Basics and Hands-on of Computer Vision

Intensive course at University of Helsinki – day 2

Jorma Laaksonen

Aalto University
Department of Computer Science
Espoo

15.5.2018

Contents

Contents

10. Morphology

- logical processing of binary images, generalizations to intensity images
- image is processed as a point set

Application areas of morphology:

- pre-processing: noise removal
- enhancement of object shapes
- qualitative description of objects

Morphological operations:

- erosion & dilation
- opening & closing
- *hit-or-miss*
- thinning & thickening

10.1 Definitions and operations (9.1/9.1.1)

- universal set in principle is the Euclidean 2-dimensional space
- universal set in practice is the discrete point set Z^2
- origin / reference point
- belongs in $\in$, subset $\subseteq$, superset $\supseteq$, intersection $\cap$, union $\cup$
- empty set $\emptyset$, complement $()^c$, subtraction $A - B = A \cap B^c$
- symmetric set or transpose or reflection or mirroring

$$\widehat{B} = \{w \mid w = -b \text{ , when } b \in B\}$$

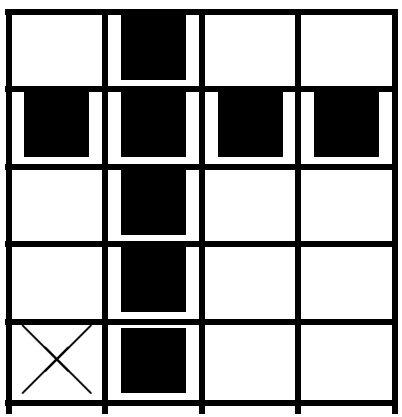
- translation or shift

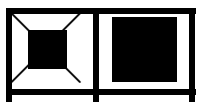
$$(B)_z = \{c \mid c = b + z \text{ , when } b \in B\}$$

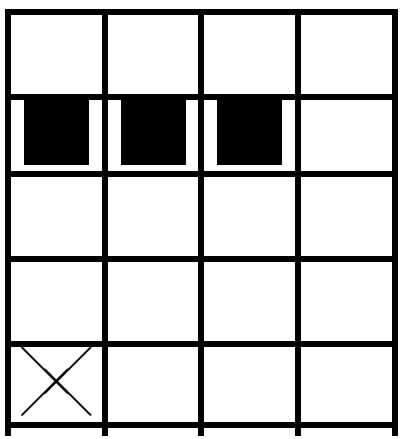
10.2 Erosion $\ominus$ (9.2.1/9.2.2)

- makes image smaller, removes details
- A is the set to be eroded, B is erosion structuring element

$$\begin{aligned} A \ominus B &= \{z \mid (B)_z \subseteq A\} \\ &= \{z \mid (B)_z \cap A^c = \emptyset\} \\ &= \{z \mid z + b \in A \text{ , for all } b \in B\} \\ &= \bigcap_{b \in B} (A)_{-b} \end{aligned}$$





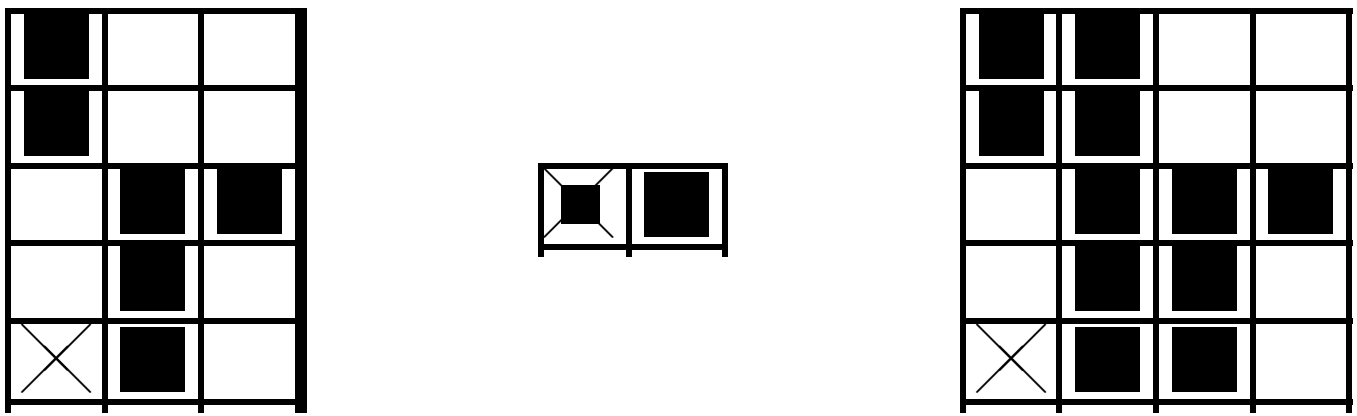


10.3 Dilation $\oplus$ (9.2.2/9.2.1)

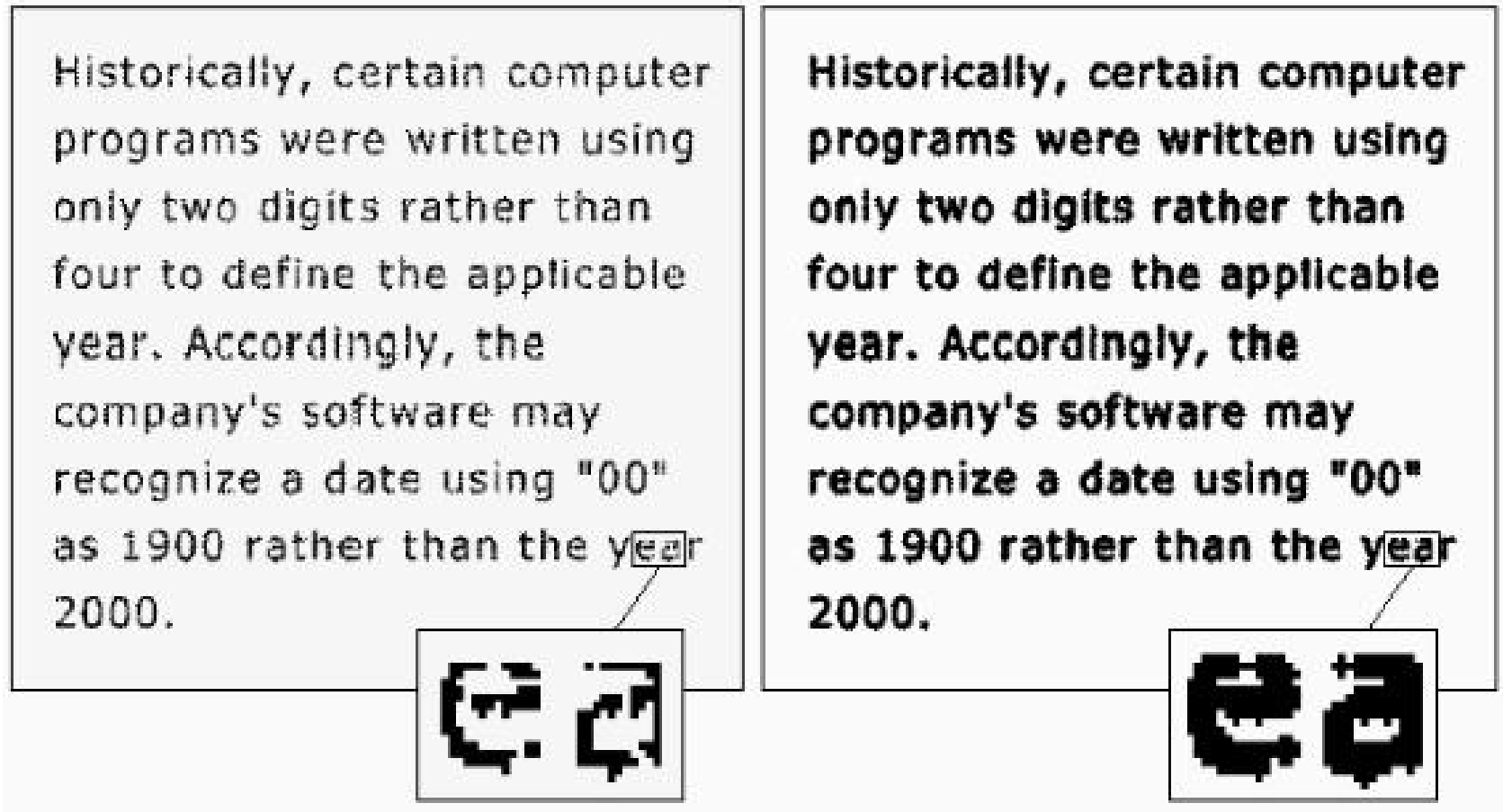
- makes image larger, fills in gaps

$$\begin{aligned} A \oplus B &= \{z \mid (\widehat{B})_z \cap A \neq \emptyset\} \\ &= \{z \mid z = a + b \text{ , for some } a \in A \text{ and } b \in B\} \\ &= \bigcup_{b \in B} (A)_b \end{aligned}$$

- comparable to convolution, in both the structuring element is reflected



An example of dilation (9.2.2/9.2.1)



10.4 Opening $\circ$ and closing $\bullet$ (9.3)

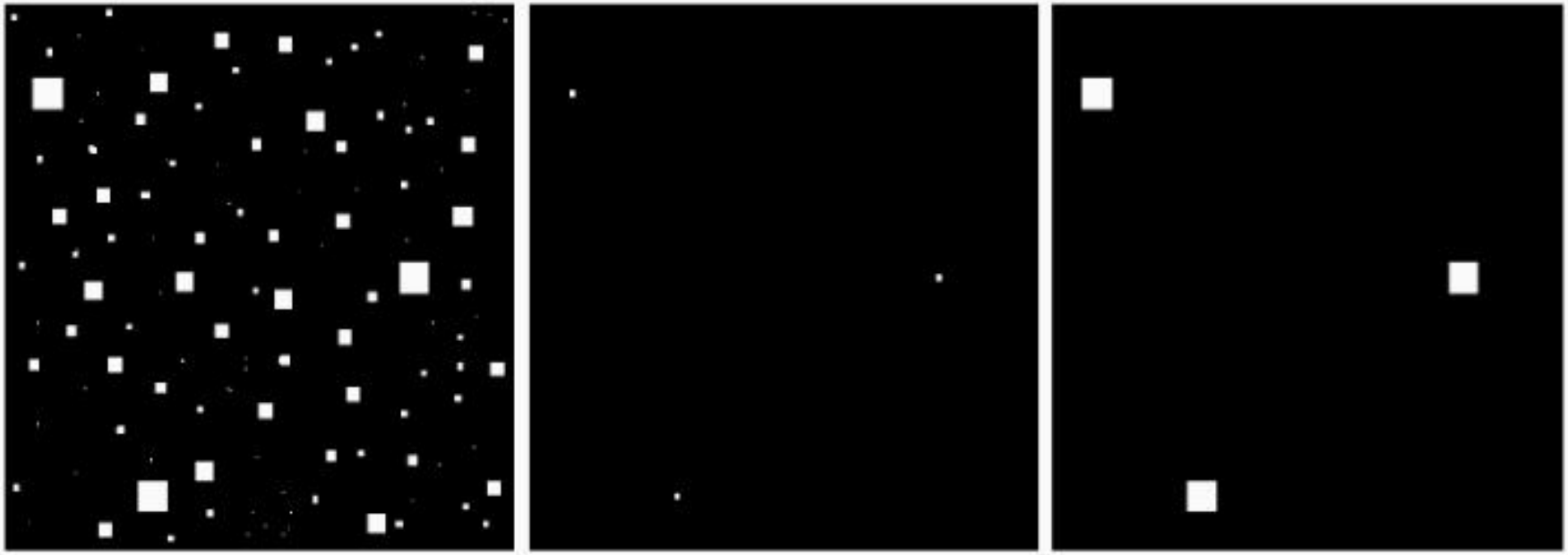
- combinations of dilation and erosion
- removal of separate points can be done with opening
- filling in holes and gaps can be done with closing
- $A \circ B = (A \ominus B) \oplus B \subseteq A \qquad A \bullet B = (A \oplus B) \ominus B \supseteq A$
- $A \circ B = \bigcup \{(B)_z \mid (B)_z \subseteq A\}$
- operations are duals: $(A \bullet B)^c = A^c \circ \widehat{B}$
- opening and closing are idempotent

$$A \circ B = (A \circ B) \circ B$$
$$A \bullet B = (A \bullet B) \bullet B$$

- it can be said that A is open / closed with respect to B

An example of opening (9.3)

original | after erosion | after dilation



Contents

7. Data structures

7.1 Introduction (4.1/3.1)

- data structures pass information from one abstraction level to another
- different information abstractions call for different data structures
- data structures and algorithms are always coupled
  - "iconic" pixel image
  - edges detected in image to match object borders
  - image segmented in regions
  - geometric representations
  - relational models

7.2 Traditional data structures (4.2/3.2)

- matrices
  - spatial and neighborhood relations
  - binary / grayscale / multichannel
  - use of different resolutions leads to hierarchic structure
  - co-occurrence matrix, integral image matrix
- chain codes
- topological descriptions
- relational structures

7.3 Hierarchic data structures (4.3/3.3)

- matrix pyramids and tree pyramids
- quadtrees

7.4 Co-occurrence matrix (4.2.1/3.2.1)

- 2-dimensional generalization of histogram
- joint distribution of graylevel values of neighboring pixels

$$C_r(z_1, z_2) = \#\{f(x_1, y_1) = z_1, f(x_2, y_2) = z_2, (x_1, y_1)r(x_2, y_2)\}$$

- if relation r is $=$, then $C_r(z, z)$ is histogram
- typically r is a shift: $x_2 = x_1 + \Delta x, \quad y_2 = y_1 + \Delta y$
- often r is assumed symmetric $\rightarrow C_r$ is symmetric
- measurement of edges of specific orientation and values
- texture analysis
- intermediate representation for feature extraction

7.5 Integral image matrix (4.2.1/3.2.1)

- Calculating sums like

$$S(x_0, y_0, x_1, y_1) = \sum_{x=x_0}^{x_1} \sum_{y=y_0}^{y_1} f(x, y)$$

  exhaustively over a range of x_0, y_0, x_1, y_1 is time consuming.

- An efficient solution is to use *integral image*:

$$iif(x, y) = \sum_{i=0}^{x} \sum_{j=0}^{y} f(i, j)$$

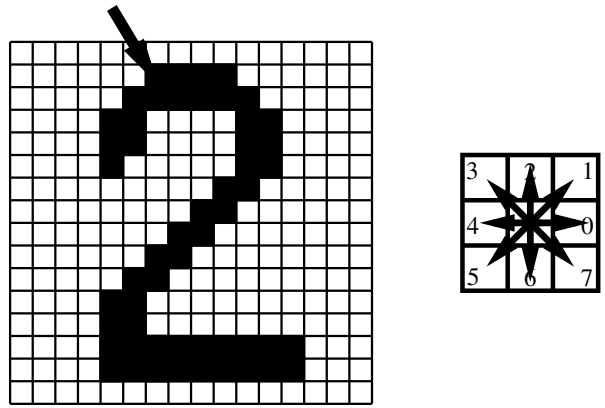
- Then the calculation of $S()$ reduces to three additions:

$$\begin{aligned} S(x_0, y_0, x_1, y_1) = {} & iif(x_1, y_1) - iif(x_1, y_0 - 1) \\ & - iif(x_0 - 1, y_1) + iif(x_0 - 1, y_0 - 1) \end{aligned}$$

- Used successfully eg. in Viola-Jones face detector.

7.6 Chain structures (4.2.2/3.2.2)

- used often for describing object boundaries
- chain-coded object is not bound to any specific location
- chain code aka Freeman code aka F-code
  - directions between adjacent 4- or 8-neighbor pixels
  - starting point has to be fixed

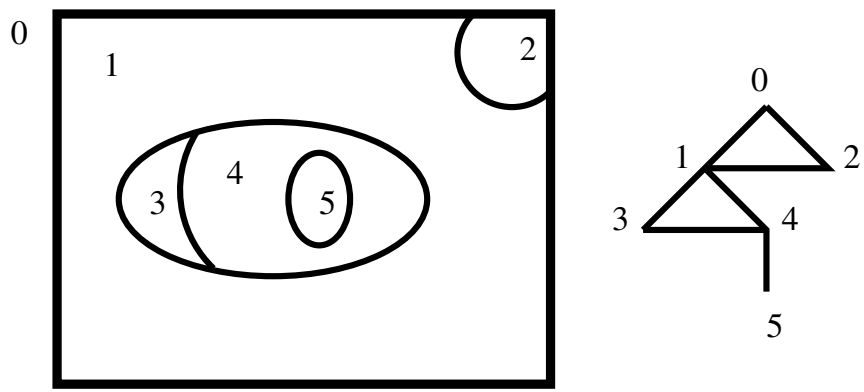


0007766555555660000006...

- vectors between chained pixels can also be longer than one
- chains can be either closed or open
- rotations are easy to implement with chain codes

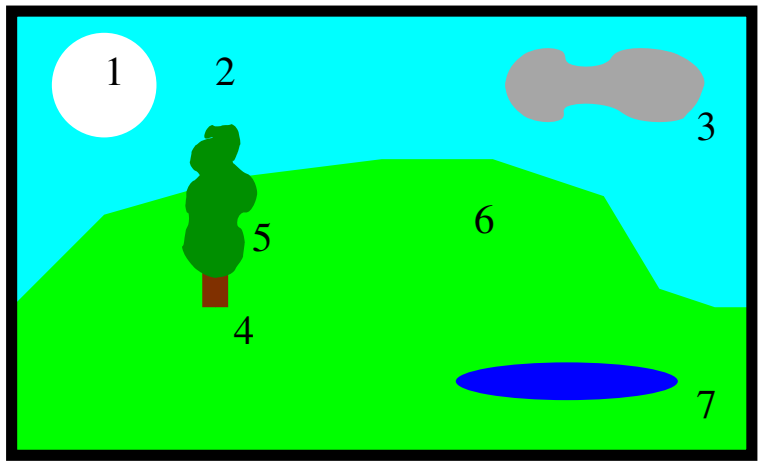
7.7 Topological data structures (4.2.3/3.2.3)

- graphs $G = (V, E)$
- nodes $V = \{v_1, v_2, \cdots, v_n\}$
- edges $E = \{e_1, e_2, \cdots, e_m\}$
- degree of a node
- **weighted or evaluated graph**: costs associated to nodes and edges
- **region adjacency graph** and **region map**



7.8 Relational database structures (4.2.4/3.2.4)

Nr.	Object	Color	Start row	Start column	Inside of
1	sun	white	5	40	2
2	sky	blue	0	0	-
3	cloud	gray	20	180	2
4	tree trunk	brown	95	75	6
5	tree crown	green	53	63	-
6	hill	light green	97	0	-
7	pond	blue	100	160	6



7.9 Hierarchical data structures (4.3/3.3)

Pyramids (3.3.1/4.3.1)

- matrix or M pyramids
  - series of matrices $\{M_L, M_{L-1}, \cdots, M_0\}$
  - $M_L =$ original image
  - $M_0 = 1$ pixel
  - $M_{i-1} = \frac{1}{4}$ of M_i
- tree or T pyramids
  - graph where nodes are placed in layers
  - each layer matches a matrix in M pyramid
  - each node has 4 child nodes
  - method for calculating values in parent nodes is needed
  - total number of nodes $N^2(1 + \frac{1}{4} + \frac{1}{16} + \cdots) \approx 1.33N^2$

Quadtrees (4.3.2/3.3.2)

- resemble T pyramids, but:
- only heterogeneous nodes are divided
- non-balanced tree
- sensitive to small changes in input images
- bounding to object coordinates
- paths can be coded as symbol strings

Contents

The gradient as a sharpening method (3.6.4/3.7.3)

The gradient vector is defined in a general form:

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

The length of the gradient vector (often called "gradient") is:

$$\text{mag}(\nabla f) = |\nabla f| = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$

Robert's cross-gradient, $g_x = z_9 - z_5$, $g_y = z_8 - z_6$:

-1	0
0	1

and

0	-1
1	0

Sobel operators: $G_x =$

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

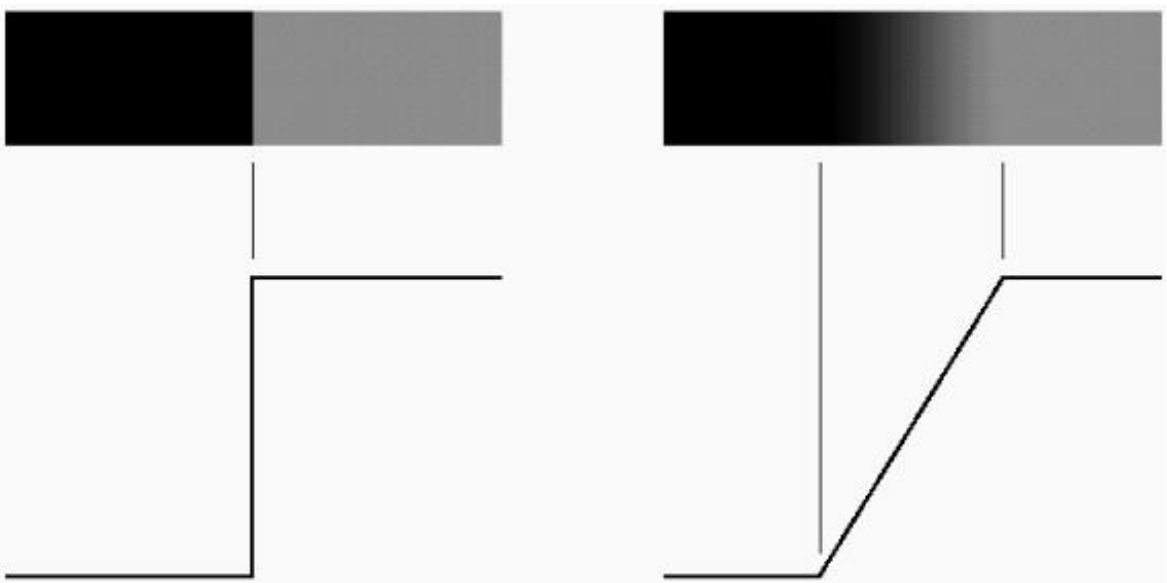
and $G_y =$

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

Edge detection (10.2.4/10.1.3)

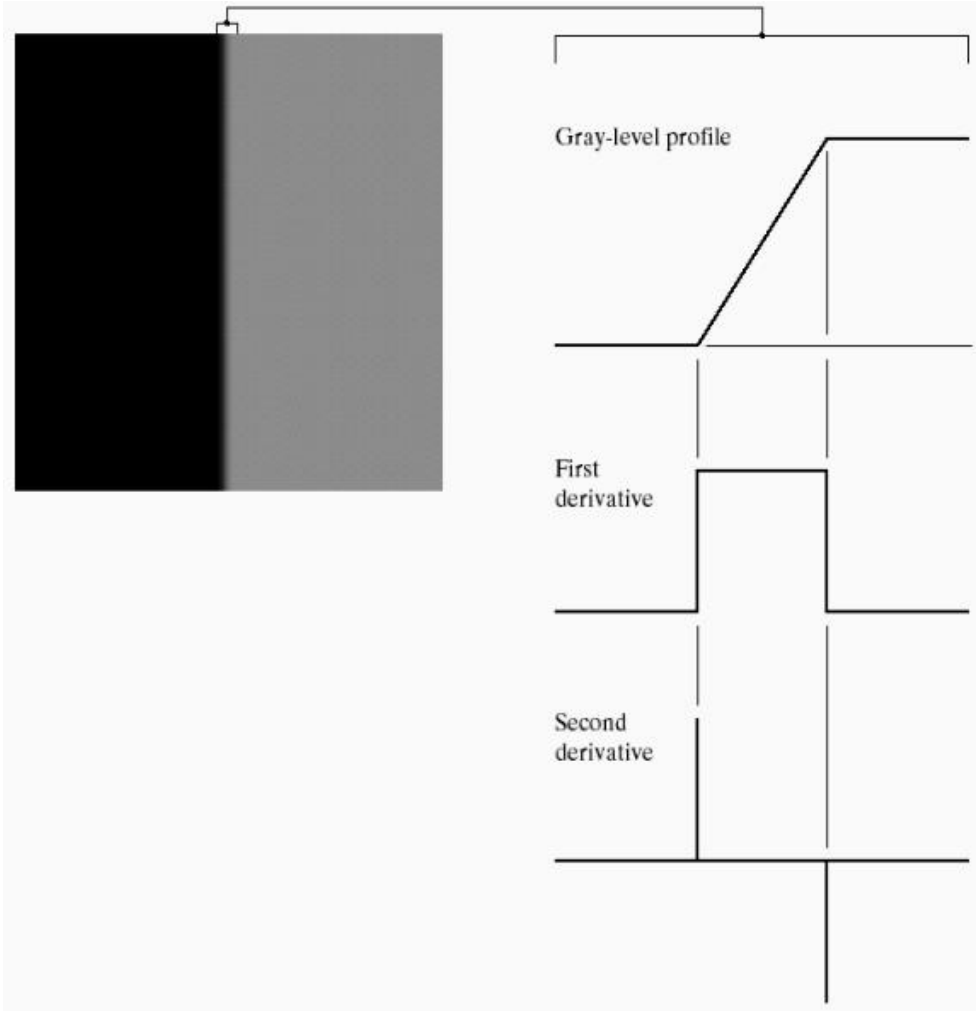
Edge detection is the most important discontinuity detection task, because detection of isolated points or one-pixel lines is seldom needed in intensity and color images.

Idealized edge models (10.2.4/10.1.3)



step edge ramp edge

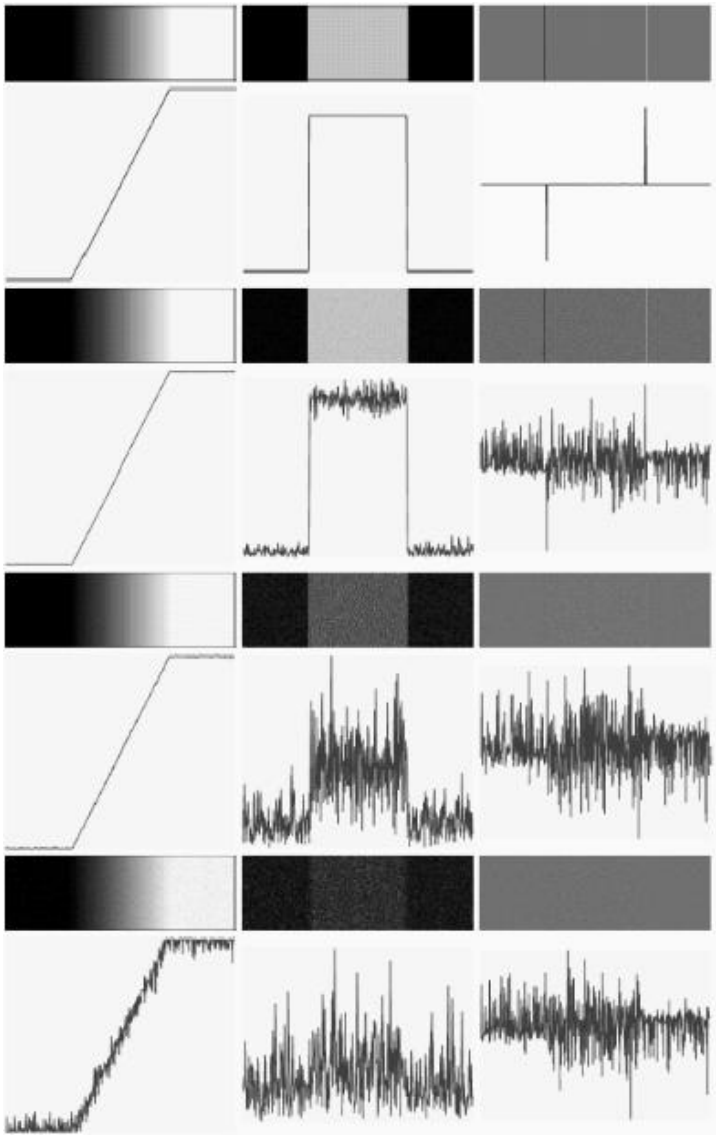
Derivatives of the ramp edge in the ideal case (10.2.4/10.1.3)



We can see the ramp edge between the dark and bright areas in the one-dimensional continuous case.

The shape of the edge is characteristic also in the first and second derivatives as *maximum values* and *zero crossings*, respectively.

Derivatives of ramp edge in noise (10.2.4/10.1.3)



When noise is added, the ideal shapes of the ramp edge derivatives are severely corrupted.

Especially the second derivative and its zero crossings become useless.

The situation can be eased by first low-pass filtering the noisy and then performing the derivation.

Gradient operators (10.2.5/10.1.3)

Edge detection in real two-dimensional discrete images is much more difficult, but derivation can be applied for it too. The most important discrete derivates are *gradient operators* and the *discrete Laplace operator*.

−1	0	0	−1
0	1	1	0

Roberts

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

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

Sobel

The most common gradient operators in image processing are Roberts, Prewitt and Sobel masks.

The value of a discrete gradient is a two-dimensional vector that indicates the rate of change in horizontal and vertical directions.

Combining the gradient to a scalar tells the absolute magnitude value, but not the direction of the change:

$$\text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \approx |g_x| + |g_y|$$

An example of gradient operators (10.2.5/10.1.3)

original

lowpass filtered

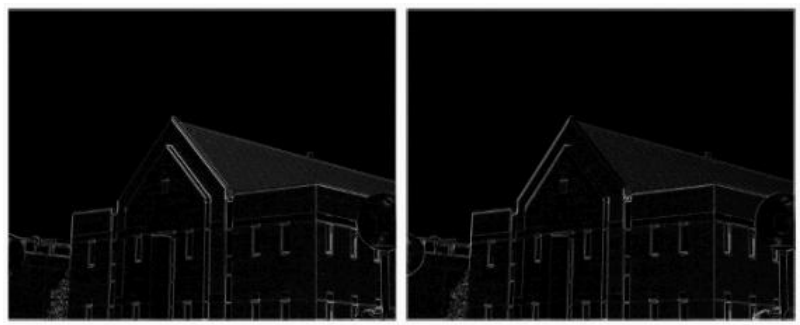
0	1	1	−1	−1	0
−1	0	1	−1	0	1
−1	−1	0	0	1	1

Prewitt

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

Sobel

$\Longrightarrow$



Laplace operator and Marr-Hildreth detector (10.2.6/10.1.3)

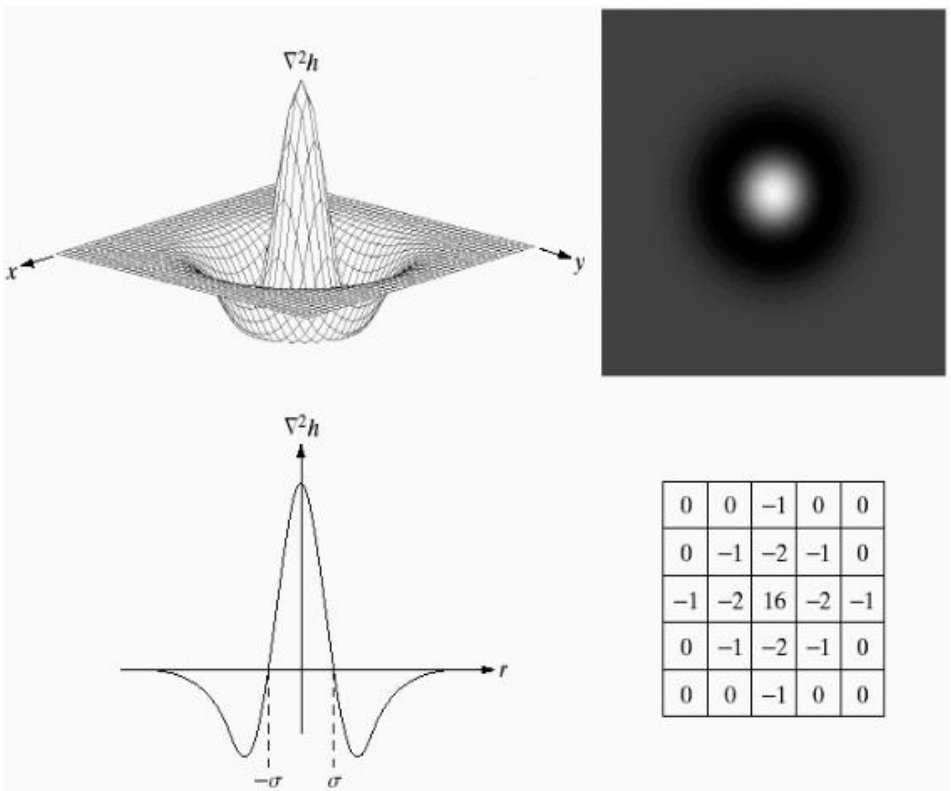
The Laplace operator approximates the two-dimensional second derivative:

$\nabla^2 \approx$

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

or

-1	-1	-1
-1	8	-1
-1	-1	-1



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

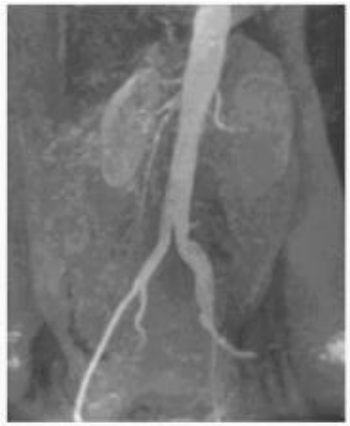
The Laplace operator can also be made larger than 3×3 pixels by combining it with low-pass filtering. This approach is called *Laplace of Gaussian* or LoG filtering and it is used in the Marr-Hildreth edge detector:

$$G(r) = -e^{-\frac{r^2}{2\sigma^2}}$$

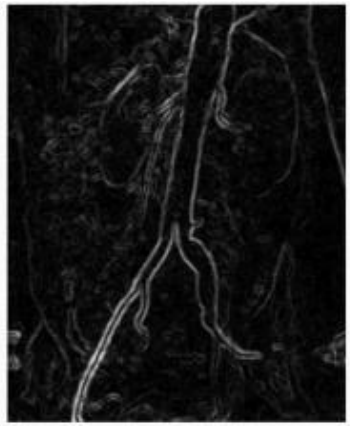
$$\nabla^2 G(r) = -\frac{r^2-\sigma^2}{\sigma^4} e^{-\frac{r^2}{2\sigma^2}}$$

An example of LoG filtering and zero crossings (10.2.6/10.1.3)

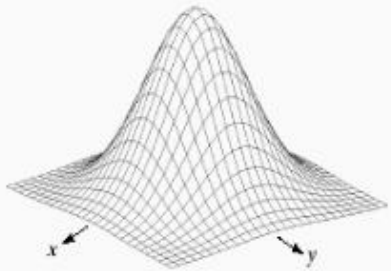
original



Sobel gradient



lowpass



Laplace mask

−1	−1	−1
−1	8	−1
−1	−1	−1

LoG

zero crossings

8.6 Local neighborhood in edge detection (5.3.2/4.3.2)

- gradient has direction and magnitude

$$|\text{grad } g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \qquad \psi = \arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right)$$

- Laplacian

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

- image sharpening, unsharp masking

$$f(i,j) = g(i,j) + CS(i,j)$$

- approximation of derivatives
- zero-crossings of the second derivative
- parametric fitting

8.7 Edge detection by derivative approximation (5.3.2/4.3.2)

- Roberts

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

- Prewitt

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \cdots$$

- Sobel

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

- Laplace

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix} \quad \begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix}$$

8.8 Marr-Hildreth edge detector (5.3.3/4.3.3)

- edge detection from second derivative zero-crossings
- image smoothing by a Gaussian kernel

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

- calculation of Laplace image

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

- association order of the operators is changed: Laplace of Gaussian, LoG

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

- algebraic solution of the second derivative

$$G''(r; \sigma) = \frac{1}{\sigma^2}\left(\frac{r^2}{\sigma^2} - 1\right) e^{-\frac{r^2}{2\sigma^2}}$$

$$h(x, y; \sigma) = c\left(\frac{x^2 + y^2 - \sigma^2}{\sigma^4}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- "mexican hat" function
- proper mask size is approximately $6\sigma \times 6\sigma \cdots 10\sigma \times 10\sigma$
- operator can be separated in x- and y-directions
- resembles the operation of the human eye
- $\nabla^2 G$ can be approximated by the difference of two Gaussians, DoG

8.14 Corner and interest point detection (5.3.10/4.3.8)

- Moravec detector

$$MO(i,j) = \frac{1}{8} \sum_{k=i-1}^{i+1} \sum_{l=j-1}^{j+1} |g(k,l) - g(i,j)|$$

- the facet model can be used for detecting corner points

$$f(x,y) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2 + c_7 x^3 + c_8 x^2 y + c_9 xy^2 + c_{10} y^3$$

  - Zuniga-Haralick operator

$$ZH(i,j) = \frac{-2(c_2^2 c_6 - c_2 c_3 c_5 - c_3^2 c_4)}{(c_2^2 + c_3^2)^{\frac{3}{2}}}$$

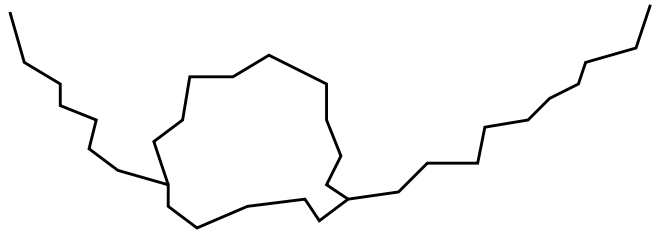
  - Kitchen-Rosenfeld operator

- Harris corner detector

Contents

8.9 Scale-space methods (5.3.4/4.3.4)

- smoothing parameter, eg. σ, is varied to produce a family of images
- 1) curves can be analyzed at multiple scales



- 2) scale-space filtering: $f(x, y)$ image to a set of $F(x, y, \sigma)$ images
  - convolution with Gaussian function (1-dim. case)

$$G(x, \sigma) = e^{-\frac{x^2}{\sigma^2}} , \quad F(x, \sigma) = G(x, \sigma) * f(x)$$

  - edges from second derivative's zero-crossings

$$\frac{\partial^2 F(x, \sigma_0)}{\partial x^2} = 0 , \quad \frac{\partial^3 F(x, \sigma_0)}{\partial x^3} \neq 0$$

  - different qualitative information with different σ, **interval tree**

8.10 Canny edge detector (5.3.5/4.3.5)

- optimal for step-shape edges in additive white noise
  - **detection**: edges are not missed, no spurious responses
  - **localization**: located and actual positions near each other
  - **uniqueness**: single edge doesn't produce multiple responses

1. image f is convolved with σ-scale Gaussian function
2. local edge's normal direction is estimated in each pixel

$$\mathbf{n} = \frac{\nabla(G * f)}{|\nabla(G * f)|}$$

3. 2nd derivative's zero-crossings are located in the normal direction

$$\frac{\partial^2}{\partial \mathbf{n}^2} G * f = 0, \quad \textbf{non-maximal suppression}$$

4. edge thresholding with **hysteresis**
    - generalization of thresholding with **high** and **low thresholds**
    - weak edge pixels are supported by strong nearby edge pixels
    - only strong changes are detected, increased signal-to-noise ratio
5. edge information is collected with different σ values
    - edge **feature synthesis** from small σ to large σ
    - differences between prediction and reality give true information

Contents

11. Segmentation

- splitting image into semantically meaningful regions
- complete segmentation
  - disjoint regions correspond uniquely with objects in the image
  - information from higher-level processing stages
- partial segmentation
  - similarity between pixels and regions, homogeneity
- segmentation methods
  - thresholding: global knowledge concerning the whole image
  - edge-based segmentation
  - region-based segmentation
  - template matching

11.1 Thresholding methods in segmentation (6.1/5.1)

- complete segmentation in S regions $R_1, \ldots, R_S$:

$$R = \bigcup_{i=1}^{S} R_i \quad \text{and} \quad R_i \cap R_j = \emptyset, \ \forall i \neq j$$

- selecting of a global threshold T
- values larger / smaller than T are background / object
- difficult to find an efficient global solution
- $\Rightarrow$ segmentation in partial images
- background or object can also be a range of values
- creation of an edge image with a narrow range of values
- many simultaneous value ranges

Thresholding can be based on

- grayvalues
- gradient
- texture
- motion
- something else

Threshold selection methods (6.1.1/5.1.1)

- histogram analysis and filtering, possibly in many scales
- is the total area of the objects known?
- uni-, bi- or multi-modal histogram?
- local maxima, minimum distance between them
- histogram extracted from small gradient pixels only
- uni-modal histogram from large gradient pixels

Optimal thresholding (6.1.2/5.1.2)

- model of the distribution needed
- fitting of normal distributions in the histogram
- iterative selection of the parameters
- initial guess for the background from image corners

Segmentation of multi-spectral images (6.1.3/5.1.3)

- each channel segmented separately
  - channel-wise histogram peaks with lower and upper limits
  - union of boundaries from all channels
  - iterative division of the created regions
- multi-dimensional histograms
- classification of n-dimensional pixels

Thresholding in hierarchical data structures (6.1.4/-)

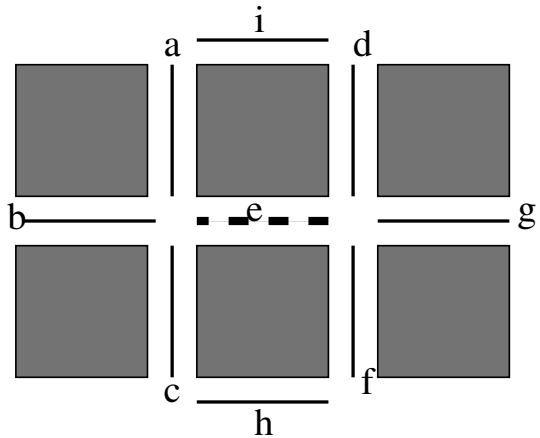
- computational efficiency
- removal of noise
- lowering in data pyramid onto higher-resolution level
- detection of important pixels on all levels in 3×3-size
- threshold is fixed on higher levels
- same or updated threshold is used on lower levels

11.2 Edge-based segmentation (6.2/5.2)

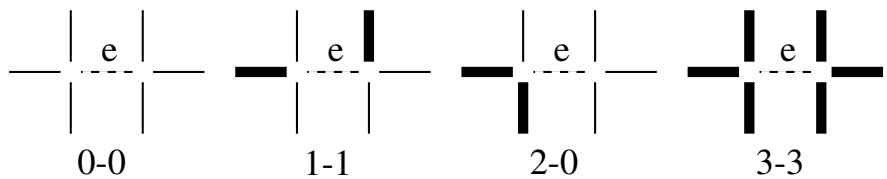
- edge-based methods are important historically and in practice
- edges need to be developed to boundaries and regions
- importance of *a priori* knowledge
- comparison between detected edges and model predictions
- methods and topics:
  - thresholding of gradient magnitude or something
  - non-maximal suppression
  - hysteresis
  - relaxation
  - border tracing
  - use of location information
  - region construction from borders

Edge relaxation (6.2.2/5.2.2)

- comparison of edge information between pixels
- iteration until coherence between neighboring pixels reached
- crack edges



- types of crack edges



- each crack edge has a confidence value $0 \leq c^{(k)}(e) \leq 1$

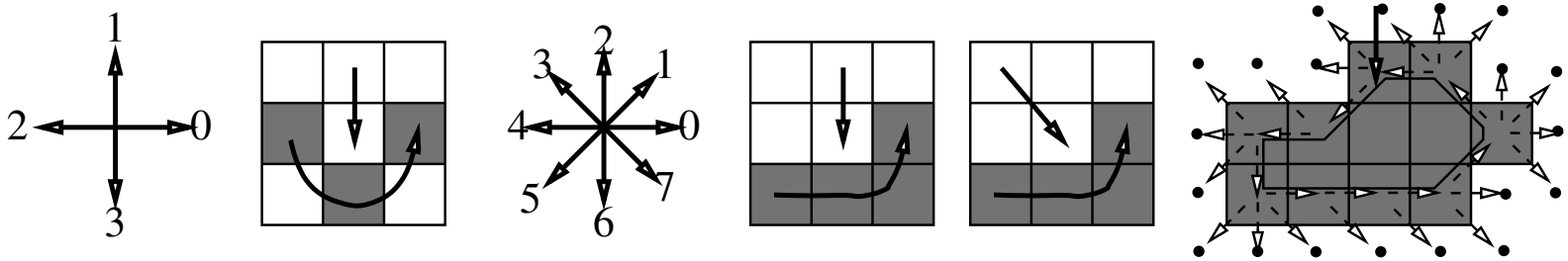
- classification of crack edge types for each end (a, b, c and d, g, f)

$$a \geq b \geq c, \quad m = \max(a, 0.1)$$
$$\text{type}(i) = \max_{k=0,1,2,3} \text{type}(k)$$
$$\text{type}(0) = (m-a)(m-b)(m-c)$$
$$\text{type}(1) = a(m-b)(m-c)$$
$$\text{type}(2) = ab(m-c)$$
$$\text{type}(3) = abc$$

- modification of each crack edge confidence value
  - 0–0, 0–2, 0–3, $c^{(k)}(e)$ decreases
  - 0–1, $c^{(k)}(e)$ increases little
  - 1–1, $c^{(k)}(e)$ increases very much
  - 1–2, 1–3, $c^{(k)}(e)$ increases quite much
  - 2–2, 2–3, 3–3, no change
- iteration should not be too long, possibly non-linear roundings towards 0 and 1

Border tracing (6.2.3/5.2.3)

- aim is to find a route around the object
- simple for binary images: $\approx$ change of presentation to chain code
- difficult for gray-scale images
- inner/outer border/boundary have different lengths
- 8- and 4-neighbors mixed on the opposite sides of the boundary
- starting from the top-left corner of the object



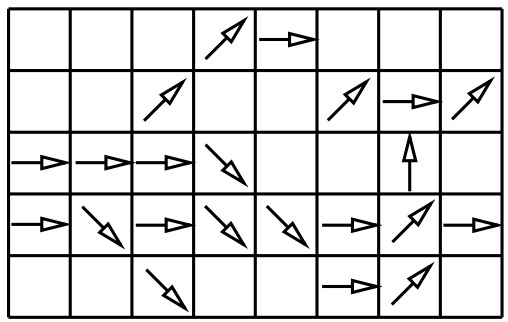
- outer boundary from the tested non-object pixels
- one pixel can belong to the boundary more than once: length?

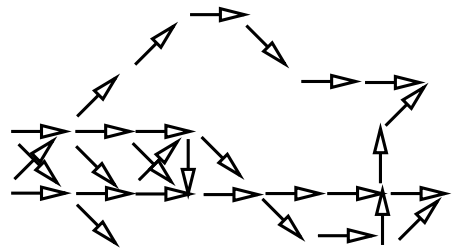
Border detection in gray-scale images

- much more difficult than for binary images
- gradient or other edge image is created first
  - current boundary direction is continued to locate more edge pixels
  - gradient directions are compared between neighboring pixels
  - gray-value tells whether we are inside or outside the object
- turns and weak edges cause difficulties
- closed boundary can remain undetected
- heuristic search
  - starting from the strongest edges
  - continuation in backward and forward directions

11.3 Border detection as graph searching (6.2.4/5.2.4)

- "A-algorithm"
- some amount of *a priori* knowledge needed: starting and end points assumed known for detecting optimal path between them
- directed weighted graph





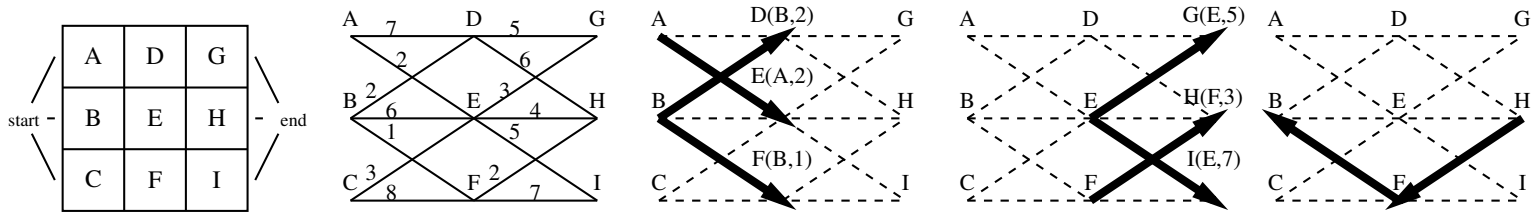
- list of all open nodes, each node listed at most once
- full path cost estimate $f(n_i) = \tilde{g}(n_i) + \tilde{h}(n_i)$
- the lowest cost estimate $f(n_i)$ expanded to new nodes
- if forward direction is known, one tries to follow it
- straightening of the path and image with geometric warping?

Optimal and heuristic search

- generally $\tilde{g}(n_i)$ is the real cost upto that node
- estimate $\tilde{h}(n_i)$ can effect the speed of search
- if $\tilde{h}(n_i) = 0$, optimal result is guaranteed
- if $\tilde{h}(n_i) > h(n_i)$, some result is obtained fast
- if $0 < \tilde{h}(n_i) < h(n_i)$, optimal result, iff $c(n_p, n_q) \geq \tilde{h}(n_p) - \tilde{h}(n_q)$
- if $\tilde{h}(n_i) = h(n_i)$, optimal result with minimal computation
- cost function $f()$ can contain
  - strength of edges, inverse of gradient
  - difference between gradient directions of succeeding nodes
  - distance to *a priori* assumed location
  - distance to end point

Border detection as dynamic programming (6.2.5/5.2.5)

- **principle of optimality**: all subpaths of an optimal path are also optimal
- one can use a set of starting and end nodes
- entering direction and cumulated cost stored in each node



$$C(x_k^{m+1}) = \min_i (C(x_i^m) + g^m(i,k))$$

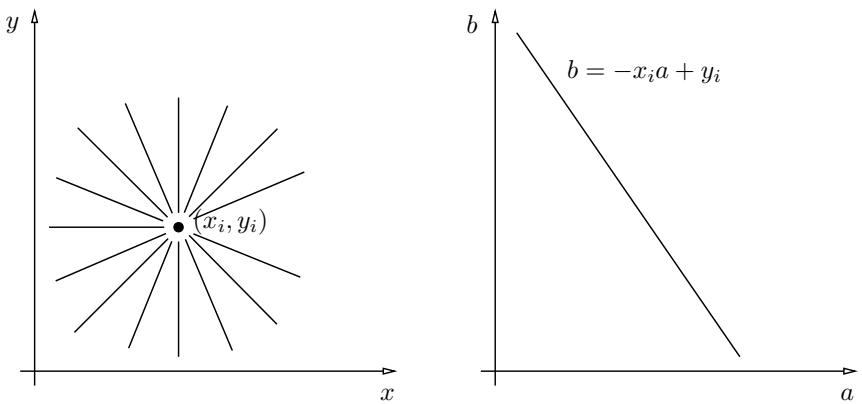
$$C(x_k^{m+1}) = \min_{i=-1,0,1} (C(x_{k+i}^m) + g^m(i,k))$$

$$\min(C(x^1, x^2, \cdots, x^M)) = \min_{k=1,\ldots,n} (C(x_k^M))$$

13.4 Hough transform (10.2.7/10.2.2)

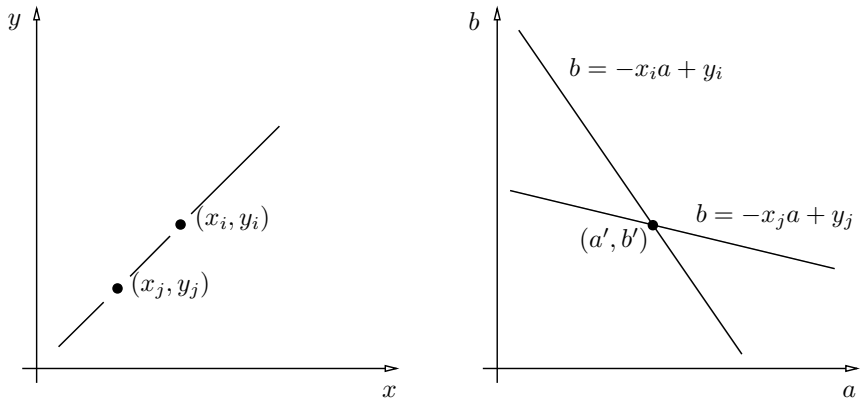
Hough [haff] **transform** is the most important global method for detecting boundaries. Hough transform uses as a linking criterion whether the edge pixels reside along a predefined *parametric curve* or not.

All found edge pixels are used simultaneously and they are matched with different combinations of the curve parameter values. The edge pixels do not need to be adjacent, it suffices if they are on the same curve, typically on the same line.



Edge pixel (x_i, y_i) is known. All lines passing through it satisfy $y_i = ax_i + b$ or $b = -x_i a + y_i$.

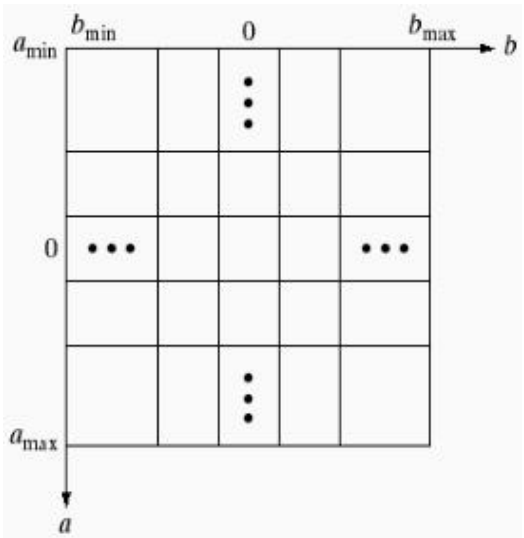
(x_i, y_i) define one line in the ab parameter plane. Each point on the ab line corresponds to different line passing through (x_i, y_i) in the image.



Similarly another edge pixel (x_j, y_j) is parameterized:

$$b = -x_j a + y_j$$

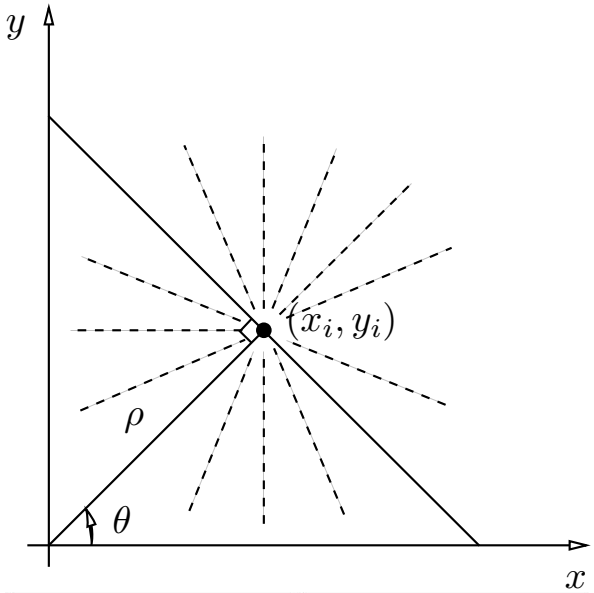
The intersection point (a', b') of these lines can be calculated. and it defines the one line passing through both (x_i, y_i) and (x_j, y_j).

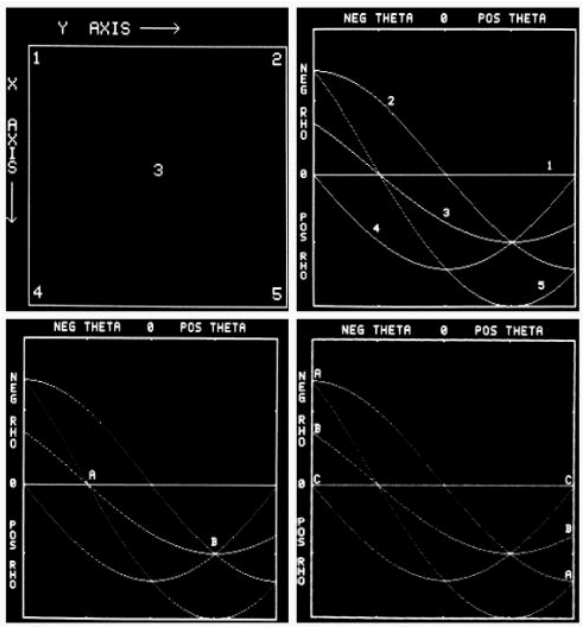


In practice the ab parameter space is quantized as a two-dimensional matrix of *accumulator cells*.

The parameter lines are "drawn" in the accumulator matrix and incrementing the cell values by one.

The maximum values in the accumulator matrix determine the most probable line parameters.





A severe problem with the linear parameterization are lines that have infinite slope, i.e., $a = \infty$ and $b = \infty$. This can be avoided by parameterizing the lines in the $\rho\theta$ plane instead of the ab plane:

$$x \cos\theta + y \sin\theta = \rho$$

Then each image pixel corresponds to a piece of trigonometric curve in the parameter plane. The line parameters are again found with the accumulator matrix.

Hough transform can be used for detecting also more complex curves than lines, e.g., circles with three parameter values.

11.5 Region-based segmentation (6.3/5.3)

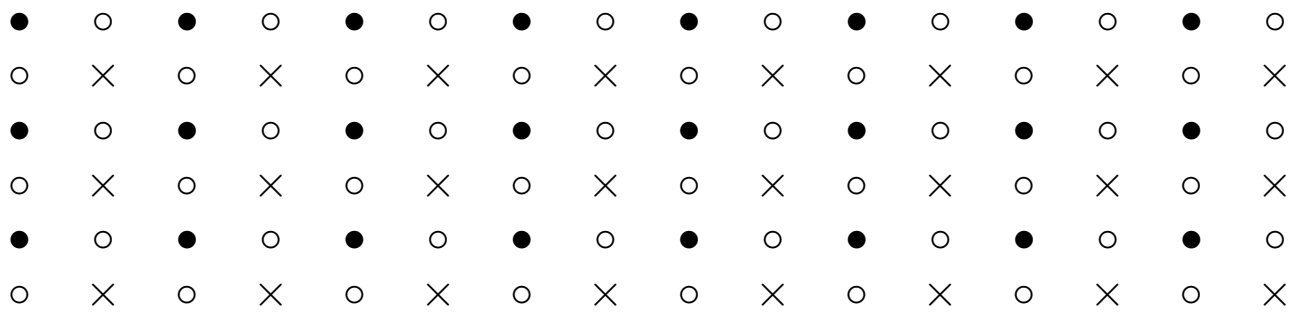
- important to define **homogeneity** criterion for a region
- homogeneity inside regions, heterogeneity between them

Region merging (6.3.1/5.3.1)

- initial state: separate pixels, each in its own segment
- combining first neighboring pixels with same gray-scale values
- conditional merging of adjacent regions

Region merging as state space search

- super grid, ×=image pixel, ○=crack edge:



- thresholding of significant edges

$$v_{ij} = \begin{cases} 0, & \text{if } |f(\mathbf{x}_i) - f(\mathbf{x}_j)| < T_1 \\ 1, & \text{otherwise} \end{cases}$$

- counting the number W of *weak edges* on the separating boundary
- removing (melting) separating boundary if

$$\frac{W}{\min(l_i, l_j)} \geq T_2 \quad \text{or} \quad \frac{W}{l} \geq T_3 \quad \text{or} \quad W \geq T_4$$

Region splitting (6.3.2/5.3.2)

- starting from the whole image in one segment
- segments are split into smaller ones according to some criteria
- criteria can include eg. histogram peaks and existing prominent edges

Splitting and merging (6.3.3/5.3.3)

- quadtrees
- how merging of nodes 03, 1, 30 and 310 is implemented in Fig 6.46/5.47?
- use of overlapping trees
  - each child linked to the most probable parent
  - content of the parents recalculated after reassingment of children

Contents

Hands-on work and home assignment

- Instructions in Google Drive
- Also hands-on work can be done at home
- Report all code and images of the hands-on and the home assignment in the same PDF
- Report also how long time the assignments took
- Email the PDF before midnight to the lecturer