BLG453E COMPUTER VISION

Fall 2018 Term

Week 6

Istanbul Technical University
Computer Engineering Department

Instructor: Prof. Gözde ÜNAL

Teaching Assistant: Enes ALBAY

Learning Outcomes of the Course

Students will be able to:

1. Discuss the main problems of computer (artificial) vision, its uses and applications
2. Design and implement various image transforms: point-wise transforms, neighborhood operation-based spatial filters, and geometric transforms over images
3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images
4. Construct least squares solutions to problems in computer vision
5. Describe the idea behind dimensionality reduction and how it is used in data processing
6. Apply object and shape recognition approaches to problems in computer vision

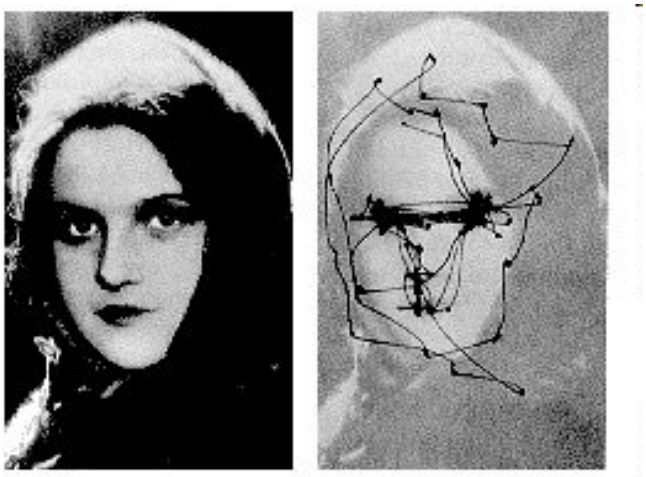
Week 6: Edge Detection Operators (Neighborhood Image Processing)

At the end of Week 6: Students will be able to (Learning Objectives of the week):

2. Design and implement various image transforms: point-wise transforms, neighborhood operation-based spatial filters, and geometric transforms over images

3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images

Edges: Important cues in Human Visual System



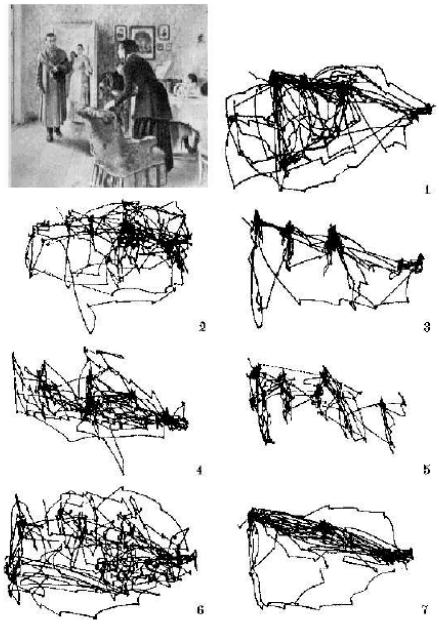
Alfred Yarbus: Eye movement maps

revealed strategies that our visual system uses to inspect objects and hence for their perception.

Eye Movements

EyeTracking Device

Eye Movements : context is important

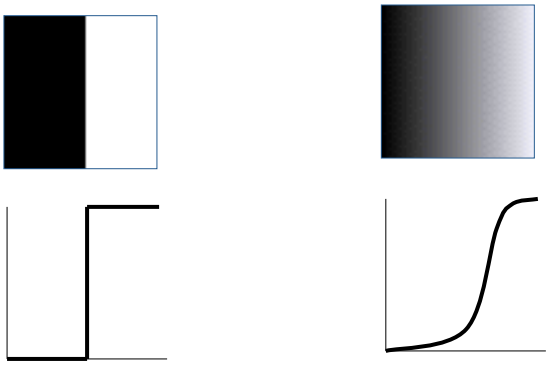


- Free Examine
- What are the material circumstances (economic conditions) of the family?
- What are their ages?
- What were they doing before arrival?
- Remember the clothes
- Remember object and person positions
- How long has the unexpected visitor been away?

[Yarbus 1967]

What is an Edge?

Observation: Boundaries of objects show up as intensity discontinuities



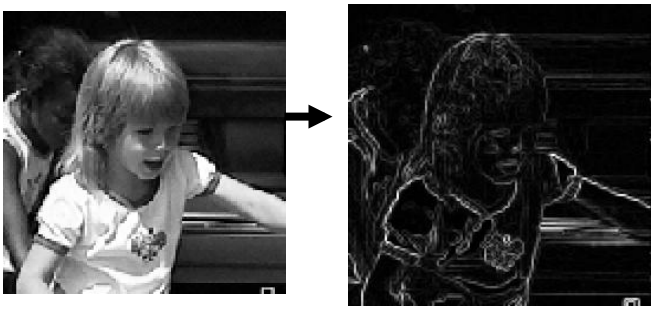
1D profile of Brightness vs. Spatial Coordinates

- Edges are those places in an image that correspond to object boundaries
- Edges are pixels where image brightness changes abruptly

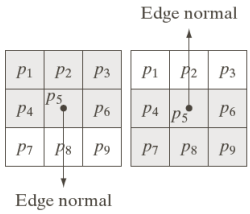
Extraction of Edge Information from Images

An edge is a property attached to an individual pixel and is calculated from the image function behavior in a neighborhood of the pixel

* It is either a scalar variable (an edge map) such as in this example:

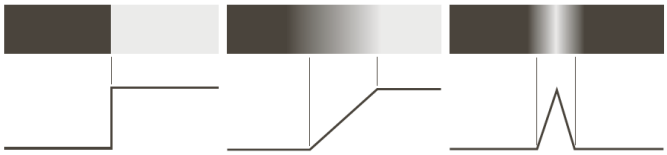


* Or a vector variable (magnitude of the gradient, direction of an edge) such as shown on the right:



Typically: we extract Magnitude and Direction

Edge Models

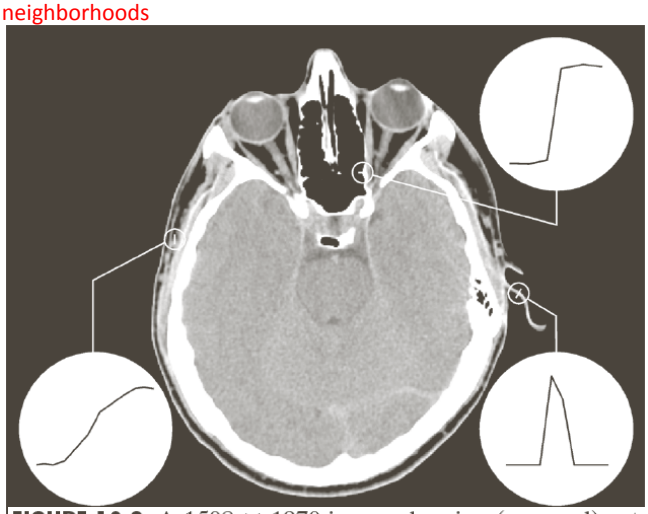


a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

- Step edge: ideal case
- Ramp edge: in practice, images have blurred edges
- Roof edge: model of a line

Edge information in an image is found by looking at the relationship a pixel has with its neighborhoods



If a pixel's gray-level value is similar to those around it, there is probably not an edge at that point.

If a pixel has neighbors with widely varying gray levels, it may present an edge point.

FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas indicated by the short line segments shown in the small circles. The ramp and "step" profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

Edge Detection Operators

Gradient and difference based operators

Detect and measure local discontinuities in intensity or its gradient

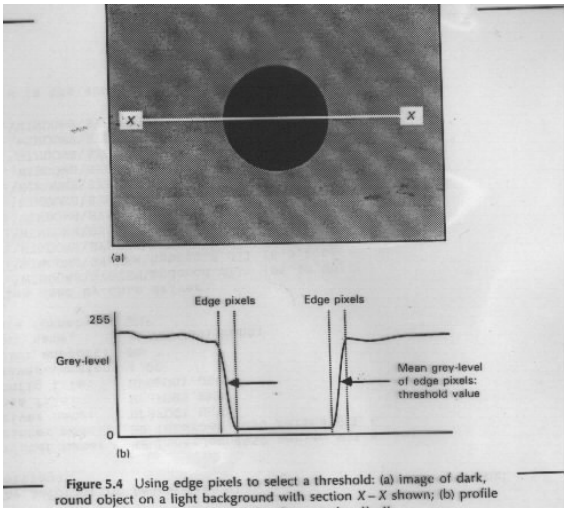
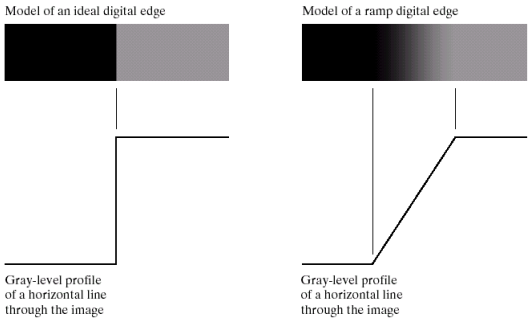


Figure 5.4 Using edge pixels to select a threshold: (a) image of dark, round object on a light background with section X – X shown; (b) profile

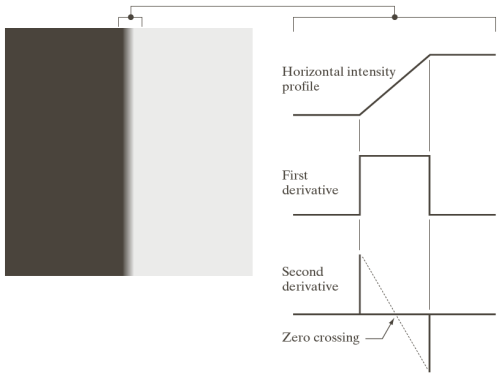
If we define a local edge in an image to be a transition between two regions of significantly different intensities

then the magnitude of the gradient (derivatives) of the image, which measure the rate of change in image brightness will have large values in these transitional boundary areas.



a b
FIGURE 10.5
(a) Model of an ideal digital edge. (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.

Edge Detection



a b
FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal vertical ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, together with its first and second derivatives.

- First derivative: We can check its MAGNITUDE
- Second derivative: We can check its ZERO-CROSSINGS
- Note that 2nd derivative produces double response (2 values) for an edge
- Same idea carries from 1D profile to 2D (on a profile perpendicular to the edge)

Edge Detection Methods

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

- Many are implemented with convolution mask and based on discrete approximations to differential operators
- Kernels are typically determined by discretized image derivatives

Discretization: First derivative Approx.

Forward Difference Equation

Expand $f(x)$ in a Taylor series about x_0.

$$f(x) = f(x_0) + (x - x_0)f'(x_0) + O(h)$$

Then evaluate at $x = x_0 + h$.

$$f(x_0 + h) = f(x_0) + hf'(x_0) + O(h)$$

First order forward difference:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}$$

Discretization: First derivative Approx.

Backward Difference Equation

Replace h with $-h$ in the previous derivation

$$f(x_0 - h) = f(x_0) - hf'(x_0) + O(h)$$

First order backward difference:

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h}$$

Discretization: First derivative Approx.

Centered Difference Equation

Subtract the first order expansions:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \ldots$$

$$-$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \ldots$$

$$=$$

$$f(x_0 + h) - f(x_0 - h) = 2hf'(x_0)$$

Divide by $2h$ to get the centered difference:

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

Discretization: First derivative Approx.

Centered Difference Equation

The second order terms in the expansion cancel, so the remainder is $O(h^2)$

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + O(h^2)$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) + O(h^2)$$

For images : Use h = 1 pixel.

Discretization: Second derivative Approx.

Second Centered Difference Equation

We can approximate the **second derivative** by adding two second order expansions:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + O(h^2)$$

+

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) + O(h^2)$$

=

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + h^2 f''(x_0) + O(h^2)$$

Discretization: Second derivative Approx.

Second Centered Difference Equation

Rearrange

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + h^2 f''(x_0) + O(h^2)$$

to get the (second order) second centered difference

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}$$

THQ: Calculate Derivatives

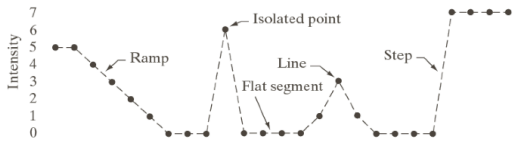
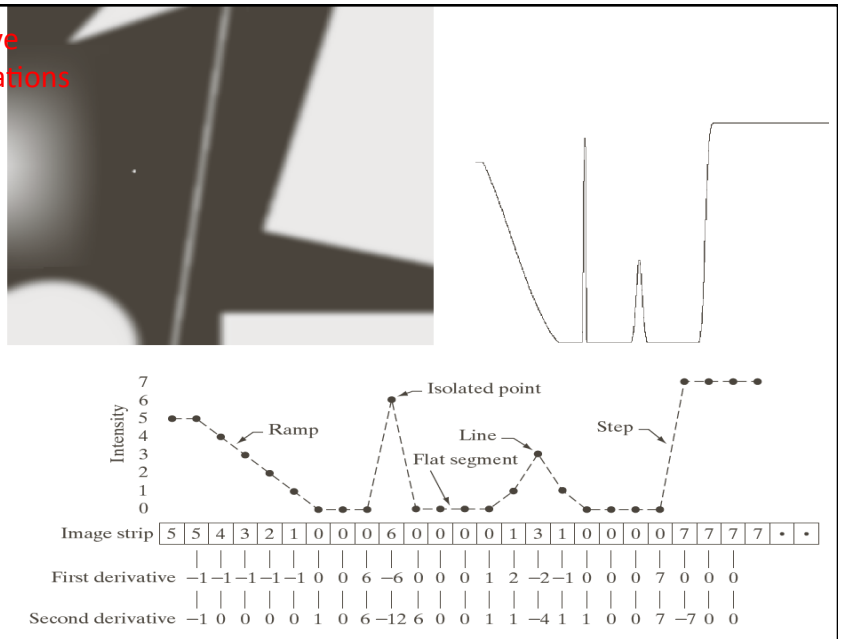


Image strip	5	5	4	3	2	1	0	0	0	6	0	0	0	0	1	3	1	0	0	0	0	7	7	7	7	•	•
First derivative										?																	
Second derivative										?																	

Digital Image Processing: Gonzalez and Woods Book: Edge Detection section

Derivative Computations

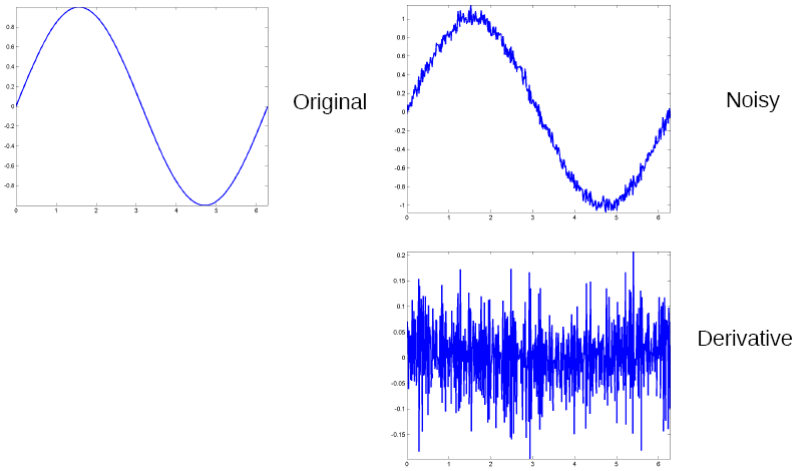


a b
c

FIGURE 10.2 (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).

Digital Image Processing: Gonzalez and Woods Book: Edge Detection section

Derivatives are noise sensitive!



Derivatives are noise sensitive!

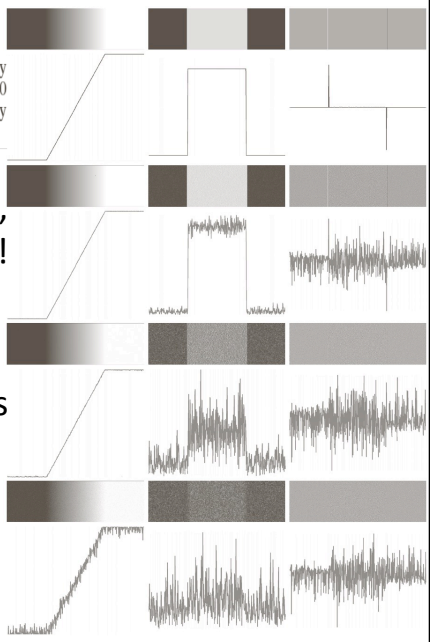


FIGURE 10.11 First column: Images and intensity profiles of a ramp edge corrupted by random Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

- Second derivative, as expected, is even more sensitive to noise!
- Therefore, image smoothing before computing derivatives is REQUIRED!

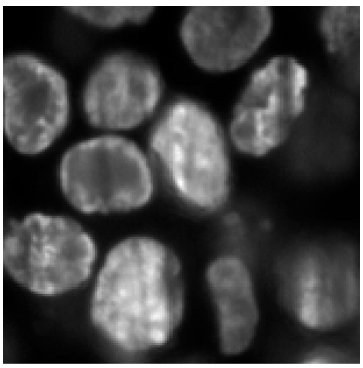
Image Gradient

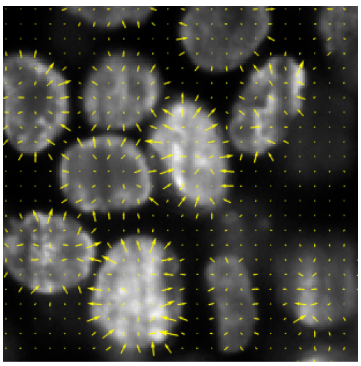
We said some (many) edge operators only return information about the existence of an edge at each point

Some edge operators return magnitude and orientation information, i.e. a vector result :

e.g. image gradient

$$\nabla I(x,y) = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$$



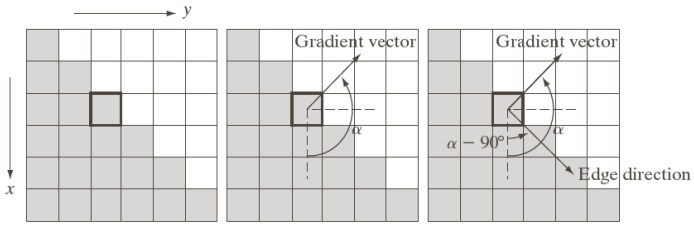


Basic Edge Detection

- Compute the image gradient vector

$$\nabla I(x,y) = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$$

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



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.

Spatial Derivatives can be computed with Spatial Masks/Filters

FIGURE 10.3 A general 3 × 3 spatial filter mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Recall: the response of the mask is R over image I at a given pixel (x,y) is:

$$R(x,y) = \sum_{k=1}^{9} w_k I_k$$

For edge detection operators:
Sum of the kernel weights S = 0

$$S = \sum_{k=1}^{9} w_k = 0$$

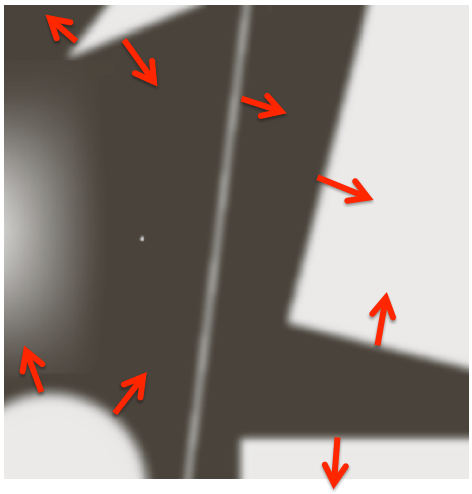
b/c the mask response will be zero in areas of constant intensity as desired.

Ex: a filter mask for a horizontal edge detector

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

Q: what about S for smoothing masks?

Want to calculate the Edge information as a VECTOR at each pixel coordinate point. Note the ORTHOGONALITY (RIGHT angle) of the edge vector to the edges in the image:



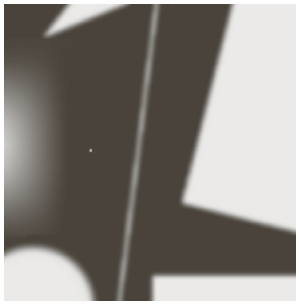
Magnitude of the vector indicates strength of the edge (not in the above figure, which is only for illustration of edge directions.

Let's look at different Edge Masks (Operators): Roberts Operator

−1	0	0	−1
0	1	1	0

Roberts

I_x I_y



Filter the image by these 2x2 masks:

Diagonal differences preferred

Roberts Operator

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- First form of Roberts Operator

$$\sqrt{[I(r,c) - I(r-1,c-1)]^2 + [I(r,c-1) - I(r-1,c)]^2}$$

- Second form of Roberts Operator

$$|I(r,c) - I(r-1,c-1)| + |I(r,c-1) - I(r-1,c)|$$

Roberts Operator

−1	0	0	−1
0	1	1	0

Roberts

These 2x2 masks:

- Not symmetric around the center point: therefore edge direction information ?
- Primary disadvantage:
  - Few pixels are used to approximate the gradient
  - High sensitivity to noise

Prewitt Operator

- Looks for edges in both horizontal and vertical directions, then combine the information into a single metric.

$$w^y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad w^x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Rx and Ry: response to masks w^x and w^y, i.e. :

$$R_x = \sum_k w_k^x I_k$$

Edge Magnitude = $\sqrt{R_x^2 + R_y^2}$

Edge Direction = $\tan^{-1}\left(\frac{R_y}{R_x}\right)$

This can be used as the image gradient magnitude

Sobel Operator

- Similar to the Prewitt, with different mask coefficients:
- The masks are as follows:

$$y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Edge Magnitude = $\sqrt{R_x^2 + R_y^2}$

Edge Direction = $\tan^{-1}\left(\frac{R_y}{R_x}\right)$

Kirsch Compass Masks

- Taking a single mask and rotating it to 8 major compass orientations: N, NW, W, SW, S, SE, E, and NE.
- The edge magnitude = The maximum value found by the convolution of each mask with the image.
- The edge direction is defined by the mask that produces the maximum magnitude.

Kirsch Compass Masks

- The Kirsch masks are defined as follows:

$$N=\begin{bmatrix}-3&-3&-3\\-3&0&-3\\5&5&5\end{bmatrix}\quad W=\begin{bmatrix}-3&-3&5\\-3&0&5\\-3&-3&5\end{bmatrix}\quad S=\begin{bmatrix}5&5&5\\-3&0&-3\\-3&-3&-3\end{bmatrix}\quad E=\begin{bmatrix}5&-3&-3\\5&0&-3\\5&-3&-3\end{bmatrix}$$

$$NW=\begin{bmatrix}-3&-3&-3\\-3&0&5\\-3&5&5\end{bmatrix}\quad SW=\begin{bmatrix}-3&5&5\\-3&0&5\\-3&-3&-3\end{bmatrix}\quad SE=\begin{bmatrix}5&5&-3\\5&0&-3\\-3&-3&-3\end{bmatrix}\quad NE=\begin{bmatrix}-3&-3&-3\\5&0&-3\\5&5&-3\end{bmatrix}$$

- EX: If NW produces the maximum value, then the edge direction is Northwest

Robinson Compass Masks

- Similar to the Kirsch masks, with mask coefficients of 0, 1, and 2:

$$W = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad N = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad S = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad E = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$SE = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix} \quad SW = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \quad NE = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} \quad NW = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

Line Detection Masks/Filters

- The maximum responses will occur at those desired locations with horizontal line features

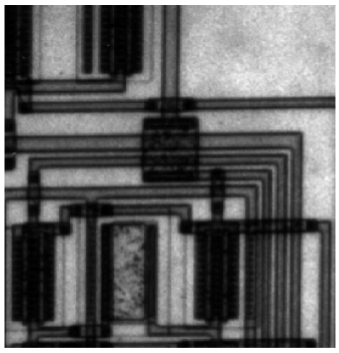
−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

FIGURE 10.6 Line detection masks.

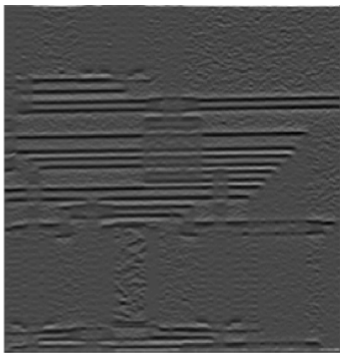
Question: Which filter mask above will give the maximum response and help you detect the lines in this image?

Answer: the horizontal filter on the left.

Line Detection: Line Filter



Original.



Horizontal Line Filter.

What are the edge maps of this image and its noisy version?

Original

With Noise

Edges via Gradient Magnitude

$$\left|\nabla I\right| = \sqrt{I_x^2 + I_y^2}$$

$$\nabla I(x,y) = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix} = \begin{pmatrix} I_x \\ I_y \end{pmatrix}$$

Gradient magnitude original.

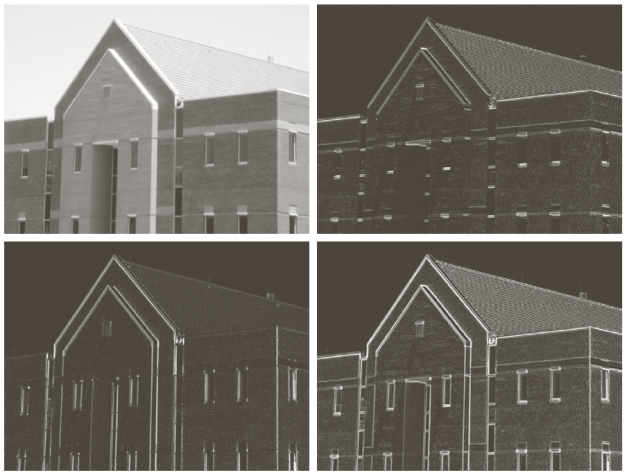
Gradient magnitude noisy.

Edges via Operators

Prewitt, noisy.

Sobel, noisy.

Edges



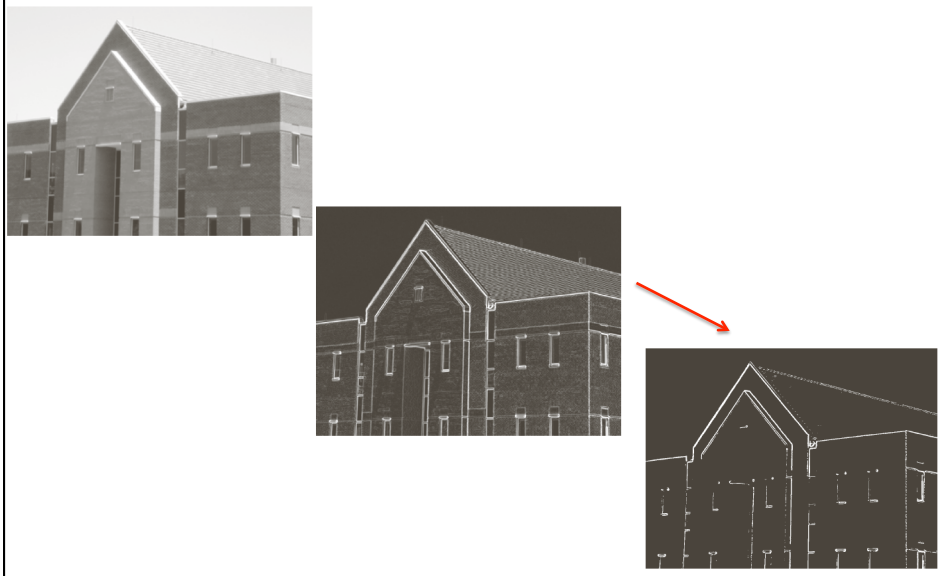
a b
c d

FIGURE 10.18
Same sequence as in Fig. 10.16, but with the original image smoothed using a 5×5 averaging filter prior to edge detection.

a b
c d

FIGURE 10.16
(a) Original image of size 834×1114 pixels, with intensity values scaled to the range $[0, 1]$. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

Binary Edge Map: Threshold the gradient magnitude image



Second-derivative based operator

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

$$L(x,y) = I(x,y) - \frac{1}{4}\Big(I(x,y+1) + I(x,y-1) + I(x+1,y) + I(x-1,y)\Big)$$

0	−1	0
−1	4	−1
0	−1	0

This is called the Laplacian operator:

$$\Delta I(x,y) = \nabla^2 I(x,y) = \frac{\delta^2 I(x,y)}{\delta x^2} + \frac{\delta^2 I(x,y)}{\delta y^2} = I_{xx} + I_{yy}$$

Point Detection

- Based on the 2nd derivatives, i.e. zero crossings using a 4-pt neighborhood

0	1	0
1	−4	1
0	1	0

- Can include diagonal terms in the mask, using an 8-pt neighborhood

1	1	1
1	−8	1
1	1	1

*Recall: for a derivative mask, the coefficients in the mask sum to 0 → the mask response will be zero in areas of constant intensity as desired.

$$\sum_{k=1}^{9} w_k = 0$$

e.g. for a 3x3 mask:

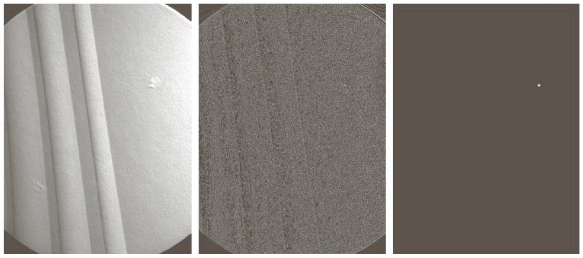
Point Detection

At each pixel coordinate: Compare the absolute value of the mask response with a threshold

if $|R| > T$, $P = 1$

otherwise (o/w) , $P = 0$

1	1	1
1	−8	1
1	1	1



a
b c d

FIGURE 10.4 (a) Point detection (Laplacian) mask. (b) X-ray image of turbine blade with a porosity. The porosity contains a single black pixel. (c) Result of convolving the mask with the image. (d) Result of using Eq. (10.2-8) showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

Laplacian Operators (*)

- Laplacian Masks for 4 and 8 neighborhoods

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \rightarrow$$

−1	−1	−1
−1	8	−1
−1	−1	−1

- Mask with stressed significance of the central pixel or its neighborhood

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \qquad \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix}$$

Advantage of Laplacian over first derivative methods

It is an isotropic filter (i.e. invariant to rotation): i.e. This means that:

If the Laplacian is applied to an image and the image is then rotated:

the same result would be obtained if the image were rotated first and the Laplacian applied second.

Notes on the Laplacian operator

The Laplacian generally is not used in its original form for edge detection:

- As a second-order derivative, the Laplacian typically is unacceptably sensitive to noise
- The magnitude of the Laplacian produces double edges (this complicates segmentation)
- The Laplacian is unable to detect edge direction.

More advanced techniques for Edge Detection

- Earlier operators we discussed have fixed kernel size (e.g. 3x3 or 5x5 or 2x2)
- To take into account the scale of the features, size of the operators should change!
- Typically we UTILIZE GAUSSIAN function to pre-filter
- Small operators detect sharply focused fine edges
- Large operators detect blurry edges

Laplacian of Gaussian (LoG)

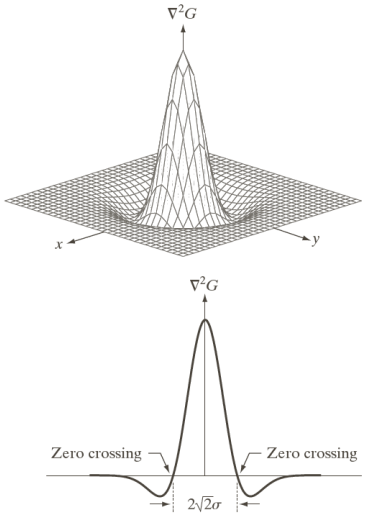
- Use the filter with mask: second derivative of the Gaussian, i.e. the Laplacian of Gaussian
- The Gaussian function:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Laplacian of the Gaussian (LoG):

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Also known as Marr-Hildreth edge detector



0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

a b
c d

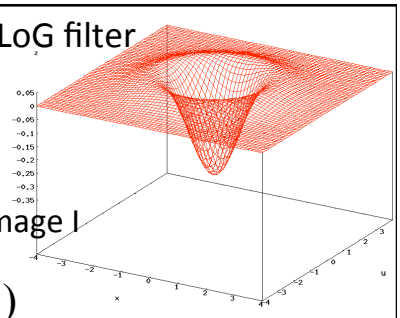
FIGURE 10.21 (a) Three-dimensional plot of the *negative* of the LoG. (b) Negative of the LoG displayed as an image. (c) Cross section of (a) showing zero crossings. (d) 5 × 5 mask approximation to the shape in (a). The negative of this mask would be used in practice.

Laplacian of Gaussian (LoG)

- Masks of arbitrary size can be generated by
  - sampling Gaussian function to the desired nxn size
  - Convolving it with a Laplacian mask
- Nice property:
  - Gaussian part blurs the image, reducing noise

Laplacian of Gaussian (LoG)

LoG filter



- Convolve the LoG filter with input image I

$$J(x,y) = \left[\nabla^2 G(x,y)\right] * I(x,y)$$

- Then find the zero-crossings of J to determine edges in I(x,y)
- Linear operator, so we can rewrite as:

$$J(x,y) = \nabla^2\left[G(x,y) * I(x,y)\right]$$

→ That is equivalently: we smooth the image with a Gaussian filter then compute the Laplacian of the result

Laplacian of Gaussian (LoG)

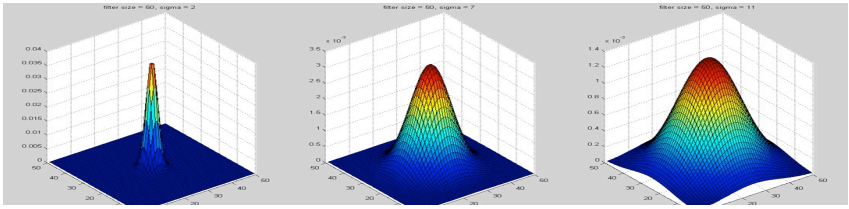
Summary of LoG Edge detection algorithm:

1. Filter the input image with an nxn Gaussian lowpass filter (how to choose n and sigma, see next)
2. Compute the Laplacian of the resulting image from step 1 using the 3x3 Laplacian mask on slide (*)
3. Find the zero-crossings of the image resulting from Step 2 (see the slide 2 next to the following)

Gaussian Function:
How to choose sigma σ (the right scale)?

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1. Filter image with various σ

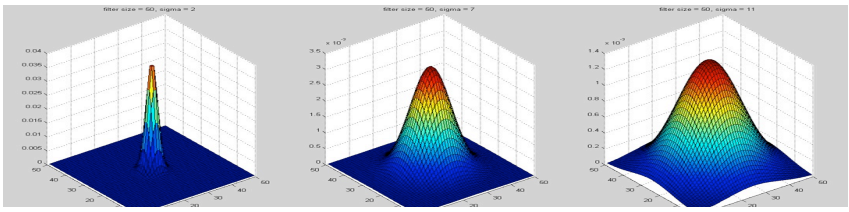


2. Combine the resulting zero-crossings edge maps by keeping only edges that are common to all maps

Gaussian Function:
How to choose sigma σ (the right scale)?

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1. Filter image with various σ



σ increasing →

2. Analyze the image at different scales (i.e. Sigma)
e.g. Apply the edge filters: Combine the resulting edge maps by keeping only edges that are common to all maps

Ties to Multiscale Processing of Images →

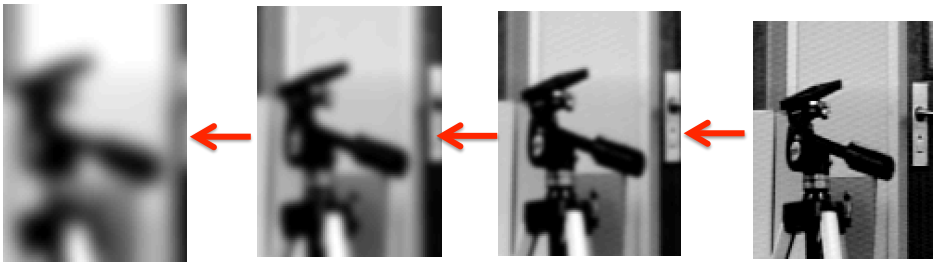
Note: Scale Space Processing Idea

σ

E.g. a large sigma is used

The Gaussian smoothing operation results in the blurring of edges and other sharp discontinuities in an image. The amount of blurring depends on the value of sigma

A larger sigma results in better noise filtering but at the same time loses important edge information, which affects the performance of the edge detector



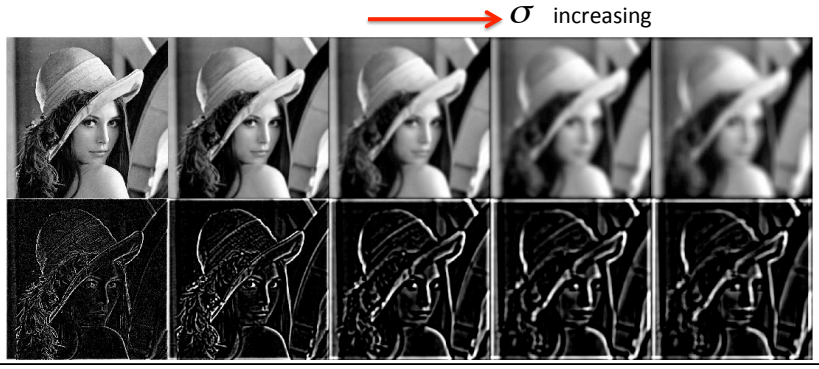
It is reasonable to think of sigma as setting the *scale* at which we preserve information in the convolved image.

Structure on a scale small compared with sigma will be removed, while structure on a larger scale is retained.

Trade-off: If a small filter is used, there is likely to be more noise due to insufficient averaging. For large filters, edges which are close to each other may get merged by smoothing and may be detected as only a single edge.

Scale Space Idea

* Many approaches are developed which apply filtering masks of multiple sizes and then analyze the behaviour of edges at these different *scales* of filtering.
* The basic idea is to exploit the fact that at higher scales, larger filtering masks result in robust but displaced edges. The location of these edges can be determined at smaller scales.



Note on the Gaussian filter design

To specify size of the nxn discrete LoG filter:

Use the fact from probability/statistics:

99.7% of the volume under a 2D Gaussian lies between ±3σ about the mean

A rule of thumb: n is the smallest odd integer greater than or equal to 6σ.

Continue with zero-crossing detection (LoG Edge detection)

To detect zero-crossings of the resulting image J

Use a 3x3 neighborhood centered around each pixel (x,y)

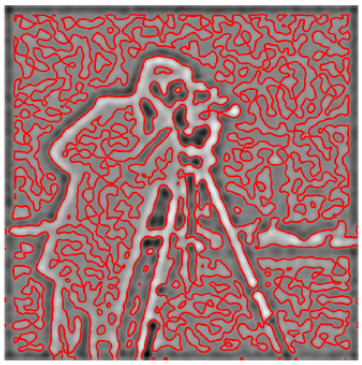
A zero-crossing implies that :

1. signs of at least two of its opposing neighboring pixels must differ:
   e.g. Up/Down, Left/Right, Two diagonals

2. Also use a threshold to compare these, i.e. not only must the signs differ, but absolute value of their numerical difference must exceed a threshold

Laplacian of Gaussian (LoG)

Original.



Noisy.

- Edges are at zero crossings of the LoG output
- Need to be thresholded based on edge strength to remove spurious edges

Laplacian of Gaussian

Small variance.

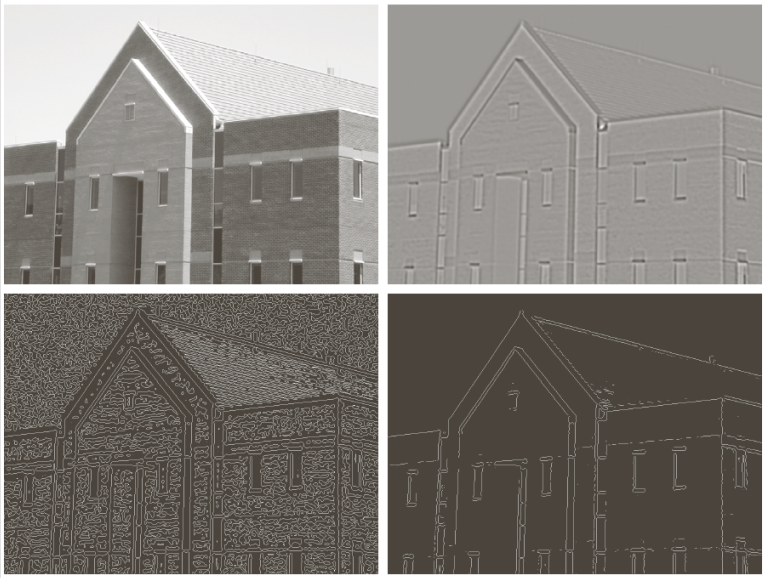
Large variance.

Laplacian of Gaussian

Low threshold.

High threshold.

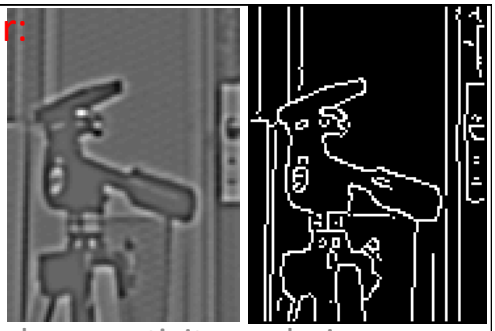
Laplacian of Gaussian



a b
c d

FIGURE 10.22
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]. (b) Results of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

An advanced Edge Detector: The Canny edge detector



Steps:

1. Smoothing with Gaussian
2. Gradient computation
3. Non-maxima suppression
4. Hysteresis thresholding and connectivity analysis
5. Edge thinning

1-2 combines Gaussian smoothing with the simple horizontal and vertical difference operators to give the total smoothed gradient at each pixel.

3 Positions where the gradient is a *local maximum* along gradient direction are then found, and then a thresholding operation is used to find those lines where the gradient is above some limit.

Canny Edge Detection Steps 1-2

1. Filter the image with a Gaussian first: $J(x,y) = [G(x,y) * I(x,y)]$

2. Over image J: compute the image gradient vector: $\vec{\nabla} J(x,y) = (J_x, J_y)$

- Compute the gradient magnitude M $M(x,y) = \sqrt{J_x^2 + J_y^2}$
- and the gradient angle α $\alpha(x,y) = \tan^{-1}\left[\frac{J_y}{J_x}\right]$

Recall the discretized derivatives:

$$J_x = \frac{\partial J}{\partial x} = \frac{J(x+1,y) - J(x-1,y)}{2}$$

$$J_y = \frac{\partial J}{\partial y} = \frac{J(x,y+1) - J(x,y-1)}{2}$$

Canny Algorithm: Nonmaxima Supression (Step 3)

- Non-maxima supression - Retain a point as an edge point if:
  - its gradient magnitude is higher than a threshold
  - its gradient magnitude is a local maxima in the gradient direction

simple thresholding will compute thick edges

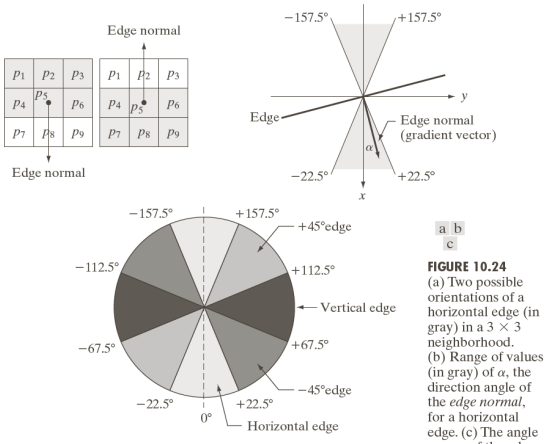
Canny Edge Detection: Non-maxima Suppression (Step 3)

- Idea: Quantize α to 4 basic edge directions --→ d1, d2, d3, d4
- Find direction dk closest to α (x,y)

- If M(x,y) is less than at least one of its two neighbors along dk,

  g(x,y) =0 (suppress)

  Else g(x,y) = M(x,y)
- Threshold g(x,y), which is the output edge image.



a b
c

FIGURE 10.24
(a) Two possible orientations of a horizontal edge (in gray) in a 3 × 3 neighborhood. (b) Range of values (in gray) of α, the direction angle of the *edge normal*, for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3 × 3 neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.

The Canny edge detector Steps

1. Smoothing with Gaussian
2. Gradient computation
3. Non-maxima suppression
4. Hysteresis thresholding and connectivity analysis
5. Edge thinning

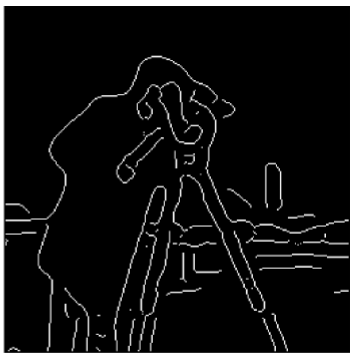
4-5. Not covered in class, but in case you're curious:
Hysteresis Thresholding relates to Connectivity (Linking) Edge Points
Uses an Upper and a Lower threshold to determine both strong enough edge points and those other edge points connected to the current one

Canny Edge Detector

This *edge map* often forms the basis for subsequent processing. For example, simple shapes like straight lines and ellipses can be found, particularly in the images of industrial objects to which computer vision is likely to be applied.

Canny Results Effect of sigma

Small variance.



Large variance.

Effect of sigma

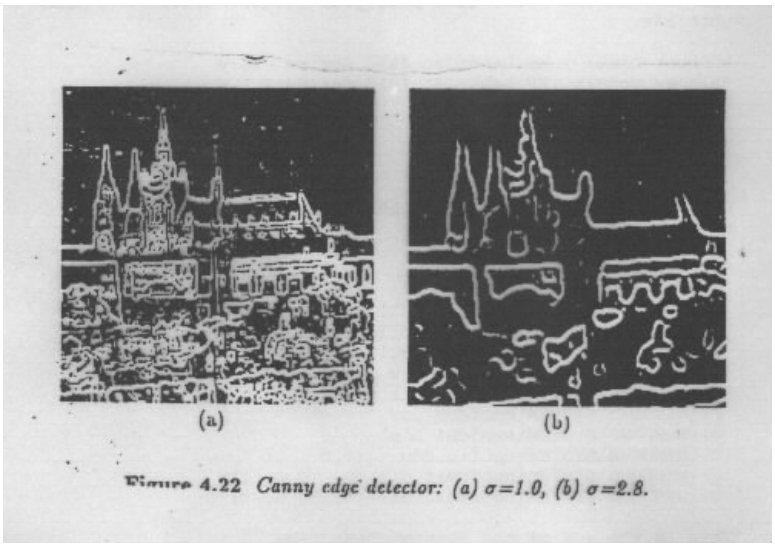
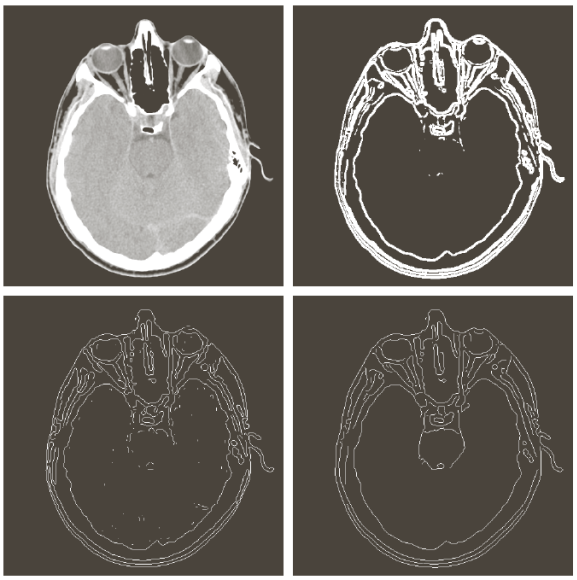
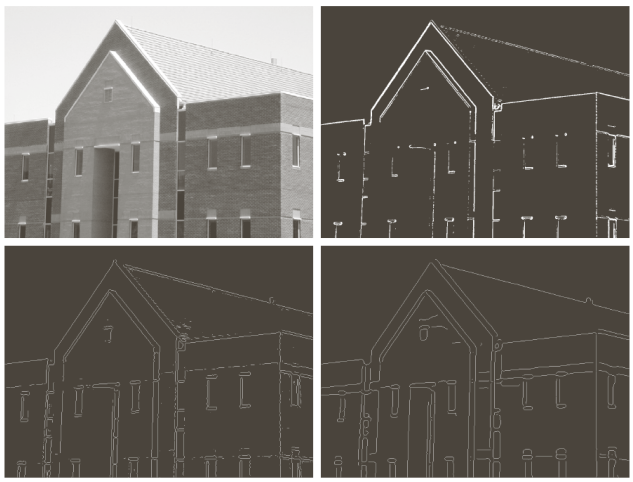


Figure 4.22 *Canny edge detector: (a) σ=1.0, (b) σ=2.8.*



a b
c d

FIGURE 10.26
(a) Original head CT image of size 512 × 512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)



a b
c d

FIGURE 10.25
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm.
Note the significant improvement of the Canny image compared to the other two.

Canny Results

Effect of threshold

Low threshold.

High threshold.

Example:  contains Gaussian noise with a standard deviation of *15*.

Neither the Roberts Cross nor the Sobel operator are able to detect the edges of the object while removing all the noise in the image.

is the result of applying the Sobel operator and thresholding the output at a value of *150*

Applying the Canny operator using a standard deviation of *1.0* yields:

In Summary: we've studied various derivative mask-based and gradient-based edge filters and operators

Remember: Gradient-based edge detection schemes suffer from a number of problems

e.g. issue of how to choose threshold values and the width of your mask, or variance (sigma) of your Gaussian blurring filters

These problems are common to all gradient based edge detectors

but they are still the most commonly used by the computer vision community

NOTE: I kept extra slides for your reference after the END OF LECTURE slide.

END OF LECTURE

Recall Learning objectives of Week 7: Students will be able to:

2. Design and implement various image transforms: point-wise transforms, neighborhood operation-based spatial filters, and geometric transforms over images
3. Define and construct segmentation, feature extraction, and visual motion estimation algorithms to extract relevant information from images

In your HW assignment: you'll get a chance to play with these Image Edge Detection Operations

Reading Assignments:

Your Lecture Notes

Also read relevant sections from [Klette Book] and [Gonzalez and Woods]:

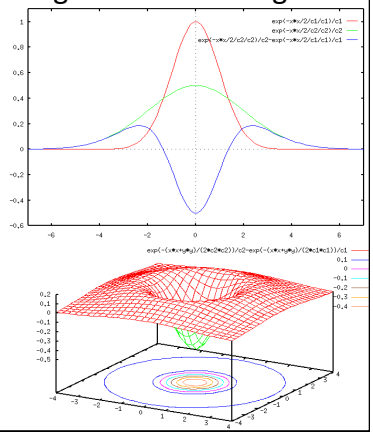
Extra Slides

Difference of Gaussians (DoGs)

DoG is a good approximation to the Laplacian of the Gaussian.

We can look at the form of the 1-D DoG by generating two 1-D masks and subtracting one from the other. It turns out that the best approximation to the Laplacian occurs if the larger mask has a sigma about 1.6 times that of the smaller.

$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

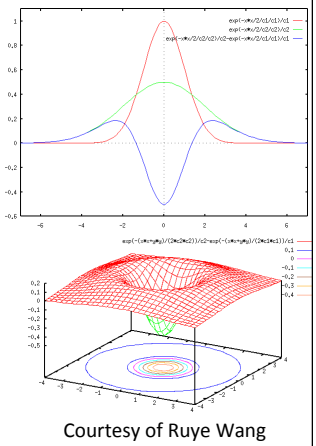


Difference of Gaussians (DoG)

$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

If $\sigma = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right]$ and ampl. equal

approx. = 1.75:1 ratio



Courtesy of Ruye Wang

Then LoG and DoG will have similar profiles

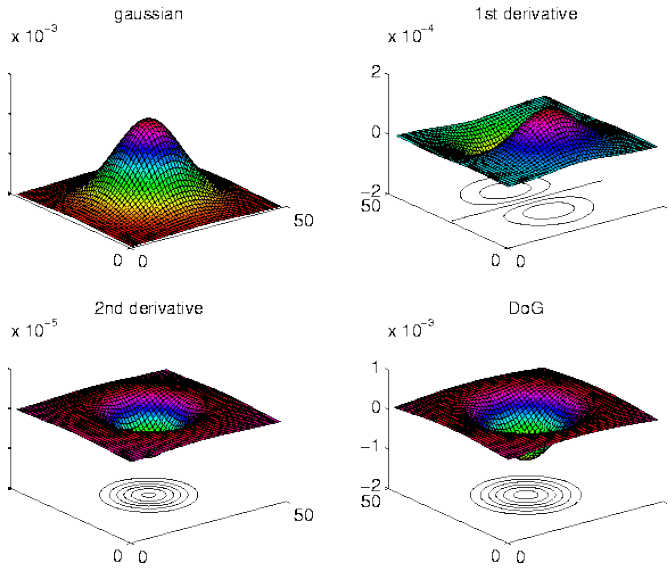
Both LoG and DoG can be implemented with 1D convolutions

2D: n^2MN ops. image size:MxN

1D: 2nMN ops. Filter size: nxn

a b

FIGURE 10.23
(a) Negatives of the LoG (solid) and DoG (dotted) profiles using a standard deviation ratio of 1.75:1. (b) Profiles obtained using a ratio of 1.6:1.



Scale Space created by LoG or DoG

$\sigma = 1,$ $\sigma = 2,$ $\sigma = 4,$

The positions of edges are accurately represented in the small-scale output, which also includes a lot of detailed texture, while the large-scale output retains the main features, but has only approximate positions for the edges.
Processes analogous to DoG convolutions do operate in the early stages of biological vision

The Canny edge detector

It was designed to meet three criteria for edge detection:

1. Low error rate of detection. It should find all edges and nothing but edges.
2. Localization of edges. The distance between actual edges in the image and edges found by this algorithm should be minimized
3. Single response. The algorithm should not return multiple edge pixels when only a single edge exists.

Canny Edge Detector

The Canny operator works in a multi-stage process:

• The image is smoothed by Gaussian convolution.

• A simple 2-D first derivative operator (somewhat like the Roberts Cross) is applied to the smoothed image to highlight regions of the image with high first spatial derivatives. Edges give rise to ridges in the gradient magnitude image.

• The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output, a process known as *non-maximal suppression*.

Canny Edge Detector: Hysteresis Thresholding (step 4)

Canny introduced the idea of *thresholding hysteresis*. This involves having two different threshold values, usually the higher threshold being 3 times the lower. Any pixel in an edge list that has a gradient greater than the higher threshold value are classed as a valid edge point. Any pixels *connected* to these valid edge points that have a gradient value above the lower threshold value are also classed as edge points.

That is, once you have started an edge you don't stop it until the gradient on the edge has dropped considerably.

Canny Edge Detector

The effect of the Canny operator is determined by three parameters:

- The width of the Gaussian kernel used in the smoothing phase,
- The upper and lower thresholds used by the tracker.

Increasing the width of the Gaussian kernel reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image. The localization error in the detected edges also increases slightly as the Gaussian width is increased.

Canny Edge Detector

Usually, the upper tracking threshold can be set quite high, and the lower threshold quite low for good results.

Setting the lower threshold too high will cause noisy edges to break up.

Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output.

Canny Edge Detection: Hysteresis thresholding

- Low threshold T_L and high threshold T_H
- Canny suggests $\frac{T_H}{T_L} = 3 \text{ or } 2$
- Perform thresholding

$$g_{NH} = g_N(x,y) > T_H$$

$$g_{NL} = T_H \geq g_N(x,y) > T_L$$

- $g_{NL}(x,y)$: Weak edges
- $g_{NH}(x,y)$: Strong edges

Connectivity analysis

- g_{NH} will have gaps. Intelligently link them: $g_{NH} \cup g_{NL}$

Canny Edge Detector

Here sigma is, as usual, the scale parameter for Gaussian smoothing, while t1 and t2 are thresholds used for deciding which parts of the edges to keep. In the display, the brightest edges are those with the strongest associated grey-level gradient.

Edge Linking

Edge detectors yield pixels in an image lying on edges.
The next step is to try to collect these pixels together into a set of edges.

Problems:
• Small pieces of edges may be missing,
• Small edge segments may appear to be present due to noise where there is no real edge, *etc*.

Edge Linking

In general, edge linking methods can be classified into two categories:

Local Edge Linkers
-- where edge points are grouped to form edges by considering each point's relationship to any neighbouring edge points.

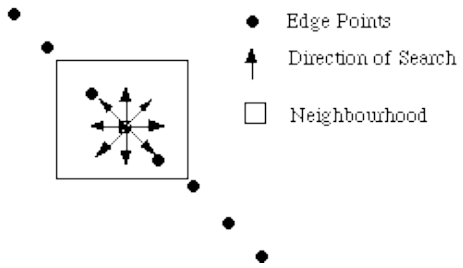
Global Edge Linkers
-- where all edge points in the image plane are considered at the same time and sets of edge points are sought according to some similarity constraint, such as points which share the same edge equation.

Local Edge Linking Methods

Most edge detectors yield information about the magnitude of the gradient at an edge point and, more importantly, the direction of the edge in the locality of the point.

Edge points in a neighbourhood which have similar gradients directions are likely to lie on the same edge.

Local edge linking methods usually start at some arbitrary edge point and consider points in a local neighbourhood for similarity of edge direction



If the points satisfy the similarity constraint then the points are added to the current edge set.

The neighbourhoods based around the recently added edge points are then considered in turn and so on.

If the points do not satisfy the constraint then we conclude we are at the end of the edge, and so the process stops.

Edge Linking: Local Processing

- A pixel with coordinates (s,t) in neighborhood S_{xy} is linked to pixel at (x,y) if both

  magnitude and direction

  $|M(s,t)-M(x,y)| \le E$ $|\alpha(s,t)-\alpha(x,y)| \le A$

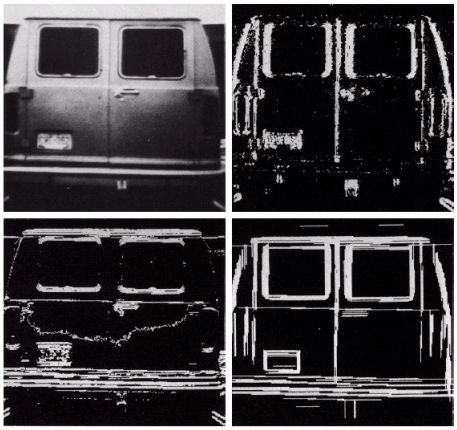
  criteria are satisfied.

- Repeat @ every location of image.

A new starting edge point is found which does not belong to any edge set found so far, and the process is repeated.

The algorithm terminates when all edge points have been linked to one edge or at least have been considered for linking once.

Edge Linking and Boundary Detection



a b
c d

FIGURE 10.16
(a) Input image. (b) G_y component of the gradient. (c) G_x component of the gradient. (d) Result of edge linking. (Courtesy of Perceptics Corporation.)

Linking all points that simultaneously had a gradient value greater than 25 and whose gradient diections did not differ by more than 15^0.