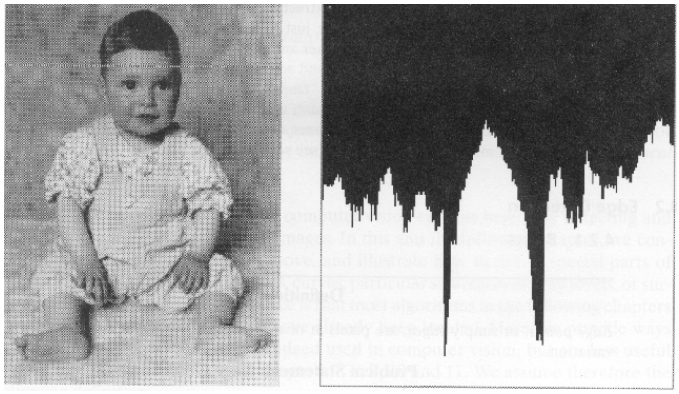
Edge detection

(Trucco, Chapt 4 AND Jain et al., Chapt 5)

• Definition of edges

- Edges are significant local changes of intensity in an image.

- Edges typically occur on the boundary between two different regions in an image.

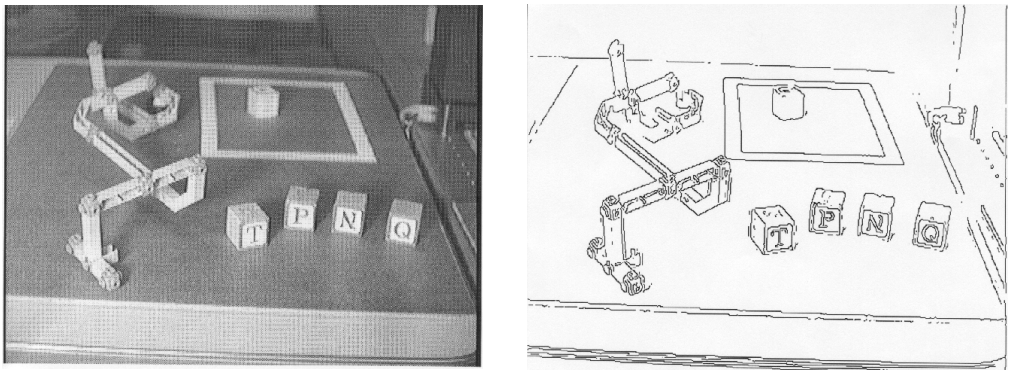


• Goal of edge detection

- Produce a line drawing of a scene from an image of that scene.

- 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).

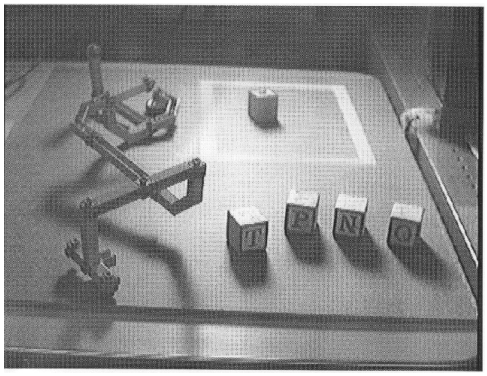


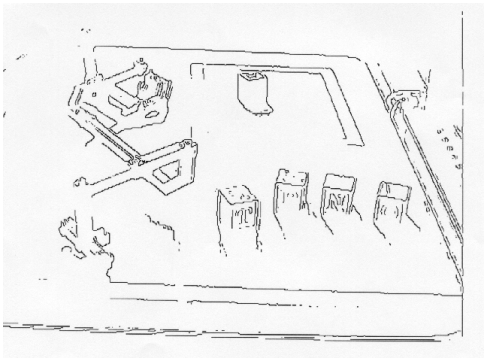
• What causes intensity changes?

- Various physical events cause intensity changes.

- Geometric events
  * object boundary (discontinuity in depth and/or surface color and texture)
  * surface boundary (discontinuity in surface orientation and/or surface color and texture)

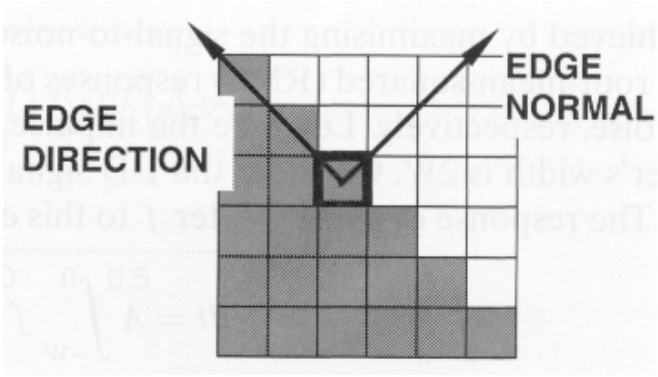
- Non-geometric events
  * specularity (direct reflection of light, such as a mirror)
  * shadows (from other objects or from the same object)
  * inter-reflections





• Edge descriptors

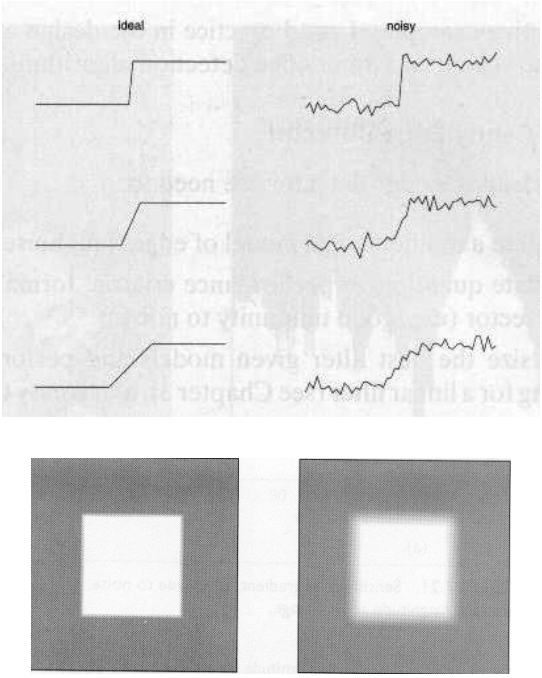
Edge normal: unit vector in the direction of maximum intensity change.
Edge direction: unit vector to perpendicular to the edge normal.
Edge position or center: the image position at which the edge is located.
Edge strength: related to the local image contrast along the normal.



• Modeling intensity changes

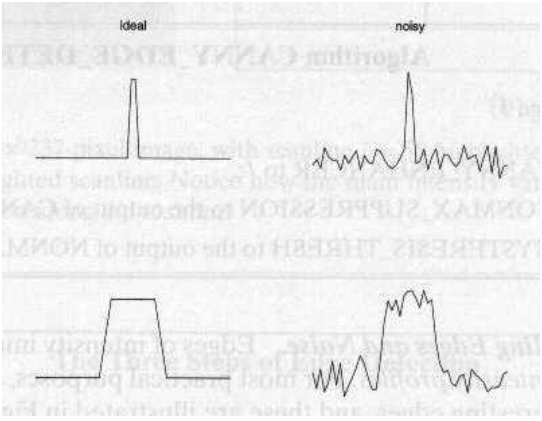
- Edges can be modeled according to their intensity profiles.

Step edge: the image intensity abruptly changes from one value to one side of the discontinuity to a different value on the opposite side.

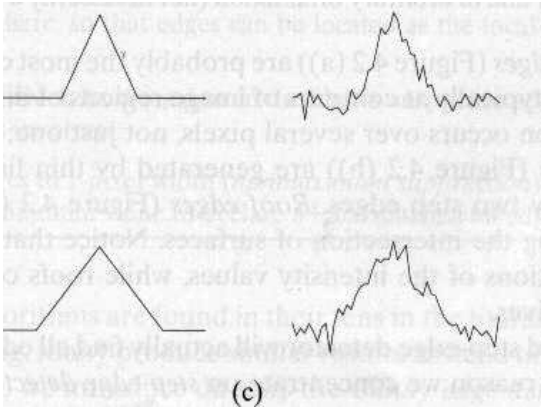


Ramp edge: a step edge where the intensity change is not instantaneous but occur over a finite distance.

Ridge edge: the image intensity abruptly changes value but then returns to the starting value within some short distance (generated usually by lines).



Roof edge: a ridge edge where the intensity change is not instantaneous but occur over a finite distance (generated usually by the intersection of surfaces).



• The four steps of edge detection

(1) Smoothing: suppress as much noise as possible, without destroying the true edges.

(2) Enhancement: apply a filter to enhance the quality of the edges in the image (sharpening).

(3) Detection: determine which edge pixels should be discarded as noise and which should be retained (usually, thresholding provides the criterion used for detection).

(4) Localization: determine the exact location of an edge (*sub-pixel* resolution might be required for some applications, that is, estimate the location of an edge to better than the spacing between pixels). Edge thinning and linking are usually required in this step.

• Edge detection using derivatives

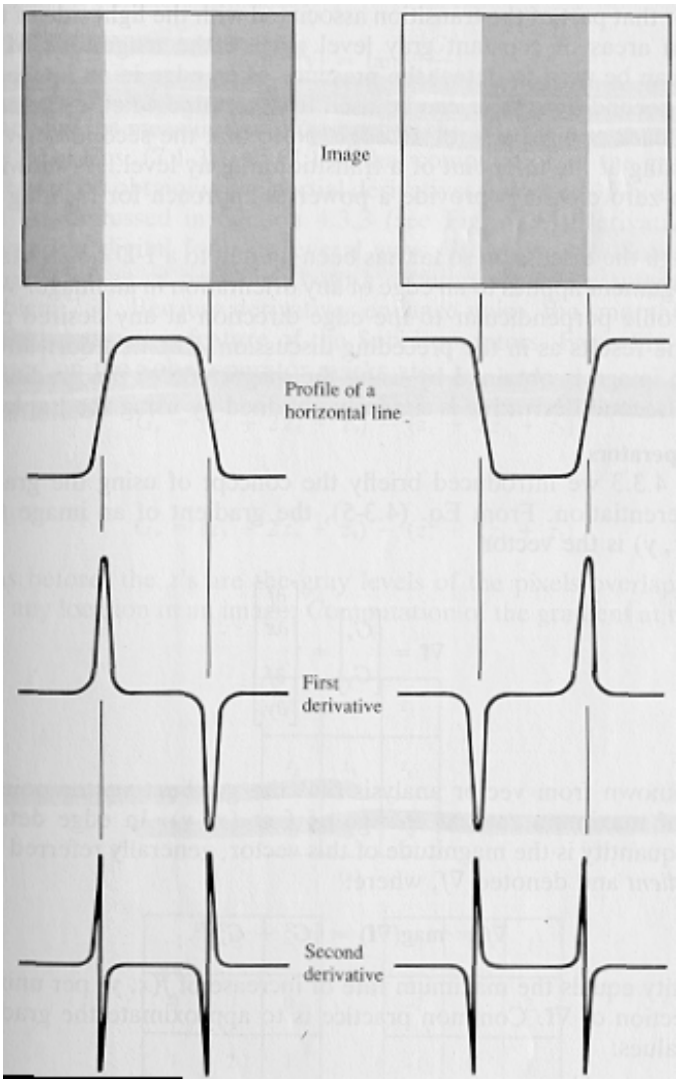
- Calculus describes changes of continuous functions using *derivatives*.

- An image is a 2D function, so operators describing edges are expressed using *partial derivatives*.

- Points which lie on an edge can be detected by:

(1) detecting local maxima or minima of the first derivative

(2) detecting the zero-crossing of the second derivative



• **Differencing 1D signals** (see also Trucco, Appendix A.2)

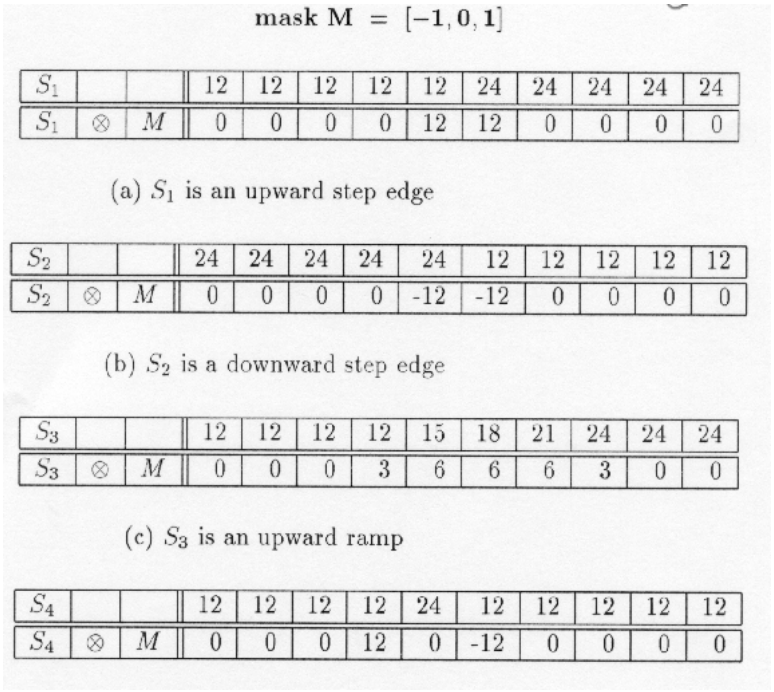
- To compute the derivative of a signal, we approximate the derivative by finite differences:

Computing the 1st derivative:

$$f'(x) = \lim_{h->0} \frac{f(x+h) - f(x)}{h} \approx f(x+1) - f(x) \quad (h=1)$$

mask: $[-1 \quad 1]$

- Examples using the edge models and the mask $[-1 \quad 0 \quad 1]$ (**centered** about x):



mask M = [−1, 0, 1]

S_1			12	12	12	12	12	24	24	24	24	24
S_1	⊗	M	0	0	0	0	12	12	0	0	0	0

(a) S_1 is an upward step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	⊗	M	0	0	0	0	-12	-12	0	0	0	0

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	⊗	M	0	0	0	3	6	6	6	3	0	0

(c) S_3 is an upward ramp

S_4			12	12	12	12	24	12	12	12	12	12
S_4	⊗	M	0	0	0	12	0	-12	0	0	0	0

Computing the 2nd derivative:

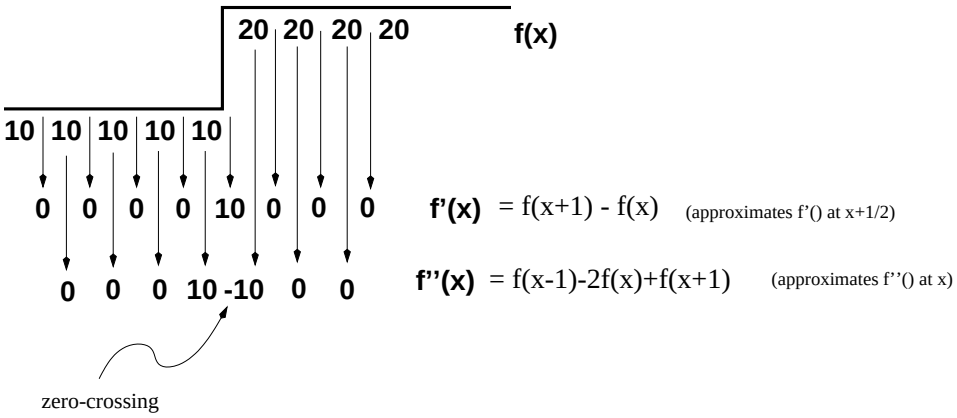
$$f''(x) = \lim_{h->0} \frac{f'(x+h) - f'(x)}{h} \approx f'(x+1) - f'(x) =$$

$$f(x+2) - 2f(x+1) + f(x) \quad (h=1)$$

- This approximation is **centered** about $x+1$; by replacing $x+1$ by x we obtain:

$$f''(x) \approx f(x+1) - 2f(x) + f(x-1)$$

mask: [1 −2 1]



- Examples using the edge models:

mask M = [−1, 2, −1]

S_1			12	12	12	12	12	24	24	24	24	24
S_1	⊗	M	0	0	0	0	-12	12	0	0	0	0

(a) S_1 is an upward step edge

S_2			24	24	24	24	24	12	12	12	12	12
S_2	⊗	M	0	0	0	0	12	-12	0	0	0	0

(b) S_2 is a downward step edge

S_3			12	12	12	12	15	18	21	24	24	24
S_3	⊗	M	0	0	0	-3	0	0	0	3	0	0

(c) S_3 is an upward ramp

S_4			12	12	12	12	24	12	12	12	12	12
S_4	⊗	M	0	0	0	-12	24	-12	0	0	0	0

Edge detection using the gradient

• Definition of the gradient

- The gradient is a vector which has certain magnitude and direction:

$$\nabla f = \begin{pmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{pmatrix}$$

$$magn(\nabla f) = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2} = \sqrt{M_x{}^2 + M_y{}^2}$$

$$dir(\nabla f) = \tan^{-1}(M_y/M_x)$$

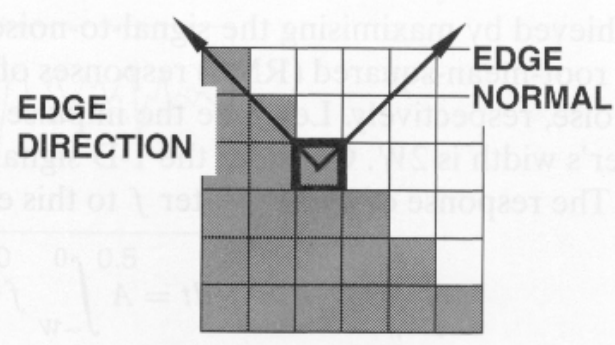
- To save computations, the magnitude of gradient is usually approximated by:

$$magn(\nabla f) \approx |M_x| + |M_y|$$

• Properties of the gradient

- The magnitude of gradient provides information about the strength of the edge.

- The direction of gradient is always perpendicular to the direction of the edge (the edge direction is rotated with respect to the gradient direction by -90 degrees).



• Estimating the gradient with finite differences

$$\frac{\partial f}{\partial x} = \lim_{h->0} \frac{f(x+h, y) - f(x, y)}{h}$$

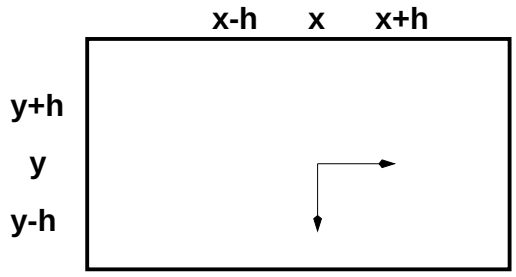
$$\frac{\partial f}{\partial y} = \lim_{h->0} \frac{f(x, y+h) - f(x, y)}{h}$$

- The gradient can be approximated by *finite differences*:

$$\frac{\partial f}{\partial x} = \frac{f(x+h_x, y) - f(x, y)}{h_y} = f(x+1, y) - f(x, y), \ (h_x=1)$$

$$\frac{\partial f}{\partial y} = \frac{f(x, y+h_y) - f(x, y)}{h_y} = f(x, y+1) - f(x, y), \ (h_y=1)$$

- Using pixel-coordinate notation (**remember:** j corresponds to the x direction and i to the negative y direction):



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

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

Example

Suppose we want to approximate the gradient magnitude at z_5

Z1	**Z2**	**Z3**
Z4	**Z5**	**Z6**
Z7	**Z8**	**Z9**

$$\frac{\partial I}{\partial x} = z_6 - z_5, \quad \frac{\partial I}{\partial y} = z_5 - z_8$$

$$magn(\nabla I) = \sqrt{(z_6 - z_5)^2 + (z_5 - z_8)^2}$$

- We can implement $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ using the following masks:

-1	**1**

1
-1

(*note*: M_x is the approximation at $(i, j + 1/2)$ and M_y is the approximation at $(i + 1/2, j)$)

• The Roberts edge detector

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

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

- This approximation can be implemented by the following masks:

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

(*note*: M_x and M_y are is approximations at $(i + 1/2, j + 1/2)$)

• The Prewitt edge detector

- Consider the arrangement of pixels about the pixel (i, j):

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & [i, j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

- The partial derivatives can be computed by:

$$M_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$
$$M_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2)$$

- The constant c implies the emphasis given to pixels closer to the center of the mask.

- Setting $c = 1$, we get the Prewitt operator:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

(*note*: M_x and M_y are approximations at (i, j))

• The Sobel edge detector

- Setting $c = 2$, we get the Sobel operator:

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

(*note*: M_x and M_y are approximations at (i, j))

• Main steps in edge detection using masks

(1) Smooth the input image ($\hat{f}(x, y) = f(x, y) * G(x, y)$)

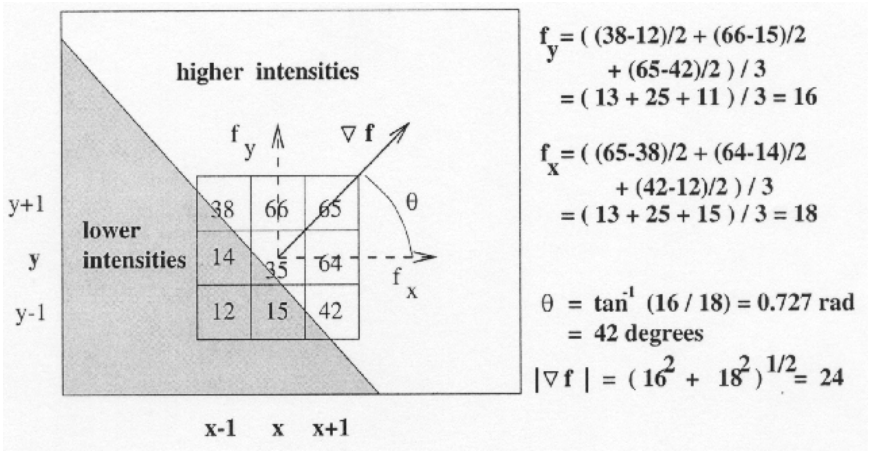
(2) $\hat{f}_x = \hat{f}(x, y) * M_x(x, y)$

(3) $\hat{f}_y = \hat{f}(x, y) * M_y(x, y)$

(4) $magn(x, y) = |\hat{f}_x| + |\hat{f}_y|$

(5) $dir(x, y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$

(6) If $magn(x, y) > T$, then possible edge point



(an example using the Prewitt edge detector - don't divide by 2)

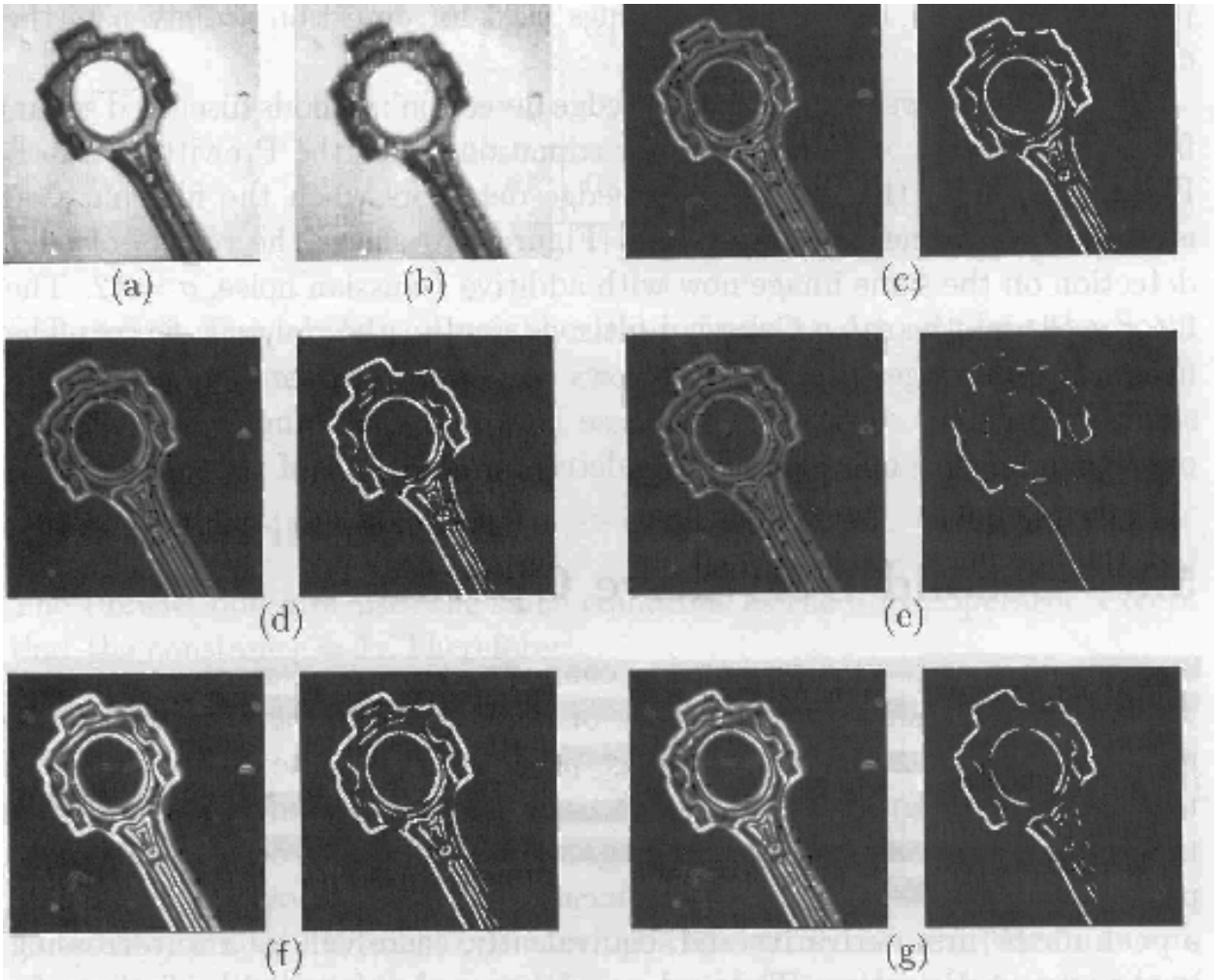


Figure 5.4: A comparison of various edge detectors. (a) Original image. (b) Filtered image. (c) Simple gradient using 1×2 and 2×1 masks, $T = 32$. (d) Gradient using 2×2 masks, $T = 64$. (e) Roberts cross operator, $T = 64$. (f) Sobel operator, $T = 225$. (g) Prewitt operator, $T = 225$.

(with noise filtering)

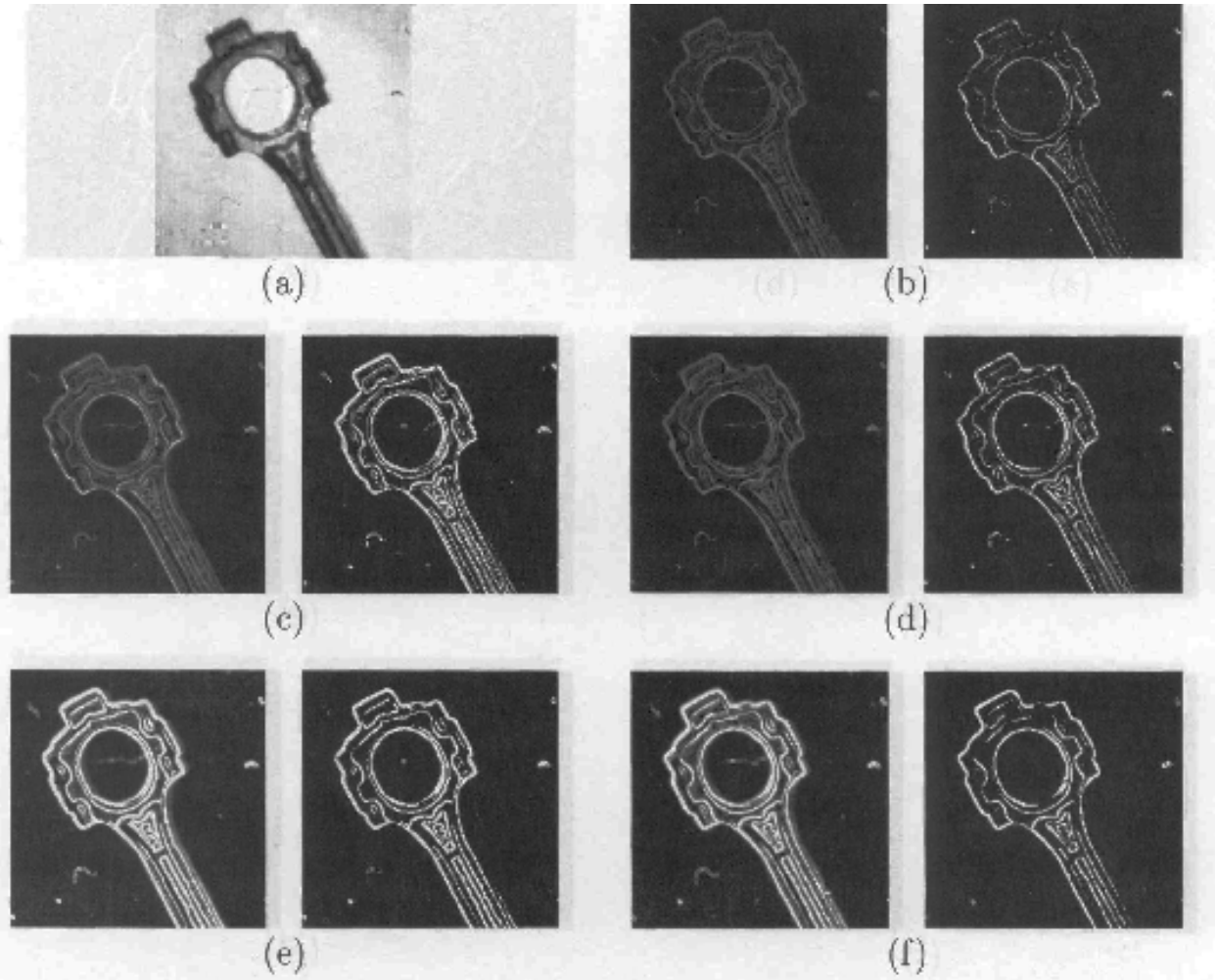
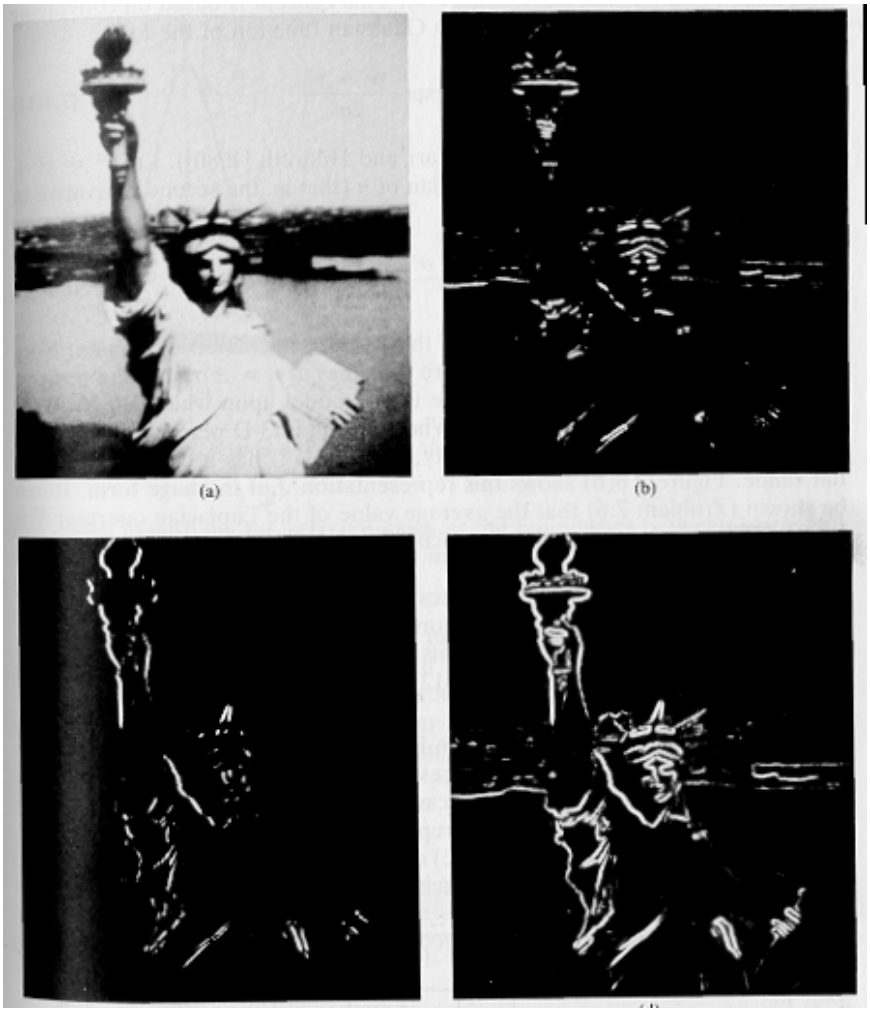


Figure 5.5: A comparison of various edge detectors without filtering. (a) Original image. (b) Simple gradient using 1×2 and 2×1 masks, $T = 64$. (c) Gradient using 2×2 masks, $T = 64$. (d) Roberts cross operator, $T = 64$. (e) Sobel operator, $T = 225$. (f) Prewitt operator, $T = 225$.

(without noise filtering)

• Isotropic property of gradient magnitude

- The magnitude of gradient is an *isotropical* operator (it detects edges in any direction !!)

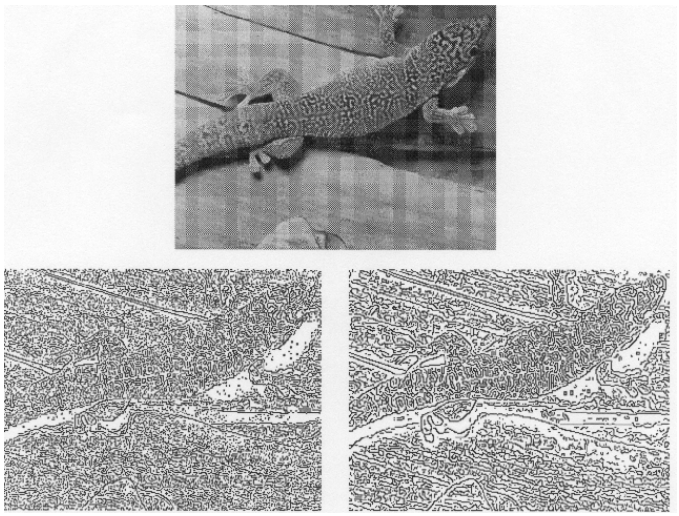


• Some practical issues

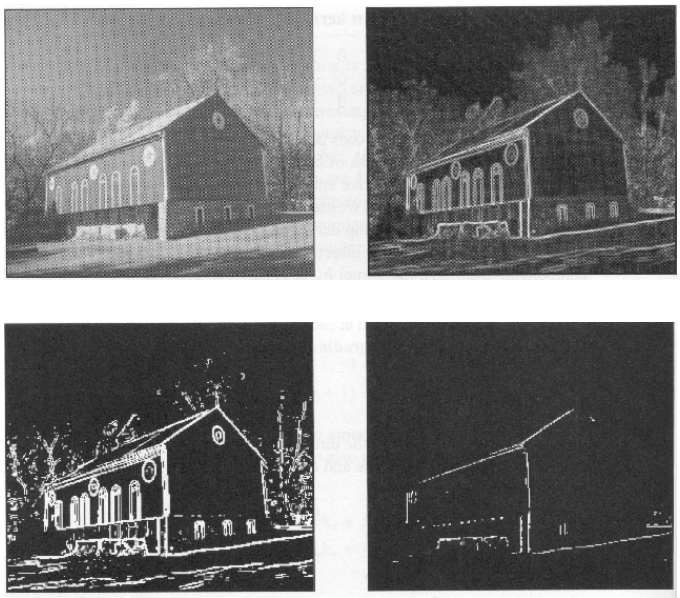
- The differential masks act as high-pass filters which tend to amplify noise.

- To reduce the effects of noise, the image needs to be smoothed first with a low-pass filter.

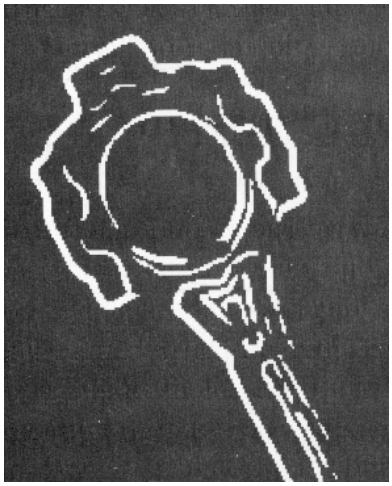
(1) The noise suppression-localization tradeoff: a lar ger filter reduces noise, but worsens localization (i.e., it adds uncertainty to the location of the edge) and vice versa.

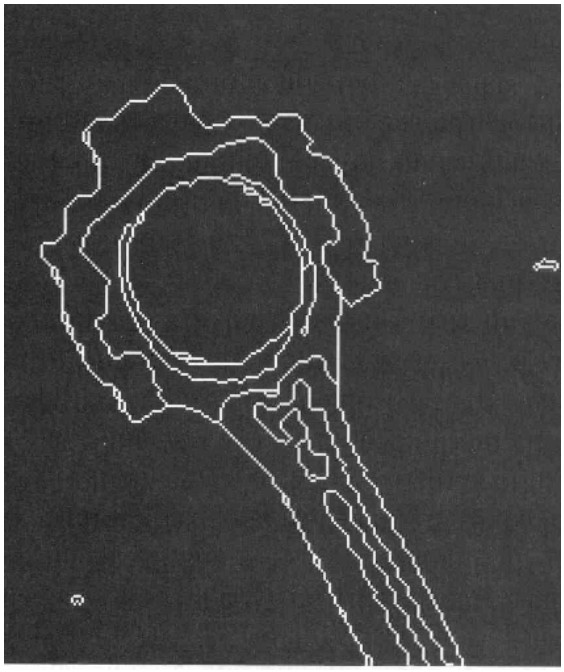


- (2) How should we choose the threshold?



- (3) Edge thinning and linking are required to obtain good contours.





• Criteria for optimal edge detection

(1) 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).

(2) Good localization: the edges detected must be as close as possible to the true edges.

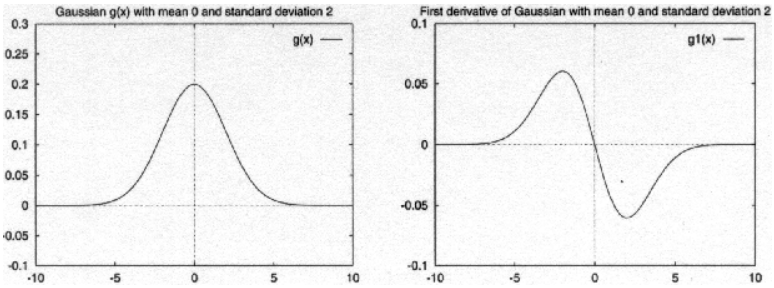
Single response constraint: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge (created by noise).

The Canny edge detector

- This is probably the most widely used edge detector in computer vision.

- 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.

- His analysis is based on "step-edges" corrupted by "additive Gaussian noise".



Algorithm

1. Compute f_x and f_y

$$f_x = \frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x} G = f * G_x$$

$$f_y = \frac{\partial}{\partial y}(f * G) = f * \frac{\partial}{\partial y} G = f * G_y$$

$G(x, y)$ is the Gaussian function

$G_x(x, y)$ is the derivate of $G(x, y)$ with respect to x: $G_x(x, y) = \frac{-x}{\sigma^2} G(x, y)$

$G_y(x, y)$ is the derivate of $G(x, y)$ with respect to y: $G_y(x, y) = \frac{-y}{\sigma^2} G(x, y)$

2. Compute the gradient magnitude

$$magn(i, j) = \sqrt{{f_x}^2 + {f_y}^2}$$

3. Apply non-maxima suppression.

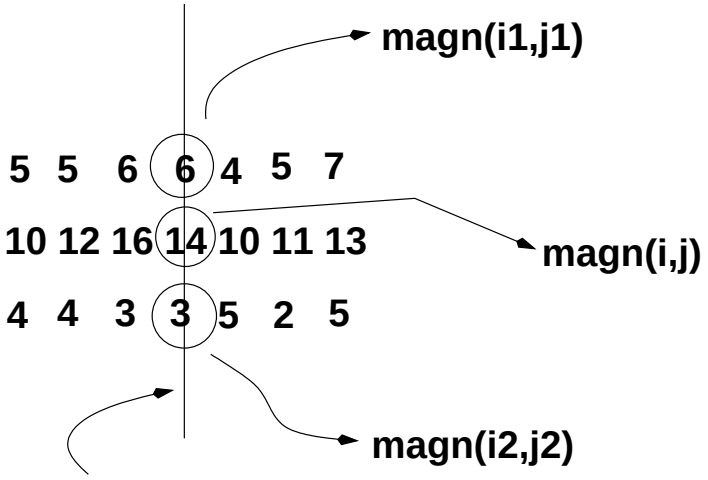
4. Apply hysteresis thresholding/edge linking.

• Non-maxima suppression

- To find the edge points, we need to find the local maxima of the gradient magnitude.

- Broad ridges must be thinned so that only the magnitudes at the points of greatest local change remain.

- All values along the direction of the gradient that are not peak values of a ridge are suppressed.



Algorithm

For each pixel (x,y) do:

if $magn(i, j) < magn(i_1, j_1)$ or $magn(i, j) < magn(i_2, j_2)$
  then $I_N(i, j) = 0$
else $I_N(i, j) = magn(i, j)$

• Hysteresis thresholding/Edge Linking

- The output of non-maxima suppression still contains the local maxima created by noise.

- Can we get rid of them just by using a single threshold?

    * if we set a low threshold, some noisy maxima will be accepted too.

    * if we set a high threshold, true maxima might be missed (the value of true maxima will fluctuate above and below the threshold, fragmenting the edge).

- A more effective scheme is to use two thresholds:

    * a low threshold t_l
    * a high threshold t_h
    * usually, $t_h \approx 2t_l$

Algorithm

1. Produce two thresholded images $I_1(i, j)$ and $I_2(i, j)$.

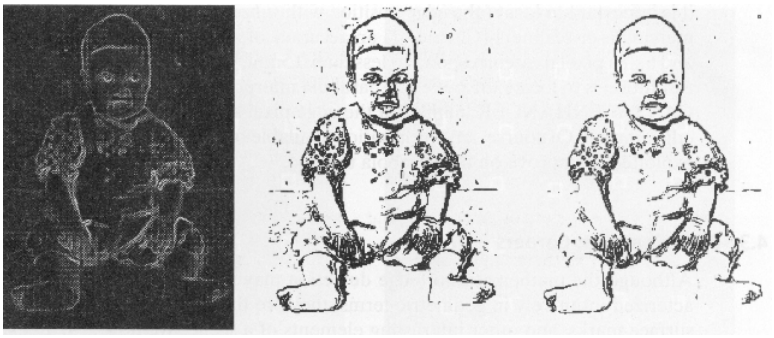
(note: since $I_2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

2. Link the edges in $I_2(i, j)$ into contours

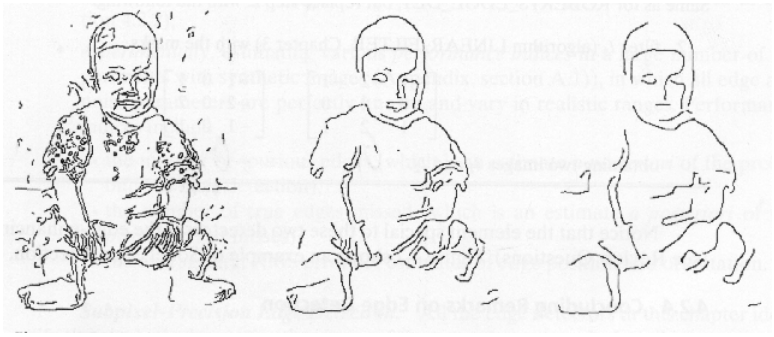
    2.1 Look in $I_1(i, j)$ when a gap is found.

    2.2 By examining the 8 neighbors in $I_1(i, j)$, gather edge points from $I_1(i, j)$ until the gap has been bridged to an edge in $I_2(i, j)$.

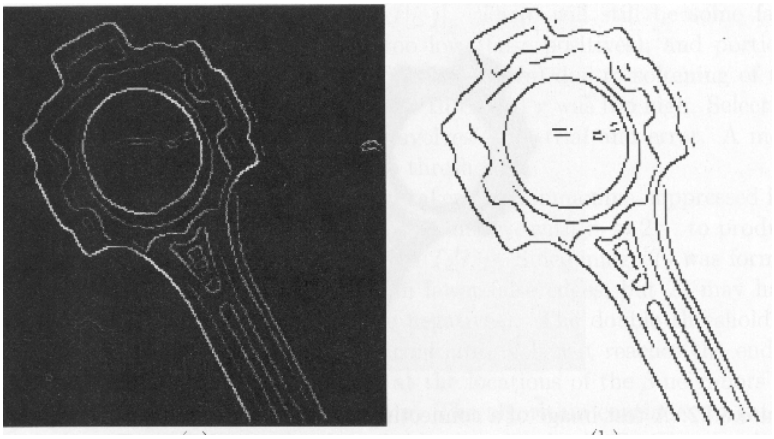
- The algorithm performs edge linking as a by-product of double-thresholding !!



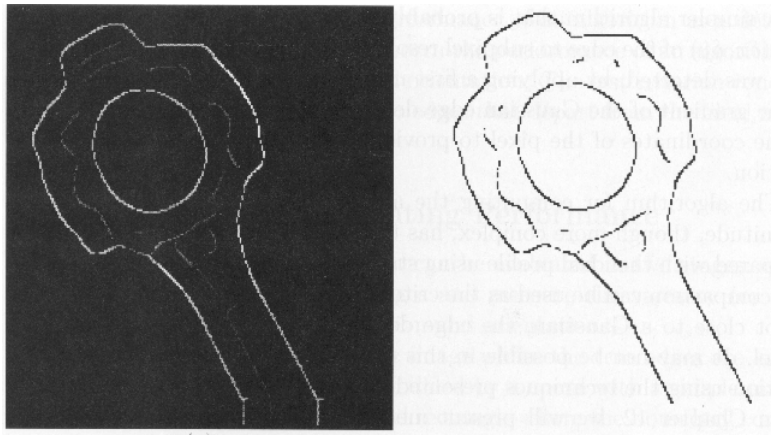
(left:Sobel, middle: thresh=35, right: thersh=50)



(Canny - left:σ=1, middle: σ=2, right: σ=3)



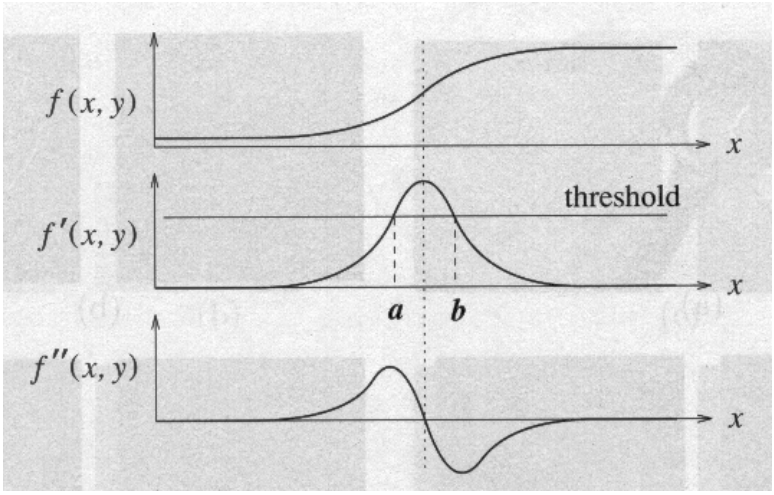
(Canny - 7x7 Gaussian, more details)



(Canny - 31x31 Gaussian, less details)

Edge detection using the second derivative

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



- There are two operators in 2D that correspond to the second derivative:

  * Laplacian
  * Second directional derivative

• The Laplacian

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

- Approximating $\nabla^2 f$:

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

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

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

Example:

Z1	**Z2**	**Z3**
Z4	**Z5**	**Z6**
Z7	**Z8**	**Z9**

$$\nabla^2 f = -4z_5 + (z_2 + z_4 + z_6 + z_8)$$

- The Laplacian can be implemented using the mask shown below:

0	**1**	**0**
1	**-4**	**1**
0	**1**	**0**

Example:

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	-
-	-	-	-	-	-

• Properties of the Laplacian

- It is an isotropic operator.

- It is cheaper to implement (one mask only).

- It does not provide information about edge direction.

- It is more sensitive to noise (differentiates twice).

• The Laplacian-of-Gaussian (LOG)

- To reduce the noise effect, the image is first smoothed with a low-pass filter.

- In the case of the LOG, the low-pass filter is chosen to be a Gaussian.

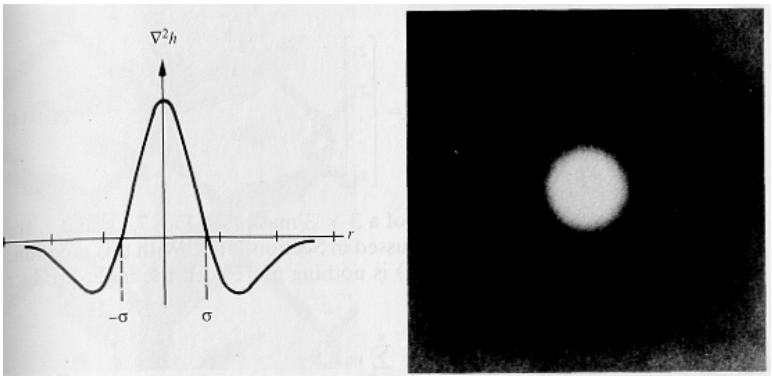
$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

(σ determines the degree of smoothing, mask size increases with σ)

- It can be shown that:

$$\nabla^2[f(x, y) * G(x, y)] = \nabla^2 G(x, y) * f(x, y)$$

$$\nabla^2 G(x, y) = (\frac{r^2 - \sigma^2}{\sigma^4}) e^{-r^2/2\sigma^2}, (r^2 = x^2 + y^2)$$

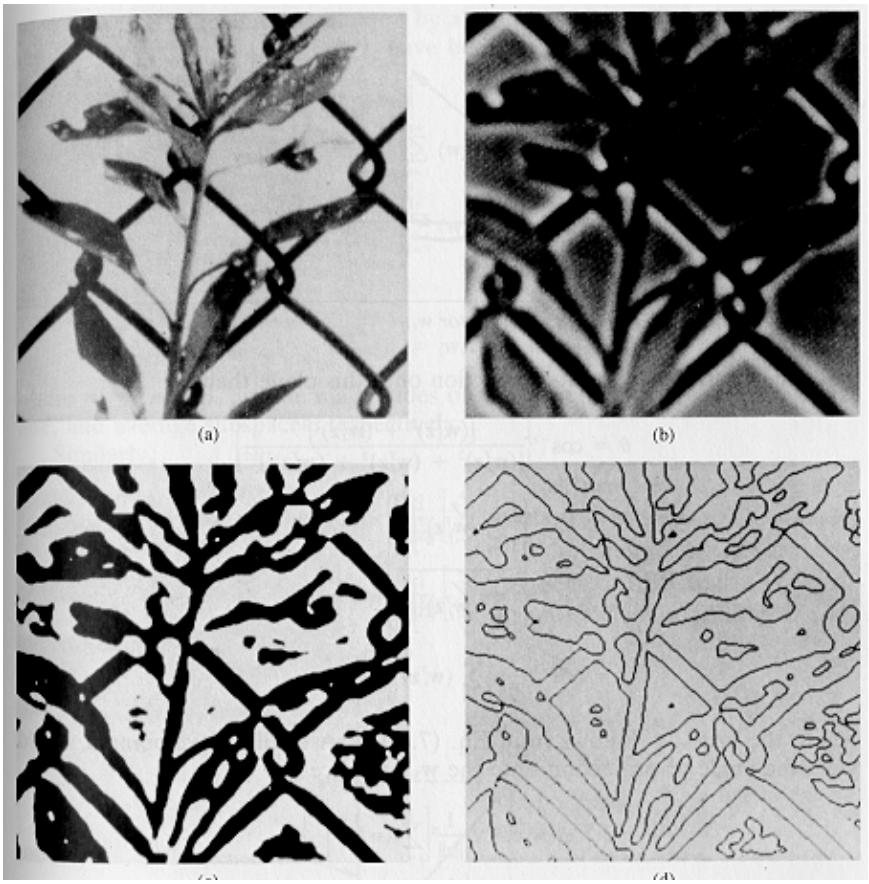


5×5 Laplacian of Gaussian mask

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

17×17 Laplacian of Gaussian mask

0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	4	12	21	24	21	12	4	−3	−3	−3	−1	−1
−1	−1	−3	−3	−2	2	10	18	21	18	10	2	−2	−3	−3	−1	−1
−1	−1	−3	−3	−3	0	4	10	12	10	4	0	−3	−3	−3	−1	−1
0	−1	−2	−3	−3	−3	0	2	4	2	0	−3	−3	−3	−2	−1	0
0	−1	−1	−2	−3	−3	−3	−2	−3	−2	−3	−3	−3	−2	−1	−1	0
0	0	−1	−1	−2	−3	−3	−3	−3	−3	−3	−3	−2	−1	−1	0	0
0	0	−1	−1	−1	−2	−3	−3	−3	−3	−3	−2	−1	−1	−1	0	0
0	0	0	0	−1	−1	−1	−1	−1	−1	−1	−1	−1	0	0	0	0
0	0	0	0	0	0	−1	−1	−1	−1	−1	0	0	0	0	0	0



• Gradient vs LOG: a comparison

- Gradient works well when the image contains sharp intensity transitions and low noise

- Zero-crossings of LOG offer better localization, especially when the edges are not very sharp

2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8
2	2	2	2	2	8	8	8	8	8

A sample image containing a vertical step edge.

0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0
0	0	0	6	−6	0	0	0

2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8
2	2	2	2	2	5	8	8	8	8

A sample image containing a vertical ramp edge.

0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0
0	0	0	3	0	−3	0	0

• The second directional derivative

- This is the second derivative computed in the direction of the gradient.

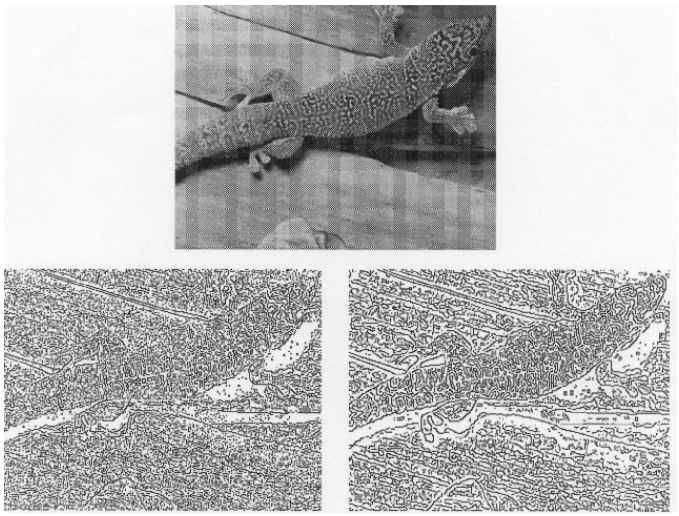
$$\frac{\partial^2}{\partial n^2} = \frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}}{f_x^2 f_y^2}$$

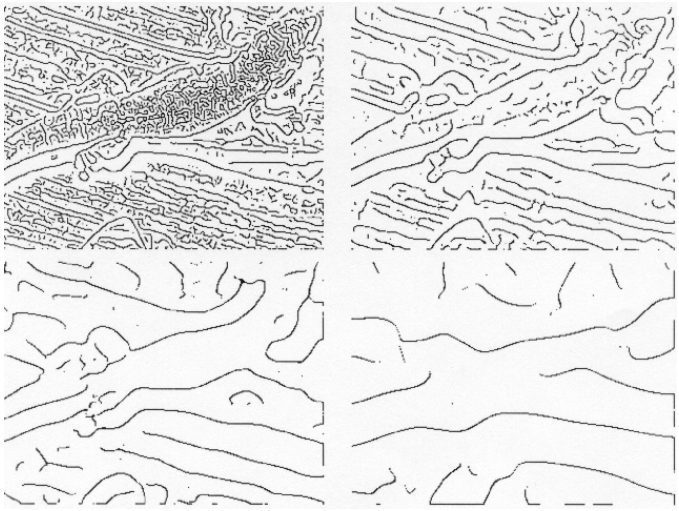
• Multiscale processing (scale space)

- A serious practical problem with any edge detector is the matter of choosing the *scale* of smoothing (e.g.,the value of σ using a Gaussian).

- For many applications, it is desirable to be able to process an image at multiple scales.

- We determine which edges are most significant in terms of the range of scales over which they are observed to occur.





(Canny edges at multiple scales of smoothing, σ=0.5, 1, 2, 4, 8, 16)