Image Processing and Imaging
Edge Detection

Dominik Söllinger
Fachbereich AIHI
Universität Salzburg

Wintersemester 2022/23

UNIVERSITÄT SALZBURG

PARIS LODRON UNIVERSITY SALZBURG

Outline

Edge Detection Introduction

- Edges are pixels, in which the image intensity function changes its magnitude
- Crack edges are a virtual edge entity between pixels

The are three different types of gradient operators:

1. Operators approximating the derivative of the image intensity function by differences: $\frac{\partial f}{\partial x} = f(x+1) - f(x)$
2. Operators approximating the zero-crossings of the second derivative of the image intensity function: $\frac{\partial^2 f}{\partial^2 x} = f(x+1) + f(x-1) - 2f(x)$
3. Operators mapping the image intensity function to a parameterised edge model

a b
c

FIGURE 10.2
(a) Image.
(b) Horizontal intensity profile that includes the isolated point indicated by the arrow.
(c) Subsampled profile; the dashes were added for clarity. The numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10-4) for the first derivative and Eq. (10-7) for the second.

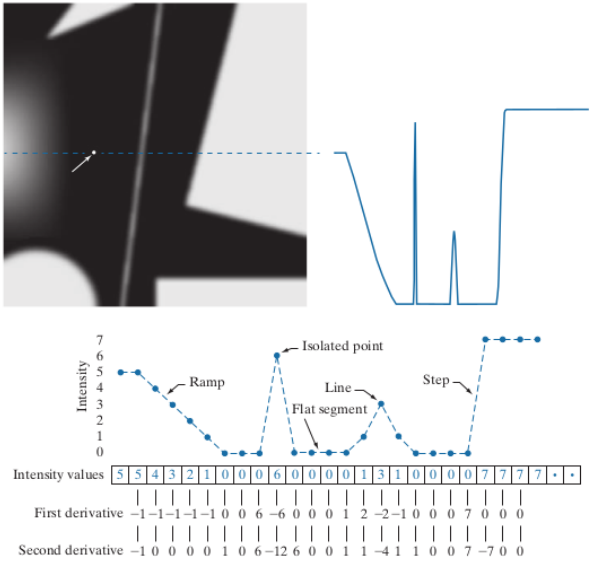


Figure: Numerics: 1. vs. 2. derivative

Outline

The Roberts operator uses a 2×2 neighbourhood with two convolution masks, thereby not considering the orientation of the edges

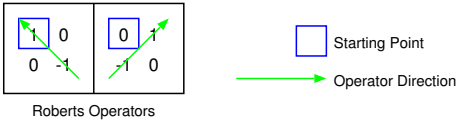


Figure: Roberts Operator

Disadvantage:
- High sensitivity against noise, since only a small number of pixels is used in the approximation
- Not symmetric around the center

A better approximation of the 1st order derivative can be achieved using a **Prewitt** or **Sobel operator**.

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Figure: Prewitt operator

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Figure: Sobel operator

Note: It can be demonstrated that using a 2 in the center location of the Sobel kernels provides image smoothing (better noise supression).

Computing the gradient and edge direction (1)

The tool of choice for finding **edge strength** and **direction** at an arbitary location (x, y) of an image f is the gradient vector ∇f.

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

We can determine the magnitude $M(x, y)$ of the gradient vector and the direction $\alpha(x, y)$ at point (x, y) as follows:

$$M(x, y) = ||\nabla f(x, y)|| = \sqrt{g_x^2(x, y) + g_y^2(x, y)}$$

$$\alpha(x, y) = tan^{-1}\left(\frac{g_x(x,y)}{g_y(x,y)}\right)$$

Computing the gradient and edge direction (2)

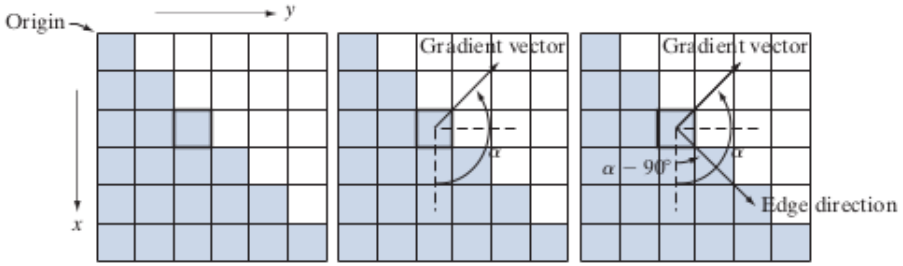
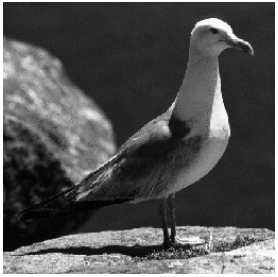


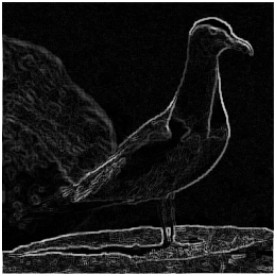
Figure: Using the gradient to determine edge strength and direction at a point. The edge direction is perpendicular to the direction of the gradient vector at the point where the gradient is computed.

Examples

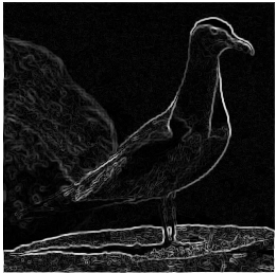
Source



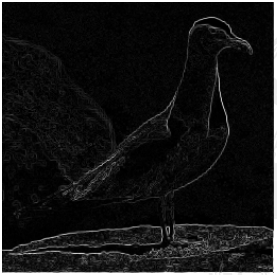
Prewitt operator



Sobel operator



Roberts operator



Compass Operators (1)

Compass Edge Detection is an alternative approach to the differential gradient edge detection. The used kernels are designed to edge magnitude **AND** direction in different directions (a.k.a. compass directions). The are two well-known compass operators — Robinson and Kirsch operator

Approach: Convolve the image with a set (in general 8) convolution kernels. For each pixel, the local edge gradient magnitude is estimated with the maximum response of a kernel at certain pixel location. There are 8 possible edge directions for every location.

Compass Operators (2)

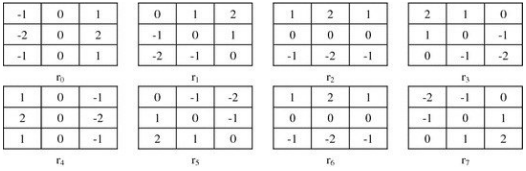


Figure: Robinson operator

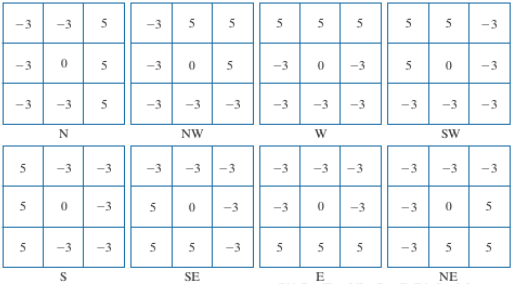


Figure: Kirsch operator

Outline

Disadvantages of 1st order approaches

- Sensitivity to noise
- Sensitivity to the size of the object and scale / type of the edge (step edge vs. ramp edge)
  In many cases it is easier to locate zero-crossings than maxima or minima (see figure about numerics)

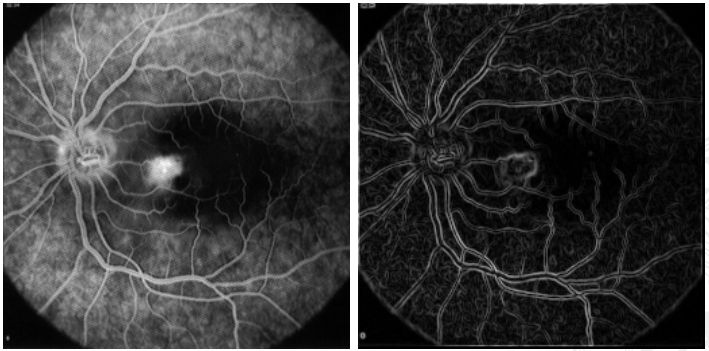


Figure: Sobel operator example

Laplace Operator

Laplace operator is a good choice if:

- If an application only requires the magnitude of the gradient regardless of its orientation
- Laplace operator is rotation invariant
- Recall: it can be computed efficiently in the Fourier domain

$$\nabla^2(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{1}$$

In most cases, it is approximated using 3×3 masks for 4- und 8-neighbourhoods:

$$h_4 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \qquad h_8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Mexican Hat Operator (1)

Mexican Hat Operator (also denoted as Marr-Hildreth Operator) uses a two-dimensional Gauss filter as a smoothing operator:

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

Standard deviation σ is proportional to the size of the neighbourhood the filter is operating on. To compute the second derivative (in order to identify zero-crossings) the Laplace operator is applied to the smoothed image:

$$\bigtriangledown^2 \left(G(x,y,\sigma) * f(x,y) \right) \overset{\text{linear}}{\rightarrow} \left(\bigtriangledown^2 G(x,y,\sigma) \right) * f(x,y)$$

$\bigtriangledown^2 G$ is image independent and thus can be computed in advance.

$$r^2 = x^2 + y^2$$

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

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

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

Replacing r^2 by $x^2 + y^2$

- We obtain the *Laplacian of Gaussian* (LoG)
- Resembles the shape of a sombrero, i.e. Mexican Hat:

$$h(x,y) = \frac{1}{\sigma^2} \left(\frac{x^2+y^2}{\sigma^2} - 1 \right) e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (2)$$

Mexican Hat Operator (2) - Graphical Representation

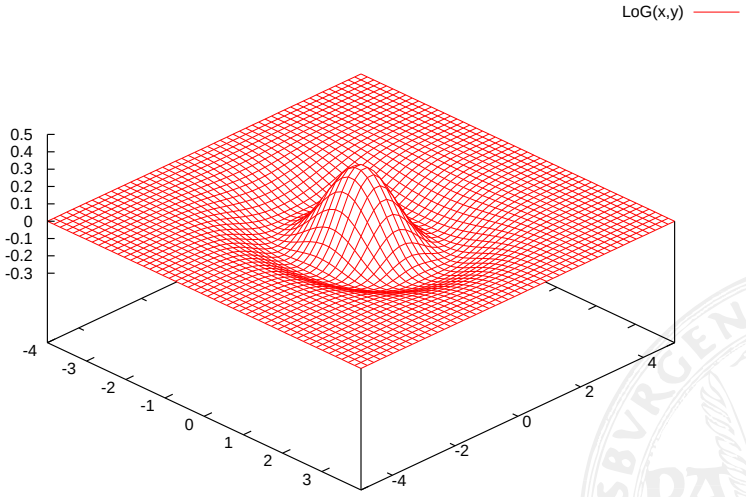


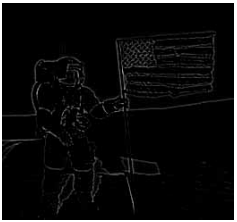
Figure: Mexican Hat

Mexican Hat Operator (3)

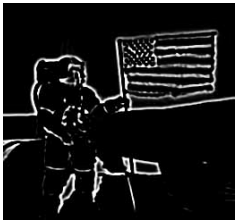
Examples for increasing values of σ:
The larger those values, the more responses we get from coarse edges (large σ in the Gauss mask leads to a strong smoothing effect).

Original

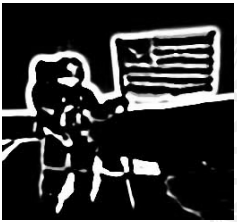
small σ



increasing σ



large σ



Mexican Hat Operator (4)

Remark: $\bigtriangledown^2 G$ can be approximated efficiently by a difference of two Gauss-masks with different *sigma* (*Difference of Gaussian*).

Advantage:
By selecting σ, it is possible to set the scale with respect to which the edge property should be determined.

Disadvantages:

- significant smoothing
- trend to form closed curves (*plate of sphagetti*)

Outline

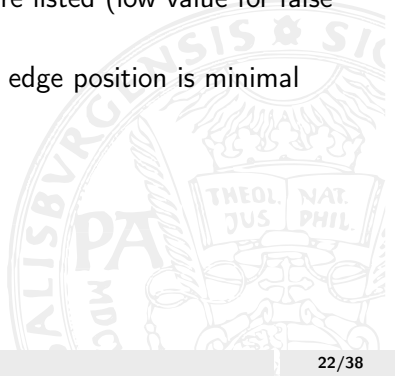
Canny Edge Detector - Introduction

- Has been developed in 1986
- Has been theoretically proven to be optimal with respect to the following criteria for noisy *step edges*:

Detection no important edges are missed and no false edges are listed (low value for false negative and false positive edges)

Localisation distance among actual edge position and computed edge position is minimal

One response multiple responses to one edge are minimised



Canny Edge Detector - Techniques

Canny Edge Detector achieves these results by applying the following techniques:

thresholding with hysteresis improves detection performance in the presence of noise. Edge pixels have to satisfy the following conditions:

- edge-magnitude $>$ *high threshold*
- edge-magnitude $>$ *low threshold* and is connected to an edge pixel $>$ *high threshold*

non-maximal suppression only local maxima of the edge image are processed further

feature synthesis approach :

- all edges with respect to a small scale (i.e. fine detail edges, small σ) are marked
- A predictor for larger σ (see Mexican Hat) is used to predict edge pixels for a larger σ:
  - Smoothing of the small scale edges
  - The predicted values are then compared to the actual ones and only those actual values are kept as additional values, which are significantly larger than the prediced ones
  - This procedure is conducted for several values of σ

Canny Edge Detector - Examples Thresholding

Original

t1=255, t2=1

t1=255, t2=220

t1=128, t2=1

Canny Edge Detector - Algorithm

1. iterate 2 until 5 for increasing values of σ
2. convolve the image with Gaussian with σ
3. compute gradient magnitude and direction
4. find position of the edges (non-maximal suppression)
5. thresholding with hysteresis
6. feature synthesis approach

Figure: Canny Edge Detector: different σ

Outline

Outline

Simple Kernels for line detection (1)

A **line** is a curve that does not bend sharply. In case a line has a width of one or two pixels, a line can be detected using the following kernels:

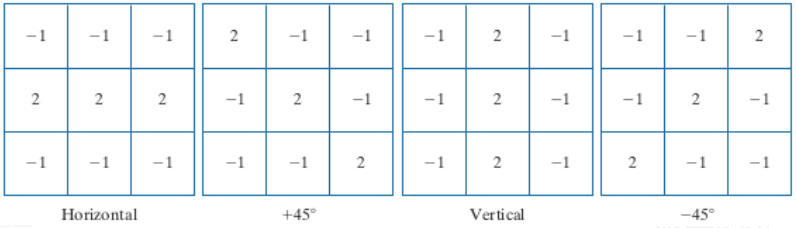
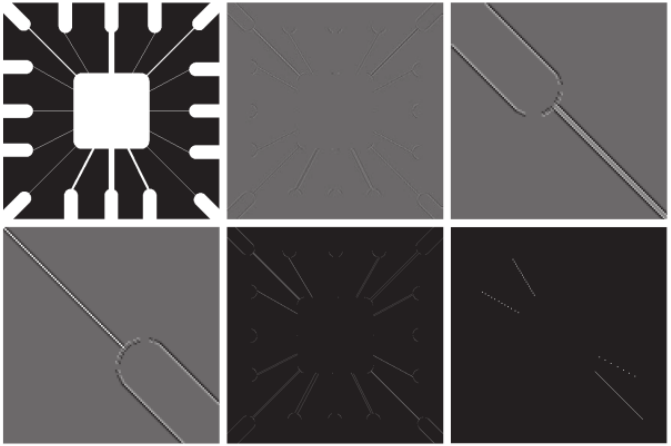


Figure: Line detection kernels

Simple Kernels for line detection (2)



a b c
d e f

FIGURE 10.7 (a) Image of a wire-bond template. (b) Result of processing with the +45° line detector kernel in F
10.6. (c) Zoomed view of the top left region of (b). (d) Zoomed view of the bottom right region of (b). (e) The ima
in (b) with all negative values set to zero. (f) All points (in white) whose values satisfied the condition $g > T$, whe
g is the image in (e) and $T = 254$ (the maximum pixel value in the image minus 1). (The points in (f) were enlarg
to make them easier to see.)

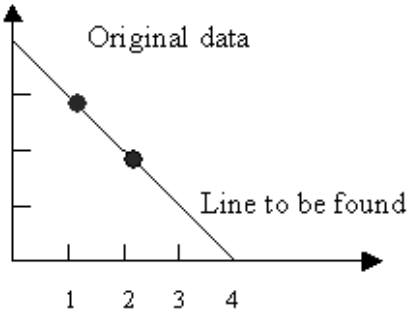
Outline

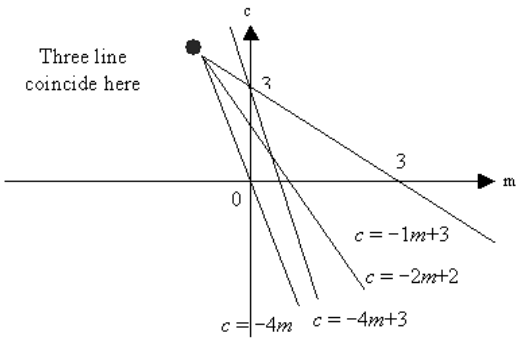
Hough Transformation (1)

- Maps edge pixels to a parametric model of a curve of a certain type
- To be applied, we have to know which curves we are looking for:

Straight line:

- Is defined by two points $A = (x_1, y_1)$ and $B = (x_2, y_2)$
- All lines passing through A are given by $y_1 = mx_1 + c$, for arbitrary m and c (this equation is associated with the parameter space m, c)
- In this representation, all lines through A are given by $c = -x_1 m + y_1$
- Lines through B are given by $c = -x_2 m + y_2$
- Only common point of those two lines in the m, c parameter space is the point representing the line connecting A and B
- Each line the the image is represented by a single point in the m, c parameter space





The points (1,3), (2,2) and (4,0) are situated on the line we are looking for, while (4,3) is not.

y	x	Gives	Transposed
3	1	3 = m 1 + c	c = -1 m + 3
2	2	2 = m 2 + c	c = -2 m + 2
0	4	0 = m 4 + c	c = -4 m
3	4	3 = m 4 + c	c = -4m + 3

Hough Transformation (3)

Based on the line definition as discussed so far, we obtain the following procedure:

1. determine all potential line pixels (edge detection)
2. determine all lines passing through these line pixels
3. transform these lines into (m, c)-parameter space
4. determine point (a, b) in parameter space, which is the result of the Hough transform of the line $y = ax + b$ ((a, b) is the only tupel that occurs several times)

Remark: Only a limited number of lines is considered passing through the line pixels. Collecting the parameters of all admissible lines results in an *accumulator array*, the elements of which are denoted *accumulator cells*.

For each line pixel, potential lines which point into an admissible direction are determined, the parameters m and c are recorded and the value of the corresponding accumulator cell $A(m, c)$ is incremented. Lines in the image are found by taking local maxima of the accumulator array.

Hough Transformation (4)

Advantage:

- Robust against noise

Advantage and disadvantage:

- Missing line parts are interpolated

Disadvantages:

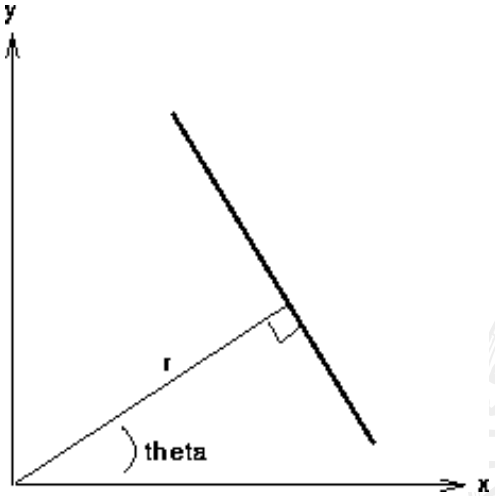
- Brute force approach with high complexity: $\mathcal{O}(A^{m-2})$ with A ... size of the image space and m ... number of parameters (e.g. 2 in case of a straight line)
- High memory consumption. Was improved for several cases in the literature
- Instead of one target line, several similar lines are found. Plateaus instead of peaks in parameter space. These plateaus need to be combined to a single peak

Check out the visualisation:
`http://liquiddandruff.github.io/hough-transform-visualizer/`

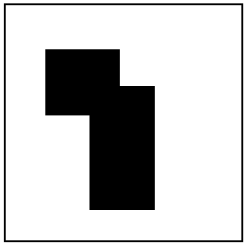
In the general setting the curve equation $f(x, a) = 0$ with a being the vector of curve parameters is used.

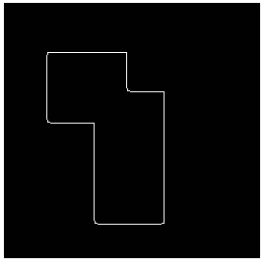
Hough Transformation - Alternate Line Equation

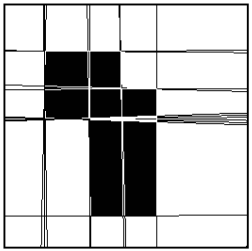
In actual implementations the usage of the common line equation $y = mx + c$ for $m \to \infty$ is sub-optimal in case of large slope. The line equation $r = x\cos(\theta) + y\sin(\theta)$ is better in such cases.

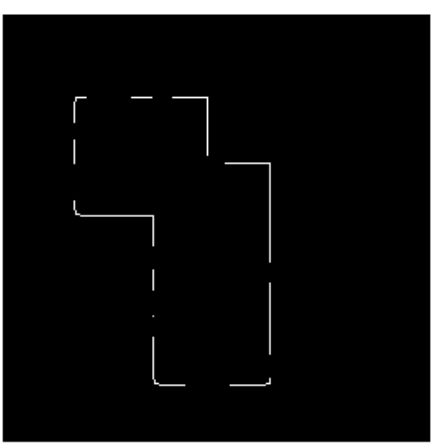


Hough Transformation - Examples 1









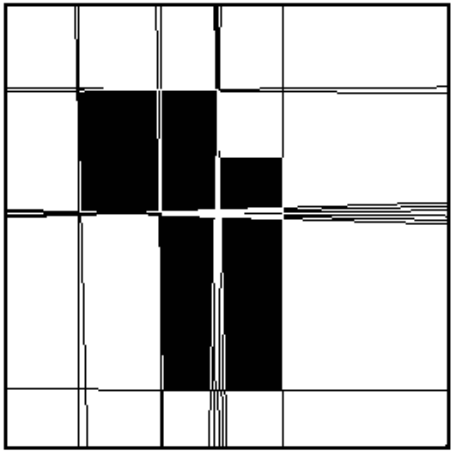


Figure: Example for Hough transform

Hough Transformation - Algorithm

1. Quantisation of the parameter space, dimension n of this space is the number of parameters in a
2. Generate the n-dimensional accumulator array $A(a)$ and set $A(b) = 0$
3. $\forall(x_1, y_1)$ in the appropriately thresholded gradient image increase accumulator cell $A(a)$ in case of $f(x, a) = 0$
4. Local maxima in the accumulator array are the curves we have been looking for

- Circles: $(x_1 - a)^2 + (y_1 - b)^2 = r^2$ three parameters and a corresponding three dimensional accumulator array are required
- By analogy: for more complicated curves, high dimensional accumulator cells are used
- Due to the high complexity, often local variants of the Hough transform are used