



UNIVERSITÀ DI PARMA

Edges Detection Lines Detection

- What is an edge?
 - Are they important?
- Edge Detection
 - Canny Edge Detector
- Lines
 - Hough Transform
 - Generalized Hough Transform

Edge filtering



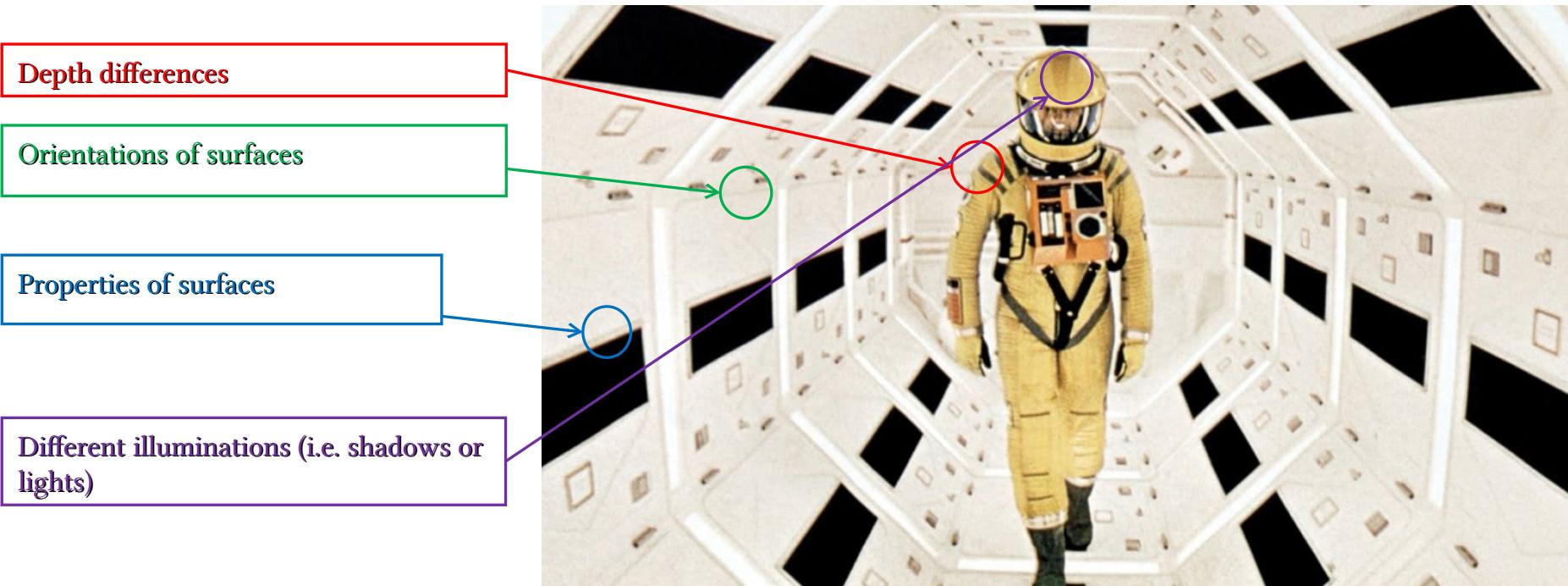
- Q: What is an edge? A: A discontinuity in intensity!



Edge detection



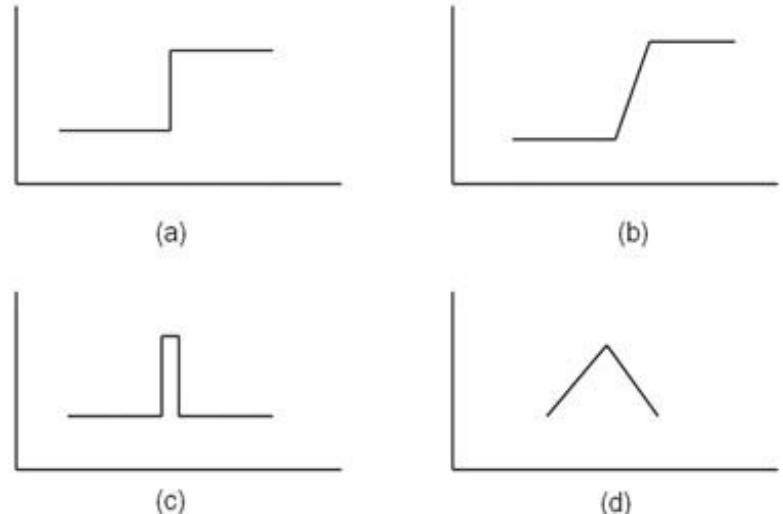
- Q: From where intensity discontinuities (aka edges) come from?



Types of edges



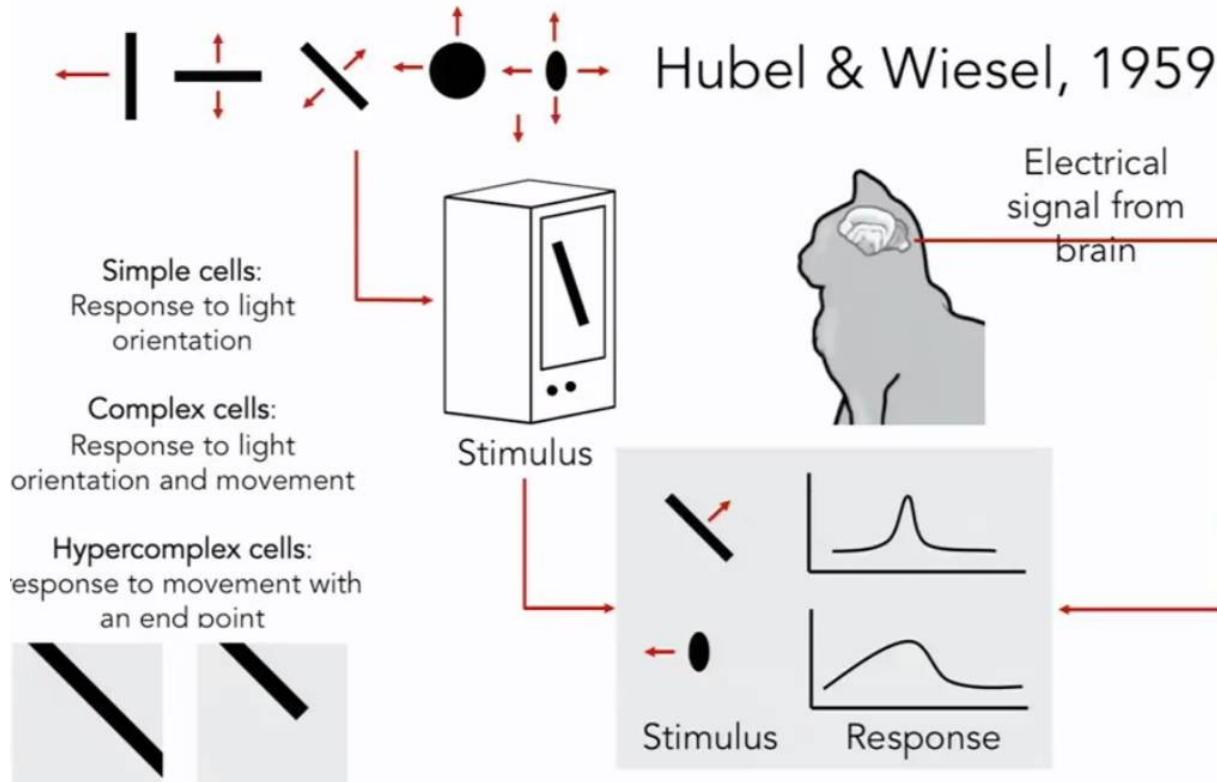
- (a) Step edge: abrupt change of intensity
- (b) Ramp edge: smooth change
- (c) Ridge/line edge: abrupt change followed by a return to starting value
- (d) Roof edge: a ridge edge with a smoother change



Edge Importance



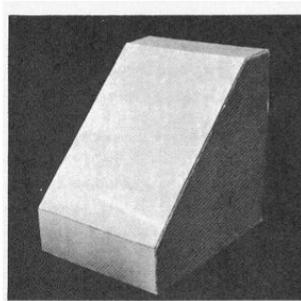
- Why edges are important?



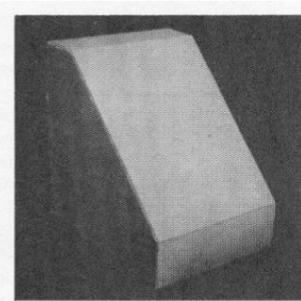
3D from edges



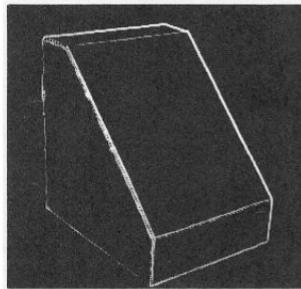
- Why edges are important?



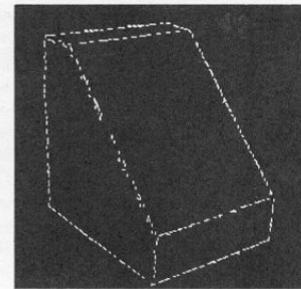
(a) Original picture.



(b) Computer display of picture
(reflected by mistake).



(c) Differentiated picture.



(d) Feature points selected.

Larry Roberts, 1963

Edge Detection

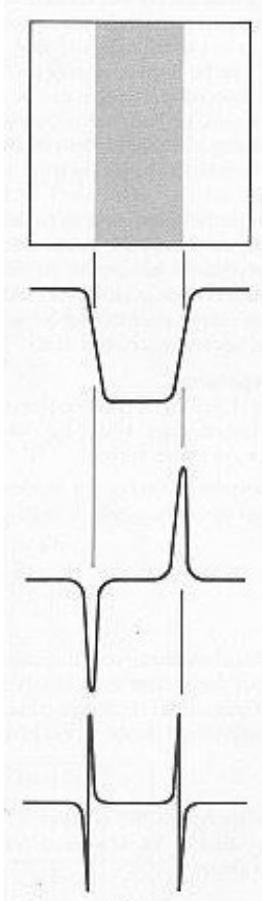


- Edges can help:
 - Extract information
 - Detect objects
 - Detect object shapes

Simple Edge Detection



- Already (partially) discussed
- Small recap:
 - Smoothing → to reduce noise
 - Derivative Horizontal & Vertical filtering → extract gradient



Image

Horizontal Intensity

First derivattive

Second derivattive

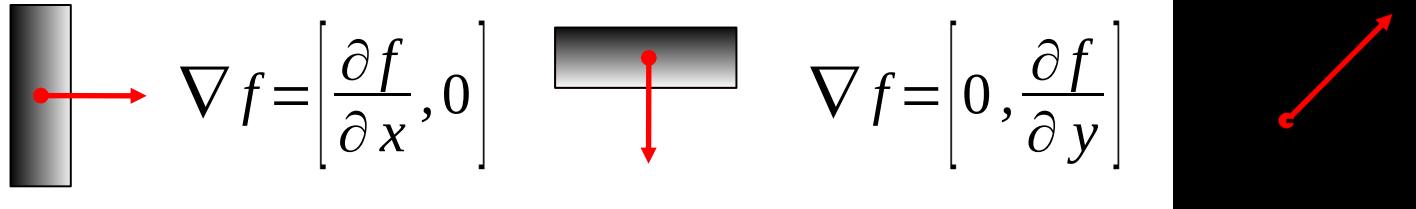
Edges are the maximum/minimum
in first derivative

Edges corresponds to 0 crossings in
second derivative

How we can compute gradient?



- Gradient of image:



- Gradient is a vector:
 - Magnitude/Intensity
 - Direction

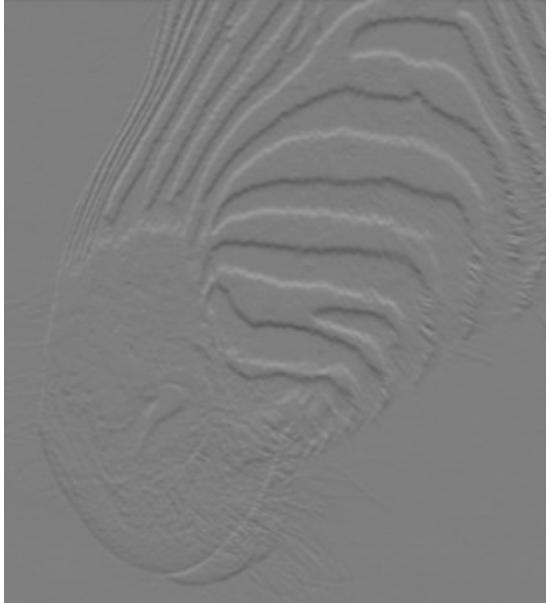
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$
$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$



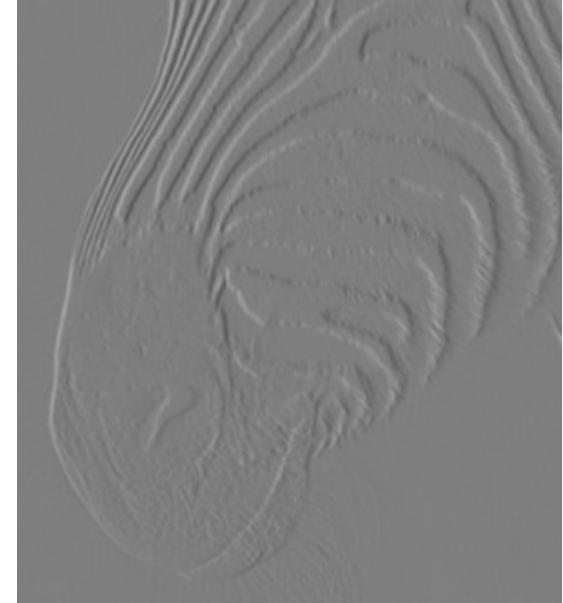
Example



f



$$\frac{\partial f}{\partial y}$$

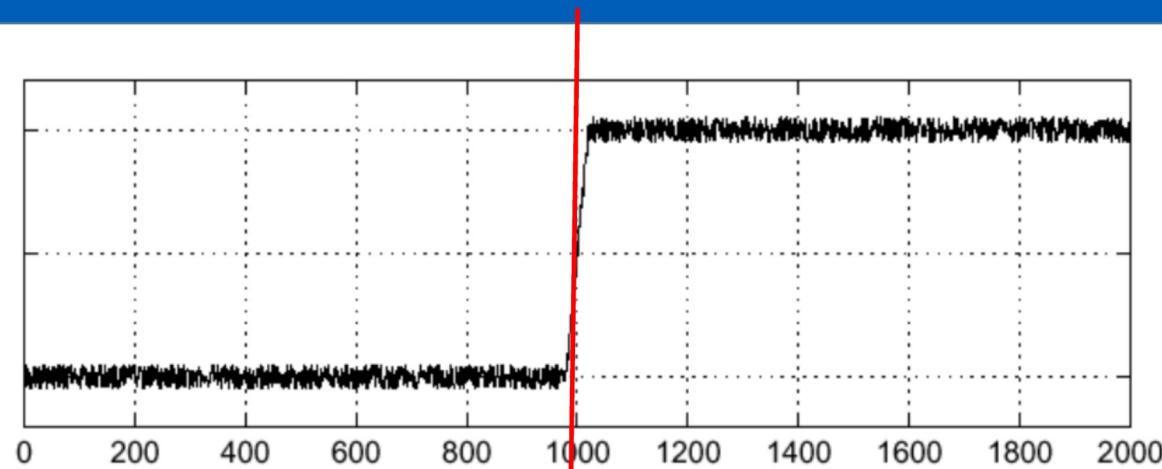


$$\frac{\partial f}{\partial x}$$

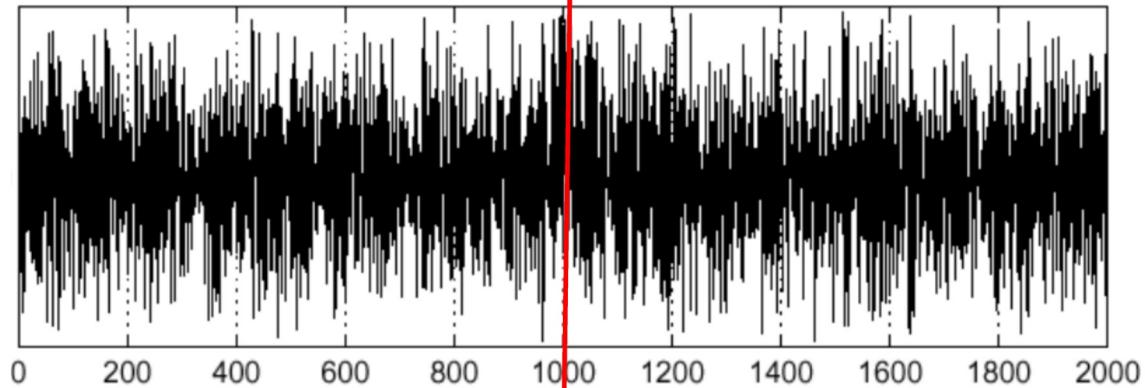
Gradient and Noise



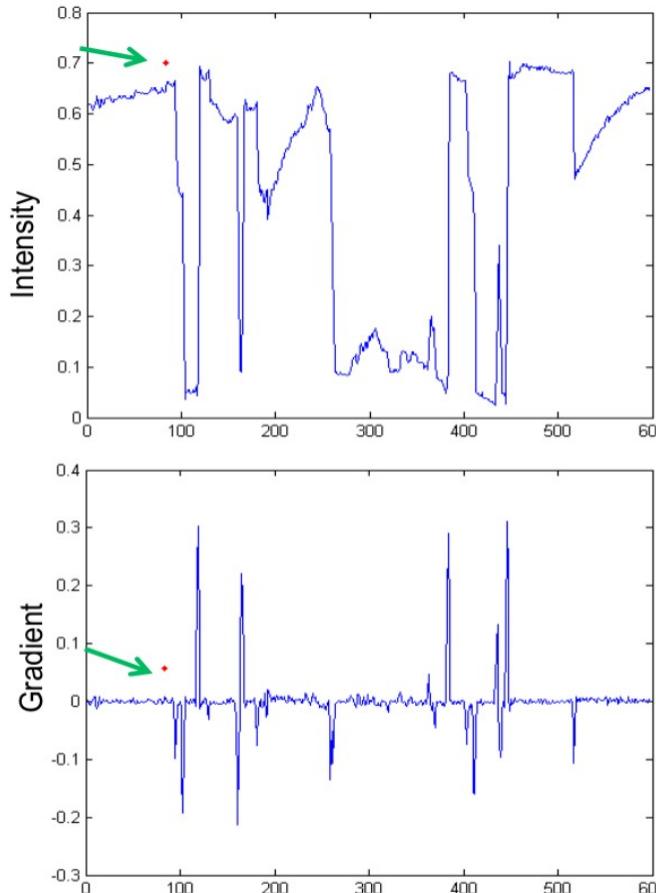
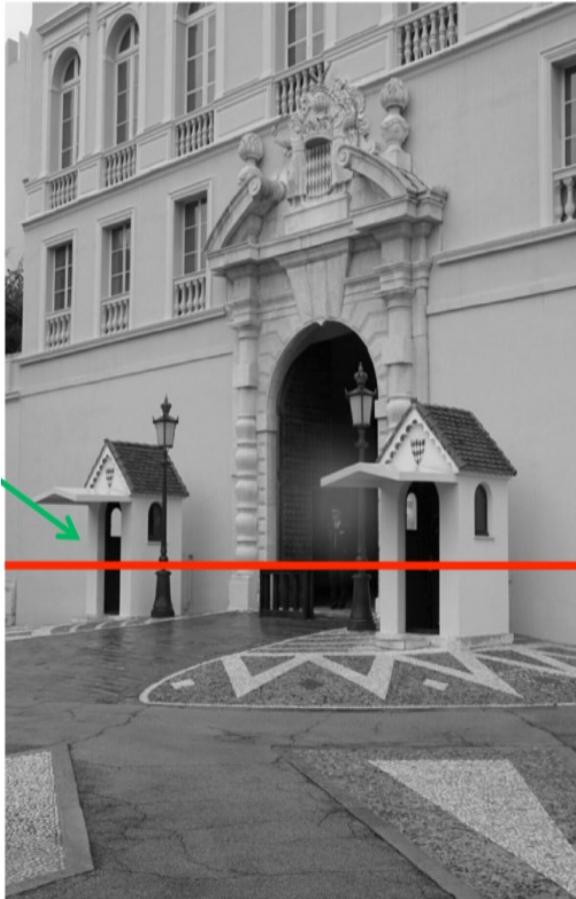
$f(x)$



$\frac{d}{dx}f(x)$



Gradient in real images

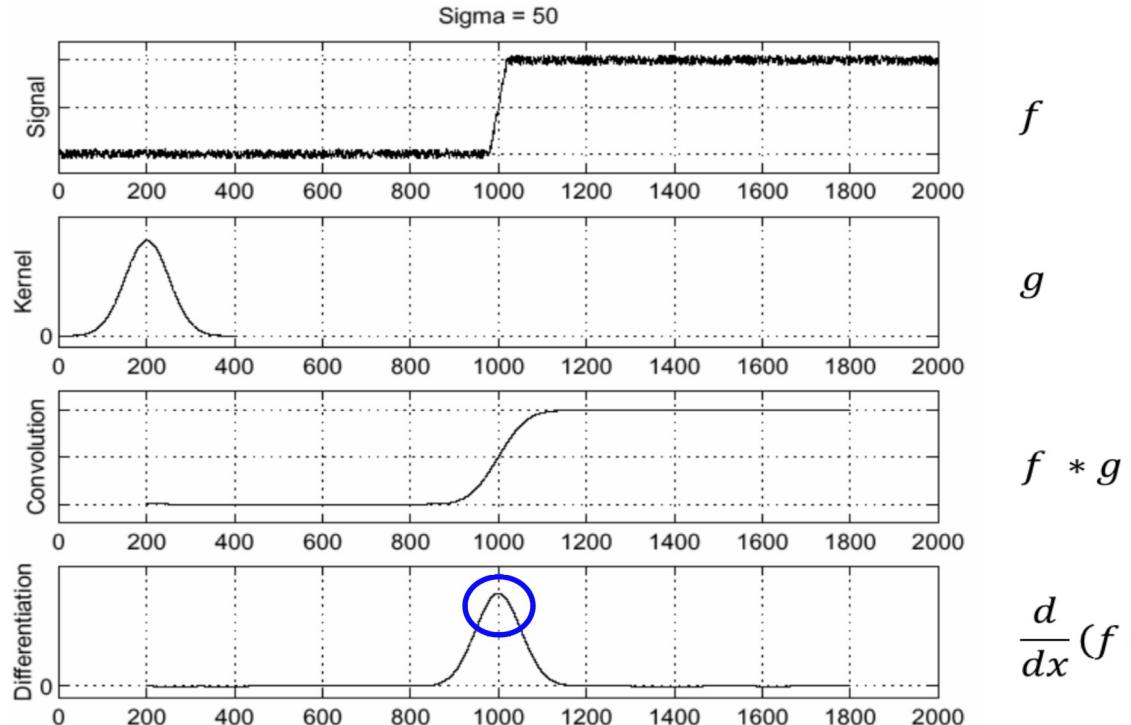


We have large peaks but also some noise

From: Fei Fei Li

- Derivative filter have a great sensitivity to noise
 - Noise → uniform areas are not so uniform
 - Pixels can be very different wrt neighbours
- Smoothing filtering allows to reduce noise effects

Smoothing & Derivative filtering



Edges correspond to peaks in derivative of f and g convolution

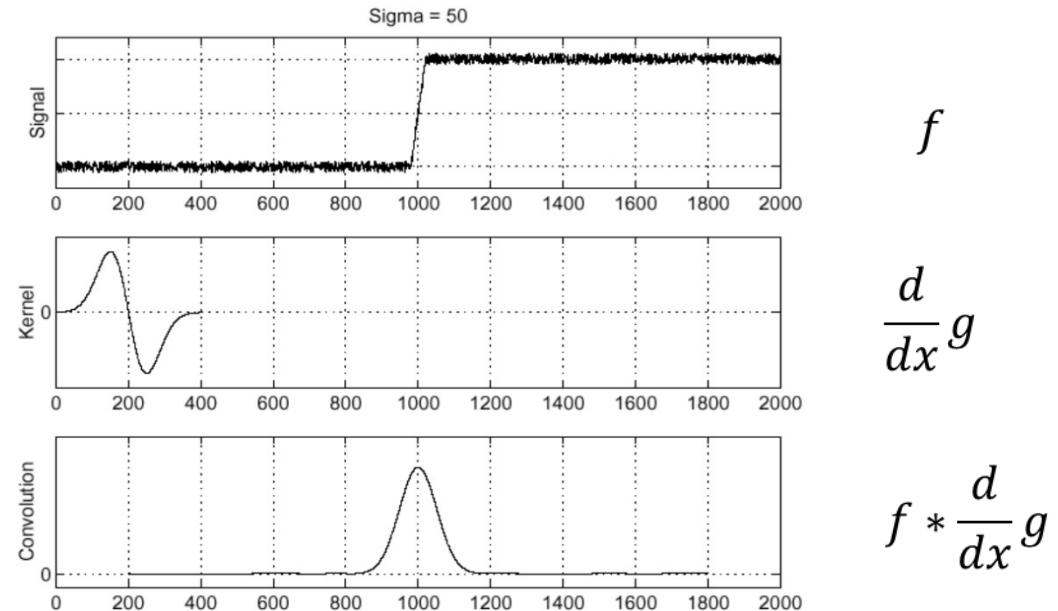
From: S. Seitz

Convolution properties



- Convolution is an associative operation
- We can optimize

$$\frac{\partial}{\partial x} (f * g) = f * \frac{\partial}{\partial x} g$$

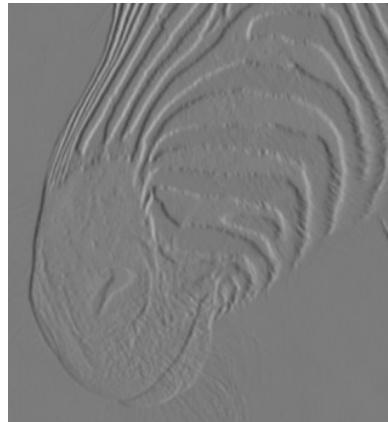




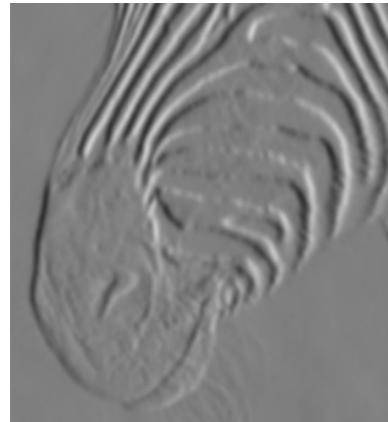
- Between smoothing and localization



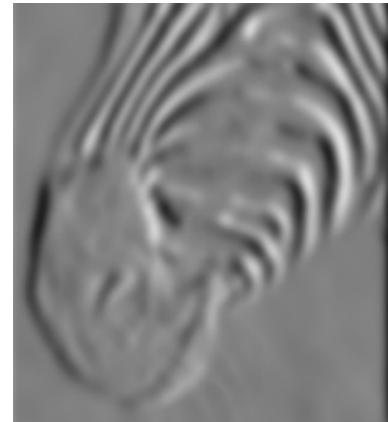
Input image



$\sigma = 1px$



$\sigma = 3px$



$\sigma = 7px$

From: Forsyth & Ponce – *Computer Vision A Modern Approach*

Sobel Edge Detector

- Composition of a derivative operator and an averaging one
- Anyway a little rough for high frequency variations

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

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

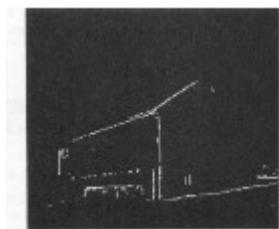
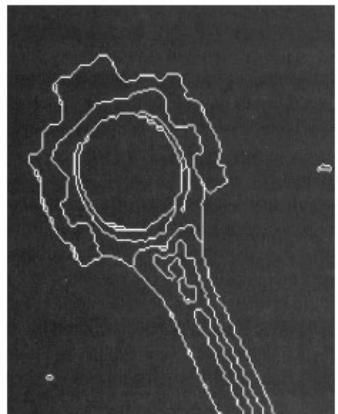
$$\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} * [1 \quad 2 \quad 1]$$

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * [-1 \quad 0 \quad 1]$$

Canny Edge Detector



- Sobel can be a quick&dirty solution
 - Basically it is only a first step
 - Threshold issues
 - Fat edges
- There is a better approach?
- Yes → Canny Edge Detector
 - Gradient
 - Thinning
 - Thresholding



- Widely used
 - It performs very well!
- Criteria
 - Accuracy: reduce false positives (also negatives)
 - Localization: detected edges should match real edges as much as possible
 - Uniqueness: only one detection for each real edge

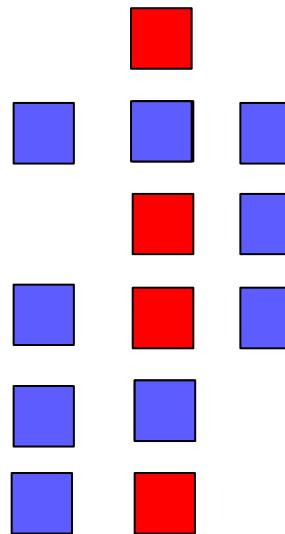
Canny Edge Detection



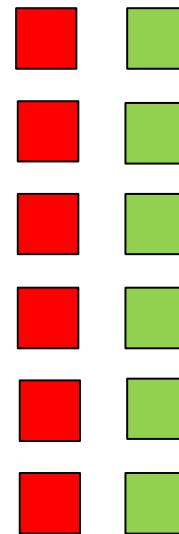
- Examples:



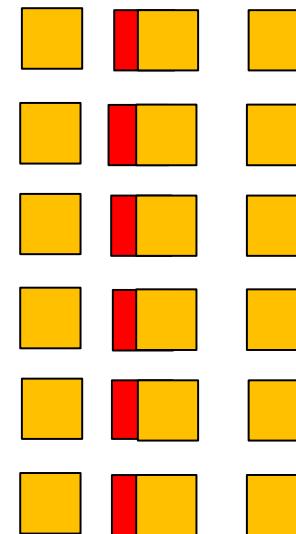
Real edge



Accuracy



Localization



Uniqueness

Canny Edge Detector

- 3 Main steps
 - Gradient detection: compute horizontal & vertical derivatives
 - Magnitude & Direction
 - Thinning: non local maxima suppression
 - 1 pixel wide gradients
 - Thresholding: hysteresis approach
 - Final result

Input



Gradient Detection



- Sobel is ok!



Horizontal Gradient G_x

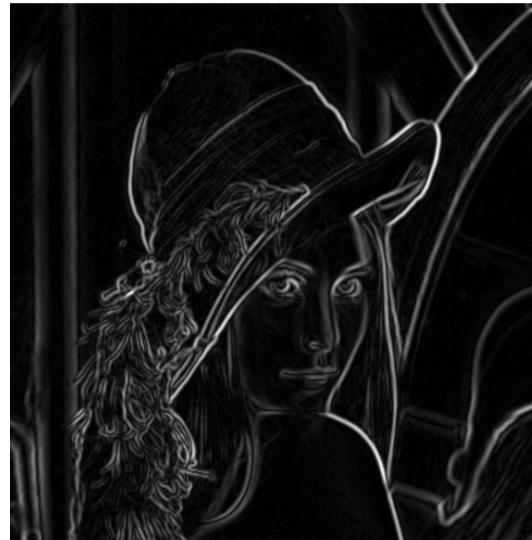


Vertical Gradient G_y

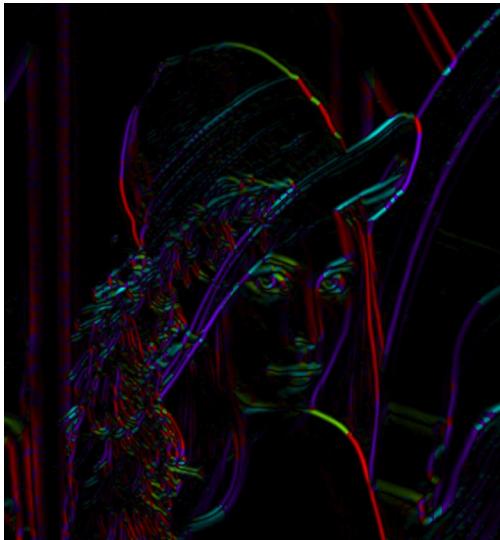
Gradient Detection



- We have 2 results: gradient magnitude and gradient direction



Magnitude



Orientation

$$M = \sqrt{G_x^2 + G_y^2}$$

$$\Theta = \tan^{-1} \left(\frac{G_y}{G_x} \right) = \text{atan2}(G_x, G_y)$$

0	0	0	0	1	1	1	3
0	0	0	1	2	1	3	1
0	0	2	1	2	1	1	0
0	1	3	2	1	1	0	0
0	3	2	1	0	0	1	0
2	3	2	0	0	1	0	1
2	3	2	0	1	0	2	1

- Non Maxima Suppression
 - Only local maxima must survive
 - 3x3 region can be used
 - Orientation and gradient are used to detect local maxima

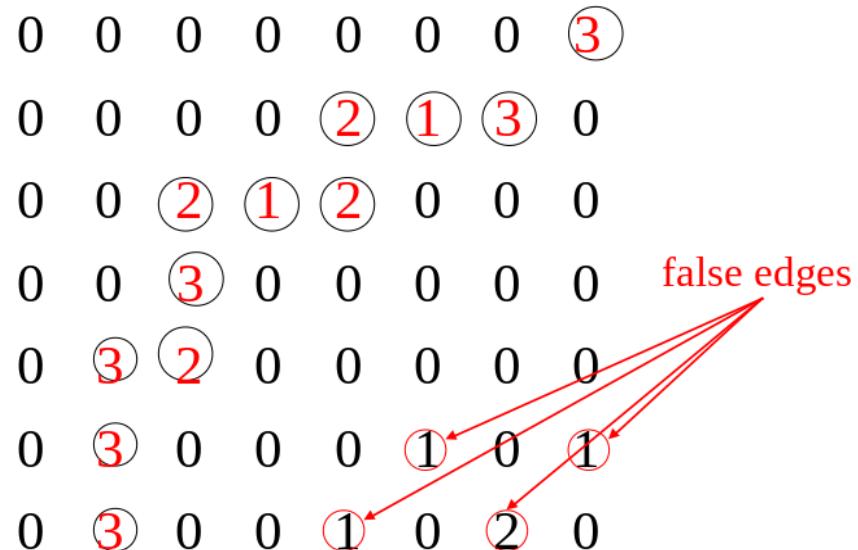
Thinning



0	0	0	0	0	1	1	1	3
0	0	0	0	1	2	1	3	1
0	0	2	1	1	2	1	1	0
0	1	3	2	1	1	1	0	0
0	3	2	1	0	0	1	0	0
2	3	2	0	0	1	0	1	1
2	3	2	0	1	0	2	1	1

- Non Maxima Suppression
 - Only local maxima must survive
 - 3x3 region can be used
 - Orientation and gradient are used to detect local maxima

Thinning



- Non Maxima Suppression
 - Only local maxima must survive
 - 3x3 region can be used
 - Orientation and gradient are used to detect local maxima

Non Maxima Suppression

UNIVERSITÀ
DI PARMA



Thresholding



- Local Maxima does not mean “strong edges”
- One threshold is not enough
- Two thresholds:
 - Under T_{low} → eliminate edge
 - Over T_{high} → keep edge
 - Between T_{low} and T_{high} ?

- Edges above T_{high} are iteratively grown using adjacent edges above T_{low}
 - Similar to a labeling technique
- At the end edges $< T_{high}$ are removed

Hysteresis



Beyond edges



- Edges are not lines... still a low level feature



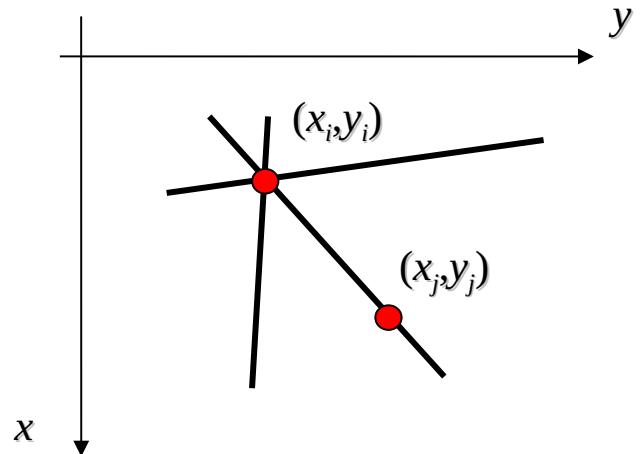
- First approach: determine pixel alignment
 - Expensive!
 - We have to match each pixel to other pixels → straight lines
 - Check whether other pixels are on those lines
 - For n pixels we have $n(n-1)/2 \sim n^2$ potential straight lines
 - $(n-2)[n(n-1)/2] \sim n^3$ matches
- Real approach: Hough Transform

Hough Transform

- The family of straight lines that pass through a given point (x_i, y_i) is defined as:

$$y_i = ax_i + b$$

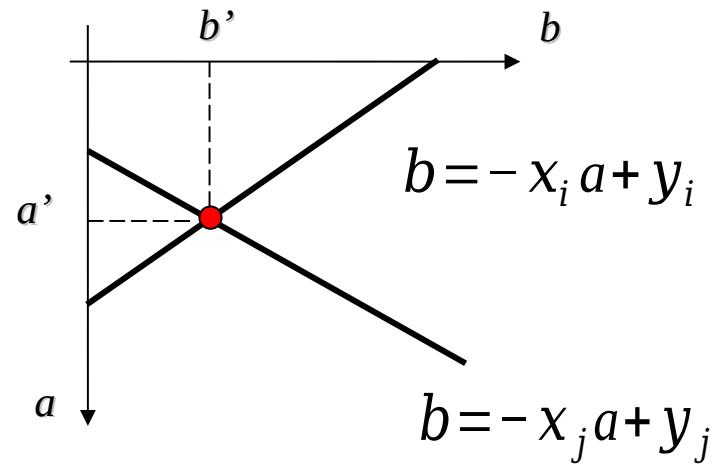
- For each parameter set (a, b) we have a given straight line



Hough Transform



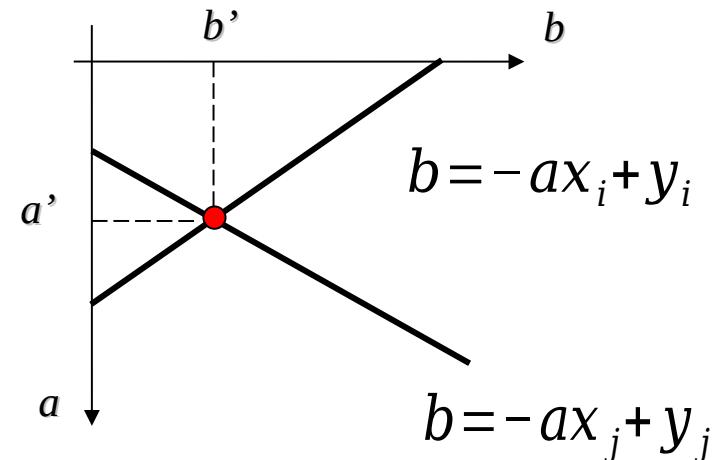
- The previous equation can be written as:
$$b = -x_i a + y_i$$
- That represents parameters values for a given point (x_i, y_i)
 - Still a straight line but in parameter space!
- This can be done also for other points, i.e. given point (x_j, y_j)
- The two straight lines intersect in (a', b')



Hough Transform

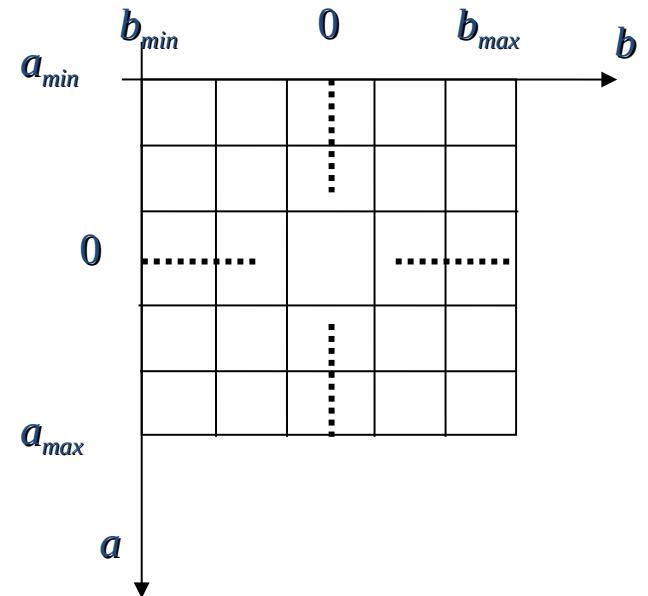


- (a', b') are the parameters that define the straight line that pass through both (x_i, y_i) and (x_j, y_j)
- More generally, all points that lie on the same line in the x,y space generate straight lines that intersect in a specific point in the a,b space
- This is the underlying rationale of the Hough Transform



Hough Transform

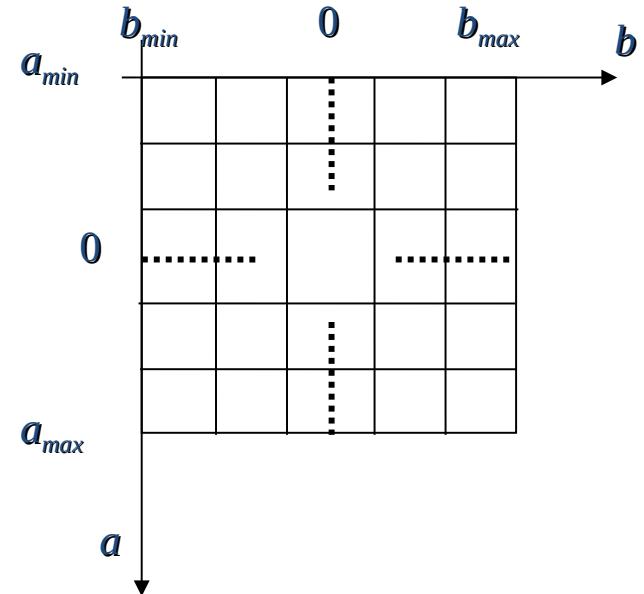
- Main steps:
 - Define a discrete accumulator matrix for the a, b space
 - For each (x_k, y_k) compute all possible b values using
 - $b = -x_k a + y_k$
 - Increase the corresponding a, b cells in the accumulator matrix



Hough Transform



- At the end of the process each accumulation cell gives us information on how much pixels lie on the same straight line
- i.e. if a cell (a_i, b_i) contains the value M this means that in the image we have M points that lie on the $y = a_i x + b_i$ line
- A threshold is generally used to extract straight lines



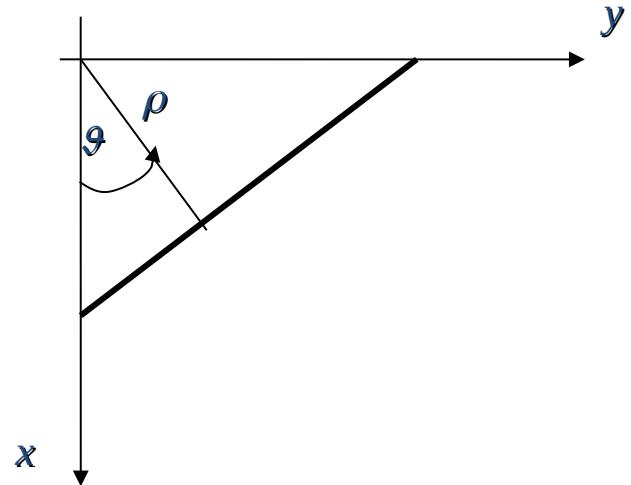
- If the a axis is subdivided in k intervals for each (x, y) point we get k values for b
- For n points we need to compute n^*k values
 - $O(n)$ better than $O(n^3)$
- Open issues:
 - Which value for k?
 - What range for a?
 - Vertical lines?

Hough Transform

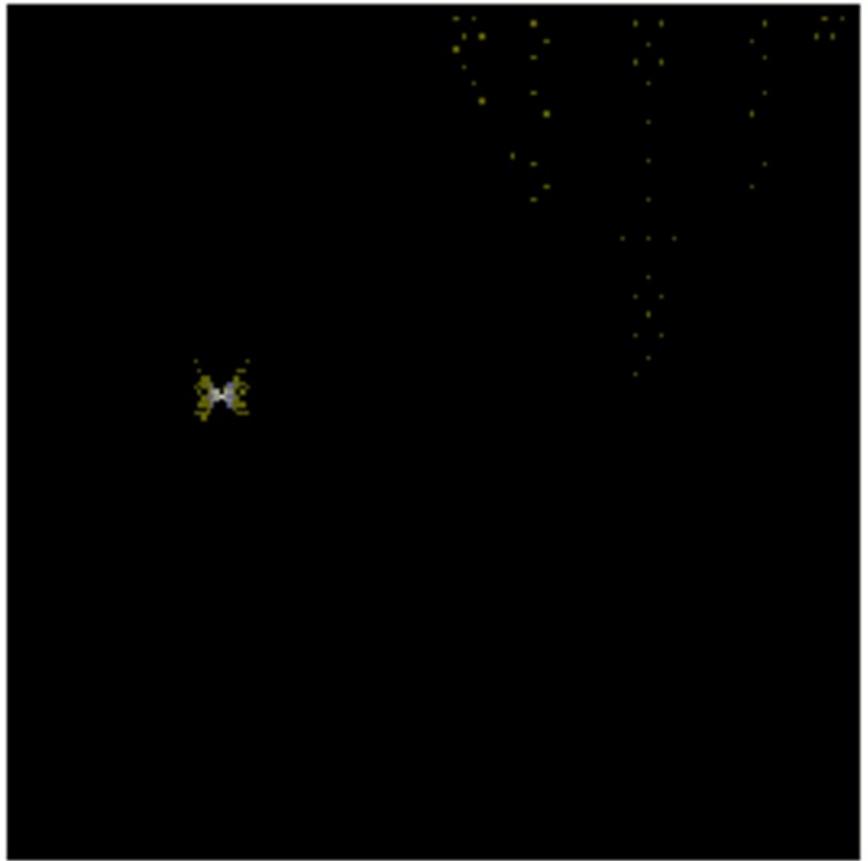
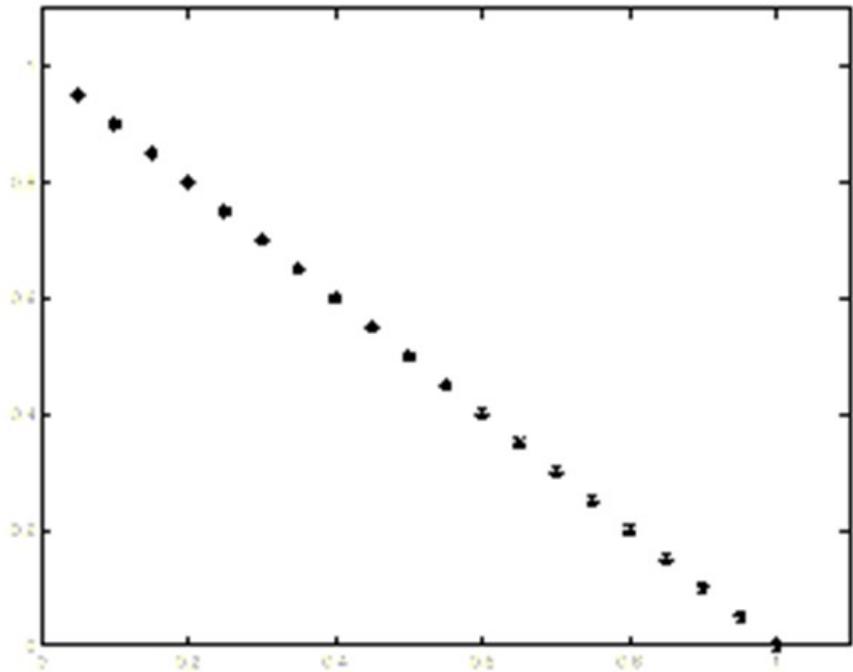
- To solve some issues Euclidean space is used instead of Cartesian one

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

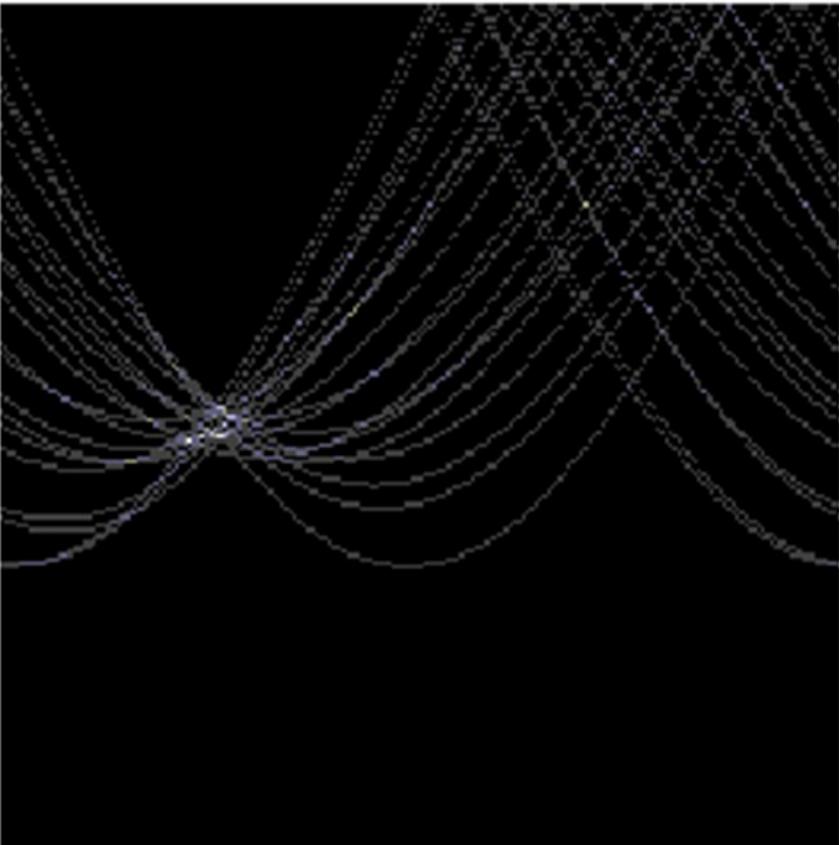
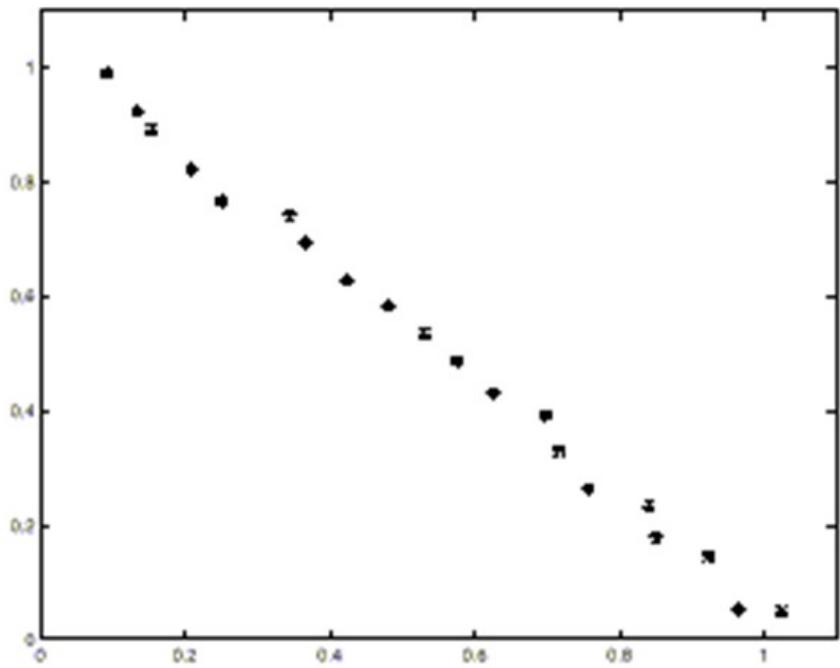
- Both parameters have a limited range
 - θ is an angle
 - ρ is limited by image diagonal
- Similar procedure as before
- In the parameters space we have curves



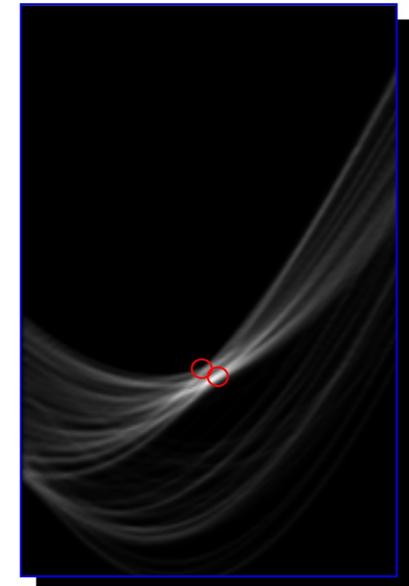
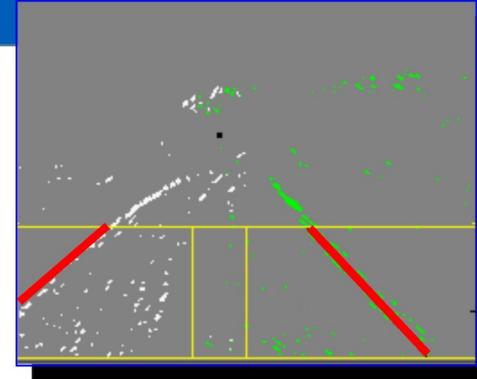
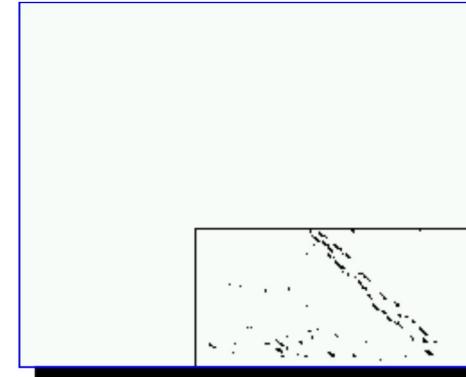
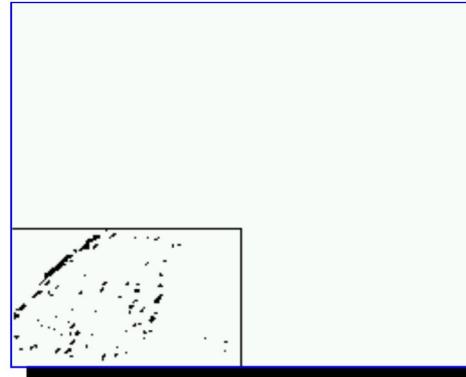
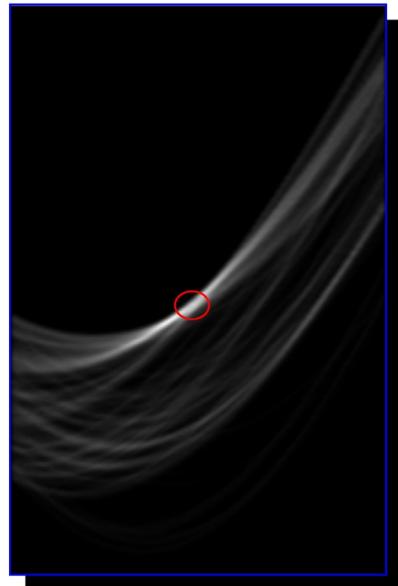
Hough Transform



Hough Transform



Hough Transform



Generalized Hough Transform

- Described approach is for straight lines
- Anyway it can be applied to any analytical curve
- The only differences are the number and ranges of parameters
 - i.e. for a circle $(x-c_x)^2+(y-c_y)^2=r^2$
 - We need to use a 3D accumulator
 - Similar approach ($O(n^2)$)