Convolution and Image Derivatives
Moving Average

- Can add weights to our moving average
- \( Weights \ [\ldots, 0, 1, 1, 1, 1, 1, 1, 0, \ldots] / 5 \)
Weighted Moving Average

- bell curve (gaussian-like) weights [..., 1, 4, 6, 4, 1, ...]
Moving Average In 2D

What are the weights $H$?

$$
\begin{array}{cccccccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 0 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 90 & 90 & 90 & 90 & 90 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 90 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
$$

$H[u, v]$

$F[x, y]$
A Gaussian kernel gives less weight to pixels further from the center of the window

\[
F[x, y] = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}
\]

This kernel is an approximation of a Gaussian function:
Mean vs. Gaussian filtering
Important filter: Gaussian

Weight contributions of neighboring pixels by nearness

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

\[
\begin{array}{cccccc}
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\
0.022 & 0.097 & 0.159 & 0.097 & 0.022 \\
0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\
0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\
\end{array}
\]

5 x 5, \( \sigma = 1 \)

Slide credit: Christopher Rasmussen
Gaussian Kernel

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

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

Source: K. Grauman
Gaussian filters

$\sigma = 1$ pixel

$\sigma = 5$ pixels

$\sigma = 10$ pixels

$\sigma = 30$ pixels
Choosing kernel width

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

Source: K. Grauman
Practical matters

How big should the filter be?

Values at edges should be near zero

Rule of thumb for Gaussian: set filter half-width to about $3 \sigma$
Cross-correlation vs. Convolution

cross-correlation: \[ G = H \otimes F \]

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

A convolution operation is a cross-correlation where the filter is flipped both horizontally and vertically before being applied to the image:

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

It is written:

\[ G = H \ast F \]

Convolution is **commutative** and **associative**
Convolution

Adapted from F. Durand
Convolution is nice!

- Notation: \( b = c * a \)

- Convolution is a multiplication-like operation
  - commutative: \( a * b = b * a \)
  - associative: \( a * (b * c) = (a * b) * c \)
  - distributes over addition: \( a * (b + c) = a * b + a * c \)
  - scalars factor out: \( \alpha a * b = a * \alpha b = \alpha(a * b) \)
  - identity: unit impulse \( e = [...] , 0 , 0 , 1 , 0 , 0 , ... \)
    \[ a * e = a \]

- Conceptually no distinction between filter and signal

- Usefulness of associativity
  - often apply several filters one after another: \( (((a * b_1) * b_2) * b_3) \)
  - this is equivalent to applying one filter: \( a * (b_1 * b_2 * b_3) \)
Gaussian and convolution

• Removes “high-frequency” components from the image (low-pass filter)
• Convolution with self is another Gaussian

![Convolution result](image)

– Convolving twice with Gaussian kernel of width $\sigma$
  $=\sigma\sqrt{2}$

Source: K. Grauman
Image half-sizing

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?
Throw away every other row and column to create a $1/2$ size image - called *image sub-sampling*. 
Image sub-sampling

1/2

1/4 (2x zoom)

1/8 (4x zoom)

Aliasing! What do we do?
Sampling an image

Examples of GOOD sampling
Undersampling

Examples of BAD sampling -> Aliasing
Gaussian (lowpass) pre-filtering

Solution: filter the image, \textit{then} subsample

- Filter size should double for each $\frac{1}{2}$ size reduction. Why?
Subsampling with Gaussian pre-filtering

Gaussian 1/2  G 1/4  G 1/8

Slide by Steve Seitz
Compare with...

1/2

1/4  (2x zoom)

1/8  (4x zoom)
More Gaussian pre-filtering
Iterative Gaussian (lowpass) pre-filtering

Filter the image, *then* subsample

- Filter size should double for each $\frac{1}{2}$ size reduction. Why?
- How can we speed this up?
Image Pyramids

Idea: Represent NxN image as a “pyramid” of 1x1, 2x2, 4x4, ..., 2^k x 2^k images (assuming N=2^k)

Known as a **Gaussian Pyramid** [Burt and Adelson, 1983]
- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*
A bar in the big images is a hair on the zebra’s nose; in smaller images, a stripe; in the smallest, the animal’s nose.

Figure from David Forsyth
Gaussian pyramid construction

Repeat
  • Filter
  • Subsample

Until minimum resolution reached
  • can specify desired number of levels (e.g., 3-level pyramid)

The whole pyramid is only 4/3 the size of the original image!

Slide by Steve Seitz
What are they good for?

Improve Search

- Search over translations
  - Classic coarse-to-fine strategy
  - Project 1!
- Search over scale
  - Template matching
  - E.g. find a face at different scales
Taking derivative by convolution
Partial derivatives with convolution

Image is function \( f(x,y) \)

Remember:

\[
\frac{\partial f(x, y)}{\partial x} = \lim_{\epsilon \to 0} \frac{f(x + \epsilon, y) - f(x, y)}{\epsilon}
\]

Approximate:

\[
\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x, y)}{1}
\]

Another one:

\[
\frac{\partial f(x, y)}{\partial x} \approx \frac{f(x + 1, y) - f(x - 1, y)}{2}
\]
Partial derivatives of an image

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

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

Which shows changes with respect to \( x \)?

\[ \begin{pmatrix} -1 & 1 \end{pmatrix} \]

or

\[ \begin{pmatrix} 1 & -1 \end{pmatrix} \]
The gradient of an image: \[ \nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right] \]

The gradient points in the direction of most rapid increase in intensity

- How does this direction relate to the direction of the edge?

The **edge strength** is given by the gradient magnitude

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]

The gradient direction is given by

\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]
Image Gradient

\[ \nabla f = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right) \]

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Partial Derivatives

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

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

\[ \| \nabla f \| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2} \]
Gradient Orientation

$$\theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \text{atan2}(dy, dx)$$

I’m making the lightness equal to gradient magnitude

Source: D. Fouhey
\[ \theta = \tan^{-1} \left( \frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right) \]

Now I’m showing *all* the gradients

Source: D. Fouhey
Why is there structure at 1 and not at 2?

Source: D. Fouhey
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
Solution: smooth first

- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz
Noise in 2D

Noisy Input  
Ix via [-1,01]  
Zoom

Source: D. Fouhey
Noise + Smoothing

Smoothed Input

Ix via [-1,01]

Zoom

Source: D. Fouhey
Derivative theorem of convolution

\[ \frac{\partial}{\partial x} (h \ast f) = (\frac{\partial}{\partial x} h) \ast f \]

This saves us one operation:

\[ \frac{\partial}{\partial x} h \]

\[ (\frac{\partial}{\partial x} h) \ast f \]
Derivative of Gaussian filter

\[ * [1 -1] = \]
Derivative of Gaussian filter

Which one finds horizontal/vertical edges?
Compare to classic derivative filters

Prewitt: \[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \]

Sobel: \[ M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

Roberts: \[ M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \quad ; \quad M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \]

Source: K. Grauman
Filtering: practical matters

What is the size of the output?

(MATLAB) filter2(g, f, shape) or conv2(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

Source: S. Lazebnik
Practical matters

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. Marschner
Low Pass vs. High Pass filtering

Image

Smoothened

Details
Filtering – Sharpening

Image + $\alpha$

“Sharpened” $\alpha=1$
Filtering – Sharpening

Image

Details

+α

“Sharpened” α=0
Filtering – Sharpening

Image + α

“Sharpened” α=2

=
Filtering – Sharpening

Image + \alpha \rightarrow \text{“Sharpened”} \alpha = 0
Filtering – Extreme Sharpening

Image

Details

+α

“Sharpened” α=10

=
Unsharp mask filter

\[ f + \alpha(f - f \ast g) = (1 + \alpha)f - \alpha f \ast g = f \ast ((1 + \alpha)e - \alpha g) \]

- Unit impulse
- Blurred image
- Unit impulse (identity)
- Gaussian
- Laplacian of Gaussian