Image Processing
(Year III, 2-nd semester)

Lecture XI- XII: EDGE BASED SEGMENTATION

Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy

Edge detection

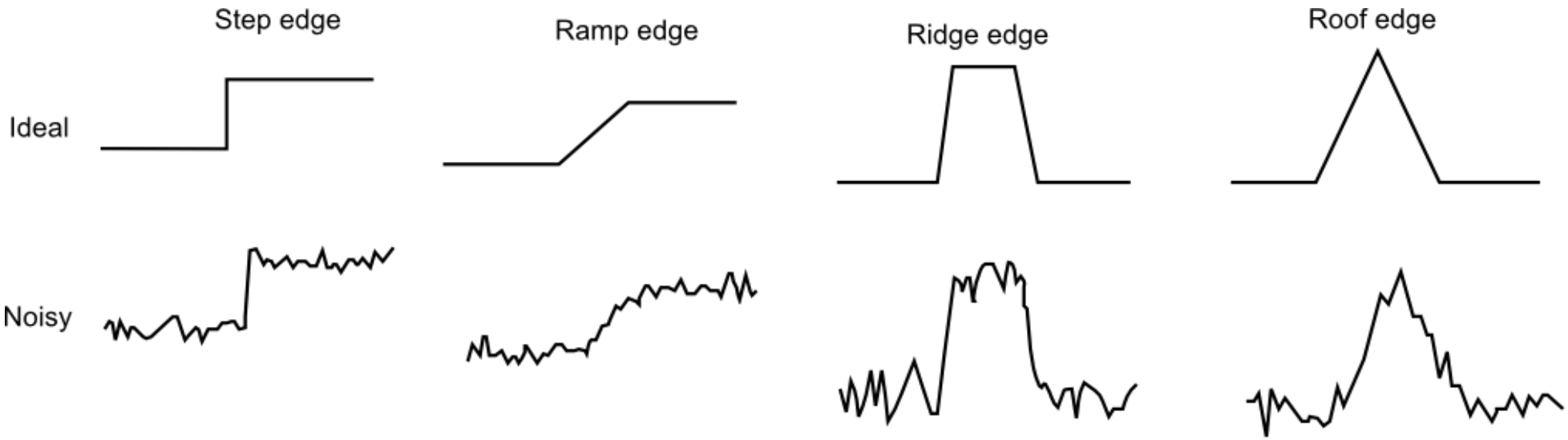
Edges are generated by:

- changes in the surface reflectance
- changes in the surface orientation
- an object being covered partially by another
- indirect covering by shadows

Edges are rapidly varying parts of the image so are represented by high spatial frequencies.

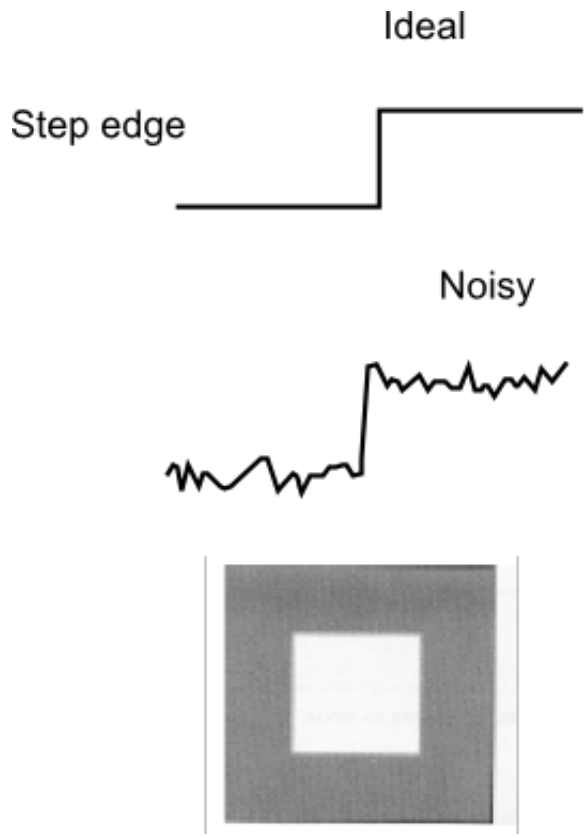
Typical edge profiles

Edges can be modeled according to their intensity profiles:



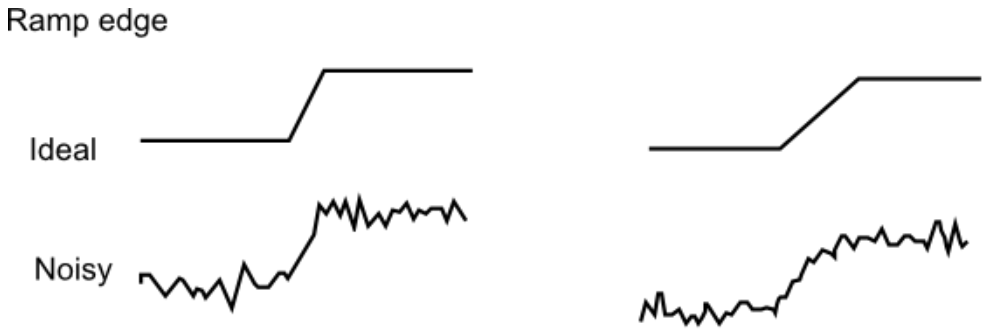
Edge types: Step edge

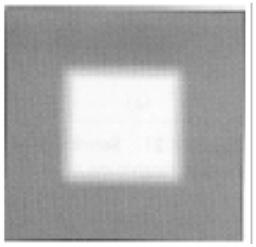
The image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side.



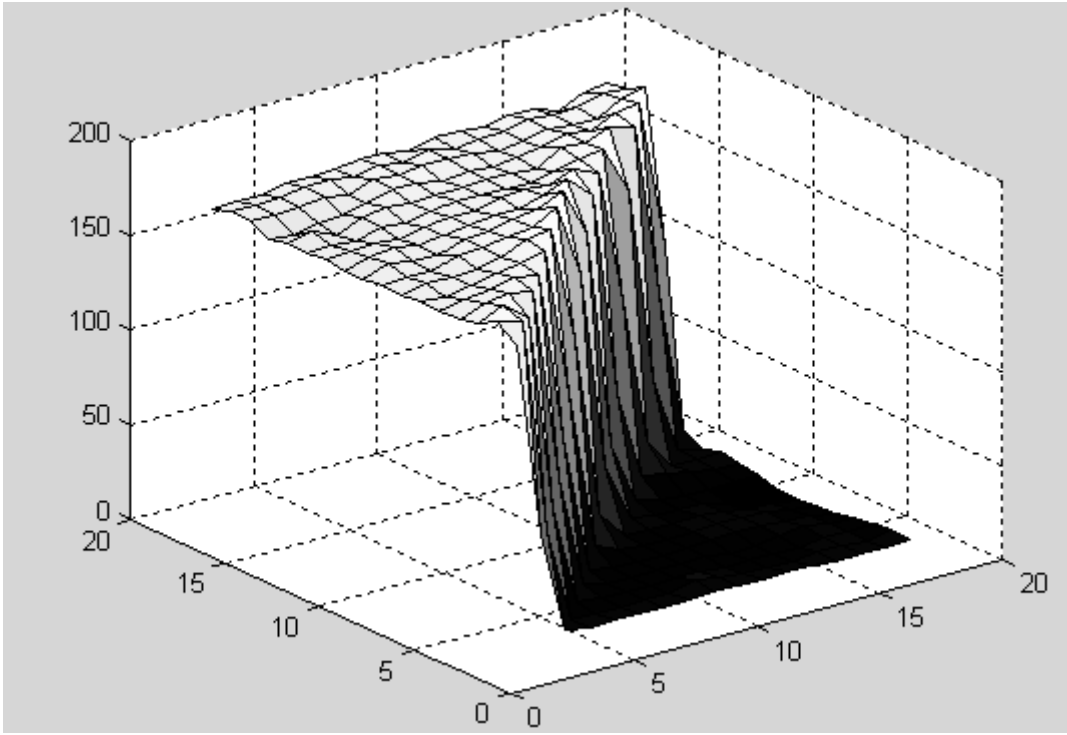
Edge types: Ramp edge

A step edge where the intensity change is not instantaneous but occurs over a finite distance.





Example:

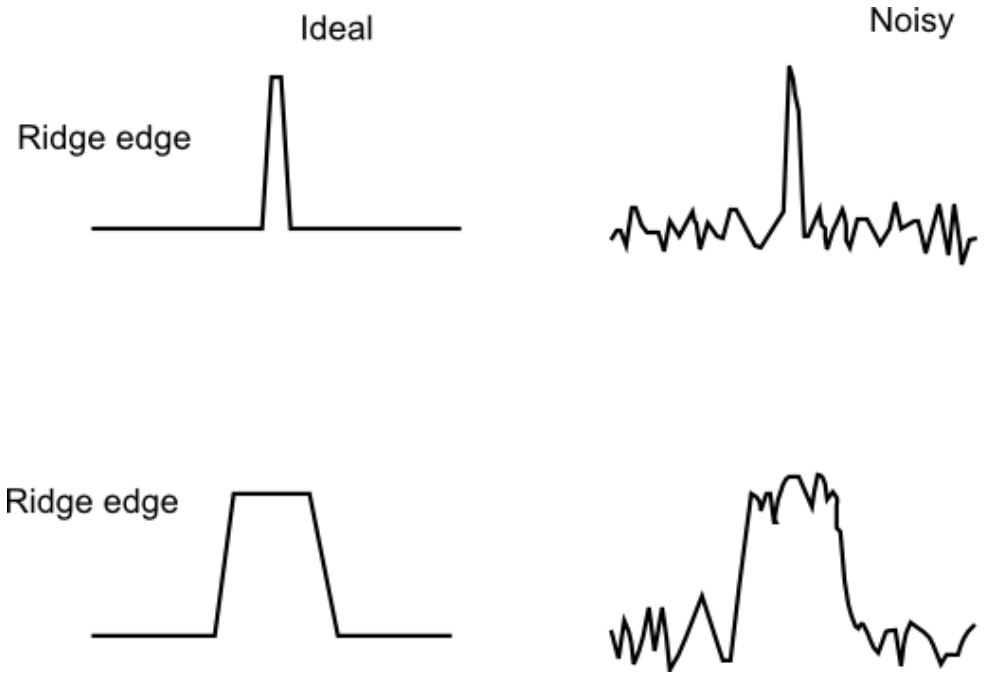


1

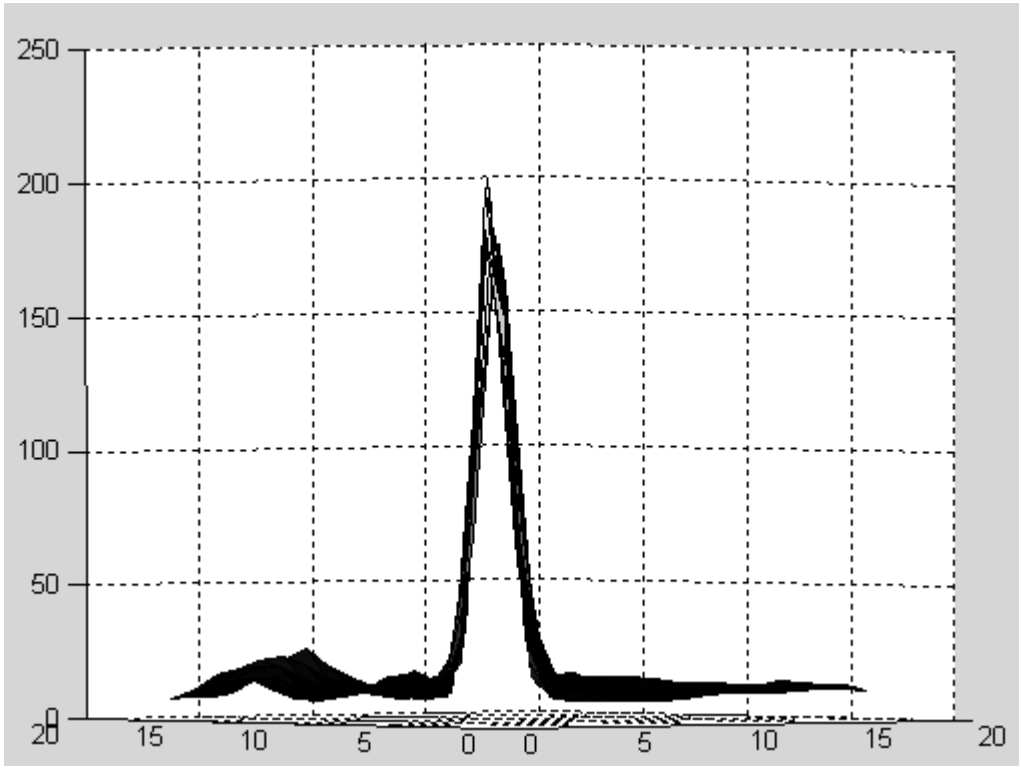
[1] Images taken from [3]

Edge types: Ridge edge

The image intensity abruptly changes value but then returns to the starting value within some short distance
– generated usually by lines



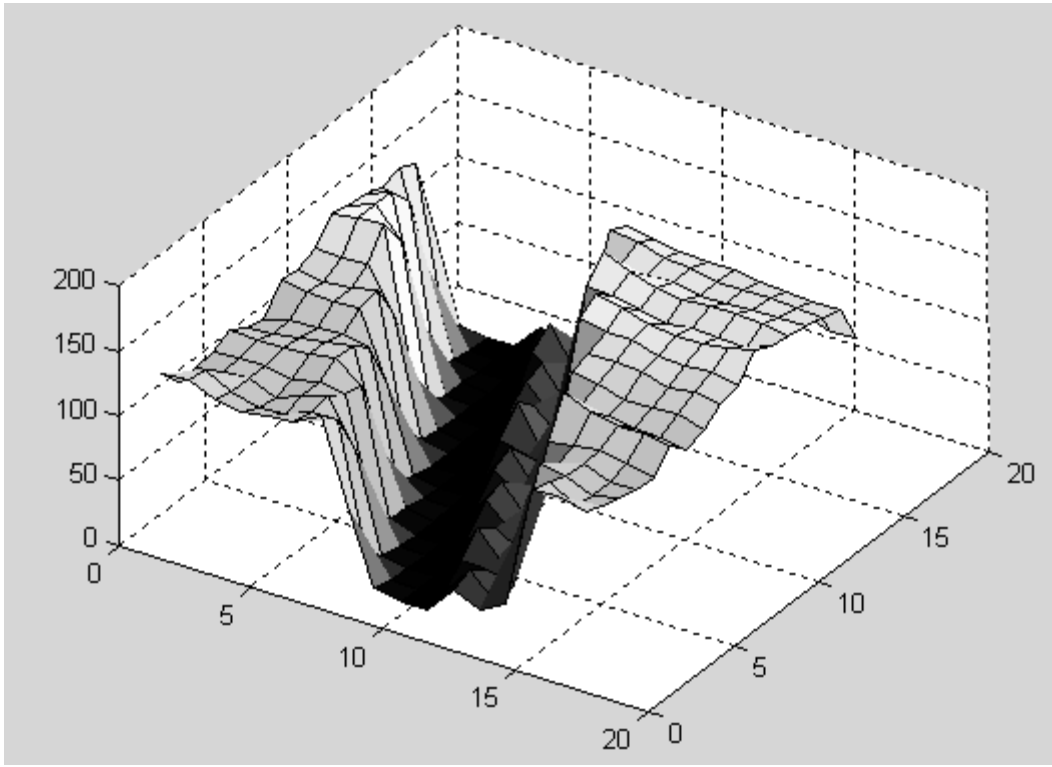
Example:



2

[2] Images taken from[3]

Example:



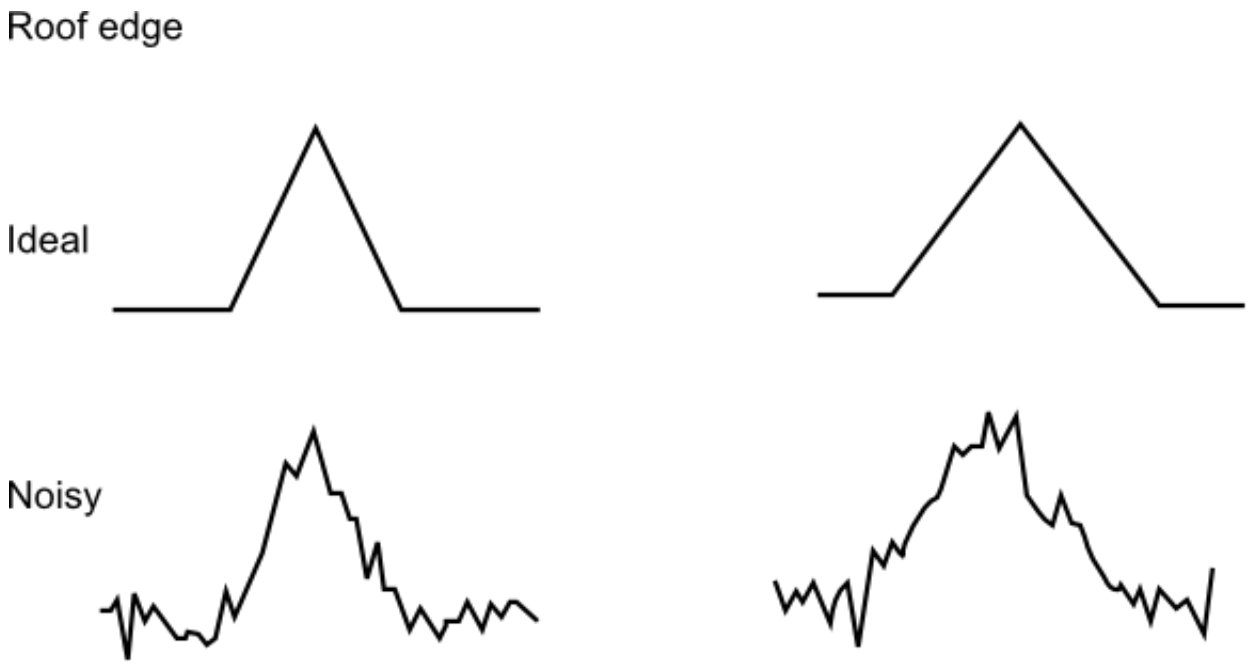
[3]

[3] Images taken from[3]

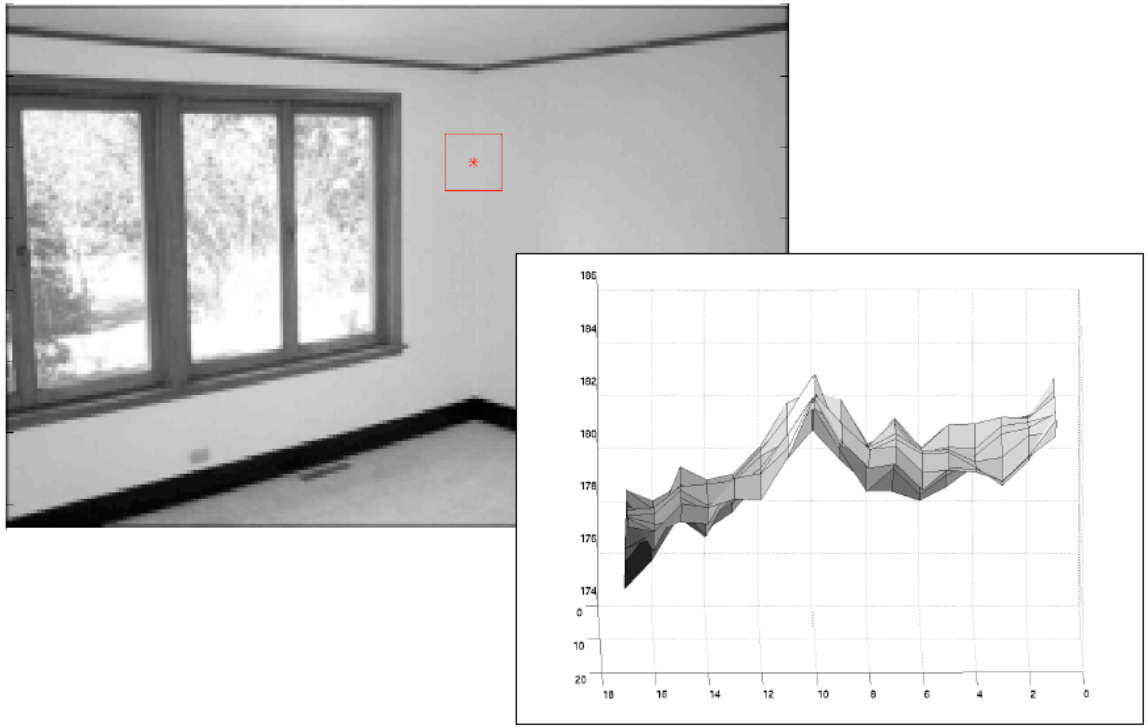
Edge types: Roof edge

A ridge edge where the intensity change is not instantaneous but occurs over a finite distance.

– generated usually by the intersection of surfaces



Example:



4

[4] Images taken from[3]

Steps in edge segmentation

The edge based segmentation process:

- *preprocessing* - noise elimination
- *edge pixels detection and identification*
- *post-processing* - improve the detection results, extract edges, close the contours.

EDGE BASED SEGMENTATION

Classification Criteria:

Principle used for edge detection:

- Methods based on applying differential operators
- Methods based on filtering followed by differential operators
- Methods based on approximation by one-dimensional surfaces having a similar profile with the edge
- Residues method

Way of performing the derivation and filtering operations

- Methods that work on discrete surfaces, using discrete approximation of differential operators by finite differences
- Methods that work on surfaces approximated by a continuous function and use the analytic form of the differential operators

Edge Detection

- The aim of all edge detection techniques is to enhance and mark edge pixels and then extract them.

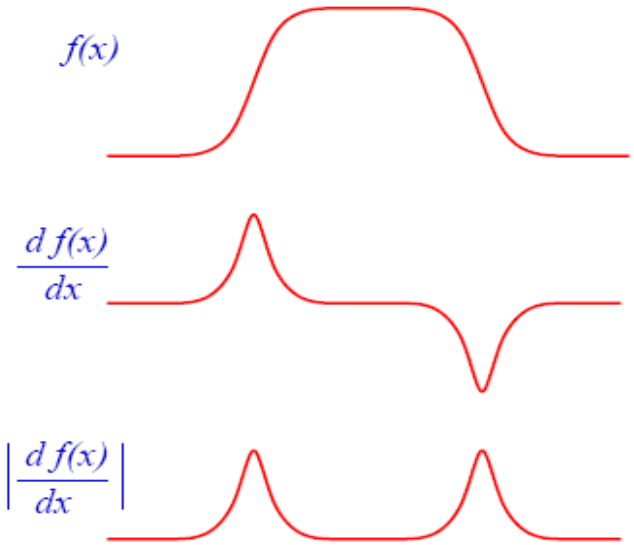
- All need some type of High-pass filter, which can be viewed as either First or Second order differentials.

First Order Differentials

The One-Dimensional case:

- Given a function f(x)
- Compute the derivative df(x)/dx

We can then detect the edge by a simple threshold of: $|\frac{\partial f(x)}{\partial x}| > T \Rightarrow Edge$



Edge detection: First Order Differentials

- In two-dimensions, being given a function f(x, y):

$$\left|\frac{\partial f(x,y)}{\partial x}\right| \Rightarrow Vertical\ Edge \quad \text{and} \quad \left|\frac{\partial f(x,y)}{\partial y}\right| \Rightarrow Horizontal\ Edge$$

- But we want to detect edges in all directions. The variation of intensity in a given direction, θ is given by:

$$\frac{\partial f(x,y)}{\partial x}\cdot\cos\theta + \frac{\partial f(x,y)}{\partial y}\cdot\sin\theta$$

- In two-dimensions, the 1st order differential, $\nabla f(x,y)$ called gradient is a vector given by:

$$\frac{\partial f(x,y)}{\partial x}\bar{i} + \frac{\partial f(x,y)}{\partial y}\bar{j}$$

- The modulus of the gradient is: $|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$
- Threshold to obtain edges:

$$|\nabla f(x,y)| > T \Rightarrow Edge\ ,\ |\nabla f(x,y)| < T \Rightarrow no\ Edge$$

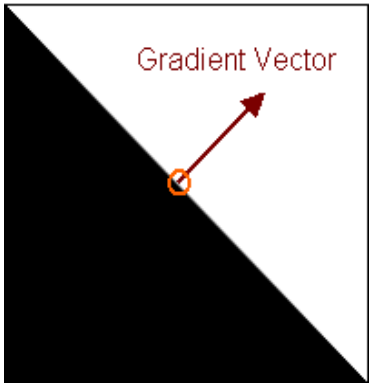
Edge detection: First Order Differentials

Digital implementation - form the first order differentials by convolution of the image with two kernels:

$$\frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i,j) \quad ,$$

$$\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \circ f(i,j)$$

$$|\nabla f(i,j)| = \sqrt{\left(\frac{\partial f(i,j)}{\partial i}\right)^2 + \left(\frac{\partial f(i,j)}{\partial j}\right)^2} \qquad (1)$$



The gradient in an edge pixel

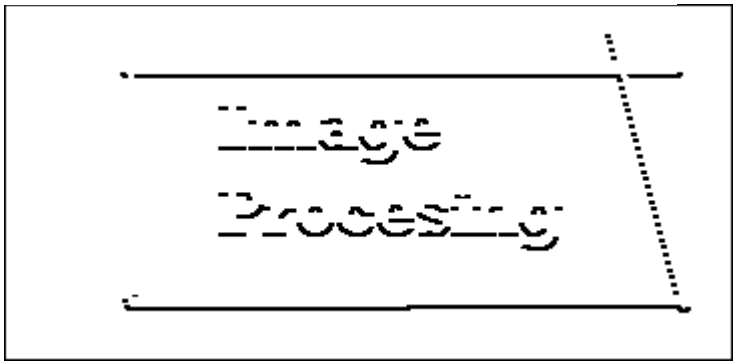
- faster approximation of the modulus of the gradient is given by:

$$|\nabla f(i,j)| = \left|\frac{\partial f(i,j)}{\partial i}\right| + \left|\frac{\partial f(i,j)}{\partial j}\right| \qquad (2)$$

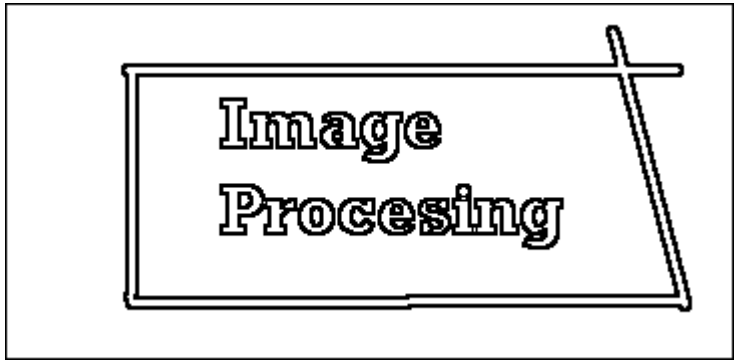
First Order Differentials Examples

Original image

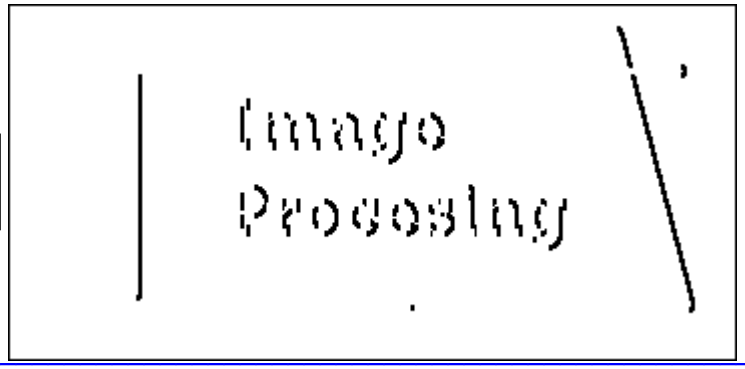
Horizontal edges



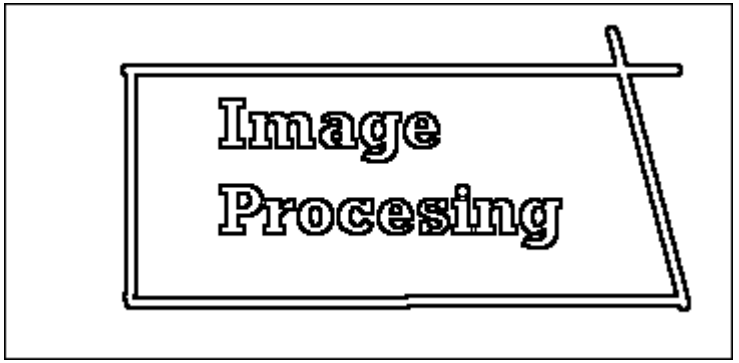
Gradient Magnitude (1)



Vertical edges

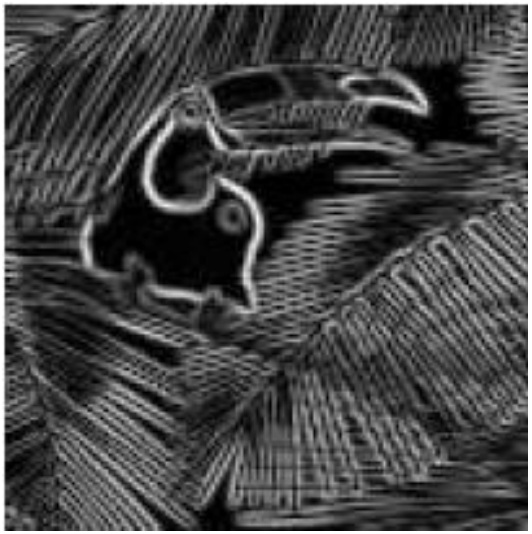


Gradient Magnitude (2)



Edge Detection: Sobel filter

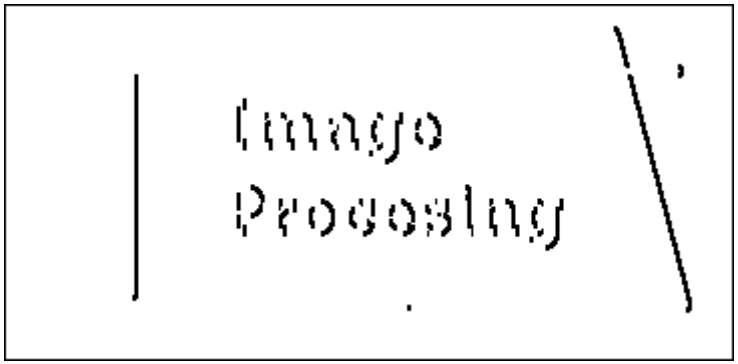
- is a slight variation on the first order differential filter :
- $\frac{\partial f(i,j)}{\partial i} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \circ f(i,j)$, and $\frac{\partial f(i,j)}{\partial j} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \circ f(i,j)$
- This is the most common simple first order differential edge detector.



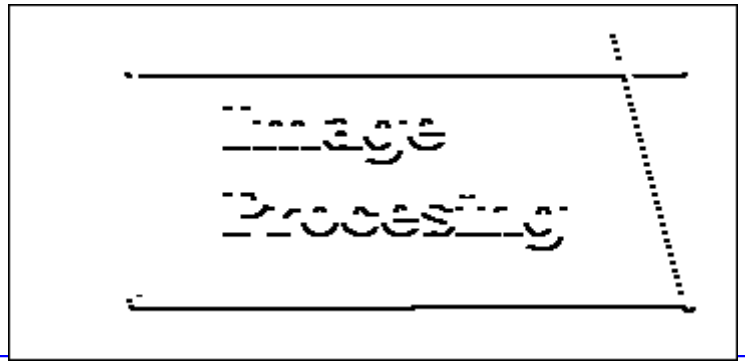
Sobel filter: Examples

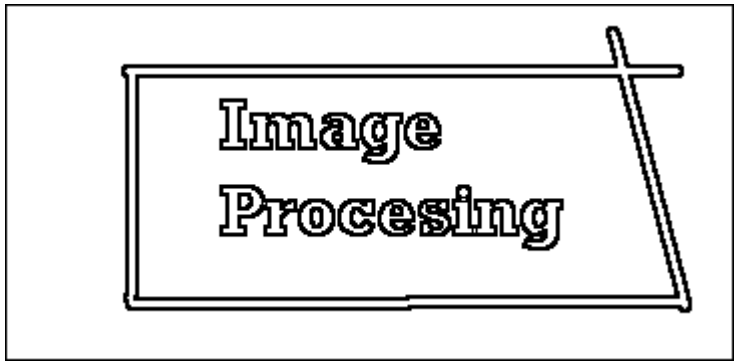
Original image

Vertical edges



Horizontal edges





Gradient magnitude

Sobel filter: Examples

Original image

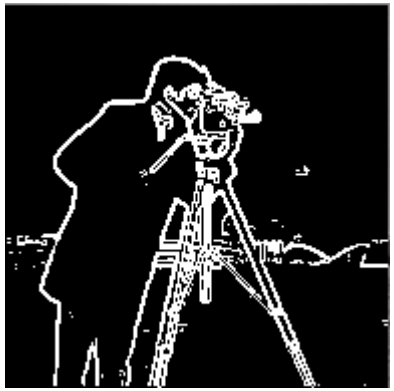
Horizontal edges

Vertical edges

Gradient magnitude

T = 20

T = 120



T = 210

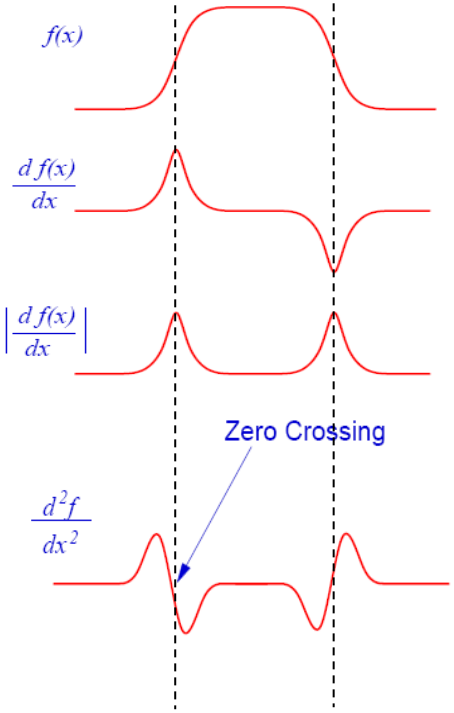
T = 250

First Order Diferentials: Problems

- **Arbitrary threshold value**: for avoiding this, the threshold is set such that x% of image is classified as edge
- **Thick edges**: if threshold is too low edges frequently are thick. Range of **edge thinning techniques** that try to thin edges to a single pixel by removing edge pixels while keeping the edges connected.
- **Broken Edges**: Range or **edge-joining techniques** to try bridge gaps also Hough Transform, to fit lines.
- **Noise Points**: Modified threshold filter to **remove isolated points** or non-connected double points.

Edge Detection: Second Order Differentials

- In one dimension the edge is located by the zero crossing:



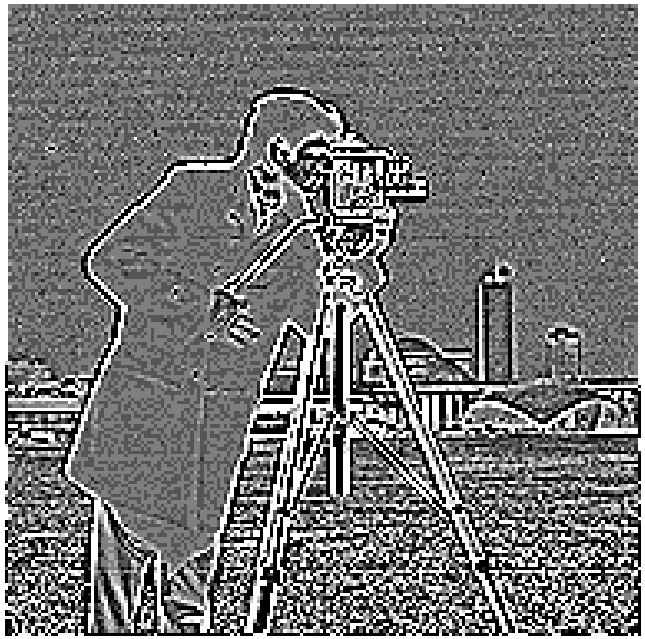
- In two dimensions one has to compute the Laplacian:

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

It can be implemented by a single convolution of :

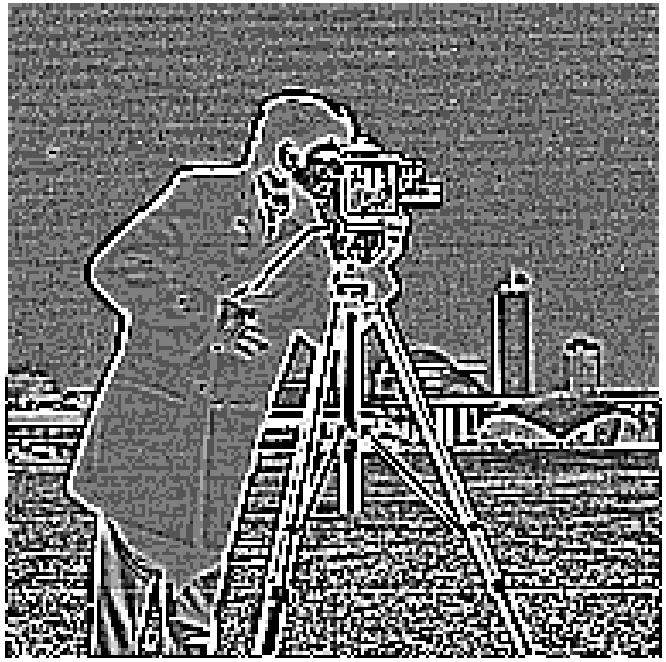
$$\nabla^2 f(x,y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ f(i,j)$$

Laplacian example



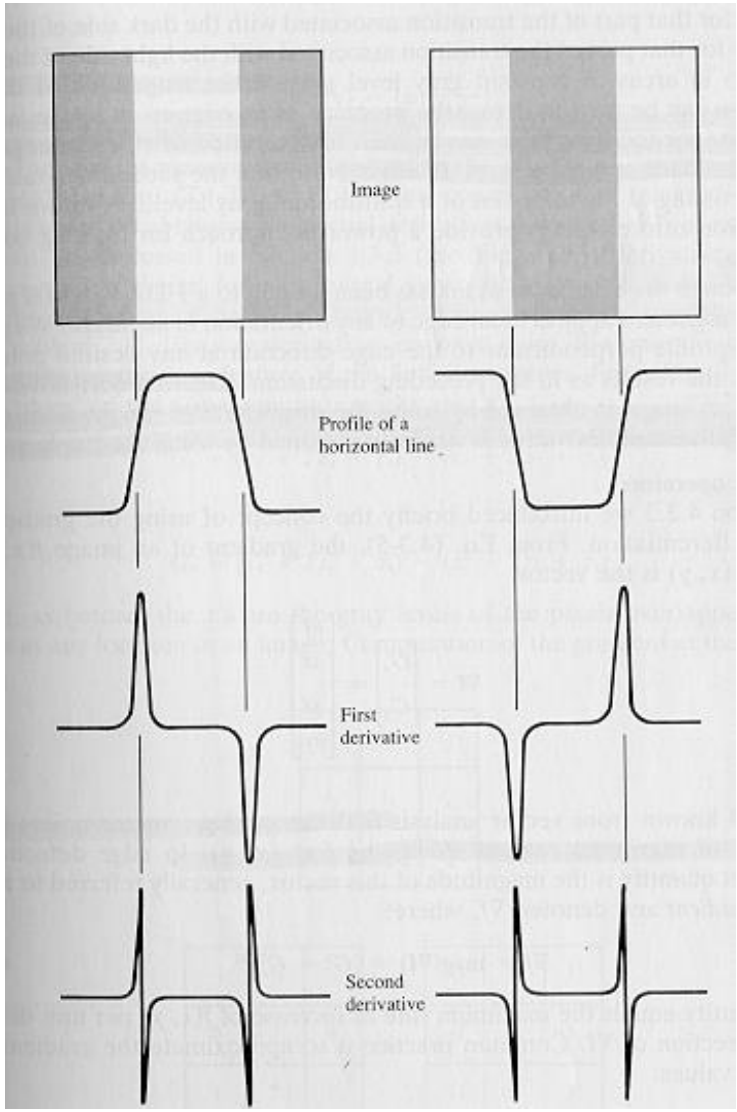
$$\nabla^2 f(x,y) = \frac{1}{8} * \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \circ f(i,j)$$

Laplacian example



$$\nabla^2 f(x,y) = \frac{1}{16} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \circ f(i,j)$$

First and second order derivatives



Second Order Differentials: Problems

- **Thin Edges:** the edges occur between pixels. Always get thin edges, but difficult to display on a digital image.
- **Closed Loops:** edges always form closed loops, reduces break-up of edges, but can cause problems as corners.
- **Noise Problems:** Laplacian is a high pass filter, so enhances high frequencies, and thus noise.

Pre Processing

- Difficult to enhance the edge image, so to reduce the effect of noise we typically want to smooth the image *before* we apply the Laplacian.

- For example use Nine point average, then Laplacian.

- Noting that the convolution is a linear operation, the two [3x3] convolutions can be implemented as a singe [5x5] convolution.

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \odot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & -2 & -1 & -2 & 1 \\ 1 & -1 & 0 & -1 & 1 \\ 1 & -2 & -1 & -2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Edge Enhancement



$$f(x,y) - \nabla^2 f(x,y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \circ f(i,j)$$

Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy

Methods based on filtering followed by differential operators

1. The 1D situation:

- The edge is represented by a step signal, and the edge point is represented by its coordinates
- A Gaussian noise is overlapped over the step signal
- The edge point is given by a maximum variation in intensity, and it can be identified by the maximum of the 1st order derivative or the zero crossings of the 2nd order derivative
- Noise generates several local maxima of close amplitude, and several zero crossings

Analysis of the 2D situation

- The edge point is characterized by coordinate and direction
- Use as filtering operator a 2D Gaussian $G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} = g(x)g(y)$
- Edge points can be emphasized by:
  1. Maximum of gradient
  2. Zero crossings of the Laplacian
  3. Zero crossings of the 2nd order directional derivative in the gradient direction

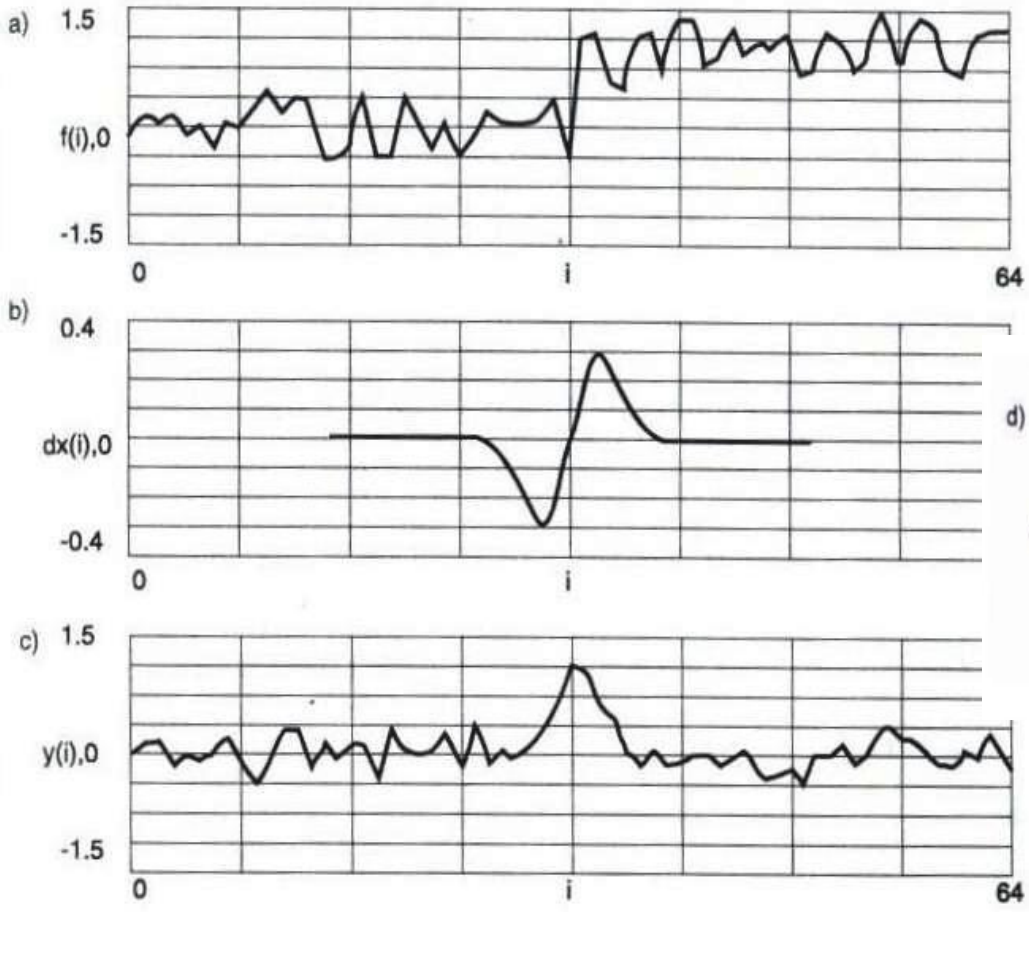
Canny edge detector

Criteria:

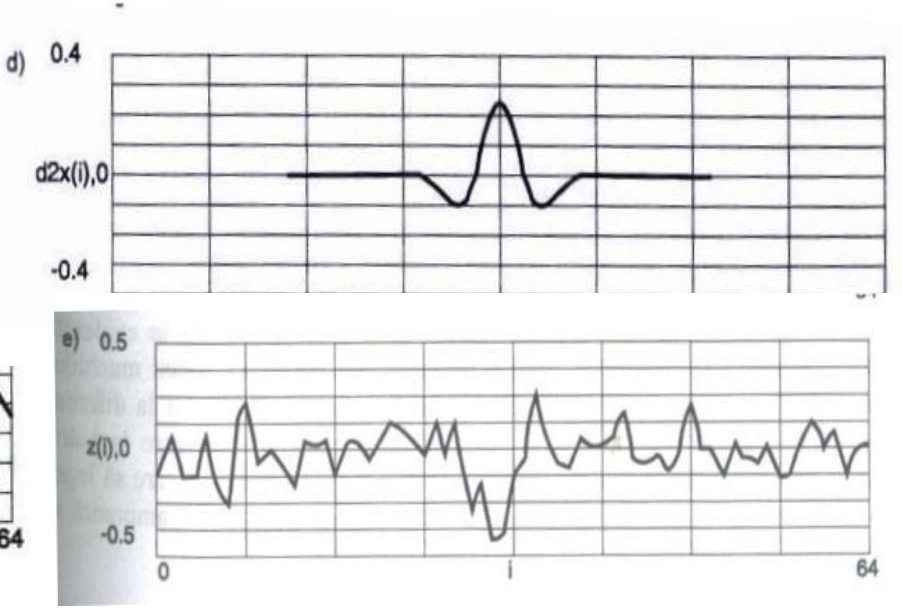
1. **Optimal detection:** the probability of marking a real edge point should be high, and the probability of marking a false edge point should be low (maximize the signal-noise ratio)
2. **Correct localization:** the points marked as belonging to an edge should be as close as possible to the real edge points
3. **Single response to one edge**

Canny gave the mathematical formulation of the three criteria, showing that the first order derivative of the Gaussian models its filter. A similar result can be found by using the 2nd order derivative of the Gaussian.

Canny – examples



a)step signal summed with white noise of mean=0 and deviation 0.5
b) first order derivative of the Gaussian
c) filter response
d) second order derivative of the Gaussian
e) the filter's response



Maximum of the gradient method

1. For each filtered image point the modulus and direction of the gradient are evaluated
2. The edge points are detected as local extremes of the gradient modulus in the gradient's direction

The filtered image:

$$G * I = I_f(x,y) = \int\int I(x-\zeta, y-\eta)\cdot G(\zeta,\eta)\cdot d\zeta \cdot d\eta = \sum_{\zeta}\sum_{\eta} I(x-\zeta, y-\eta)\cdot g(\zeta)\cdot g(\eta)$$

$$\frac{\partial I_f(x,y)}{\partial x} = \int\int I(\zeta,\eta)\cdot g_x'(x-\zeta)\cdot g(y-\eta)d\zeta\cdot d\eta = g_x'(x) * I(x,y) * g(y)$$

$$\frac{\partial I_f(x,y)}{\partial y} = \int\int I(\zeta,\eta)\cdot g(x-\zeta)\cdot g_y'(y-\eta)d\zeta\cdot d\eta = g_y'(y) * I(x,y) * g(x)$$

the gradient: $$\nabla I_f(x,y) = \frac{\partial I_f(x,y)}{\partial x}\vec{i} + \frac{\partial I_f(x,y)}{\partial y}\vec{j}$$

Maximum of the gradient method

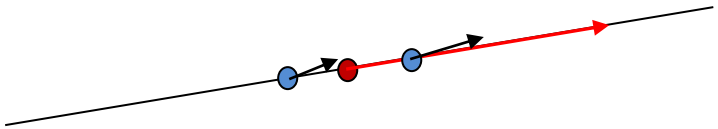
the gradient magnitude:

$$|\nabla I_f(x,y)| = \sqrt{\left(\frac{\partial I_f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial I_f(x,y)}{\partial y}\right)^2} \quad \text{and}$$

the gradient orientation:

$$\alpha = \arctan \frac{\frac{\partial I_f(x,y)}{\partial y}}{\frac{\partial I_f(x,y)}{\partial x}} \ ,$$

- A point is a local maximum of the gradient modulus if:
  The modulus of its gradient is greater than the modulus of the gradient of two opposite points located in the direction of the gradient, at equal distance



Zero crossing of the Laplacian

1. The image is filtered with a LoG kernel representing the composition of a Gaussian low pass filter and a Laplacian 2nd order derivative filter

2. The edge points are detected as the zero crossings of the filtered image

Zero crossing of the Laplacian

The *Gaussian distribution* function in two variables: $G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma}}$

- The shape of the distribution and hence the amount of smoothing can be controlled by varying σ.
- In order to smooth an image $f(x,y)$, we convolve it with $G(x,y)$ to produce a smoothed image $s(x,y) = f(x,y)*G(x,y)$.
- Having smoothed the image with a Gaussian operator we can now take the Laplacian of the smoothed image:
- Therefore the total operation of edge detection after smoothing on the original image is: $\nabla^2(f(x,y) * g(x,y))$.
- This method of edge detection was first proposed by Marr and Hildreth at MIT who introduced the principle of the *zero-crossing* method.

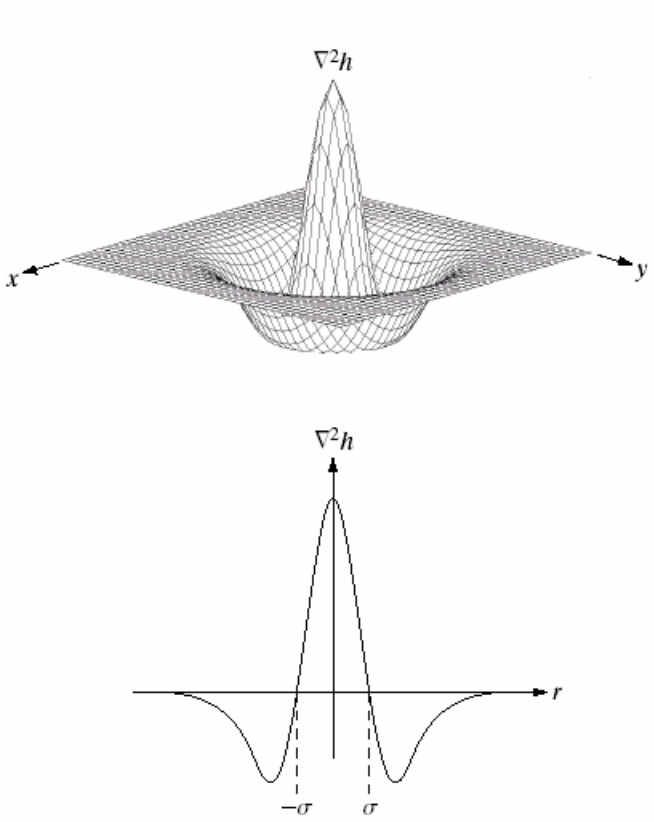
Zero crossing of the Laplacian

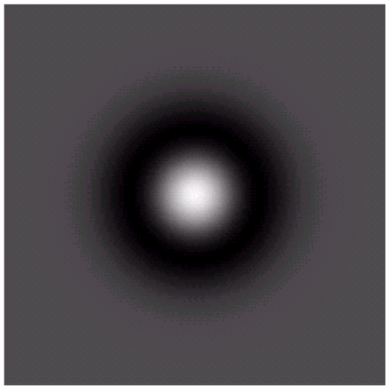
- The LoG (Laplacian of Gaussian) kernel:

$$\nabla^2 \cdot G = \frac{x^2 + y^2 - 2\cdot\sigma^2}{\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Where σ is the standard deviation of the Gaussian and it determines the scale of the filter
- The width (in pixels) of the central region of the filter is, $w = \sigma \cdot 2 \cdot \sqrt{2}$
- The useful width of the filter is $s = 3w$
- A low value for s will emphasize many edges, a high value emphasizes only high variations in intensity

Zero crossing of the Laplacian

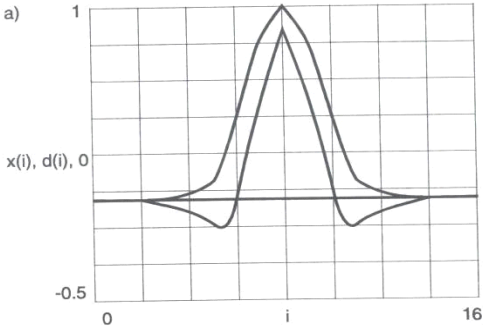


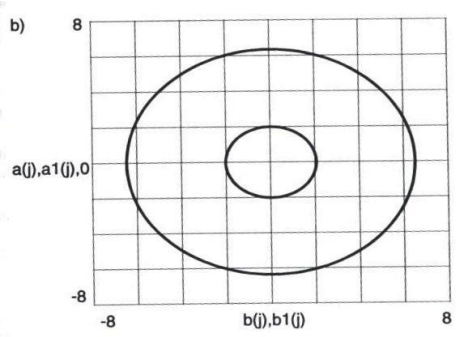


0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

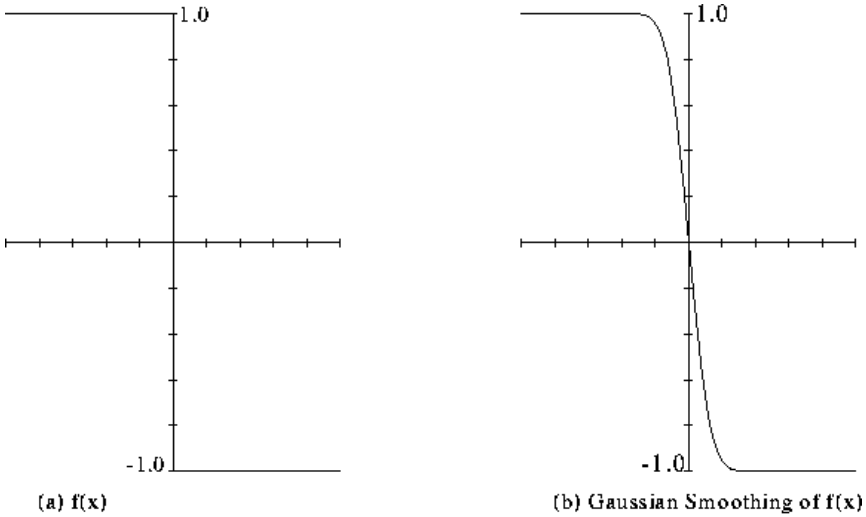
a b
c d

FIGURE 10.14 Laplacian of a Gaussian (LoG). (a) 3-D plot. (b) Image (black is negative, gray is the zero plane, and white is positive). (c) Cross section showing zero crossings. (d) 5 × 5 mask approximation to the shape of (a).

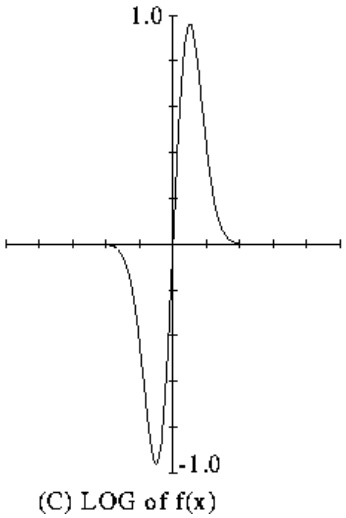




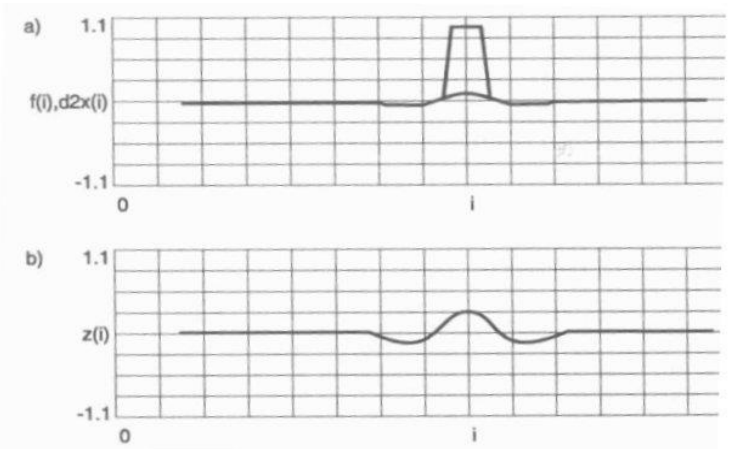
Zero crossing of the Laplacian



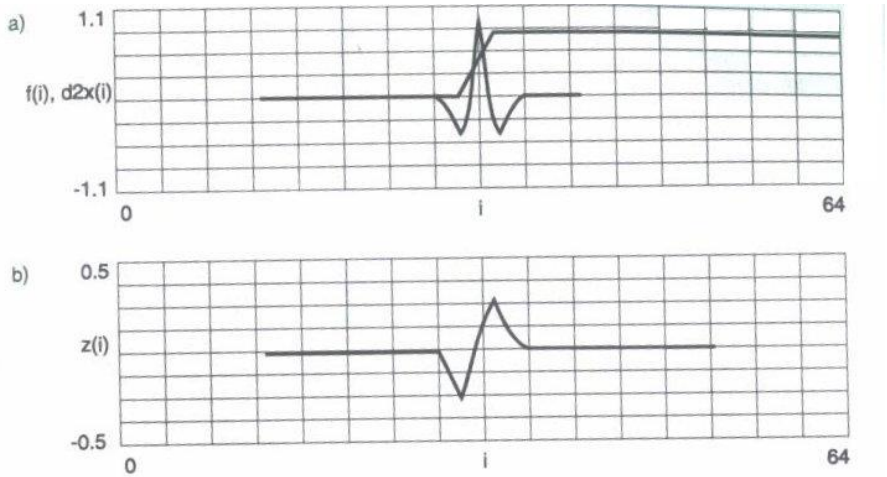
The basic principle of this method is to find the position in an image where the second derivatives become zero. These positions correspond to edge positions as shown in the figure:



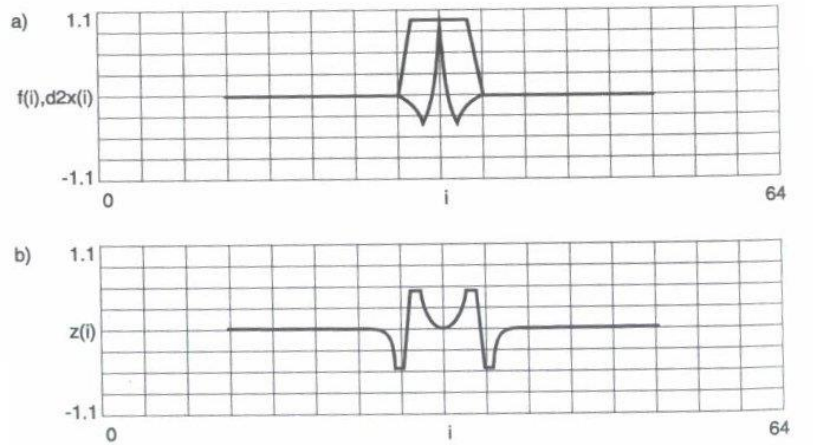
LOG for different edge types



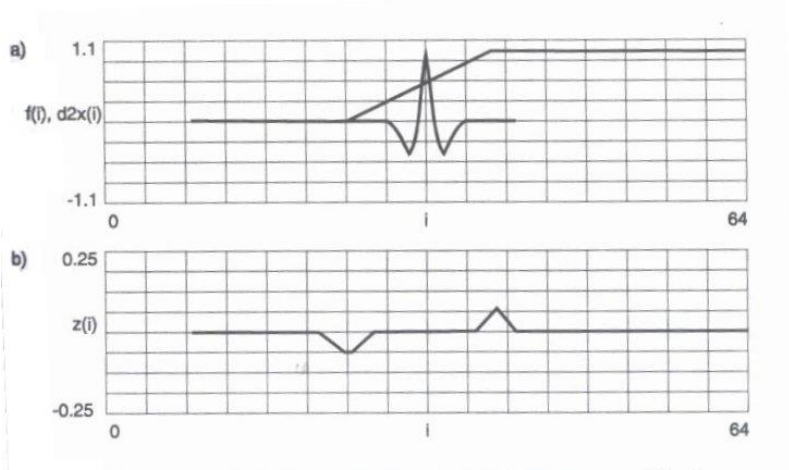
(a) muchie impuls de lățime mai mică decât w (σ=3);
(b) răspunsul filtrului LoG, se observă deplasarea trecerilor prin zero cu (w-d)/2.



(a) semnal rampă cu lățimea mai mică decât lățimea operatorului s(σ=1);
(b) răspunsul filtrului LoG, se observă o trecere prin zero corespunzătoare mijlocului rampei.



(a) muchie impuls, cu lățimea mai mare decât w și mai mică decât s (σ=1);
(b) răspunsul filtrului LoG, se observă influența reciprocă dintre muchii care se menține atâta timp cât lățimea impulsului este mai mică decât s.

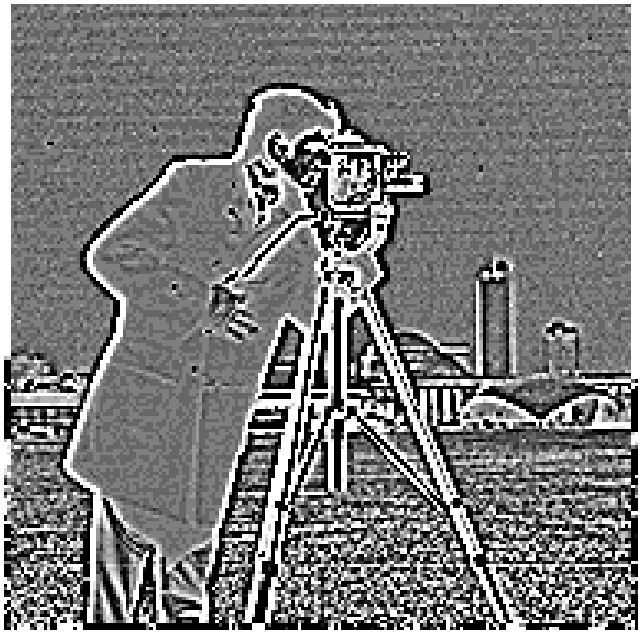


(a) muchie rampă cu lățimea mai mare decât dimensiunea operatorului (σ=1);
(b) răspunsul filtrului LoG nu pune în evidență treceri prin zero, este necesar ca lățimea rampei să fie mai mică decât lățimea operatorului.

EDGE BASED SEGMENTATION

Original Image

LoG



LoG - Shrinked

Zero crossing of the second order directional derivative, in the direction of the gradient

1. Filter the image with a Gaussian kernel, G

2. The edge points are the zero-crossings of the 2nd order directional derivative in the direction of the gradient

The 1st order directional derivative in the direction of the gradient:

$G_n = \frac{\partial G}{\partial n} = n \cdot \nabla G$ where

- $n = \frac{\nabla (G * I)}{|\nabla (G * I)|}$, and
- $\nabla (G * I) = \frac{\partial I_f(x,y)}{\partial x}\vec{i} + \frac{\partial I_f(x,y)}{\partial y}\vec{j}$

An edge point is a local maximum in the direction *n* of the operator G_n applied to the image.

Zero crossing of the second order directional derivative, in the direction of the gradient

For a local maximum: $\frac{\partial}{\partial n} G_n * I = 0$ or $\frac{\partial^2}{\partial n^2} G * I = 0$

$$\frac{\partial^2}{\partial n^2} G * I = \frac{\frac{\partial^2 I_f}{\partial x^2} \cdot (\frac{\partial I_f}{\partial x})^2 + 2 \cdot \frac{\partial I_f}{\partial x} \cdot \frac{\partial I_f}{\partial y} \cdot \frac{\partial^2 I_f}{\partial x \partial y} + \frac{\partial^2 I_f}{\partial y^2} \cdot (\frac{\partial I_f}{\partial y})^2}{(\frac{\partial I_f}{\partial x})^2 + (\frac{\partial I_f}{\partial y})^2}$$

The zero crossings of the 2nd order derivative represent a change of sign in the resulted image

Mathematics for 2nd order derivative

$$\frac{\partial^2 I_f(x,y)}{\partial x^2} = \int\int I(\zeta,\eta) \bullet g''_{xx}(x-\zeta) \bullet g(y-\eta) d\zeta d\eta =$$

$$\sum_{\zeta}\sum_{\eta} I(\zeta,\eta) \bullet g''_{xx}(x-\zeta) \bullet g(y-\eta) = \sum_{\zeta} g''_{xx}(x-\zeta) \sum_{\eta} I(\zeta,\eta) \bullet g(y-\eta) =$$

$$g''_{xx}(x) * I(x,y) * g(y)$$

$$\frac{\partial^2 I_f(x,y)}{\partial y^2} = \int\int I(\zeta,\eta) \bullet g(x-\zeta) \bullet g''_{yy}(y-\eta) d\zeta d\eta =$$

$$\sum_{\zeta}\sum_{\eta} I(\zeta,\eta) \bullet g(x-\zeta) \bullet g''_{yy}(y-\eta) = \sum_{\eta} g''_{yy}(y-\eta) \sum_{\zeta} I(\zeta,\eta) \bullet g(x-\zeta) =$$

$$g''_{yy}(y) * I(x,y) * g(x)$$

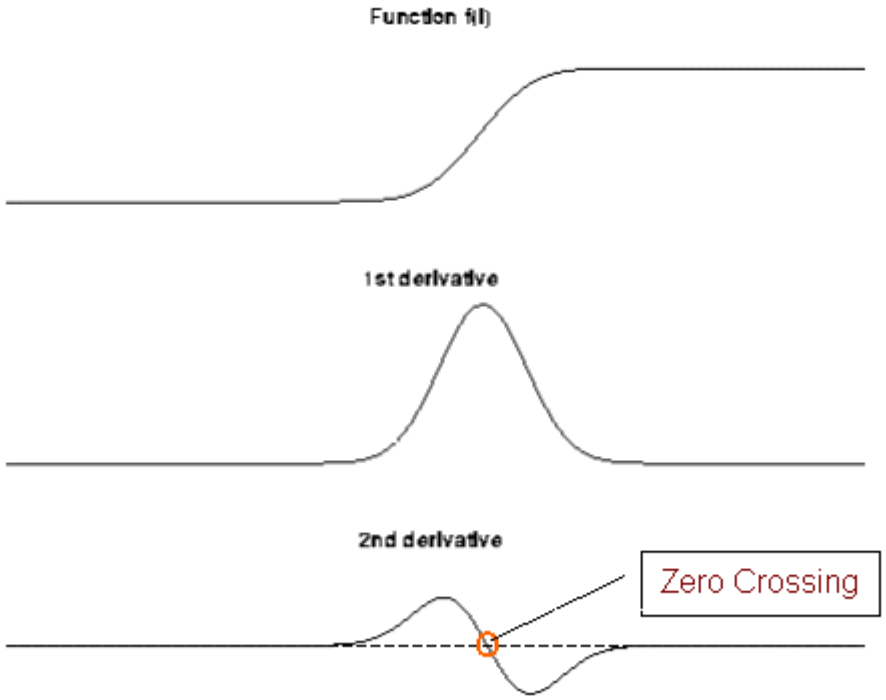
$$\frac{\partial^2 I_f(x,y)}{\partial x \partial y} = \int\int I(\zeta,\eta) \bullet g'_x(x-\zeta) \bullet g'_y(y-\eta) d\zeta d\eta =$$

$$\sum_{\zeta}\sum_{\eta} I(\zeta,\eta) \bullet g'_x(x-\zeta) \bullet g'_y(y-\eta) = \sum_{\zeta} g'_x(x-\zeta) \sum_{\eta} I(\zeta,\eta) \bullet g'_y(y-\eta) =$$

$$g'_x(x) * I(x,y) * g'_y(y)$$

EDGE BASED SEGMENTATION

Zero crossing of the second order directional derivative



Contents

- Edge based segmentation, edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy

Edge extraction

- Edge detection defines some thick contours of the real edges, and because of noise, it introduces false edges
- For high level processing, one needs to mark the edge by a curve of 1 pixel width, very close to the real position of the edge
- Methods:
  - Gradient variation:
    1. extraction of local extremes of the gradient
    2. discrimination of the local extremes
  - Zero crossing of the Laplacian or the 2^{nd} order derivative
    1. extraction of the zero crossings
    2. discrimination of the zero crossings

The gradient method – extraction of the local gradient extremes

- Let M a point in the image, having ∇M , d a given distance
- M_1 and M_2 points on a line passing through M and having the direction of the gradient of M, located in opposite positions with respect to M
- $M_1 = M + d \cdot \frac{\nabla M}{|\nabla M|}$ and $M_2 = M - d \cdot \frac{\nabla M}{|\nabla M|}$
- M is selected if $|\nabla M| \geq |\nabla M_1|$ and $|\nabla M| > |\nabla M_2|$

The gradient method – discrimination gradient extremes

- **Hysteresis thresholding**: select all extremes that have the gradient greater than a low threshold t_l and form a connected path to an extreme that has the value greater than a high threshold t_h
- **Algorithm**:

1. Find the binary images f_h and f_l obtained by thresholding the extremes of the gradient with t_h and t_l and compute the image $f_p = f_l - f_h$

2. Expand all the non-zero points in f_h to form connected chains with all the non-zero points of f_p (i.e. find the adjacency graph of the non-zero points in f_p and select the connected components with non-zero points in f_h

3. Select all the connected components that have a length greater than l_{min}

Zero crossings methods

1. Extraction of the zero crossings:

Let A a point of coordinates (x, y), having as neighbors B(x+1, y), C(x, y+1),D(x, y-1),E(x-1, y)

a. If $A \cdot B<0$ and $A \cdot C<0$, then there is a zero-crossing of coordinates [x + interpolation(A,B),y + interpolation(A,C)]

b. If $A \cdot B<0$ and $A \cdot C>=0$ then there is a zero-crossing of coordinates [x + interpolation(A,B),y]

c. If $A \cdot B>=0$ and $A \cdot C<0$ then there is a zero-crossing of coordinates [x, y + interpolation(A,C)]

Interpolation(A,B) returns a value in [0,1], based on the linear interpolation of the zero point between A and B

2. **Discrimination of the zero crossings** eliminates the zero-crossings generated by noise:

- In the case of the zero crossings of the LoG use the slope condition or the gradient's amplitude for discrimination.
- In the case of the 2nd order directional derivative a first condition is the similitude of the gradient's direction and the transition direction.

Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy

Edge closing

For eliminating false edges use high thresholds and apply an algorithm for tracking and closing the contours.

Algorithm: identifies the end points and for each

1. it determines the direction of the contour (the tangent to the contour in the end point)
2. select the point having the maximum gradient compared to the gradient values of 3 neighboring pixels, in the direction of the contour with a deviation of +/- 45^0
3. check the stop conditions
4. if the stop conditions are not satisfied, go to step 1 otherwise go to another end point

The stop conditions are:

- no more points for which to continue the search
- contour is closed
- obtain a maximum length

Contents

- Edge detection
- Methods based on filtering followed by differential operators
- Edge extraction
- Edge closing
- Algorithm for edge detection, tracking and extraction based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy

Algorithm for edge detection, tracking and extraction, based on the zero crossings of the 2nd order derivative, with sub-pixel accuracy

Steps:

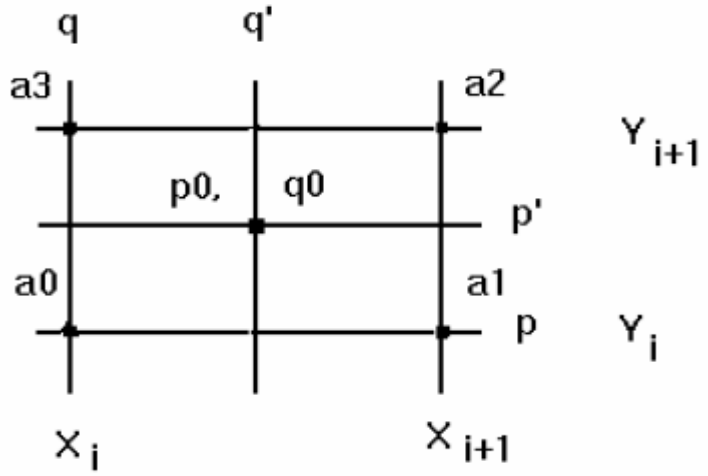
1. Filter the image
2. Find the partial derivatives of 1st and 2nd order
3. Compute the 2nd directional derivative in the gradient's direction
4. Detect the zero-crossings at sub-pixel level
5. Track, extract and close the contours

The last two steps are done simultaneously. They are generated by the detection of a start point obtained by the systematic scanning of the 2nd order directional derivative.

Algorithm (2)

- The starting points obey the conditions:
  - A change of the sign in the 2nd order directional derivative between the current point and the next point
  - The point should not have been marked previously
  - The gradient should be greater than a threshold
  - The direction of the gradient should coincide with the sense in which the 2nd order directional derivative changes
- From the start point one calls the following procedures:
  - Detect zero crossings
  - Track the contour
  - Extract the edges

Algorithm – detect zero crossings



$$a0 = \frac{\partial^2 I_f(x_i, y_i)}{\partial n^2} \qquad a1 = \frac{\partial^2 I_f(x_i + 1, y_i)}{\partial n^2}$$

$$a2 = \frac{\partial^2 I_f(x_i + 1, y_i + 1)}{\partial n^2} \qquad a3 = \frac{\partial^2 I_f(x_i, y_i + 1)}{\partial n^2}$$

The bilinear approximation of f(x, y) in the 2x2 cell of pixels, in the directions:

1 dir. Y_i: $f(x,y) = f(x_i, y_i) + [f(x_{i+1}, y_i) - f(x_i, y_i)]x = z_1(x)$

2 dir. y_{i+1}: $f(x,y) = f(x_i, y_{i+1}) + [f(x_{i+1}, y_{i+1}) - f(x_i, y_{i+1})]x = z_2(x)$

3 dir. X: $f(x,y) = [z_2(x) - z_1(x)]y + z_1(x)$

$f(x,y) = f(x_i + p, y_i + q) \cong g(p,q) = (1-p)\cdot(1-q)\cdot a0 + p\cdot(1-q)\cdot a1 + q\cdot(1-p)\cdot a3 + p\cdot q\cdot a2$, where $p, q \in [0,1]$

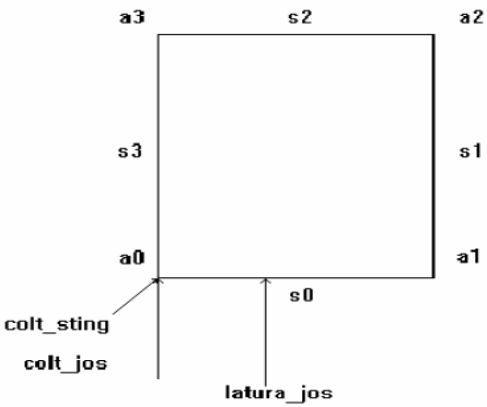
The zero crossings depend on the sign of s0=a0·a1, s1=a1·a2, s2=a2·a3, s3=a3·a0

1. If all the products are >0, the function has no zero crossings in the interpolation interval
2. If two products are <0 then there is a single contour line that intersects the corresponding edges
3. If all products are negative then the cell is intersected by two distinct contours.

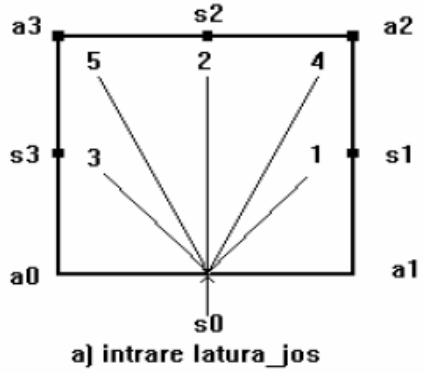
Let D= a1·a3- a0·a2

- For D>0, a contour intersects the cell through the edges 0 and 3, and another through 1, 2
- For D<0, a contour intersects the cell through the edges 0 and 1, and another through 2, 3
- If D=0, the two contours intersect and pass through opposite edges of the cell.

Algorithm– contour tracking and extraction

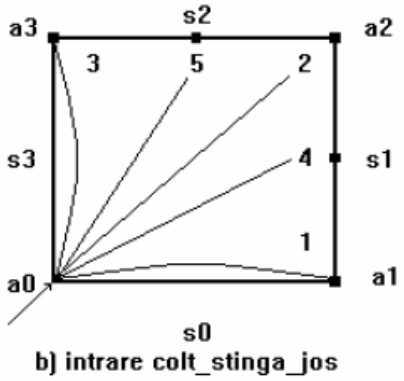


Cons. the entry point on S0 or in node a0.
Three entrance directions :
1.through edge
2.through a corner orthogonal on the edge
3.oblic in the corner



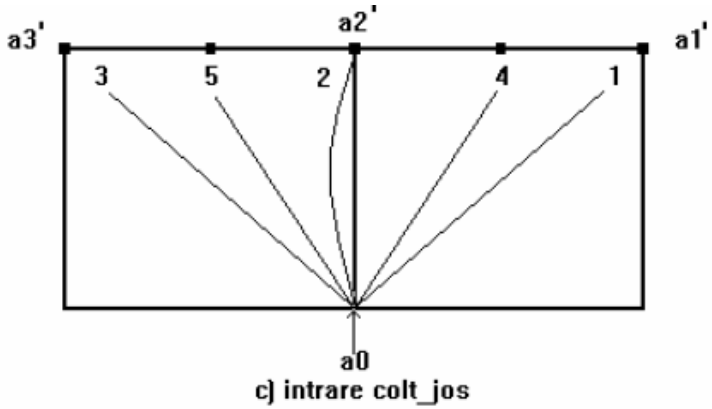
a) intrare latura_jos

Exit points: s0<0
1.if s3≥0^s1<0 v s3<0^s1<0^D<0
2. if s3>0^s1>0 v s3<0 ^s1<0^D=0
3. if s3<0 ^ s2≥0vs3<0 ^s1<0^D>0
4. if s3>0^s1=0
5. if s3=0 ^s1>0



b) intrare colt_stinga_jos

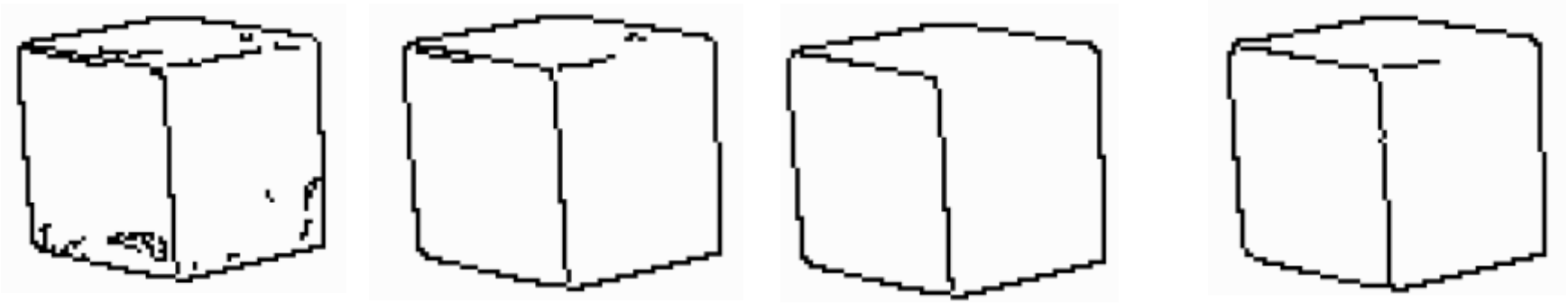
Exit points:a0=s0=s3=0
1.if s1=0^a1=0^a2≠0^a3≠0
2. if s1=0^a1≠0^a3≠0
3.if s1>0 ^a3=0
4. if s1<0 ^ s2>0
5.if s1>0 ^ s2<0



c) intrare colt_jos

Algorithm – last step

- By the linear interpolation the exact coordinate of the edge point is found.
- The coordinates are maintained in a list of contour points. In the result matrix, the contour points are marked as checked points
- If the contour point is already marked as checked point, or if the gradient in that point is less than a low threshold, the contour extraction ends. (Only contours having a length greater than a minimum length, l_{min} are considered).
- The gradient threshold can be reduces greatly (as this algorithm is based on a rigorous tracking of the contour)



References

[1] Sergiu Nedevschi: “Prelucrarea Imaginilor si Recunoasterea Formelor”, *Ed. Microinformatica*, 1997

[2] Emanuele Trucco, Alessandro Verri: Introductory Techniques for 3-D Computer Vision

[3] Robert Collins, CSE486, Penn State, “Lecture 5: Gradients and Edge Detection”