Digital Image Processing
Lecture 9

- Edge detection
  - The Marr-Hildreth edge detector
  - Canny's edge detector
- Edge linking
  - using gradient information (Simple case.)
  - using polygonial fit
- Hough transform
  - Detection of lines
  - Detection of circles
- Fourier Descriptors
- Find rotation using eigenvectors
- Gonzales & Woods:
  - Chapter 10, pp. 711-742, Chapter 11, pp. 835-839,861-868

Maria Magnusson, CVL, Institutionen för Systemteknik, Tekniska Högskolan, Linköpings Universitet

Edge models.
Edges found in a real CT-image.

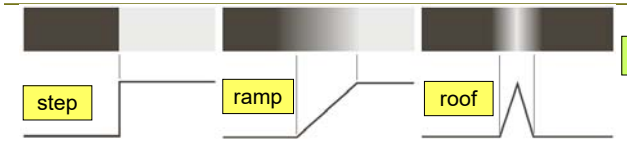


Fig. 10.8

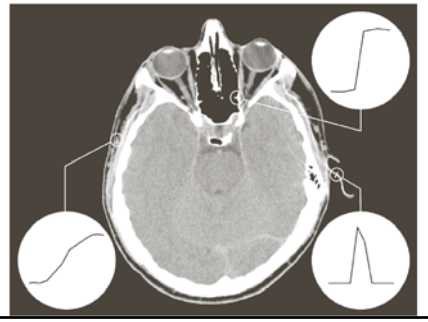


Fig. 10.9

Noise increase after derivative

Some low-pass filtering (smoothing) is often a good idea when computing derivatives!

⟶

Use e.g. the Sobel filters.

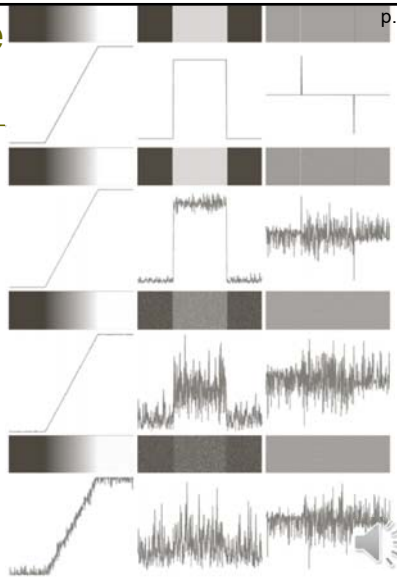


Fig. 10.11

Col 1: original	Col 2: 1:st der.	Col 3: 2:nd der.

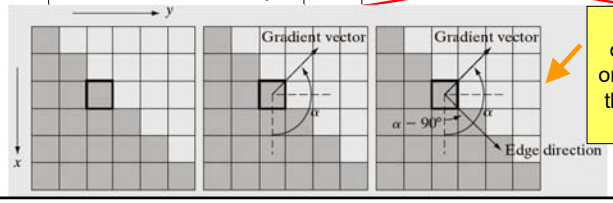
The image gradient vector

To get perfectly correct M and α, g_x and g_y must be **rotation invariant**

$$\nabla g \equiv \text{grad}(g) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} * g \\ \frac{\partial}{\partial y} * g \end{bmatrix} \quad (10\text{-}16)$$

$$M(x,y) = \text{mag}(\nabla g) = \sqrt{g_x^2 + g_y^2} \quad (10\text{-}17)$$

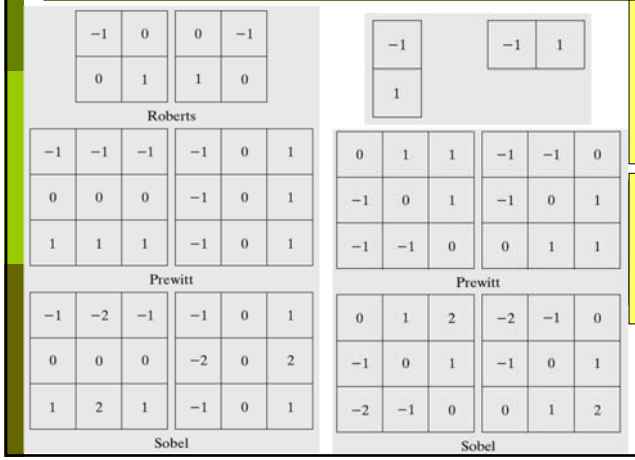
$$\alpha(x,y) = \arg(g_x, g_y) \quad \alpha(x,y) = \tan^{-1}(g_y / g_x) \quad (10\text{-}18)$$



The edge direction is orthogonal to the gradient direction

Fig. 10.42

Various gradient operators. Some you already know. Some are diagonal.



Do you remember what constitutes a good edge detector?

Fig. 10.13

Fig. 10.14

To get perfectly correct M and α, g_x and g_y must be rotation invariant

Rotational symmetric functions.

Not in G&W! Also lecture 4

$\mathcal{F}_2$

$h_x(x,y) = \frac{\partial}{\partial x} h_0(r)$

$H_x(u,v) = j2\pi u H_0(\rho)$

The derivative corresponds to $j2\pi u$. H_0 gives rotational symmetric low-pass filtering.

To check the rotation invariance of a derivative filter h_x, you can take the Fourier transform and divide by $j2\pi u$, it gives H_0, see 2, 3 and 4 slides forward.

Rotation invariance, example

Lecture 4: It is desirable that two derivative filters gx and gy give a magnitude of the gradient $(fx^2+fy^2)^{1/2}$ that is not dependent on rotation. Then the two derivative filters are rotation invariant.

Lab1: comparison sobel and simple

with-out rotation invariant opera-tors

$|\nabla f(x,y)|$ $f(x,y)$ $|\nabla f(x,y)|$

with rotation invariant opera-tors

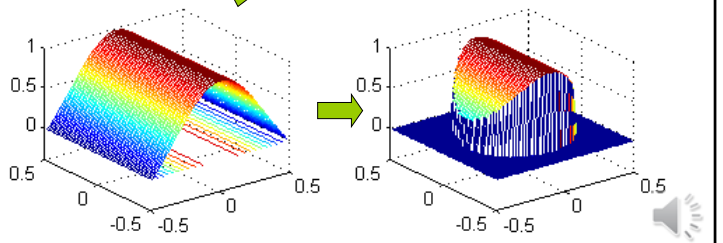
Note: simple is not an official name!

This simple pair is not especially rotation invariant

simplex $= \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{pmatrix}/2$

Take the Fourier transform and divide by $j2\pi u$.

simpley $= \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}/2$

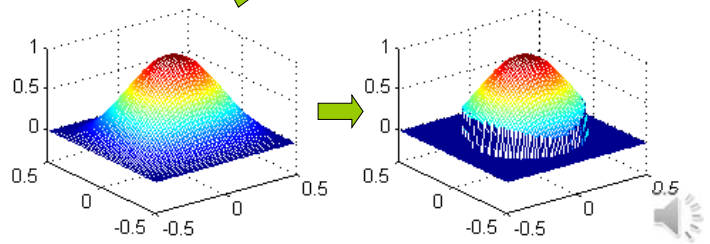


The Sobel pair is rather rotation invariant

$$\text{sobelx} = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} /8$$

Take the Fourier transform and divide by j2πu.

$$\text{sobely} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} /8$$



Sufficient smoothing (low pass filtering)

- The sobel pair already contains some smoothing
- If more smoothing is desired: Take the derivative of a Gaussian function (with the desired amount of smoothing), sample & truncate. This will also give good rotation invariance.
- If more smoothing is desired: Convolve the image with a low-pass filter (i.e. Gaussian with the desired amount of smoothing) before applying the derivative filters (i.e. sobel).

Edge operations (based on sobel)

$$|g_x|,\ |g_y|,\ |g_x| + |g_y| \approx \sqrt{g_x^2 + g_y^2}$$

See also lect.3

Fig. 10.16

Edge operations (based on sobel), orig. image smoothed

Smoothing removes noise and undesired details!

Fig. 10.17

The Laplacian of Gaussian (LoG) filter

It should be -1!

$$G(x,y) = e^{-(x^2+y^2)/2\sigma^2} \quad (10\text{-}27)$$

$$-\nabla^2 G(x,y) = -\frac{\partial^2 G(x,y)}{\partial x^2} - \frac{\partial^2 G(x,y)}{\partial y^2} \quad (10\text{-}28)$$

Filters of arbitrary size can be received by sampling (10.28) and scaling the coefficients so that they sum to 0.
Rule of thumb: Set the size of the discrete LoG filter to ≈6σ

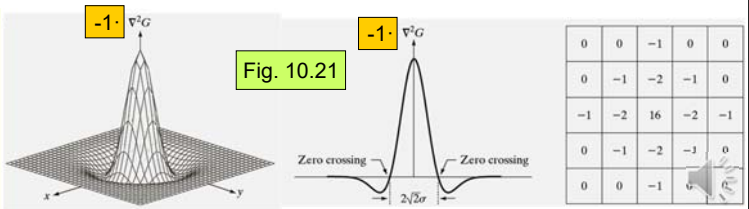


Fig. 10.21

The Difference of Gaussian (DoG) filter

Two DoG filters are an alternative to a LoG filter!

$$\mathrm{DoG}(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-(x^2+y^2)/2\sigma_1^2} - \frac{1}{2\pi\sigma_2^2} e^{-(x^2+y^2)/2\sigma_2^2}$$

(10-32)

The human vision system seems to contain DoG filters with σ1: σ2 =1.75:1

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 - \sigma_2^2} \ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right] \quad (10\text{-}33)$$

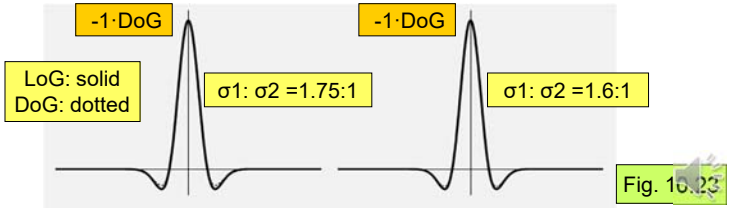


Fig. 10.23

The Marr-Hildreth edge detector

- Var 1a) Apply a LoG or a DoG to the input image.
- Var 1b) Smooth the input image with a Gaussian filter. Apply a simple negative laplacian filter.

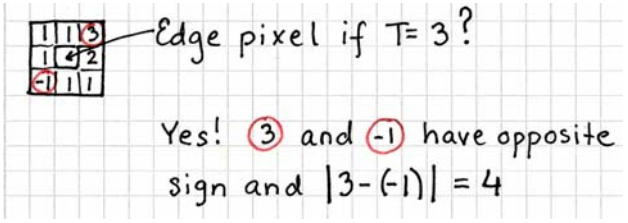
$$f_s(x,y) = [\nabla^2 G(x,y) * f(x,y)] = \nabla^2 [G(x,y) * f(x,y)]$$

(10-30), (10-31)

- 2) Find the zero-crossings. If a) and b) below are true, the pixel is an edge pixel!
  - a) At least two opposite pixels in a 3x3 neighborhood must have different sign.
  - b) Their absolute difference must also exceed a threshold value.

The Marr-Hildreth edge detector Example)

- 2) Find the zero-crossings. If a) and b) below are true, the pixel is an edge pixel!
  - a) At least two opposite pixels in a 3x3 neighborhood must have different sign.
  - b) Their absolute difference must also exceed a threshold value.



The Canny edge detector

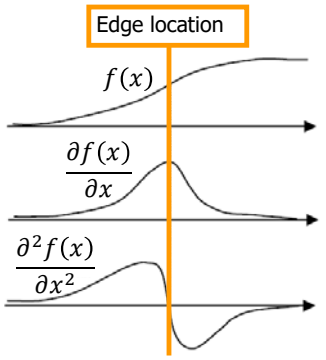
Lab 7

- 1) **Low error rate.** All edges should be found, and there should be no spurious responses.
- 2) **Edge points should be well localized**. The edges must be as close as possible to true edges.
- 3) **Single edge point response.** The detector should not identify multiple edge pixels where only a single edge point exist.

Most advanced edge detector in the book!

The Canny edge detector

- In 1D, Canny's edge detector identify an edge pixel as the position of the maximum of the absolute value of the 1:st derivative.
- In 1D, there is also the zero-crossing of the 2:nd derivative.
- In 2D, Canny's edge detector identify an edge pixel as the position of the maximum of the magnitude of the gradient.



The Canny edge detector

- 1) Smooth the input image with a Gaussian filter to remove noise.

$$G(x,y) = e^{-(x^2+y^2)/(2\sigma)} \quad (10\text{-}35)$$

$$f_s(x,y) = G(x,y) * f(x,y) \quad (10\text{-}36)$$

- 2) Compute the gradient magnitude and angle images.

$$M(x,y) = \sqrt{g_x^2 + g_y^2}, \text{ where } g_x = \partial f_s/\partial x, \; g_y = \partial f_s/\partial y \quad (10\text{-}37)$$

$$\alpha(x,y) = \arg(g_x, g_y) \quad \text{not} \quad \alpha(x,y) = \tan^{-1}(g_y/g_x) \quad (10\text{-}38)$$

- 3) Apply non-maxima suppression to the grad. magn. image.
- 4) Use double thresholding and connectivity analysis to detect and link edges, i.e. thresholding with hysteres. Canny suggest the ratio 1:2 or 1:3 between the thresholds.
- (5) A few edges thicker than 1 pixel may remain. Remove them with morphological thinning.)

Non-maxima suppression

- Let d_1,d_2,d_3 and d_4 denote the 4 basic edge directions.
- In every 3x3-region:
  - Find the direction d_k that is closest to $\alpha(x,y)$.
  - If the value of M(x,y) is less than one of its two neighbors along d_k, suppress edge.

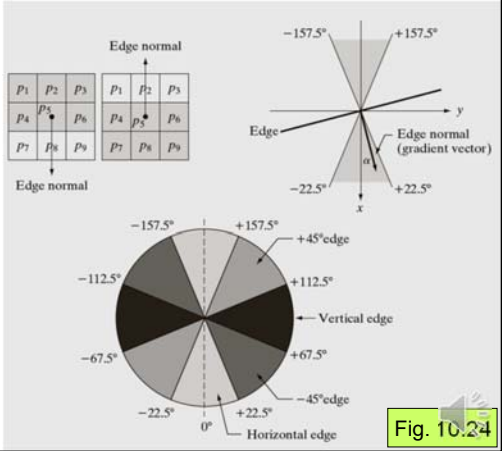
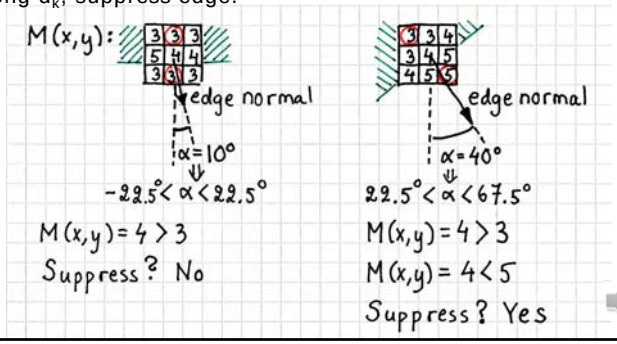


Fig. 10.24

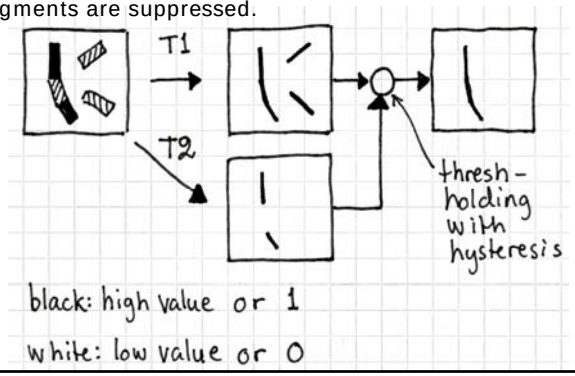
Non-maxima suppression Examples)

- Use the gradient magnitude M(x,y) and edge normal angle α.
- Find the direction d_k that is closest to α.
- If the value of M(x,y) is less than one of its two neighbors along d_k, suppress edge.



Threshold with hysteresis

- Weak line segments with supporting strong line segments are maintained.
- Weak line segments with no supporting strong line segments are suppressed.

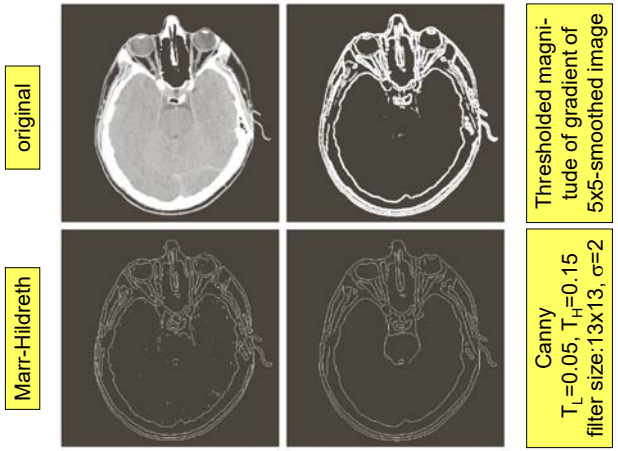


Comparison of 3 edge detection methods

Fig. 10.25

Comparison of 3 edge detection methods

Fig. 10.26



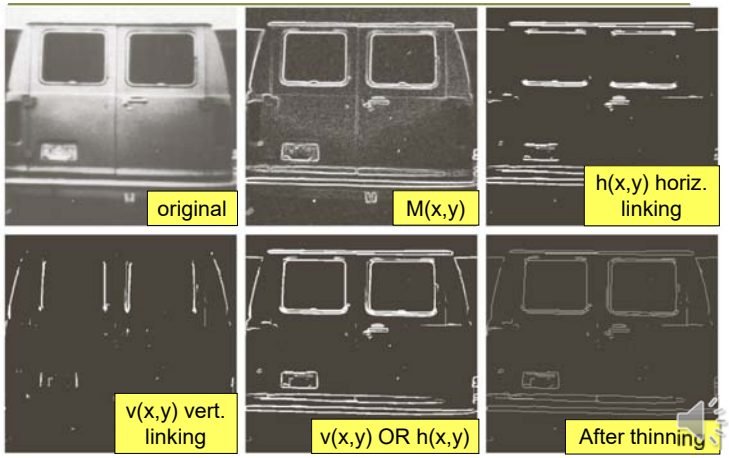
Edge linking using gradient information
(simple variant in one direction)

- 1) Compute the gradient magnitude M(x,y) and angle images $\alpha(x,y)$ of the input image f(x,y).
- 2) Thesholding: g(x,y)=1 if M(x,y)>T_M AND $\alpha(x,y)=A\pm T_A$, otherwise g(x,y)=0.
- 3) Scan the rows of g and fill gaps smaller than a specified length.
- 4) Other directions can be processed similarly by rotating the image before step 2,3 and then rotating it back.

More advanced variants exists!

Fig. 10.27

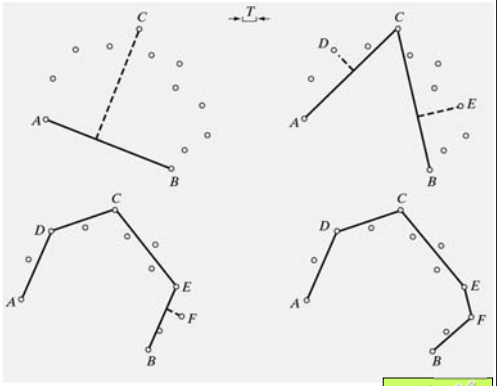
Edge linking using gradient information
(simple case in the horizontal and vertical direction)



Edge linking using polygonial fit

Polygonial approximations can capture the essential shape with a simple description!

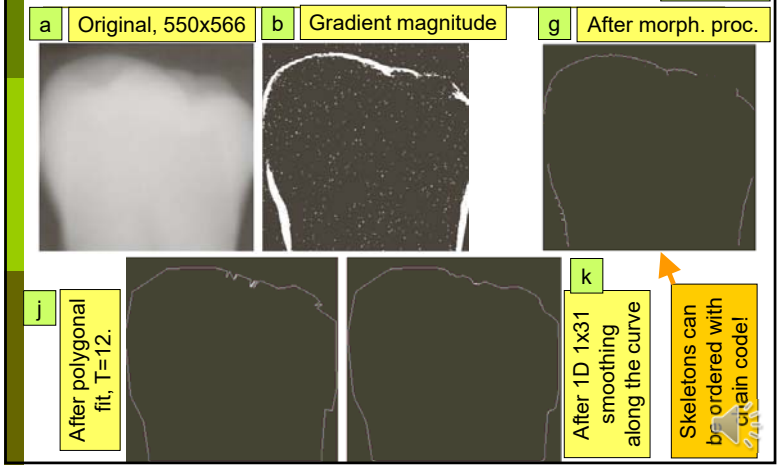
- 1) Suppose that A and B are end points and that all points are ordered.
- 2) A-B forms a line. The distance from this line to C is maximal and >T => C is a new vertex.
- 3) Process line A-C and line C-B as in step 2.
- 4) Continue in a similar way until finished.



3rd ed: Fig. 10.28

Edge linking using polygonial approximation. Applied on tooth.

3rd ed: Fig. 10.30



Global processing with the Hough transform

Lab 7

Calculation of the Hough transform and its "inverse"

$x\cos\theta + y\sin\theta = \rho$ (10-44)

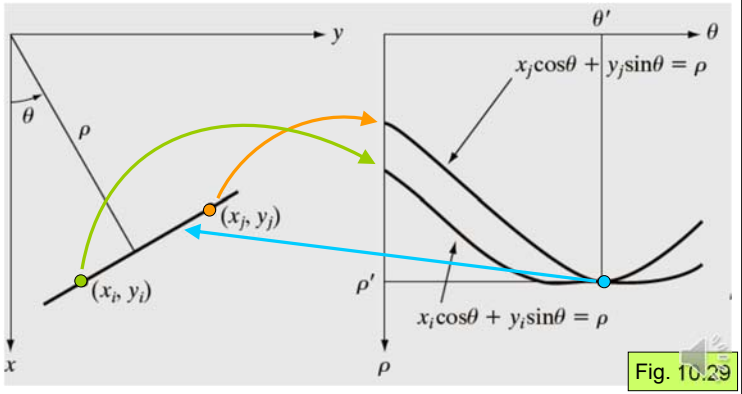


Fig. 10.29

Hough transform and inverse Hough transform

- With the Hough transform with

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

  a point (x,y) in the image becomes a sinusoid in the Hough parameter space.
- With the inverse Hough transform with

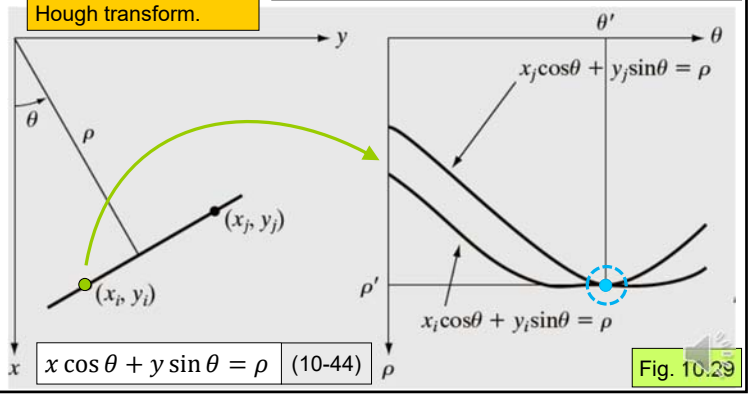
  $x\cos\theta + y\sin\theta = \rho$

  a point (ρ,θ) in the Hough parameter space becomes a line in the image.

Hough transform calculation improvements

Calculation of the gradient in the actual (green) point (xi,yi) gives θ. Insertion in (10-44) gives ρ. Only the point (ρ,θ) (blue) and a small neighborhood (blue dashed circle) in the Hough transform needs to be updated.

Faster. Less disturbing sinusoids in the Hough transform.

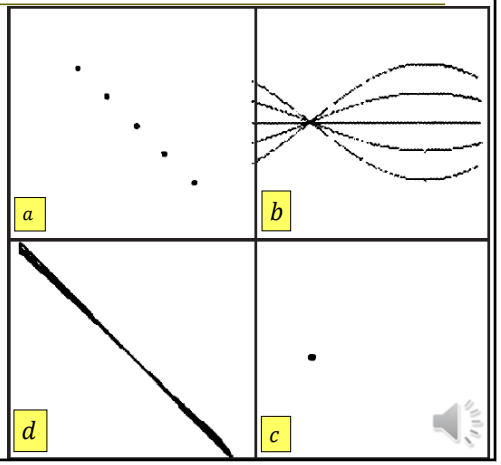


$x\cos\theta + y\sin\theta = \rho$ (10-44)

Fig. 10.29

Line detection with the Hough transform

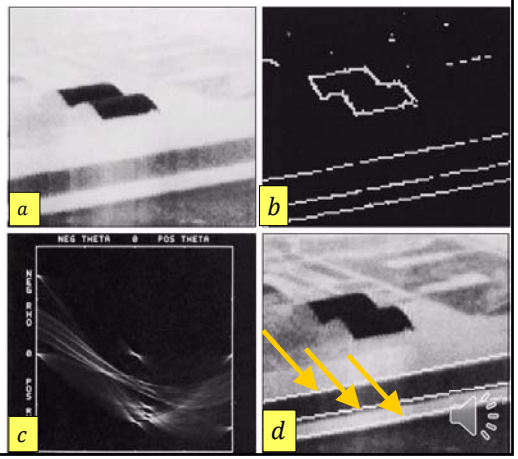
- (a) Take an image with a weak non-connected line (here only 5 points) and possibly perform thresholding.
- Hough transform gives (b)
- Thresholding gives (c)
- Inverse Hough transform gives (d)



Line detection with the Hough transform

From G&W 2:nd edition, Fig.10.21.

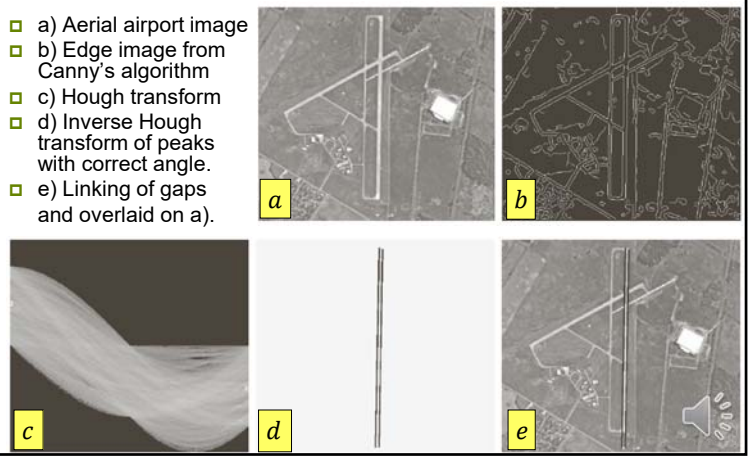
- a) Infrared image of an airport.
- b) Thresholded magnitude of the gradient.
- c) Hough transform
- d) Inverse Hough transform of the 3 strongest peeks in c) combined with linking of gaps < 5 pixels. Then overlaid on a).
- Gap linking is better illustrated in Lab7.



Line detection with the Hough transform

Fig. 10.31

- a) Aerial airport image
- b) Edge image from Canny's algorithm
- c) Hough transform
- d) Inverse Hough transform of peaks with correct angle.
- e) Linking of gaps and overlaid on a).



Hough transform and inverse Hough transform, alternative 2

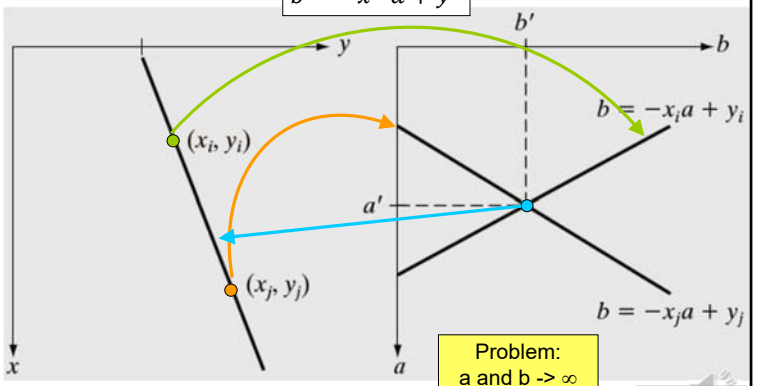
- With the Hough transform with $b = -x \cdot a + y$ a point (x,y) in the image becomes a line in the Hough parameter space.
- With the inverse Hough transform with $y = a \cdot x + b$ a point (a,b) in the Hough parameter space becomes a line in the image.

Hough transform

```
For all (x,y) where f(x,y)!=0
  For all (a,b) where b=-xa+y
    H(a,b):= H(a,b) + f(x,y)
  END
END
```

Calculation of the Hough transform and its inverse, alternative 2

$b = -x \cdot a + y$



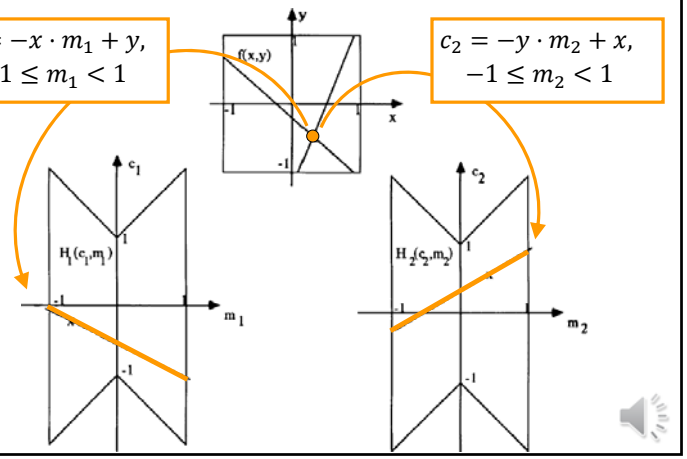
Problem:
a and b -> ∞
for some lines.

Fig. 10.28

Solution: double Hough spaces! Calculation of the Hough transform, alt. 3

$c_1 = -x \cdot m_1 + y,$
$-1 \le m_1 < 1$

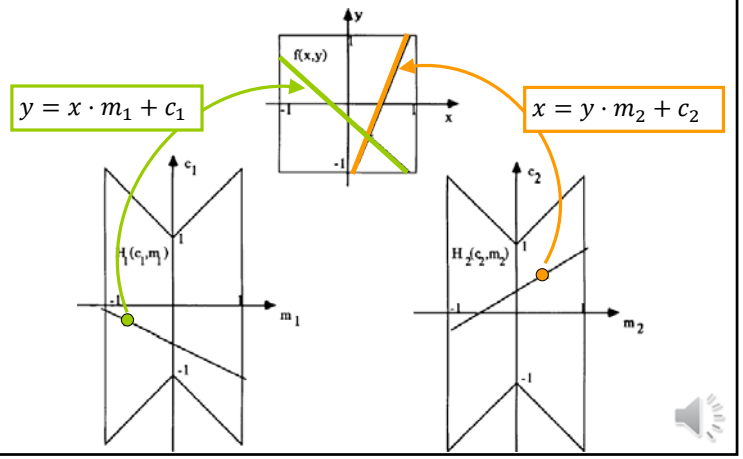
$c_2 = -y \cdot m_2 + x,$
$-1 \le m_2 < 1$



Calculation of the inverse Hough transform, alternative 3

$y = x \cdot m_1 + c_1$

$x = y \cdot m_2 + c_2$



Hough transform and inverse Hough transform of circles

- With the Hough transform $(\alpha - x)^2 + (\beta - y)^2 = r^2$ with a point (α,β) in the image becomes a cone of circles in the Hough parameter space.
- With the inverse Hough transform with $(\alpha - x)^2 + (\beta - y)^2 = r^2$ (10.2-39) a point (x,y) in the Hough parameter space becomes a cone of circles in the image.

Hough transform

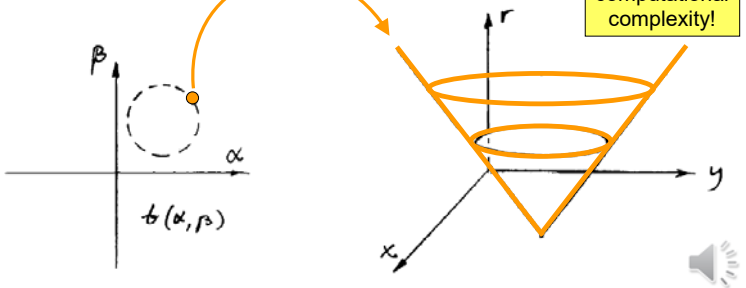
```
For all (α,β) where b(α,β)!=0
  For all (x,y,r) where (α-x)²+(β-y)²=r²
    c(x,y,r):= c(x,y,r) + b(α,β)
  END
END
```


Calculation of the Hough transform for circles

$(\alpha - x)^2 + (\beta - y)^2 = r^2$

3D out-data space!

High computational complexity!



Fourier descriptors

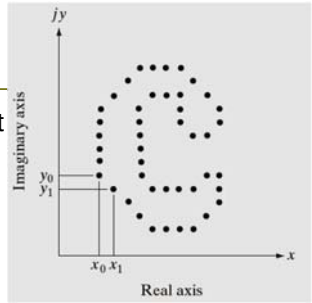


Fig. 11.18

- Let the boundary of an object be represented as the sequence of coordinates
  $s(k) = [x(k), y(k)],$
  $k = 0,1,2,\ldots,K-1$
  $s(k) = x(k) + jy(k)$ (11-7)
- The DFT of s(k) is

$$a(u) = \sum_{k=0}^{K-1} s(k)\, e^{-j2\pi uk/K}, \quad u = 0,1,2,\ldots,K-1 \qquad (11\text{-}8)$$

- The complex coefficients a(u) are called the Fourier descriptors of the boundary.

Fourier descriptors

- The inverse DFT of a(u) restores s(k):

$$s(k) = \frac{1}{K}\sum_{u=0}^{K-1} a(u)\, e^{j2\pi uk/K} \qquad (11\text{-}9)$$

- If only the first P coefficients are used, the result is an approximation to s(k), namely

$$\hat{s}(k) = \frac{1}{P}\sum_{u=0}^{P-1} a(u)\, e^{j2\pi uk/P} \qquad (11\text{-}10)$$

- See Fredrik Larsson's example (PhD from CVL) on next slide.

Keeping different number of Fourier coefficients

$P = 2 \Rightarrow$ 2 3 4 5 6 8 10

12 14 16 32 64 128 256

Properties of Fourier descriptors

Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a_t(u) = a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$a_s(u) = \alpha a(u)$
Starting point	$s_p(k) = s(k - k_0)$	$a_p(u) = a(u)e^{-j2\pi k_0 u/K}$

Tab. 11.1

Fourier descriptor Invariances

- Translation affects the DC-component (zero frequency component) only => remove DC-component
- Scaling affects the magnitude => normalize with the (remaining) signal energy
- Rotation and index-shift affects the phase of each coefficient => keep the phase and correlate, se next slide

Fourier descriptors: Correlation of contour

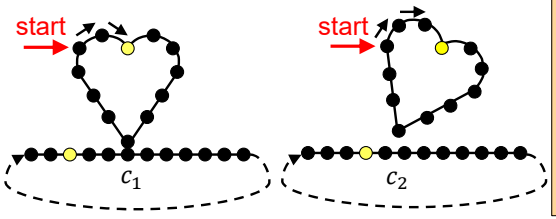
Which variant do you prefer?

Variant 1 $\left(c_1 \star_N c_2\right)(x) = \sum_{\alpha} c_1(\alpha) \cdot c_2(x+\alpha)_N$

Variant 2 $\left(c_1 \star_N c_2\right)(x) = \mathrm{DFT}^{-1}[C_1(-u) \cdot C_2(u)]$

where $C_1(u) = \mathrm{DFT}[c_1(x)]$

and $C_2(u) = \mathrm{DFT}[c_2(x)]$



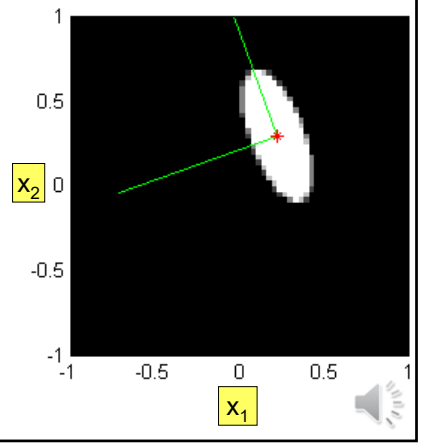
- The correlation max is the matching measure.
- max(abs( )) gives rotation independent matching
- max(real()) gives rotation dependent matching

Application of Fourier descriptors: Traffic sign Recognition

- see Fredrik Larsson's PPT-presentation "FourierDescr" at home page
- Fredrik also mention that the traditional way to use Fourier descriptors is to remove the phase. This gives worse result.

Eigenvectors

- The center of gravity (mean) of the object $\mathbf{m_x}=(m_1,m_2)$ is marked in red.
- The main axes of the object oriented along the eigenvectors $\mathbf{e_1}=(v_{11},v_{21})^T$ and $\mathbf{e_2}=(v_{12},v_{22})^T$ are marked in green.



Find translation and rotation by using eigenvectors, equations

From K vector samples, the mean vector can be approximated with:

$$\mathbf{m_x} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x_k}$$

Compare (11-47)

Similarly, the covariance matrix can be approximated with:

$$\mathbf{C_x} = \frac{1}{K}\sum_{k=1}^{K} (\mathbf{x_k} - \mathbf{m_x})(\mathbf{x_k} - \mathbf{m_x})^T = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x_k}\mathbf{x_k^T} - \mathbf{m_x}\mathbf{m_x^T}$$

Compare (11-48)

Simple equations for the previous image

Set image f(x_1,x_2) to 1 inside object and to 0 outside object

$$fsum = \sum\sum f(x_1,x_2)$$

$\mathbf{m_x}$=(m_1,m_2) is the center of gravity (mean) of the object.

$$m_1 = \frac{\sum\sum x_1 \cdot f(x_1,x_2)}{fsum} \qquad \text{var}(x_1) = \frac{\sum\sum (x_1-m_1)^2 \cdot f(x_1,x_2)}{fsum}$$

$$m_2 = \frac{\sum\sum x_2 \cdot f(x_1,x_2)}{fsum} \qquad \text{var}(x_2) = \frac{\sum\sum (x_2-m_2)^2 \cdot f(x_1,x_2)}{fsum}$$

$$\text{cov}(x_1,x_2) = \text{cov}(x_2,x_1) = \frac{\sum\sum (x_1-m_1)\cdot(x_2-m_2)\cdot f(x_1,x_2)}{fsum}$$

Find translation and rotation by using eigenvectors, equations

$$\mathbf{A^T} = \mathbf{A^{-1}} \quad \text{(11-51)}$$

$$\mathbf{C_x} = \begin{pmatrix} \text{var}(x_1) & \text{cov}(x_1,x_2) \\ \text{cov}(x_1,x_2) & \text{var}(x_2) \end{pmatrix} = \mathbf{A^T C_y A} = \mathbf{V D V^T}$$
$$= \begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix} \cdot \begin{pmatrix} d_{11} & 0 \\ 0 & d_{22} \end{pmatrix} \cdot \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix}$$

Matlab code: [V,D]=eigs(Cx)

$\begin{pmatrix} v_{11} \\ v_{12} \end{pmatrix}$ och $\begin{pmatrix} v_{21} \\ v_{22} \end{pmatrix}$ … are eigenvectors to $\mathbf{C_x}$. They gives the main axes of the object.

Find translation and rotation by using eigenvectors, MATLAB-code

The in-image is in f,
x1Mat is a matrix with x1-coordinates,
x2Mat is a matrix with x2-coordinates.

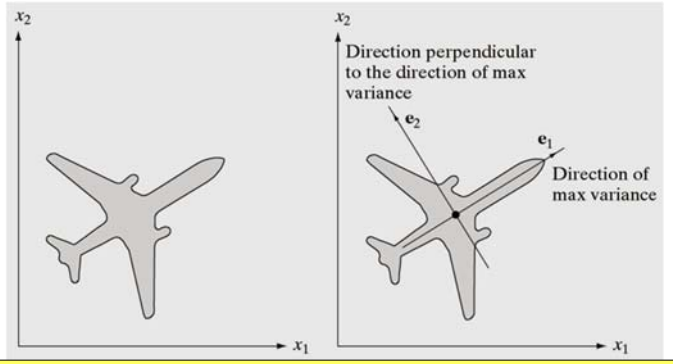
```
fsum = sum(sum(f));
m1 = sum(sum(x1Mat.*f))/fsum;
m2 = sum(sum(x2Mat.*f))/fsum;
Ixx = sum(sum((x1Mat-m1).^2.*f))/fsum;
Iyy = sum(sum((x2Mat-m2).^2.*f))/fsum;
Ixy = sum(sum((x1Mat-m1).*(x2Mat-m2).*f))/fsum;

M = [Ixx Ixy;
     Ixy Iyy]
[V D]=eigs(M)
```

The main axis is in V!

Find rotation by using eigenvectors, G&W example

Fig. 11.43



The points in the region (or its boundary) may be treated as two-dimensional vectors, $\mathbf{x}=(x_1,x_2)^T$, where x_1 and x_2 are the coordinate values along the x_1- and x_2-axis, respectively.