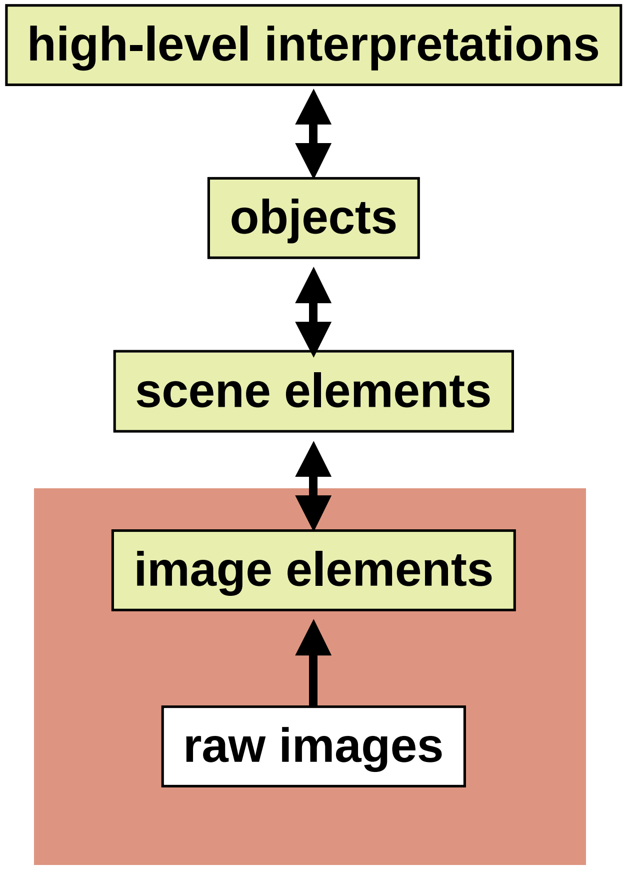
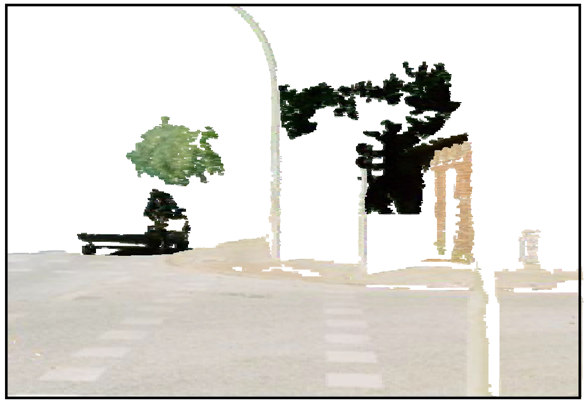
Segmentation

Segmenting the image into image elements which may correspond to meaningful scene elements

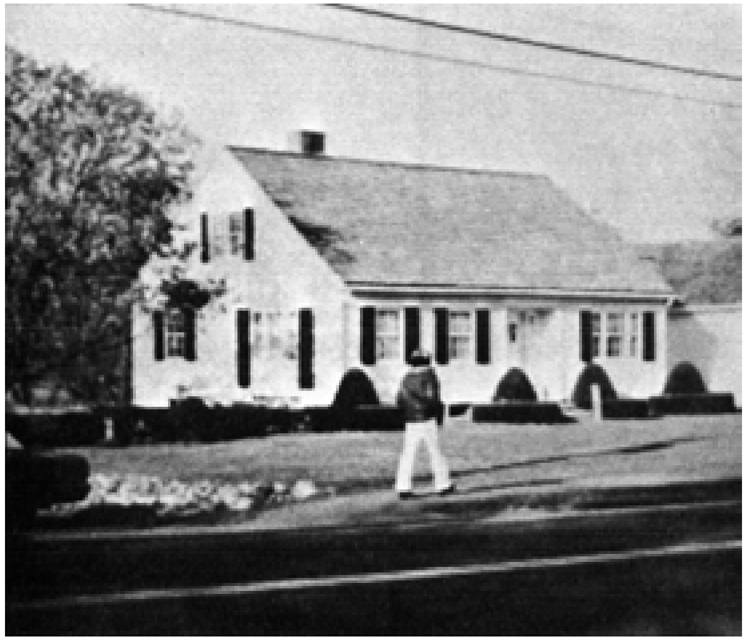


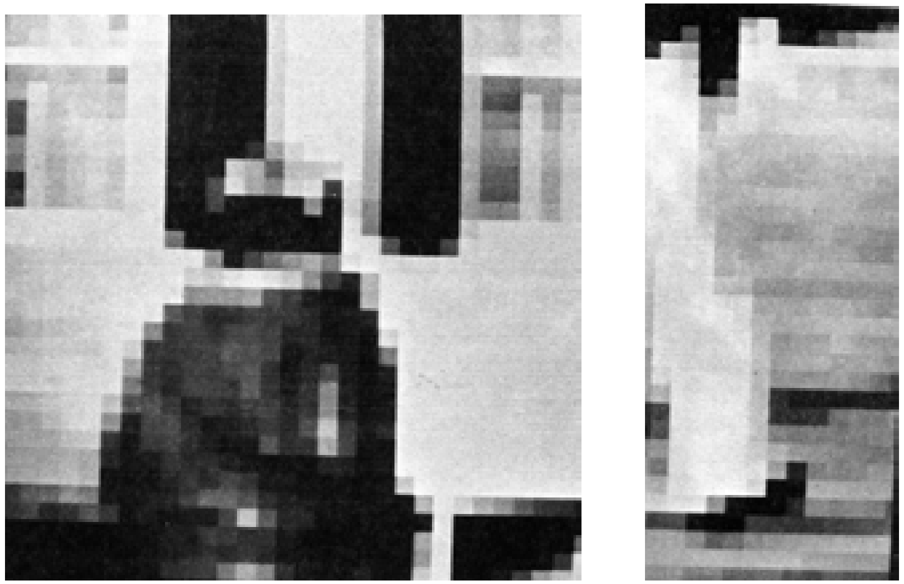
<u>Example</u>: Partitioning an image into regions which may correspond to objects



Typical results of first segmentation steps

Problems with Segmentation


landhouse scene


upper part and leg of person

Greyvalues of foreground may be indistinguishable from greyvalues of background.

In general, context knowledge is necessary for successful segmentation

Primary Goal of Segmentation

"Segmenting an image into image elements which may correspond to meaningful scene elements"

What sort of image elements may correspond to meaningful scene elements?

Answer depends on type and complexity of images: Less constrained scenes must be segmented more conservatively.

Segmentation into ...

... entire objects	e.g. for	printed character recognition industrial object recognition medical cell analysis
... edge lines	e.g. for	aerial image analysis indoor scenes
... edge elements, vertices, groupings		e.g. for natural scenes

Secondary Goals of Segmentation

- **Multiple resolutions for subsequent processes**

  <u>coarse resolution description</u> for e.g.

  - analysis of image layout (horizon, foreground, background)

  - control of attention

  - planning a detailed analysis

  <u>fine resolution description</u> e.g. for

  - details

  - stereo analysis

  - motion analysis

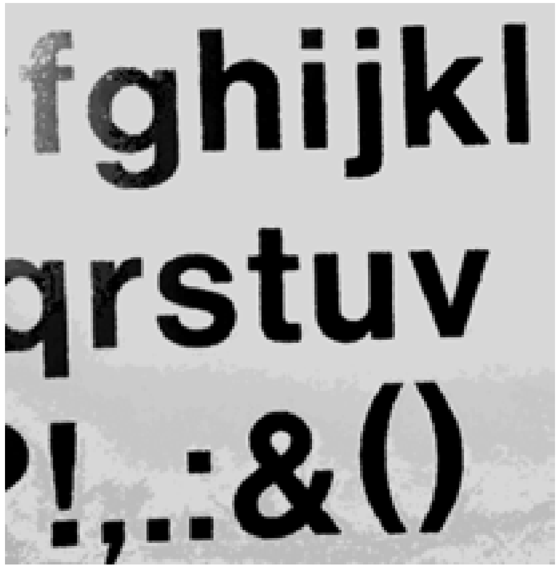
- **Data reduction**

  Because of their large data volume, raw images are inconvenient as basic data structures for image analysis

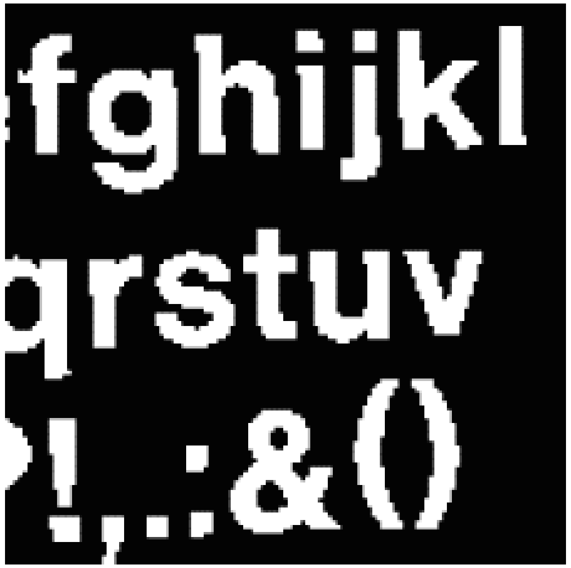
E.g.	**TV colour image**	**$3 \times 512 \times 576 \approx 7$ MB**
	10 sec TV colour images	**$10 \times 25 \times 7 \approx 1750$ MB**

Thresholding

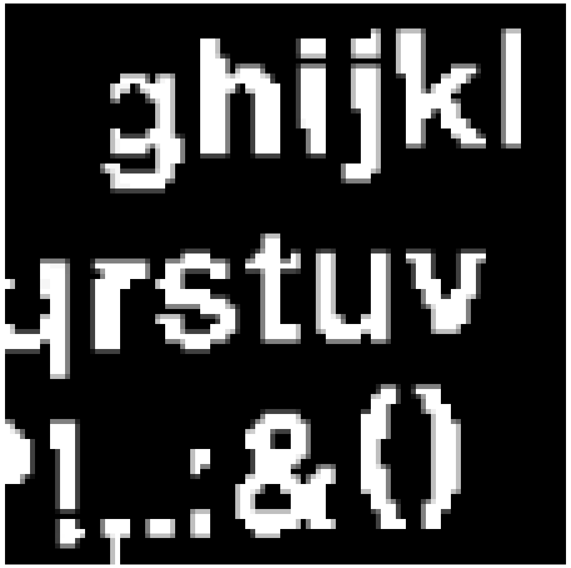
Thresholding has been introduced as a discretization technique. The same techniques can be applied for segmentation.



greyvalue image



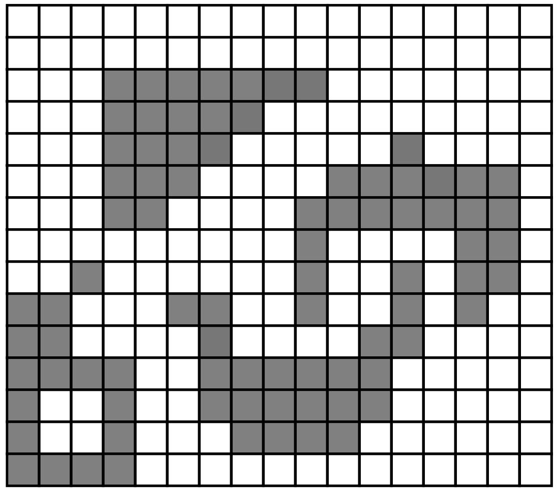
optimal threshold



threshold too high

Representing Regions

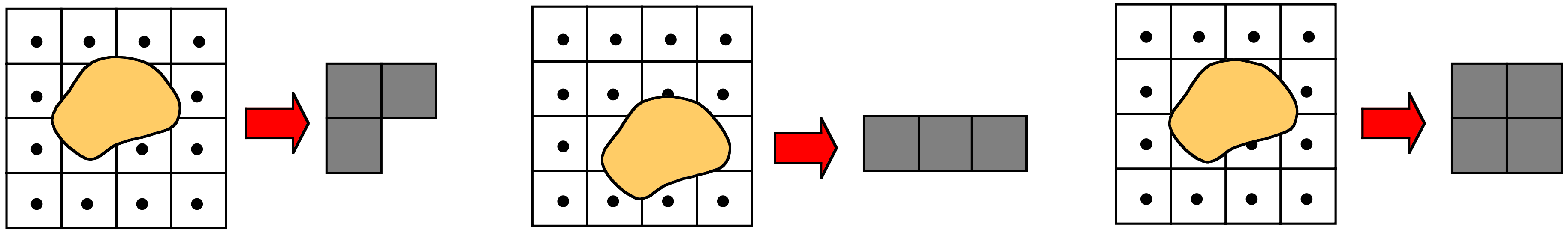
A region is a maximal 4- (or 8-) connected set of pixels.



Methods for digital region representation:

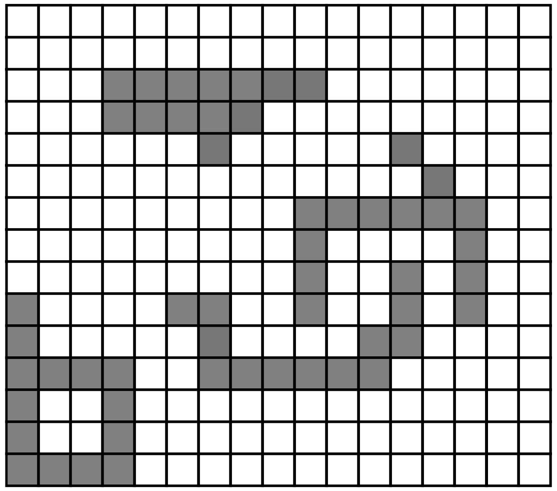
- grid occupancy
  - labelling
  - run-length coding
  - quadtree coding
  - cell sets
- boundary description
  - chain code
  - straight-line segments, polygons
  - higher-order polynomials

Note that discretizations of an analog region are not shift or rotation invariant:



Component Labelling

Determining connected regions in B/W images



Component 1
(2 3 9)(3 3 7)(4 6 6)

Component 2
(4 12 12)

Component 3
(5 13 13)(6 9 14)(7 9 9 14 14)(8 9 9 14 14)(9 9 9 14 14)

Component 4
(9 0 0)(10 0 0)(11 0 3)(12 0 0 3 3)(13 0 0 3 3)(14 0 0 3 3)

Component 5
(9 5 6 12 12)(10 6 6 11 12)(11 6 11)

In this example: component descriptions using run-length coding

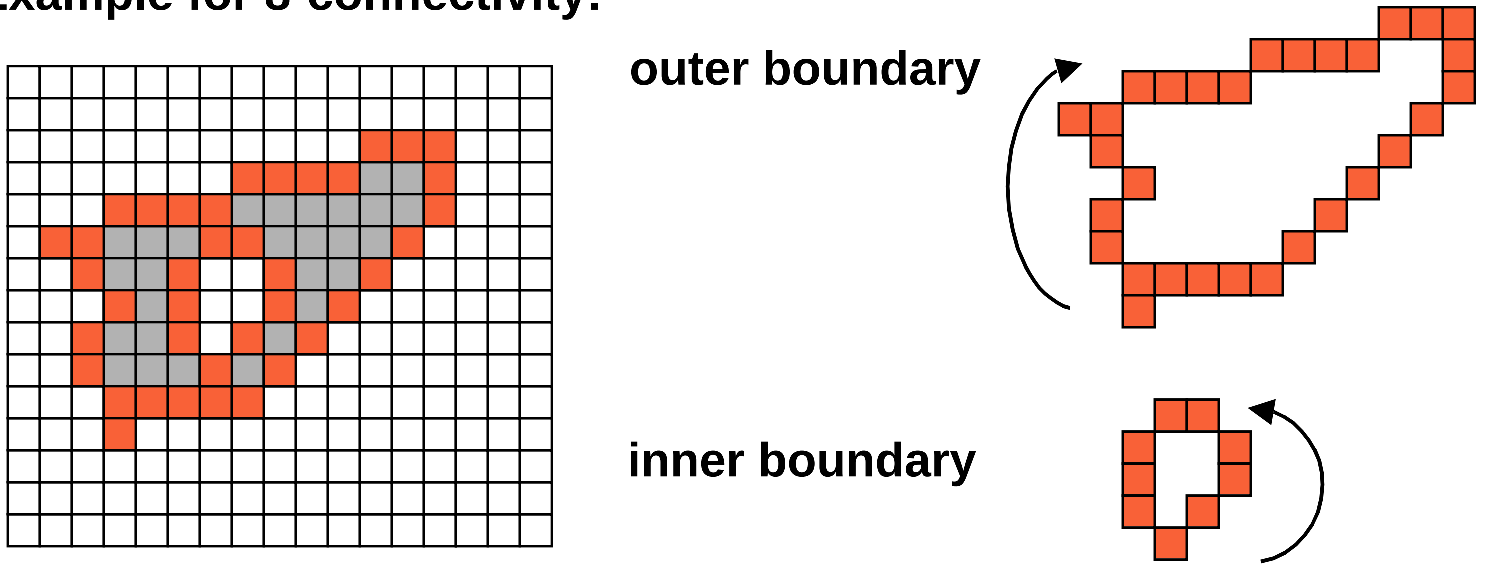
Component labelling of B/W images with 4-neighbourhood

```
Scan image left to right, top to bottom:
    if pixel is white then continue
    if pixel is black then
        if left neighbour is white and upper neighbour is white then assign new label
        if left neighbour is black and upper neighbour is white then assign left label
        if left neighbour is white and upper neighbour is black then assign upper label
        if left neighbour is black and upper neighbour is black then
            assign left label, merge left label and upper label
```

Boundaries

For a 4- (8-) connected region R the boundary is defined as the set of pixels of R which are 8- (4-) connected to the complement R^c of R.

Example for 8-connectivity:



Boundary pixels are usually ordered clockwise for outer boundaries and counter-clockwise for inner boundaries.

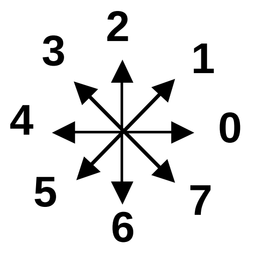
<u>Disadvantage</u> of this boundary definition:

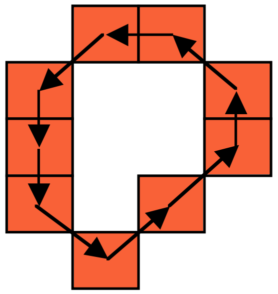
R and R^c have different boundaries - but nothing is in between.

Chain Code

Chain code represents boundaries by "chaining" direction arrows between successive boundary elements.

Chain code for 8-connectivity:





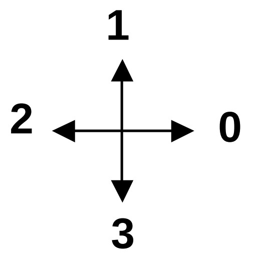
Arbitrary choice of starting point, chain code can be represented e.g. by

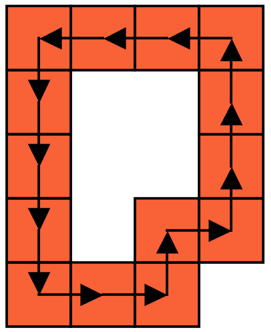
{456671123}

Normalization by circular shift until the smallest integer is obtained:

{112345667}

Chain code for 4-connectivity:





Arbitrary starting point:

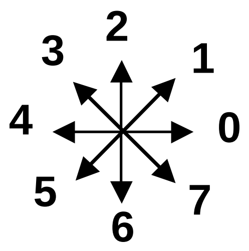
{22233330010111}

Normalized:

{00101112223333}

Chain Code Derivatives

Chain code is highly susceptible to discretization noise. Hence derived properties are usually also noisy.

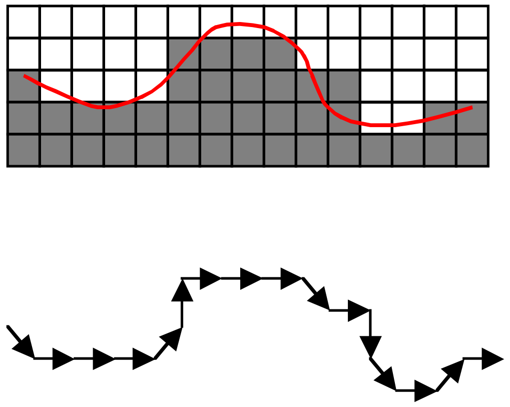


Slope:

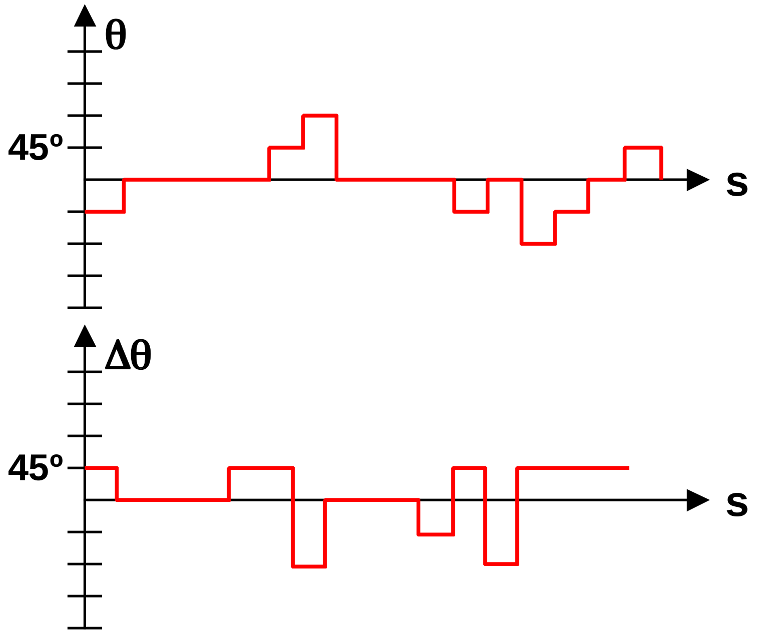
chain code	0	1	2	3	4	5	6	7
tan θ	0	1	±∞	-1	0	1	±∞	-1
θ	0	45	90	135	±180	-135	-90	-45

Curvature: $\Delta\theta = \theta_{i+1} - \theta_i$

Example:



{7000120007067010}



k-Slope and k-Curvature

Smoothed chain code slope and curvature:

L	chain code
$\{p_1 \dots p_N\}$	starting points of chain code elements

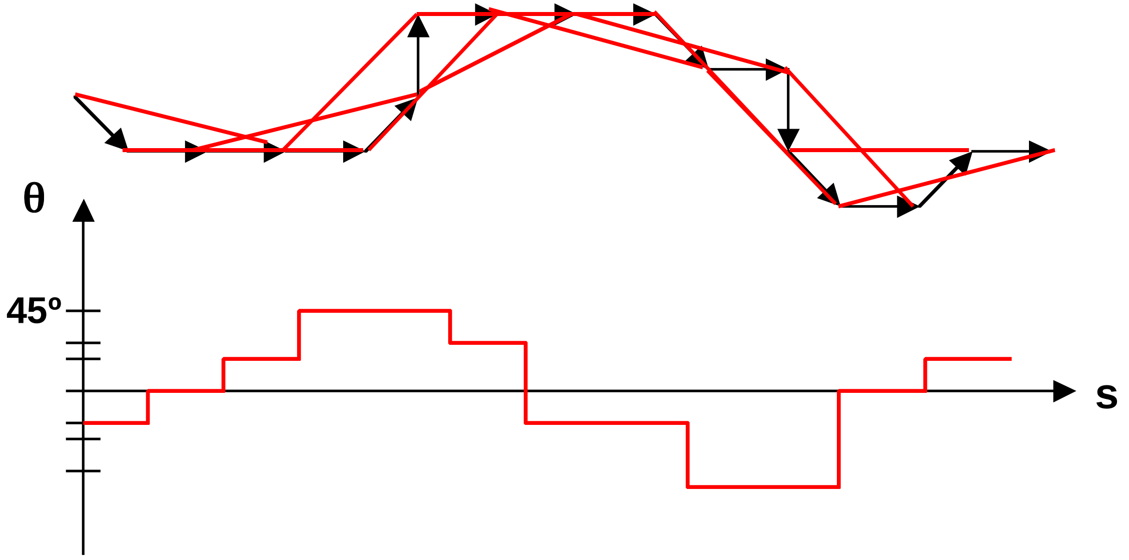
<u>right k-slope</u> of L at i, k≥1, is slope from p_i to p_{i+k}

<u>left k-slope</u> of L at i, k≥1, is slope from p_i to p_{i-k}

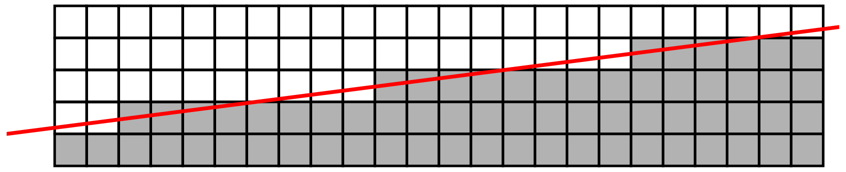
<u>k-curvature</u> at i is difference between right and left k-slope

<u>Example</u>:

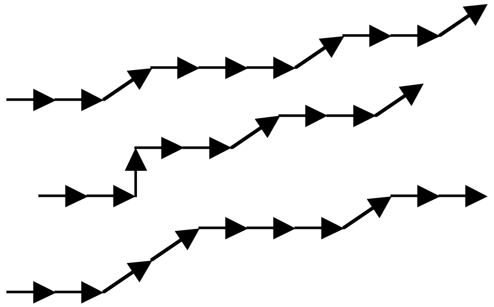
k = 3



Digital Straight Lines



What are the properties of a chain code which represents a straight line boundary?



may represent a straight line

may not represent a straight line

may not represent a straight line

Necessary and sufficient straight line properties of chain code:

1. Only 2 element types
2. Numerical difference of element types (mod 8) at most 1
3. One of the element types occurs only in runs of length 1 and is distributed "as regularly as possible".

"as regularly as possible": Assume 2 types a and b, b single. Runs of a must have lengths l_0 and l_0+1. Consider l_0-runs and l_0+1-runs as 2 chain code types and apply straight line criteria recursively.

Uniformity Assumption

Many segmentation procedures are based on a uniformity assumption:

- meaningful objects correspond to regions which satisfy a uniformity predicate

  => region finding
- object boundaries correspond to discontinuities of a uniformity predicate

  => edge finding

Typical uniformity predicates:

- greyvalues within a narrow interval (e.g. in B/W images)
- similar colour
- small greyvalue gradient
- uniform statistical properties (e.g. local distribution, texture)
- smoothness in 3D

Region Growing

Regions which satisfy a uniformity criterion may be grown from seed regions based on two criteria:

1. Merge region with new area if merged region satisfies uniformity criterion.

E.g. greyvalue variance remains limited

2. Merge region with new area if boundary area satisfies a merging criterion.

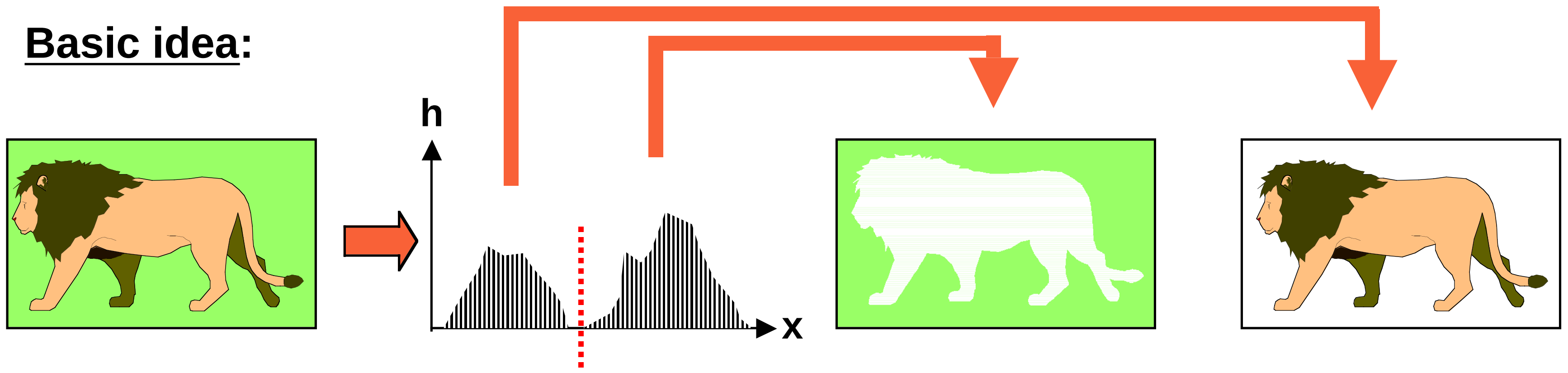
E.g. boundary area has weak edges

Problem with (1): Large regions may be merged with small patches even if the patches are distinctly different.

Problem with (2): Distinct large regions may be merged if they are connected by a weak boundary.

Segmentation into Regions Using Histograms

Basic idea:



Recursive histogram decomposition:

- compute 1D histograms of pixel features (e.g. R, G, B histograms)
- use "clearest" histogram for decomposition into regions
- apply procedure recursively to individual regions

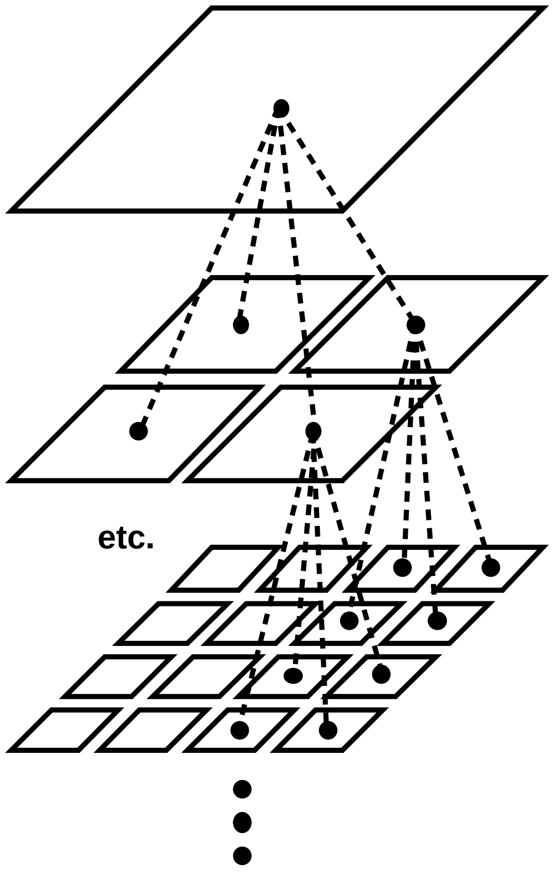
Problems:

- histograms do not reflect neighbourhood relationships
- histograms may not show multimodality clearly
- bad early decisions cannot be corrected

Region Segmentation by Split-and-merge

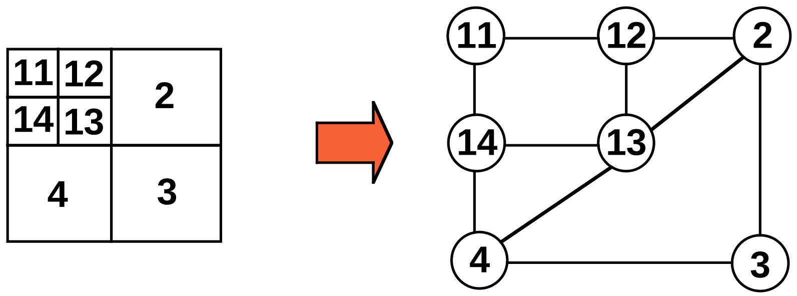
Region boundaries are determined along quadtree region boundaries.

- Begin with an arbitrary region decomposition in a quadtree plane
- Split each region which violates a uniformity predicate into its 4 quadtree sons
- Merge (recursively) all regions which jointly satisfy a uniformity criterion

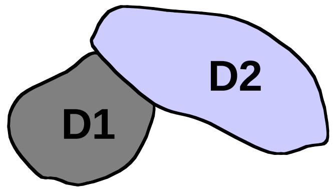


Supporting data structure:

Region adjacency graph



Maximum-likelihood Edge Finding



Hypothesis test about the likelihood of a boundary between two regions D_1 and D_2

H_0: Pixels from D_1 and D_2 stem from the same statistical source $N(\mu_0, \sigma_0)$

H_{12}: Pixels from D_1 and D_2 stem from different statistical sources $N(\mu_1, \sigma_1)$ and $N(\mu_2, \sigma_2)$, respectively.

Maximum-likelihood decision chooses hypothesis H_i for which $P(g_{ij}$ are observed | H_i is true) is maximal.

Step 1: Maximum-likelihood estimation of $\mu_0, \sigma_0, \mu_1, \sigma_1, \mu_2, \sigma_2$

$$\hat{\mu}_i = \frac{1}{|D_i|} \sum_{ij \in D_i} g_{ij} \qquad \hat{\sigma}_i^2 = \frac{1}{|D_i|} \sum_{ij \in D_i} (g_{ij} - \hat{\mu})^2 \qquad i = 0, 1, 2$$

Step 2: Determine likelihood quotient

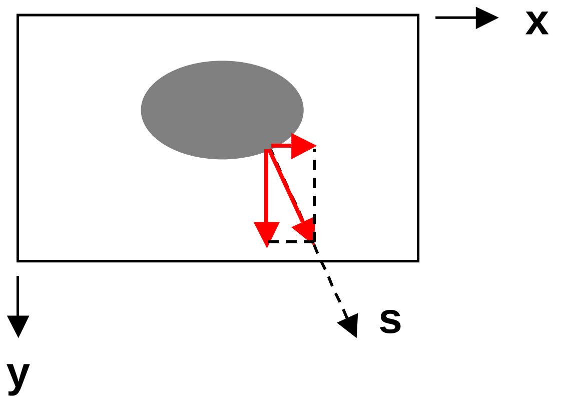
S to be determined empirically

$$\frac{\prod_{g \in D_0} P(g|H_0)}{\prod_{g \in D_1} P(g|H_{12}) \prod_{g \in D_2} P(g|H_{12})} > 1 \Rightarrow$$

Decision rule: $\frac{\hat{\sigma}_1^{|D_1|} \; \hat{\sigma}_2^{|D_2|}}{\hat{\sigma}_0^{|D_0|}} > S$

Greyvalue Discontinuities

Edges may be localized via the 1. and 2. derivative of the greyvalue function.



<u>Gradient</u>

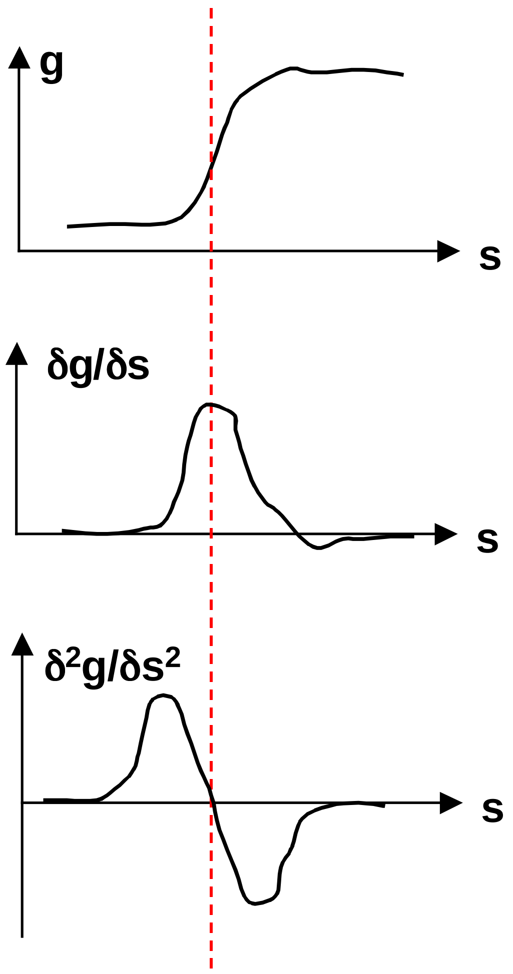
vector in the direction of steepest increase

„g(x, y) = [δg/δx δg/δy]

edges may be located at ...

... high gradient magnitudes ...

... zero crossings of the second derivative



Are Edges Object Boundaries?

Four reasons for edges in images:

1. **Discontinuities of physical object surface properties**

   e.g. colour, material, smoothness ("reflectivity")

2. **Discontinuities of object surface orientation towards observer**

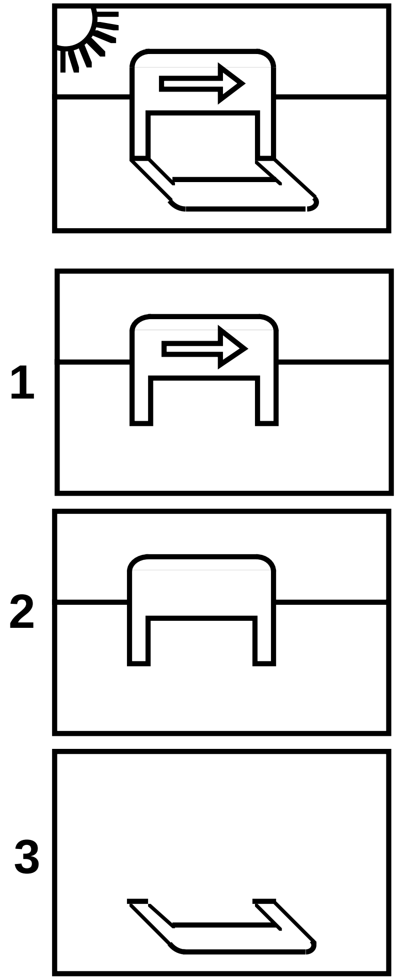
   e.g. strong curvature, 3D-edges, specularities

3. **Discontinuities of illumination**

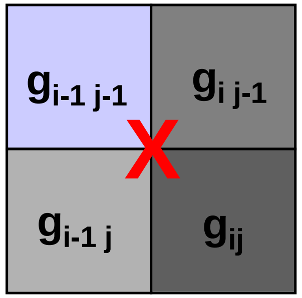
   e.g. shadows, secondary illumination

4. **Discretization effects**

   e.g. binarisation



Robert´s Cross Operator



Computes the gradient based on crosswise greyvalue differences

gradient magnitude

$$|\nabla g_{ij}| = \sqrt{(g_{i\ j-1} - g_{i-1\ j})^2 + (g_{i\ j} - g_{i-1\ j-1})^2}$$

$$\left.\begin{aligned} &\approx |g_{i\ j-1} - g_{i-1\ j}| + |g_{ij} - g_{i-1\ j-1}| \\ &\approx \max\{(g_{i\ j-1} - g_{i-1\ j}),\ (g_{ij} - g_{i-1\ j-1})\} \end{aligned}\right\}\ \text{approximations}$$

gradient direction

$$\tan\gamma = \frac{g_{ij} - g_{i-1\ j-1}}{g_{i\ j-1} - g_{i-1\ j}}$$

direction angle γ in coordinate system rotated by 45^0

Sobel Operator

Popular operator contained in most image processing software packages

$$\begin{array}{|c|c|c|} \hline g_5 & g_6 & g_7 \\ \hline g_4 & g_{ij} & g_0 \\ \hline g_3 & g_2 & g_1 \\ \hline \end{array} \longrightarrow x$$

$$\downarrow y$$

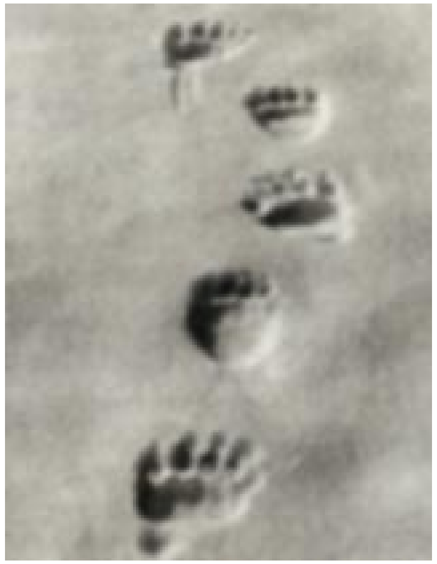
- **Computes gradient components Δx and Δy based on pixels taken from a 3x3 neighbourhood.**
- **Performs simultaneous smoothing**

$$\Delta g_x = (g_1 + 2g_0 + g_7) - (g_3 + 2g_4 + g_5)$$
$$\Delta g_y = (g_1 + 2g_2 + g_3) - (g_7 + 2g_6 + g_5)$$

$$|\nabla g_{ij}| = \sqrt{\Delta g_x^2 + \Delta g_y^2}$$

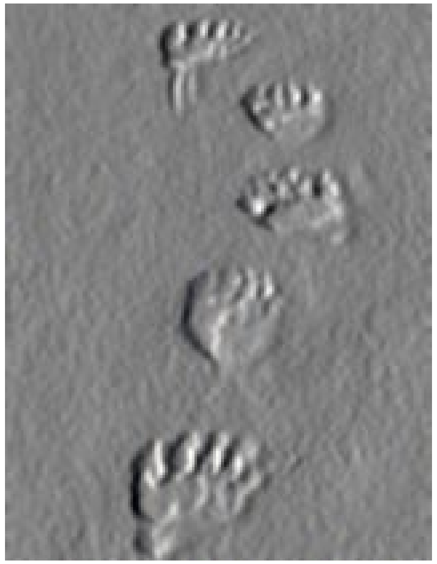
$$\tan \gamma = \frac{\Delta g_y}{\Delta g_x}$$

Example for Sobel Operator



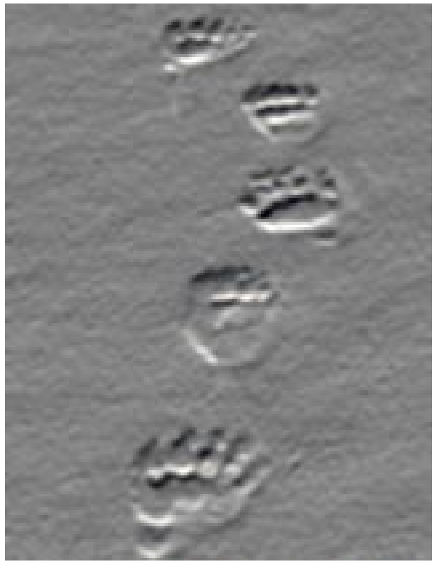
$g(x, y)$
greyvalue image

0 = black
255 = white



Δg_x
x-component of greyvalue gradient

0 = greyvalue 128



Δg_y
y-component of greyvalue gradient

0 = greyvalue 128

Kirsch Operator

g_5	g_6	g_7
g_4	g_{ij}	g_0
g_3	g_2	g_1

- Computes gradient magnitude in 8 directions, selects maximum
- Performs simultaneous smoothing

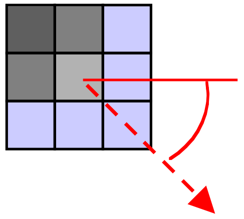
gradient magnitude

$$|\nabla g_{ij}| = \max_{k=0\ldots7 \bmod 8} \{3(g_k + g_{k+1} + g_{k+2} + g_{k+3} + g_{k+4}\} - 5(g_{k+5} + g_{k+6} + g_{k+7})\}$$

gradient direction

$\gamma = (90^0 + k_{max} \cdot 45^0) \bmod 360^0$

<u>Example</u>:



$k_{max} = 7$

$\gamma = (90^0 + 7 \cdot 45^0) \bmod 360^0 = 45^0$

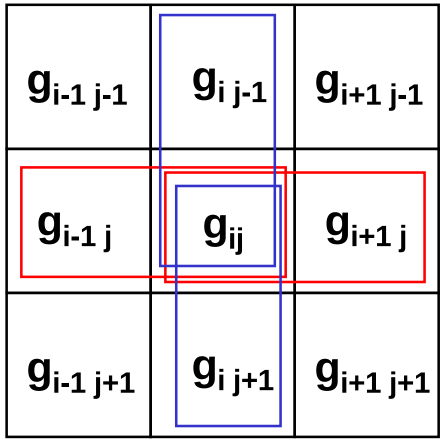
Laplacian Operator

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

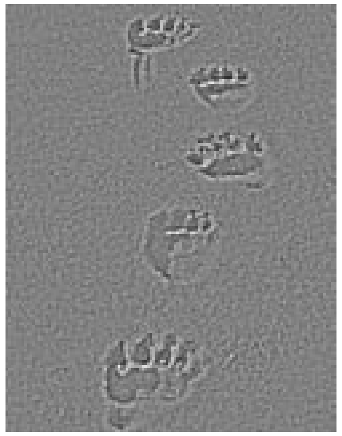
Orientation-independent measure for the strength of the second derivative of a greyvalue function

Discrete approximation by differences of differences of greyvalues:

$$\begin{aligned}\nabla^2 g_{i\,j} = {} & (g_{i+1\,j} - g_{i\,j}) - (g_{i\,j} - g_{i-1\,j}) \\ & + (g_{i\,j+1} - g_{i\,j}) - (g_{i\,j} - g_{i\,j-1}) \\ = {} & g_{i+1\,j} + g_{i-1\,j} + g_{i\,j+1} + g_{i\,j-1} - 4g_{i\,j}\end{aligned}$$



"difference between the greyvalue of a point and the average of its surrounding"



Using the Laplacian operator on raw images will typically give unacceptable results since the 2. derivative amplifies noise. (A single isolated point generates the maximal response.)

Marr-Hildreth Operator

Locates edges at zero crossings of second derivative of smoothed image

Laplacian of Gaussian (LoG): $\nabla^2\left[f(x,y,\sigma)\bullet g(x,y)\right]$

with Gaussian filter $f(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$

Interchanging the order of differentiation and convolution in the LoG gives

$$\nabla^2\left[f(x,y,\sigma)\right]\bullet g(x,y) = h(x,y)\bullet g(x,y)$$

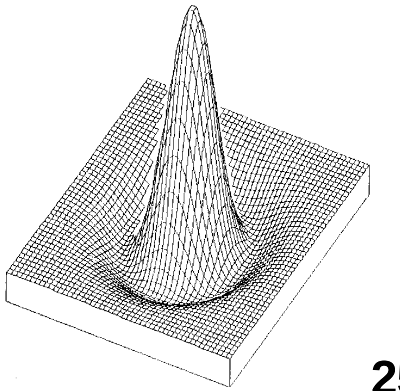
$$\boxed{h(x,y) = c\left(\frac{x^2+y^2-\sigma^2}{\sigma^4}\right)e^{-\frac{x^2+y^2}{2\sigma^2}}}$$

c normalizes the sum of mask elements to zero

discrete 5 x 5 approximation

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

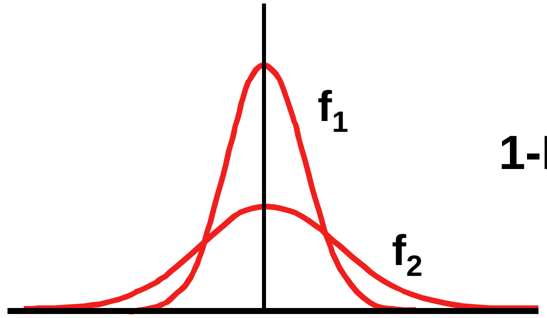
<u>Nickname</u>:
Mexican Hat Operator



Difference of Gaussians (DoG)

The Marr-Hildreth Operator can be approximated by the difference of 2 Gaussians:

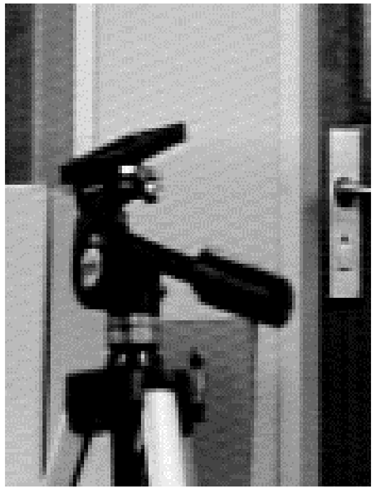
$h(x, y) = f_1(x, y) - f_2(x, y)$



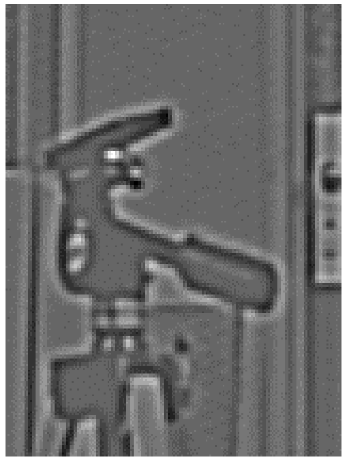
1-D DoG

The best approximation of the Laplacian is for $\sigma_2 \approx 1.6\ \sigma_1$

original image



result of DoG filtering with $\sigma_1 = 1$, $\sigma_2 = 1.6$



Canny Edge Detector (1)

Optimal edge detector for step edges corrupted by white noise.

Optimality criteria:

- **Detection of all important edges and no spurious responses**
- **Minimal distance between location of edge and actual edge**
- **One response per edge only**

Derivation for 1D results in edge detection filter which can be effectively approximated (< 20% error) by the 1rst derivative of a Gaussian smoothing filter.

Generalization to 2D requires estimation of edge orientation:

$$n = \frac{\nabla(f \bullet g)}{\left|\nabla(f \bullet g)\right|}$$

n	normal perpendicular to edge
f	Gauusian smoothing filter
g	greyvalue image

Edge is located at local maximum of g convolved with f in direction n:

$$\frac{\partial^2}{\partial n^2} f \bullet g = 0$$

"non-maximal suppression"

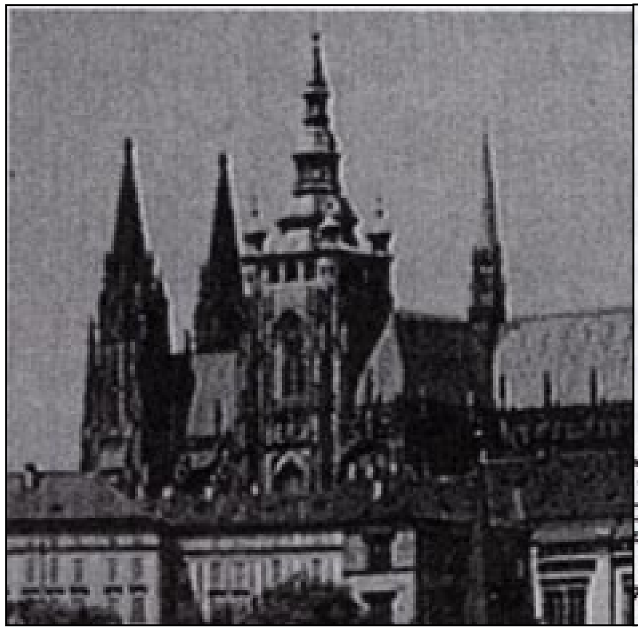
Canny Edge Detector (2)

Algorithm includes

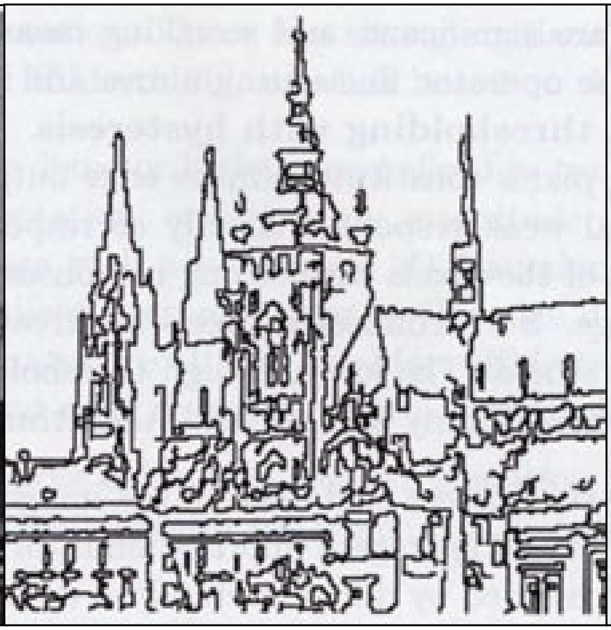
- choice of scale σ
- hysteresis thresholding to avoid streaking (breaking up edges)
- "feature synthesis" by selecting large-scale edges dependent on lower-scale support

1. Convolve image g with Gaussian filter f of scale σ
2. Estimate local edge normal direction n for each point in the image
3. Find edge locations using non-maximal suppression
4. Compute magnitude of edges by $|\nabla(f \bullet g)|$
5. Threshold edges with hysteresis to eliminate spurious edges
6. Pepeat steps (1) through (5) for increasing values of σ
7. Aggregate edges at multiple scales using feature synthesis

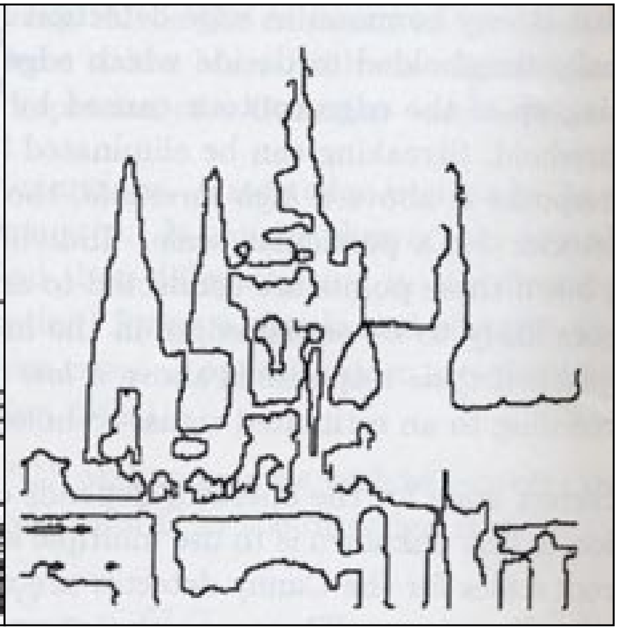
Examples for Canny Edge Detector



original



Canny operator σ = 1.0



Canny operator σ = 2.8
(without feature synthesis)