text{gradient(smoothed_image)}; \\
gmag = \text{sqrt}(dx.^2 + dy.^2); \\
gmax &= \max(\max(gmag)); \\
imshow(gmag); \\
colormap(gray(gmax));
\end{align*}
\]

**Displaying edge normal**

\[
\begin{align*}
[m,n] &= \text{size(gmag)}; \\
edges &= (gmag > 0.3 \times gmax); \\
inds &= \text{find(edges)}; \\
[posx,posy] &= \text{meshgrid}(1:n,1:m); posx2=posx(inds); posy2=posy(inds); \\
gm2 &= \text{gmag}(inds); \\
sintheta &= dx(inds) ./ gm2; \\
costheta &= -dy(inds) ./ gm2; \\
quiver(posx2,posy2, gm2 .* sintheta / 10, -gm2 .* costheta / 10); \\
hold off;
\end{align*}
\]
Effect of Smoothing Scale

- Convolution with $x$-derivative of Gaussian filter with varying scale
- Scale affects the derivative estimates as well as semantics of the edges

Gradient Magnitude Scale

- Increased smoothing:
  - Eliminates noise edges.
  - Makes edges smoother and thicker.
  - Removes fine detail
There are three major issues in edge detection:

1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trail; how do we identify the significant points?
3) How do we link the relevant points up into curves?

At different scales edges makes the edges smoother and thicker.

Review: Smoothing vs. derivative filters

- **Smoothing filters**
  - Gaussian: remove “high-frequency” components; “low-pass” filter
  - Can the values of a smoothing filter be negative?
  - What should the values sum to?
    * One: constant regions are not affected by the filter

- **Derivative filters**
  - Derivatives of Gaussian
  - Can the values of a derivative filter be negative?
  - What should the values sum to?
    * Zero: no response on constant regions
    * High absolute value at points of high contrast
Pointwise Image Operations

- Lookup table - match image intensity to the displayed brightness values

Manipulation of the lookup table - different Visual effects - mapping is often non-linear
Quantization  

Thresholding  

Histogram  

Histogram - frequency gray-level \( \rightarrow \) empirical distribution  

\( h[i] \) - number of pixels of intensity \( i \)  

Histogram equalization - making histogram flat