EE 417 Introduction to Computer Vision / EE 569 3D vision

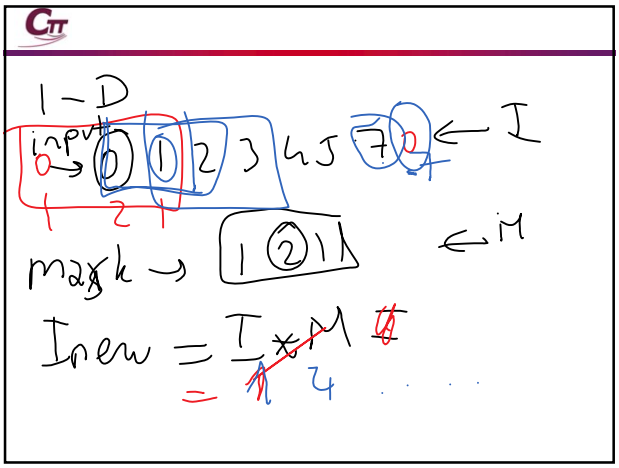
Smoothing, Sharpening and Edges

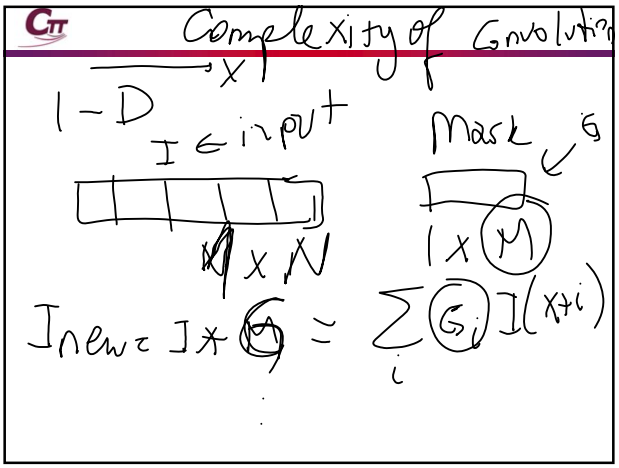
Instructor: Associate Prof. Mehmet Keskinöz

Communication Theory &Technologies (CTT) Group,
Electronics Engineering Program,
Computer Science and Engineering,
Cyber-Security Program
Faculty of Engineering and Natural Sciences

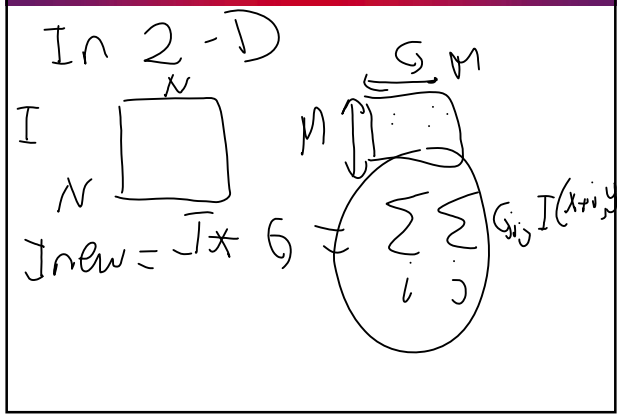
Email: keskinoz@sabanciuniv.edu

Sabancı Üniversitesi

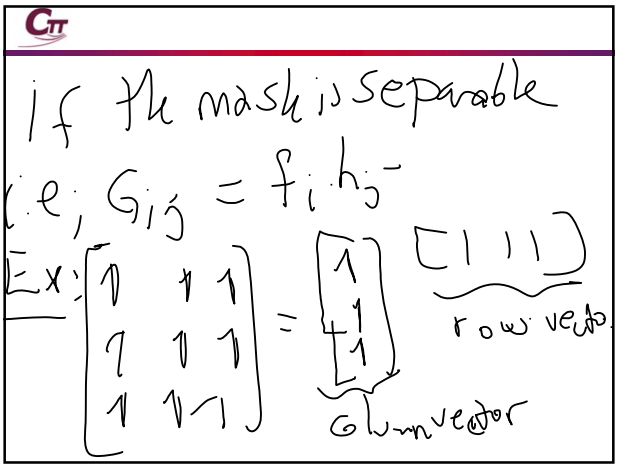


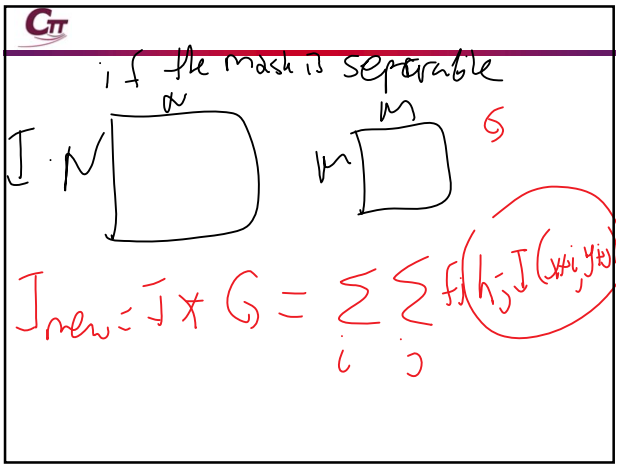


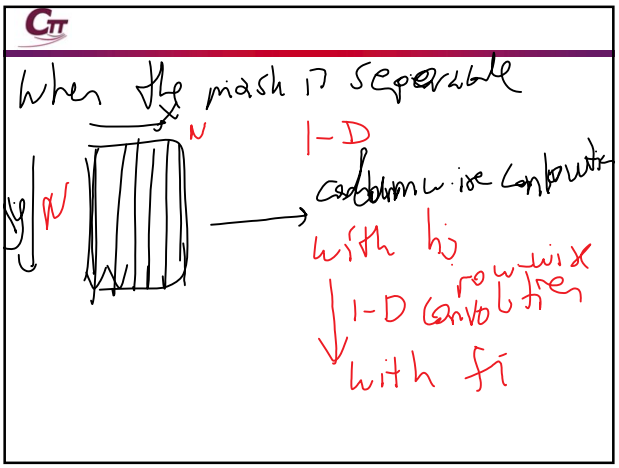
For 1-D
for each pixel (for each x)
we need M multiplication
" " M-1 addition
Complexity $O(N \times M)$

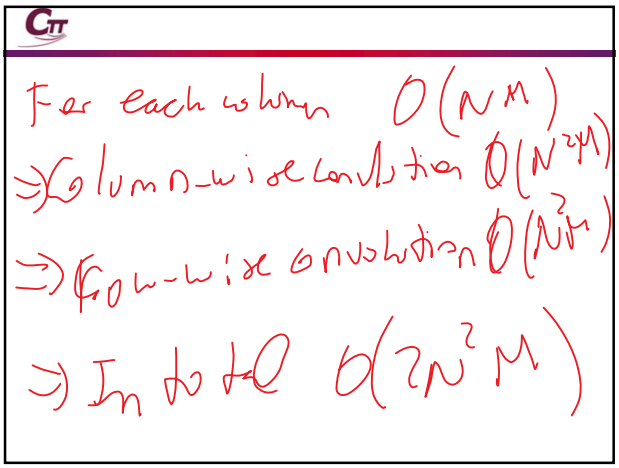


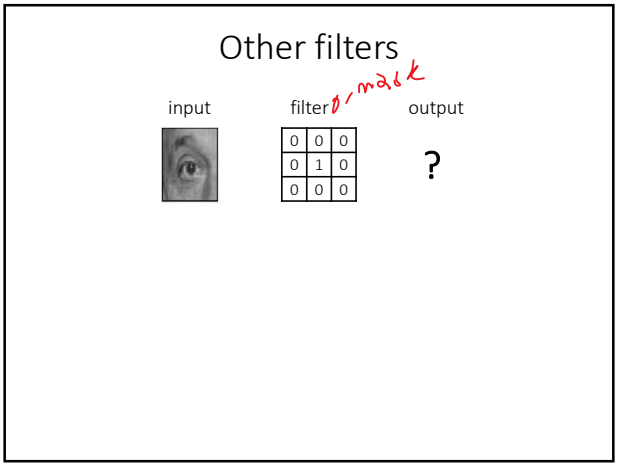
For each pixel we need
(M^2 multiplications
M^2-1 additions)
Complexity $O(N^2 M^2)$

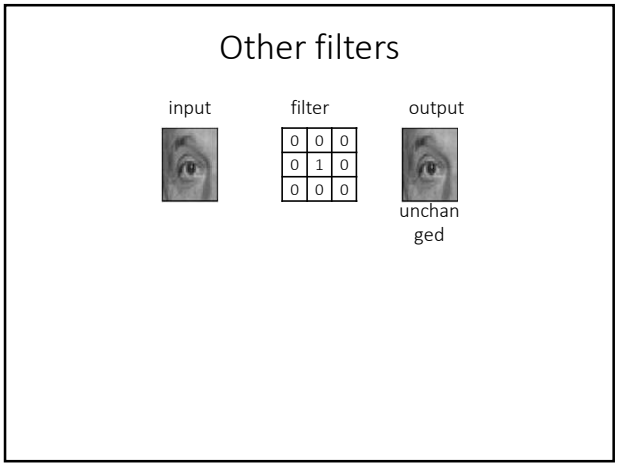

If the mask is separable
i.e., $G_{ij} = f_i h_j$
Ex:
row vector
column vector


if the mask is separable
N
M
$I_{new} = I * G = \sum_i \sum_j f_i h_j I(x_{ij}, y_{ij})$


When the mask is separable
N
M
1-D
column-wise convolution
with h_j
row-wise
1-D convolution
with f_i


For each column $O(NM)$
$\Rightarrow$ Column-wise convolution $O(N^2M)$
$\Rightarrow$ Row-wise convolution $O(N^2M)$
$\Rightarrow$ In total $O(2N^2M)$


Other filters
input
filter
0-mask
output
0 0 0
0 1 0
0 0 0
?


Other filters
input
filter
output
0 0 0
0 1 0
0 0 0
unchanged

Other filters

input | filter | output

0	0	0
0	1	0
0	0	0

unchanged

input | filter | output

0	0	0
0	0	1
0	0	0

?

Other filters

input | filter | output

0	0	0
0	1	0
0	0	0

unchanged

input | filter | output

0	0	0
0	0	1
0	0	0

shift to left by one

Box Filter and Median Operator

Image *smoothing* aims at eliminating "outliers" in image values considered to be noise in a given context

Box Filter (Mean Operator)

Local mean in a $(2k+1) \times (2k+1)$ window (as discussed before)

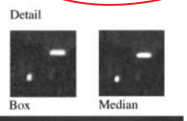
A simple option for image smoothing
Removes outliers but also reduces significantly the contrast
Often sufficient to use just a 3×3 or 5×5 filter kernel

Median Operator

Median of a $(2k+1) \times (2k+1)$ window to reference pixel p

Median of $2n+1$ values = value at position $n+1$ in sorted order
Example: 4 is median of $\{4, 7, 3, 1, 8, 7, 4, 5, 2, 3, 8\}$
Removes outliers with only insignificant changes in image contrast

Example: Box Filter and Median Operator



Bottom, left: 11×11 box filter. *Bottom, right:* 11×11 median operator

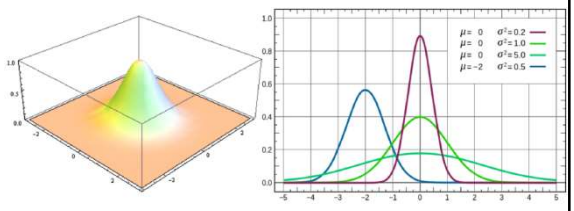
Gauss Filter

1-D Gaussian function

$$f(x) = e^{\frac{-(x-\mu)^2}{2\sigma^2}} \Big/ \sqrt{2\pi}\,\sigma$$

μ → 3 σ: standard deviation

2D Gauss Function



Left: For expected values $\mu_x = \mu_y = 0$
Right: Four examples of 1D Gauss functions

Centered Gauss Function

Zero means $\mu_x = \mu_y = 0$, thus

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) = \frac{1}{\pi s}\cdot e^{-\frac{x^2}{2\sigma^2}}\cdot e^{-\frac{y^2}{2\sigma^2}}$$

Sampled at $(2k+1)\times(2k+1)$ locations

Defines the convolution kernel of the *Gauss filter*

Parametrized by $\sigma > 0$ and $k \geq 1$

$x = i\,\Delta x$

$y = j\,\Delta y$

Filter Kernel for Gauss Filter for $k = 2$ and $\sigma = 1$

Three-sigma rule in statistics:

Sample G_σ by a kernel of size $6\sigma - 1$

$\sum_i \sum_j W_{ij} = 1$

for smoothing

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/273

Gauss Filter

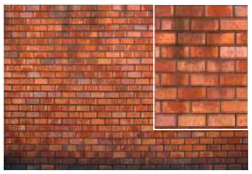
Input image I; calculate local convolution

$$L(x,y,\sigma) = [I * G_\sigma](x,y)$$

by using a sampled kernel

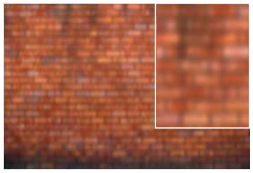
Gaussian filtering example

Gaussian vs box filtering



original

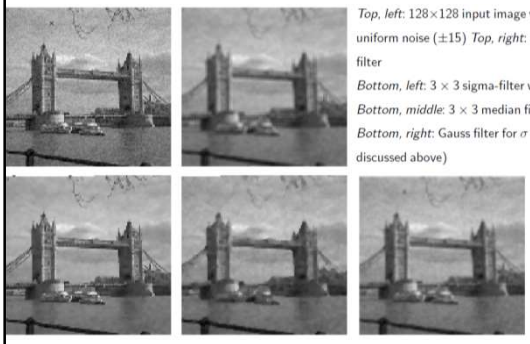
Which blur do you like better?



7x7 Gaussian

7x7 box

Comparison of Noise Removal



Top, left: 128×128 input image with added uniform noise (±15) *Top, right*: 3 × 3 box filter
Bottom, left: 3 × 3 sigma-filter with $\sigma = 30$
Bottom, middle: 3 × 3 median filter
Bottom, right: Gauss filter for $\sigma = 1$ (as discussed above)

Sigma-Filter

Often useful local operator for noise removal

Assume a $(2k+1) \times (2k+1)$ window W_p

1. Calculate the histogram of window W_p
2. Calculate the mean μ of all values in the interval $[I(p) - \sigma, I(p) + \sigma]$
3. Let $J(p) = \mu$

Example: $\sigma = 50$ for $G_{max} = 255$

Note:

$$\mu = \frac{1}{S} \cdot \sum_{u=I(p)-\sigma}^{I(p)+\sigma} u \cdot H(u)$$

$H(u)$ histogram value for W_p and $S = H(I(p) - \sigma) + \ldots + H(I(p) + \sigma)$

Sharpening

Sharpening aims at producing an enhanced image J by increasing the contrast of the given image I along edges, without adding too much noise within homogeneous regions in the image

Unsharp Masking

1. Residual $R(p) = I(p) - S(p)$ for smoothed version $S(p)$ of $I(p)$
2. Add this residual to the given image I:

$$\begin{aligned} J(p) &= I(p) + \lambda [I(p) - S(p)] \\ &= [1 + \lambda] I(p) - \lambda S(p) \end{aligned}$$

where $\lambda > 0$ is a scaling factor

Any smoothing operator may be tried to produce $S(p)$

Parameter k controls the spatial distribution of the smoothing effect

Parameter λ controls the influence of the correction signal $[I(p) - S(p)]$

Examples of Sharpening

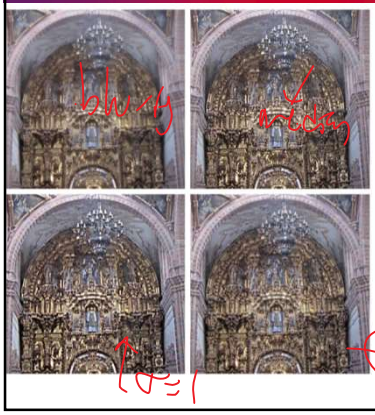
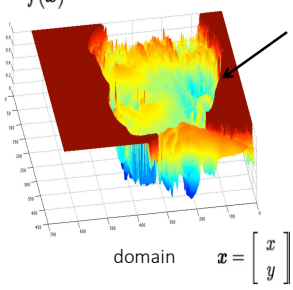


Illustration of unsharp masking for $k = 3$ and $\lambda = 1.5$
Upper left: 512×512 blurred input image
Upper right: Use of a median operator
Lower left: Use of a Gauss filter with $\sigma = 1$
Lower right: Use of a sigma filter with $\sigma = 25$

Image gradients

What are image edges?

grayscale image



Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

✓ You take derivatives: derivatives are large at discontinuities.

How do you differentiate a discrete image (or any other discrete signal)?

Detecting edges

How would you go about detecting edges in an image (i.e., discontinuities in a function)?

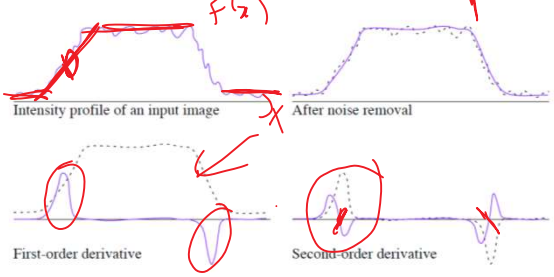
✓ You take derivatives: derivatives are large at discontinuities.

How do you differentiate a discrete image (or any other discrete signal)?

✓ You use finite differences.

Step-edge Model

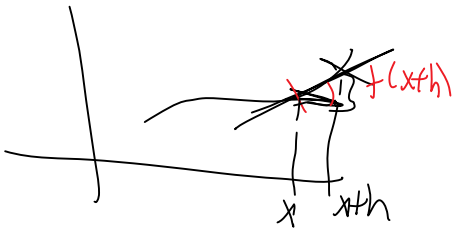
Step-edge model: Defined by
local maxima or minima of first-order derivatives or
zero-crossings of second order derivatives



Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set $h = 2$

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

What convolution kernel does this correspond to?

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

-1	0	1

1	0	-1

Finite differences

High-school reminder: definition of a derivative using forward difference

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

Alternative: use central difference

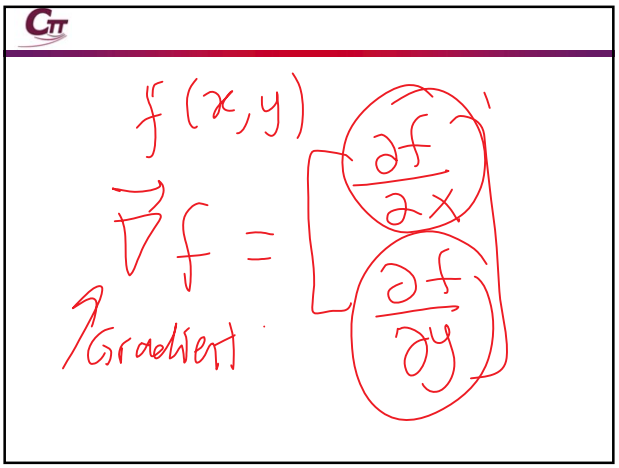
$$f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$$

For discrete signals: Remove limit and set h = 2

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

1D derivative filter

1	0	-1



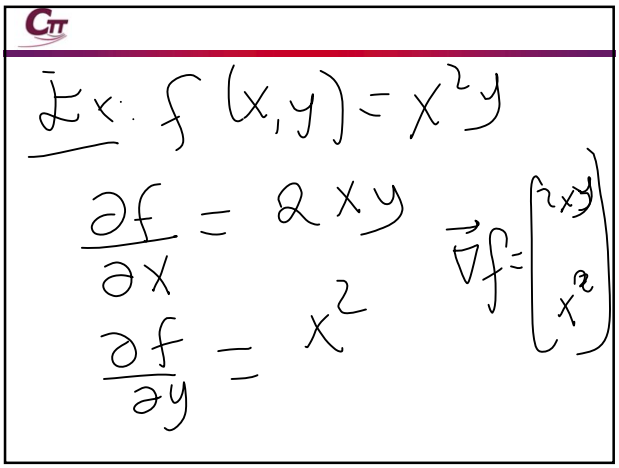
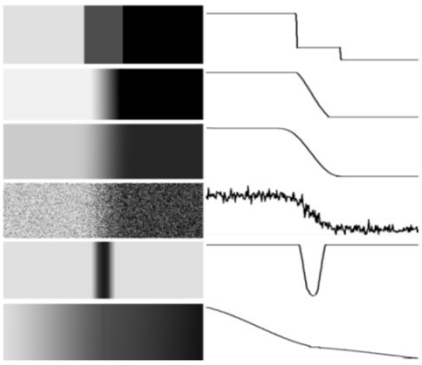


Image Profiles: Step Edges, a Line, and a Slope Change



Derivatives

The derivative of a unary function f in the continuous case

$$\frac{\mathrm{d}f}{\mathrm{d}x}(x) = f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

Partial derivatives for function with two arguments, e.g.

$$\frac{\partial f}{\partial y}(x,y) = f_y(x,y) = \lim_{\varepsilon \to 0} \frac{f(x,y+\varepsilon) - f(x,y)}{\varepsilon}$$

But

In Ω we have the smallest distance $\varepsilon = 1$ between pixel locations
No $\varepsilon \to 0$ possible in the image grid

Smoothing masks

$\rightarrow \sum_i \sum_j G_{ij} = 1$

Derivative mask

$\sum_i \sum_j G_{ij} = 0$

Discrete Derivatives

Option 1: Reduce the derivative to a difference quotient for $\varepsilon = 1$

Option 2: Use a symmetric representation for $\varepsilon = 1$

$$I_y(x,y) = \frac{I(x,y+1) - I(x,y-1)}{2}$$

Thus no asymmetric bias, but very noise-sensitive

Option 3: Larger filter masks

For example, the *Sobel operator* uses the following kernels

I_x

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} /1$$

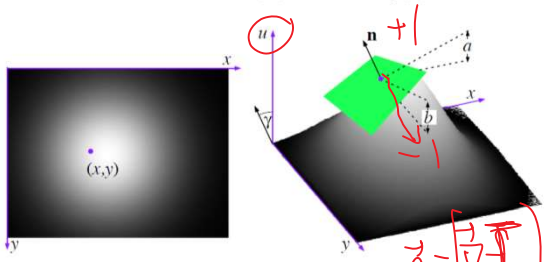
I_y

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} /1$$

Derivatives Define Gradient and Normal

Gradient $[I_x(p), I_y(p)]^\top = [a, b]^\top$

Defined by partial derivatives $I_x(p)$ in x-, and $I_y(p)$ in y-direction



Normal $\mathbf{n}(p) = [I_x(p), I_y(p), +1]^\top = [a, b, +1]^\top$

Orthogonal to tangential plane

$\vec{n} = \begin{bmatrix} \nabla f \\ +1 \end{bmatrix}$

$\nabla f \rightarrow$ magnitude

$\rightarrow$ direction

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

$\angle \nabla f = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

direction

$V = [a, b]^\top$

$V^\perp = [b \quad -a]^\top$

$\vec{n}$

edge

$f(x)$

∇f

direction $\perp$ normal

$\nabla f(x_0)$

edge direction

$\vec{n}$

which is in tangential direction

Gradient and Sobel Operator

Gradient vector $[I_x(p), I_y(p)]^\top$ at pixel location $p \in \Omega$

Magnitude of gradient

$$\sqrt{I_x(p)^2 + I_y(p)^2} = ||\mathbf{grad}\, I(p)||_2$$

identifies local maxima or minima in $I_x(p)$ or $I_y(p)$ (i.e. edges)

Sobel Operator

1. Convolution with the two masks shown on page before
2. Produces at pixel p two values, $S_x(p)$ and $S_y(p)$
3. Sum $|S_x(p)| + |S_y(p)|$ is the result of the Sobel operator

This result approximates the magnitude of the gradient at p

$$\|\vec{x}\|_2 = \sqrt{\sum_i x_i^2}$$ ← more complex

L_2 norm

$$\|\vec{x}\|_1 = \sum_i |x_i|$$ ← simpler

L_1-norm

Example for Sobel Operator

Top: Original image I and absolute value of derivative in x-direction
Bottom: Absolute value of derivative in y-direction and final result
Results shown as inverted images (i.e. pixels with value 0 are white, grey pixels in between, and finally value 255 shown as black pixel).

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image

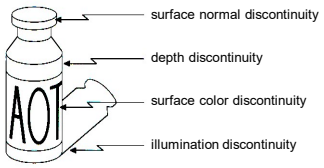
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

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

Source: D. Lowe and Debis

Edge detection

Origin of edges



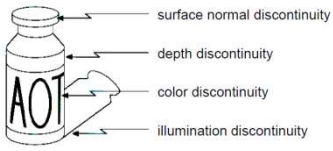
Edges are caused by a variety of factors.

Geometric events

- surface orientation (boundary) discontinuities
- depth discontinuities
- color and texture discontinuities

Non-geometric events

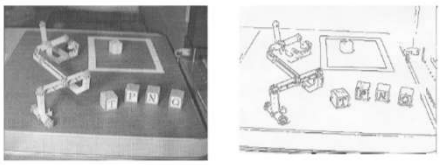
- illumination changes
- specularities
- shadows
- inter-reflections



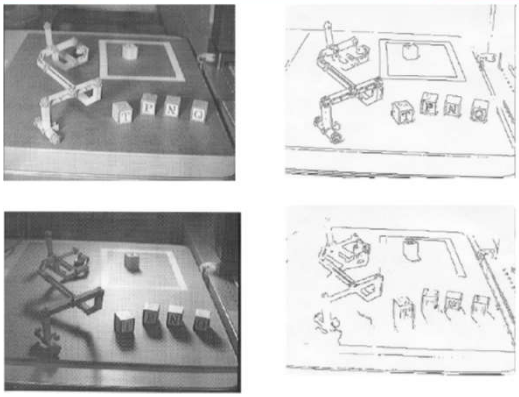
Why is Edge Detection Useful?

Important features can be extracted from the edges of an image (e.g., corners, lines, curves).

These features are used by higher-level computer vision algorithms (e.g., recognition).



Effect of Illumination

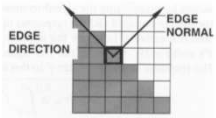


Edge Descriptors

Edge direction: perpendicular to the direction of maximum intensity change (i.e., edge normal)

Edge strength: related to the local image contrast along the normal.

Edge position: the image position at which the edge is located.



Characterizing edges

- An edge is a place of rapid change in the image intensity function

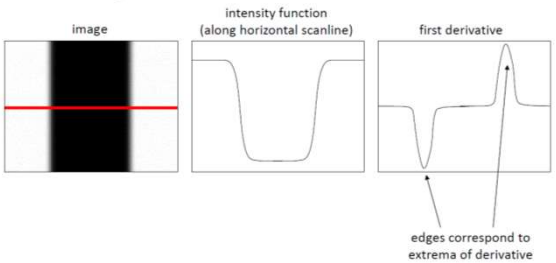
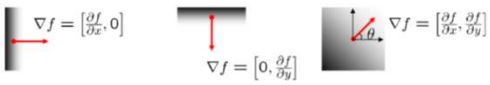


Image gradient

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

The gradient direction is given by

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

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

The *edge strength* is given by the gradient magnitude

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

Source: Steve Seitz

Differentiation and convolution

Recall, for 2D function, f(x,y):

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon, y)}{\varepsilon} - \frac{f(x,y)}{\varepsilon}\right)$$

We could approximate this as:

$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

(which is obviously a convolution)

Check!

-1	1

Source: D. Forsyth, D. Lowe

Computing image gradients

1. Select your favorite derivative filters.

$$S_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad S_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial f}{\partial x} = S_x \otimes f \qquad \frac{\partial f}{\partial y} = S_y \otimes f$$

Computing image gradients

1. Select your favorite derivative filters.

$$S_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad S_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial f}{\partial x} = S_x \otimes f \qquad \frac{\partial f}{\partial y} = S_y \otimes f$$

3. Form the image gradient, and compute its direction and amplitude.

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

gradient direction amplitude

4.) Determine gradient magnitude

L_1 norm: $\|S_x\|_1 + \|S_y\|_1$

L_2: $\|S_x\|_2 + \|S_y\|_2$

Several derivative filters

Sobel

1	0	-1
2	0	-2
1	0	-1

1	2	1
0	0	0
-1	-2	-1

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

- How are the other filters derived and how do they relate to the Sobel filter?
- How would you derive a derivative filter that is larger than 3x3?

Computing image gradients

1. Select your favorite derivative filters.

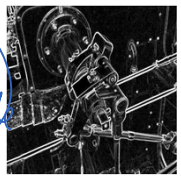
$$S_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad S_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Image gradient example

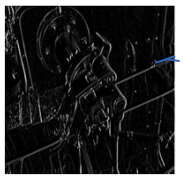
original

vertical derivative

gradient amplitude

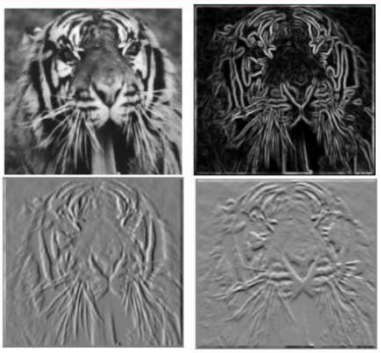


horizontal derivative



How does the gradient direction relate to these edges?

Finite differences: example



Which one is the gradient in the x-direction (resp. y-direction)?

The Sobel filter

1	0	-1
2	0	-2
1	0	-1

Sobel filter

=

1
2
1

What filter is this?

*

1	0	-1

1D derivative filter

The Sobel filter

1	0	-1
2	0	-2
1	0	-1

Sobel filter

=

1
2
1

Blurring

*

1	0	-1

1D derivative filter

In a 2D image, does this filter responses along horizontal or vertical lines?

The Sobel filter

1	0	-1
2	0	-2
1	0	-1

Sobel filter

=

1
2
1

Blurring

*

1	0	-1

1D derivative filter

Does this filter return large responses on vertical or horizontal lines?

The Sobel filter

Horizontal Sober filter:

1	0	-1
2	0	-2
1	0	-1

=

1
2
1

*

1	0	-1

What does the vertical Sobel filter look like?

The Sobel filter

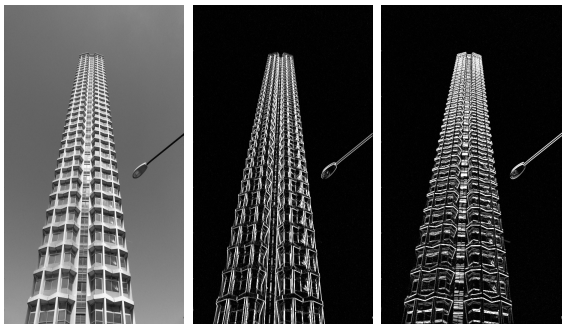
Horizontal Sober filter:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

Vertical Sobel filter:

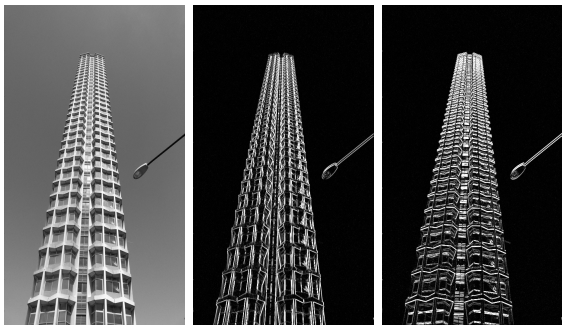
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

Sobel filter example



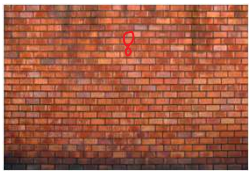
original | which Sobel filter? | which Sobel filter?

Sobel filter example



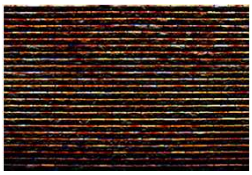
original | horizontal Sobel filter | vertical Sobel filter

Sobel filter example



original

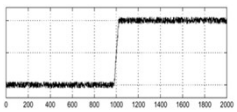
horizontal Sobel filter



vertical Sobel filter

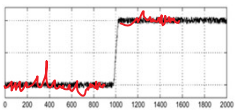
How do you find the edge of this signal?

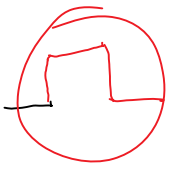
intensity plot



How do you find the edge of this signal?

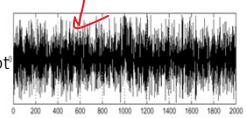
intensity plot





Using a derivative filter:

derivative plot



What's the problem here?

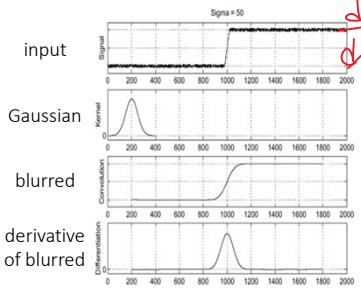
Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

Differentiation is very sensitive to noise

When using derivative filters, it is critical to blur first!

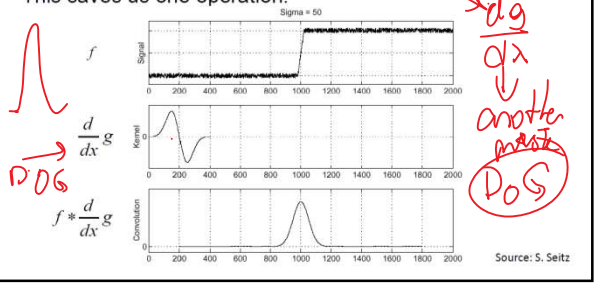


Derivative theorem of convolution

- Differentiation and convolution both linear operators: they "commute"

$$\frac{d}{dx}(f * g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

- This saves us one operation:



Source: S. Seitz

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference $f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$ → 1D derivative filter

1	0	-1

second-order finite difference $f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$ → Laplace filter ?

Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference $f'(x) = \lim_{h \to 0} \frac{f(x+0.5h) - f(x-0.5h)}{h}$ → 1D derivative filter

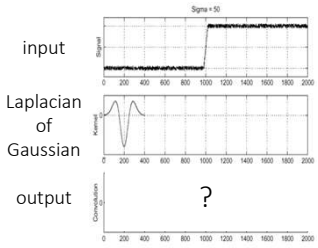
1	0	-1

second-order finite difference $f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$ → Laplace filter

1	-2	1

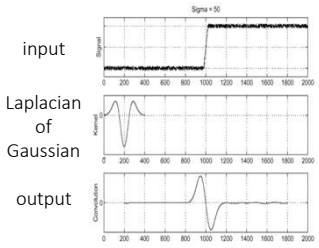
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering



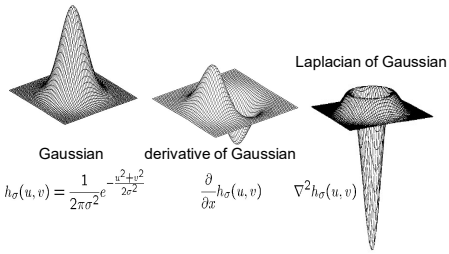
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering



"zero crossings" at edg

2D edge detection filters



$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$ $\quad \frac{\partial}{\partial x} h_\sigma(u,v)$ $\quad \nabla^2 h_\sigma(u,v)$

∇^2 is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

88

Laplacian of Gaussian (LoG) Masks

$$\frac{\partial^2 f}{\partial y^2} = f(i,j+1) - 2f(i,j) + f(i,j-1)$$

$$\frac{\partial^2 f}{\partial x^2} = f(i+1,j) - 2f(i,j) + f(i-1,j)$$

$$\nabla^2 f = -4f(i,j) + f(i,j+1) + f(i,j-1) + f(i+1,j) + f(i-1,j)$$

0	0	0
1	-2	1
0	0	0

0	1	0
0	-2	0
0	1	0

0	1	0
1	-4	1
0	1	0

Laplacian mask

89

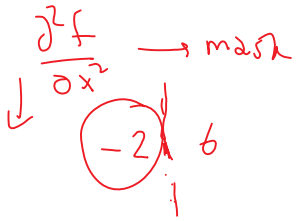
Zero crossing of a 1-D function

f(x) continuous

f(a) > 0

f(b) < 0 ⇒ f(c) = 0, c ∈ (a, b)

90



91

Laplacian of Gaussian (LoG) Masks

$\nabla^2(f * G) = f * \nabla^2 G$ → LoG.

Gaussian

0	1	0	
1	-4	1	1
0	1	0	

1	1	1	
1	-8	1	1
1	1	1	

-1	2	-1	
2	-4	2	1
-1	2	-1	

92

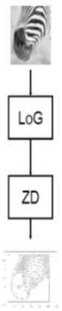
Using the LoG Function (Laplacian of Gaussian)

- The LoG function will be
  - Zero far away from the edge
  - Positive on one side
  - Negative on the other side
  - Zero just at the edge
- It has simple digital mask implementation(s)
- So it can be used as an edge operator

93

Edge detection strategy

- filter with Laplacian of Gaussian
- detect zero crossings
- mark the zero points where:
  - there is a sufficiently large derivative,
  - and enough contrast
- once again we have parameters
  - scale of Gaussian smoothing
  - thresholds
- once again no set of universal parameters
- does not seem to be better than the strategy of looking for maxima of gradient magnitude.



94

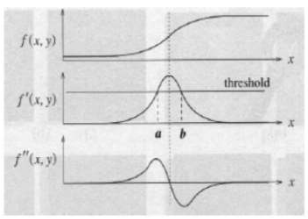
- Approximate finding maxima/minima of gradient magnitude by finding places where:

$$\frac{df^2}{dx^2}(x) = 0$$

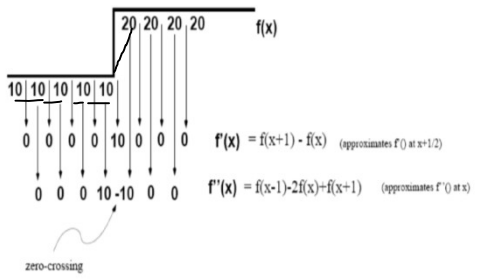
- Can't always find discrete pixels where the second derivative is zero – look for zero-crossing instead.

95

- Edge points can be detected by finding the zero-crossings of the second derivative.



96

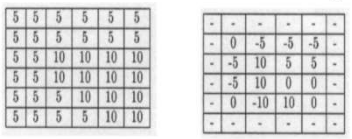


97

- Four cases of zero-crossings:

{+,-}, {+,0,-},{-,+}, {-,0,+}

- **Slope** of zero-crossing {a, -b} is: |a+b|.
- To detect "strong" zero-crossing, threshold the slope. |slope| > threshold.



5	5	5	5	5	5
5	5	5	5	5	5
5	5	10	10	10	10
5	5	10	10	10	10
5	5	5	10	10	10
5	5	5	5	10	10

-	-	-	-	-	-
-	0	-5	-5	-5	-
-	-5	10	5	5	-
-	-5	10	0	0	-
-	0	-10	10	0	-
-	-	-	-	-	-

99

Properties of Laplacian

- It is cheaper to implement than the gradient (i.e., one mask only).
- It does not provide information about edge direction.
- It is more sensitive to noise (i.e., differentiates twice).
- It is an isotropic operator (equal weights in all directions)

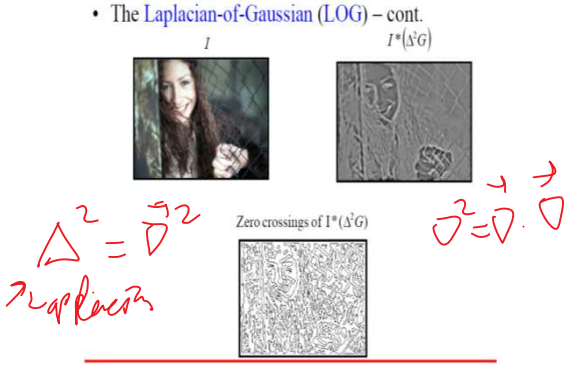
100

Marr-Hildteth (LOG) Algorithm:

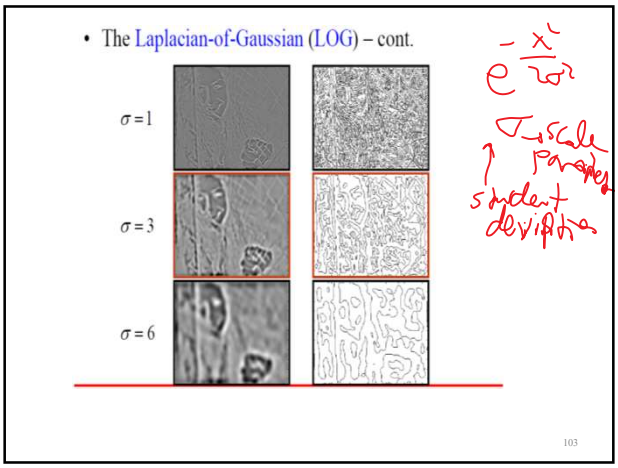
- Find zero-crossings from each row and column
- Find slope of zero-crossings
- Apply threshold to slope and mark edges

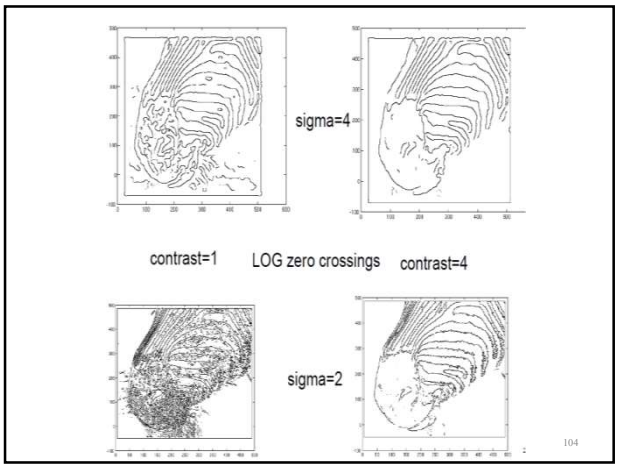
101

- The Laplacian-of-Gaussian (LOG) – cont.

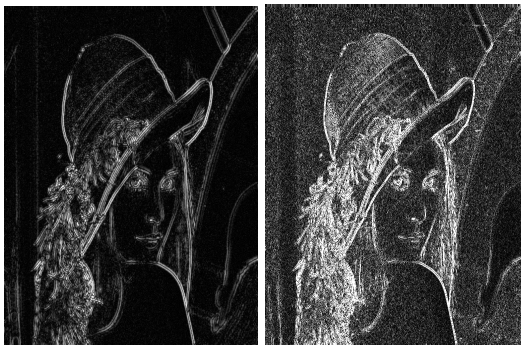


102

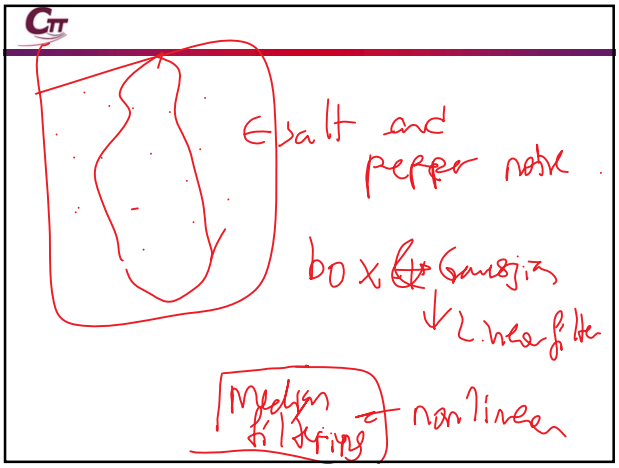




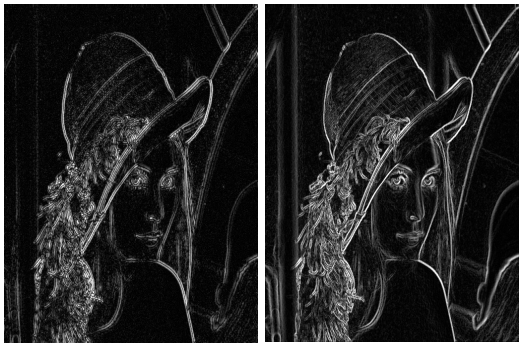
Laplace and LoG filtering examples



Laplacian of Gaussian filtering Laplace filtering

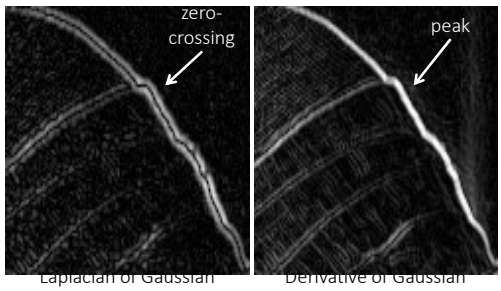


Laplacian of Gaussian vs Derivative of Gaussian



Laplacian of Gaussian filtering Derivative of Gaussian filtering

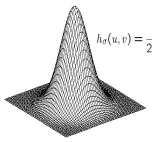
Laplacian of Gaussian vs Derivative of Gaussian



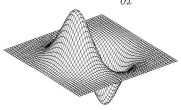
Laplacian of Gaussian filtering Derivative of Gaussian filtering

ero crossings are more accurate at localizing edges (but not very convenient

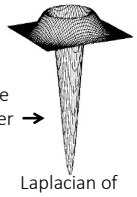
2D Gaussian filters



Gaussian



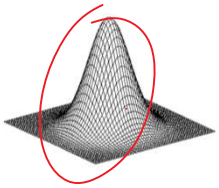
Derivative of Gaussian

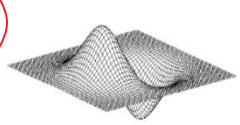


Laplacian of Gaussian

how does this relate to this lecture's cover picture? →

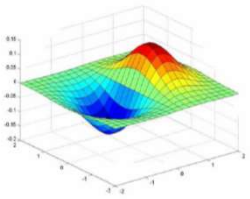
Derivative of Gaussian filter



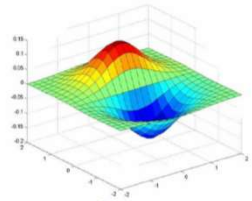


This filter is separable

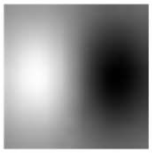
Derivative of Gaussian filter

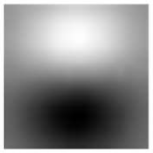


x-direction

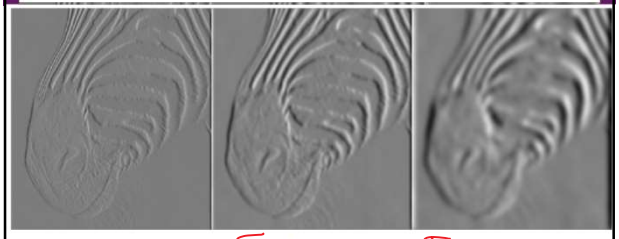


y-direction





Tradeoff between smoothing and localization



σ = 1 pixel σ = 3 pixels σ = 7 pixels

Smoothed derivative removes noise, but blurs edge.
Also finds edges at different "scales".

Source: D. Forsyth

Finite difference filters

- Other approximations of derivative filters exist:

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

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

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

Source: K. Grauman

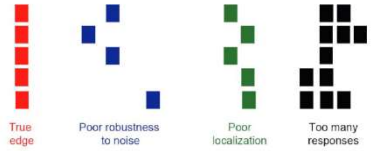
Implementation issues

- The gradient magnitude is large along a thick "trail" or "ridge", so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Designing an edge detector

- Criteria for an "optimal" edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Source: L. Fei-Fei

Canny edge detector

It is a multi-stage algorithm and we will go through each stages.

1. **Noise Reduction**

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter.

2. **Finding Intensity Gradient of the Image**

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

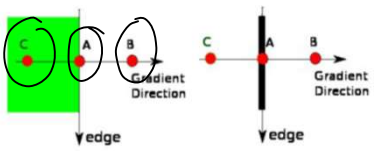
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

3. **Non-maximum Suppression**

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:



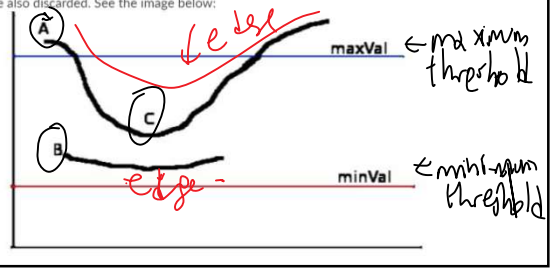
Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

you get "thin edges".

non-maximum suppression to get rid of spurious response to edge detection

4. **Hysteresis Thresholding**

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, *minVal* and *maxVal*. Any edges with intensity gradient more than *maxVal* are sure to be edges and those below *minVal* are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:



The edge A is above the *maxVal*, so considered as "sure-edge". Although edge C is below *maxVal*, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above *minVal* and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select *minVal* and *maxVal* accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

So what we finally get is strong edges in the image.

double threshold to determine potential edges

Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

Canny edge detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide "ridges" down to single pixel width
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

MATLAB: edge(image, 'canny')

Source: D. Lowe, L. Fei-Fei

Example

original image (Lena)

Example

norm of the gradient

Example

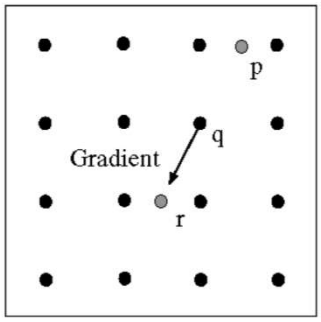
thresholding

Example

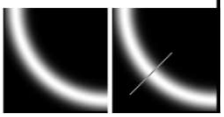
thinning

(non-maximum suppression)

Non-maximum suppression

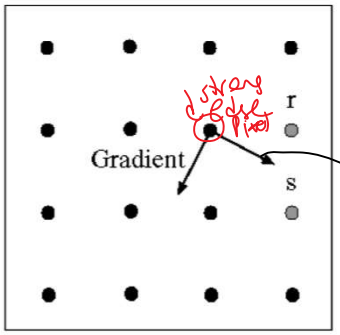


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

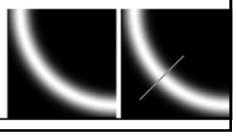


Source: D. Forsyth

Edge linking



Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Source: D. Forsyth

Hysteresis thresholding

Check that if maximum value of gradient is sufficiently large

- drop-outs? use **hysteresis**
  - use a high threshold to start edge curves and a low threshold to continue them.

Source: S. Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Source: S. Seitz

Edge detection is just the beginning…

Berkeley segmentation database:
http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/