UiO : Department of Technology Systems
University of Oslo

Lecture 3.1
Line features

Idar Dyrdal

Edges and lines

An edge is a place of rapid change of image intensity, colour or texture, representing:

- Boundaries of objects
- Shadow boundaries
- Creases
- …

Edge points (*edgels*) can be grouped into:

- Curves/contours
- Straight line segments
- Piecewise linear contours
- …

Edge operators (edge enhancement filters)

Edge pixels are found at extrema of the first derivative of the image intensity function.

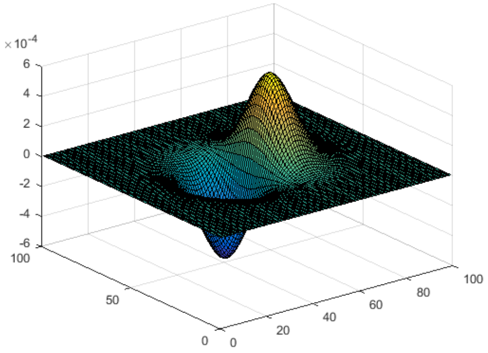
Image gradient (noisy):

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Gradient magnitude:

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

Derivative of Gaussian (smoother result):



$$\frac{\partial}{\partial u} h_\sigma(u, v)$$

$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{u^2+v^2}{2\sigma^2}\right)}$$

Prewitt operator:

$G_x =$

-1	0	1
-1	0	1
-1	0	1

$G_y =$

-1	-1	-1
0	0	0
1	1	1

Sobel operator:

$S_x =$

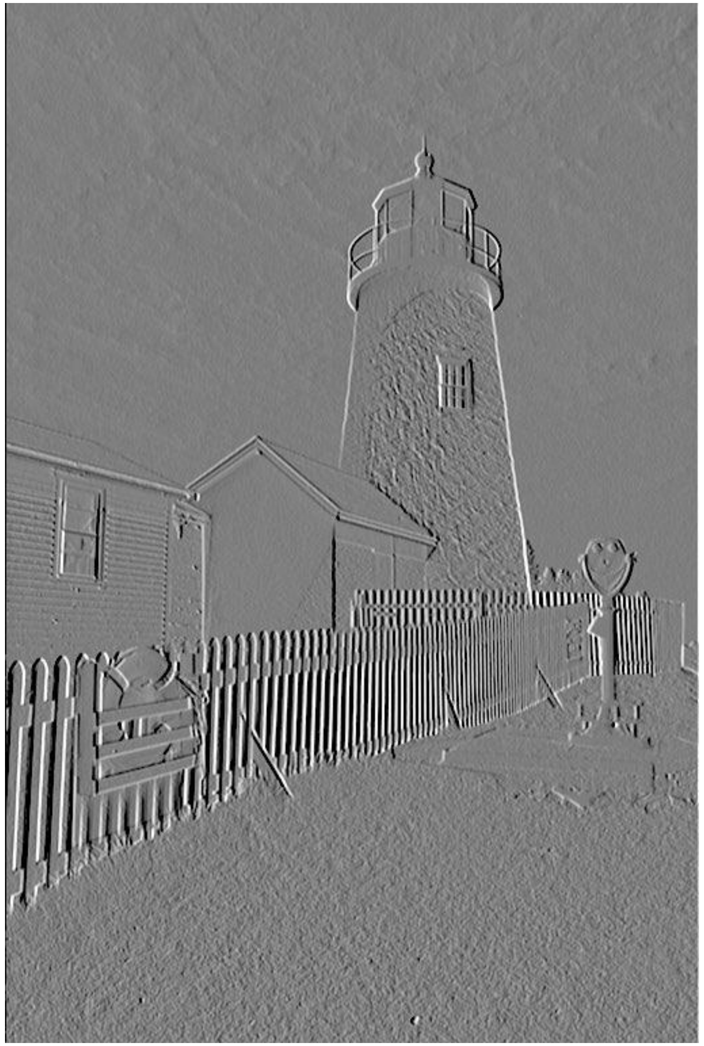
-1	0	1
-2	0	2
-1	0	1

$S_y =$

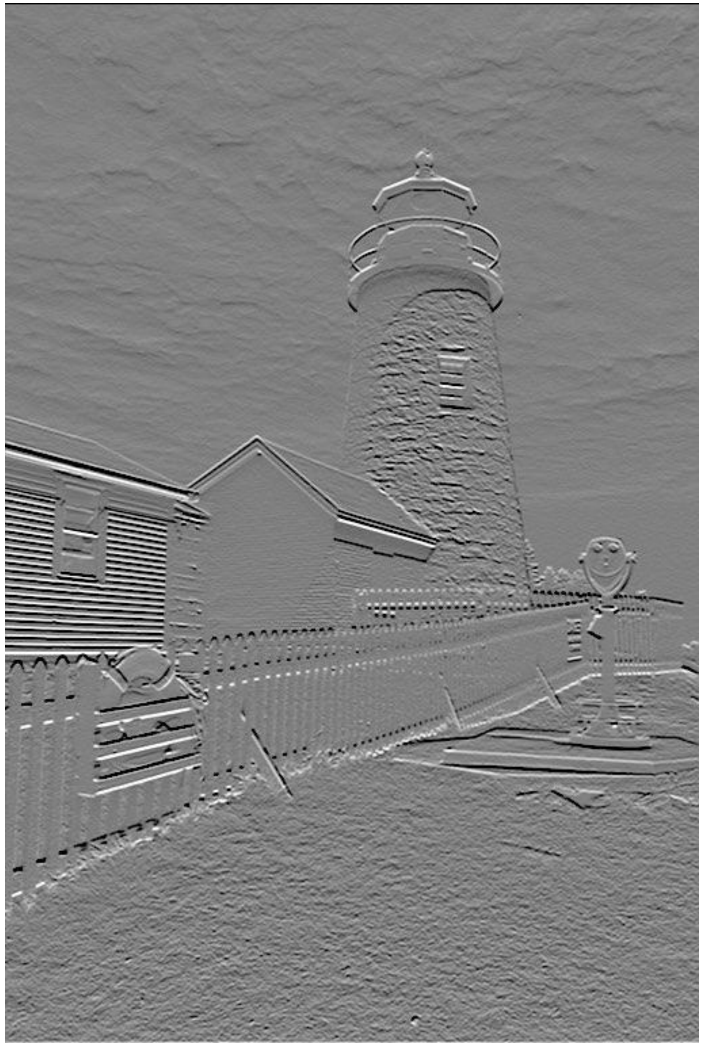
-1	-2	-1
0	0	0
1	2	1

Image derivatives - Sobel

Gray level image



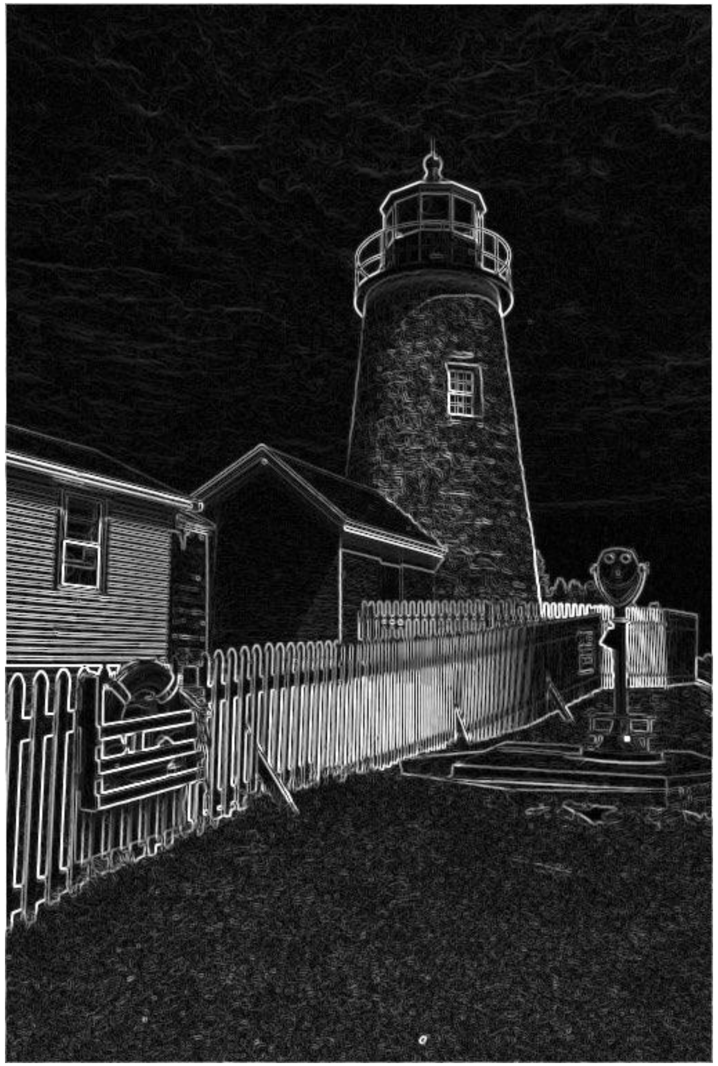
x-component



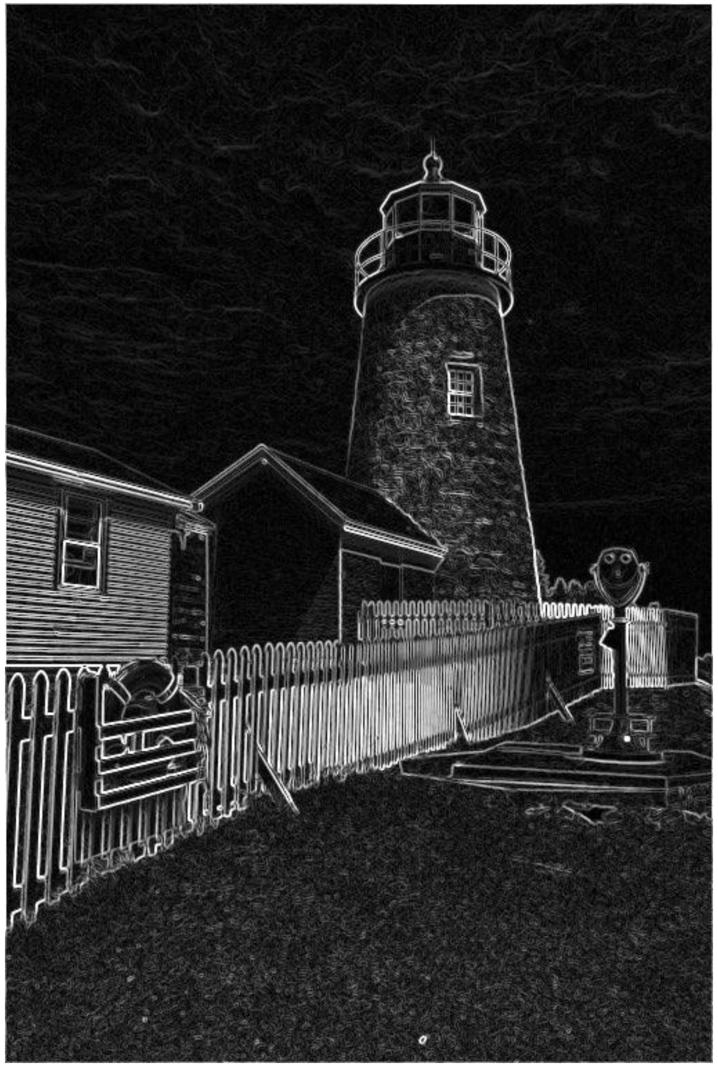
y-component

Gradient magnitude

Gray level image



Gradient magnitude - Prewitt

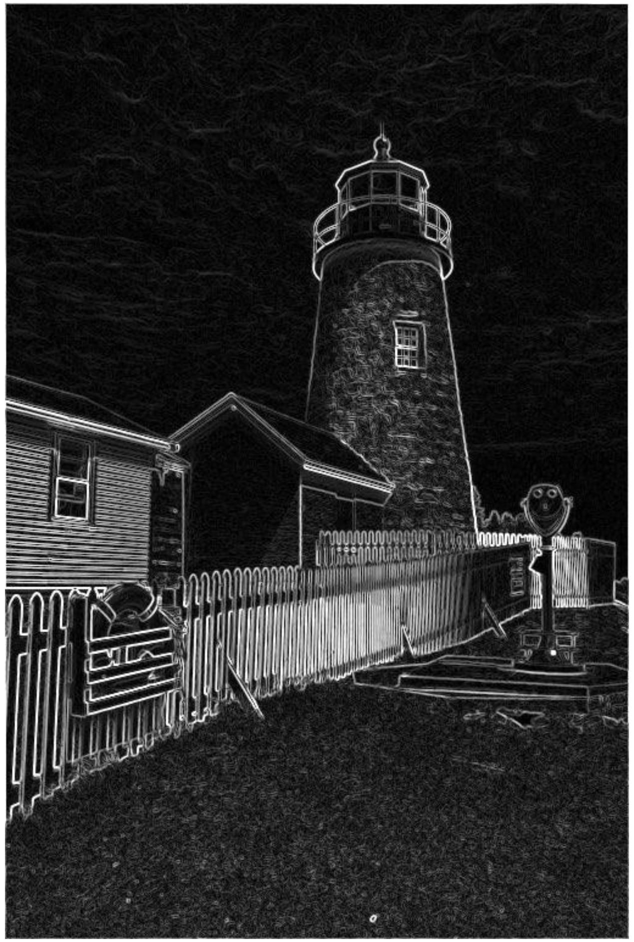


Gradient magnitude - Sobel

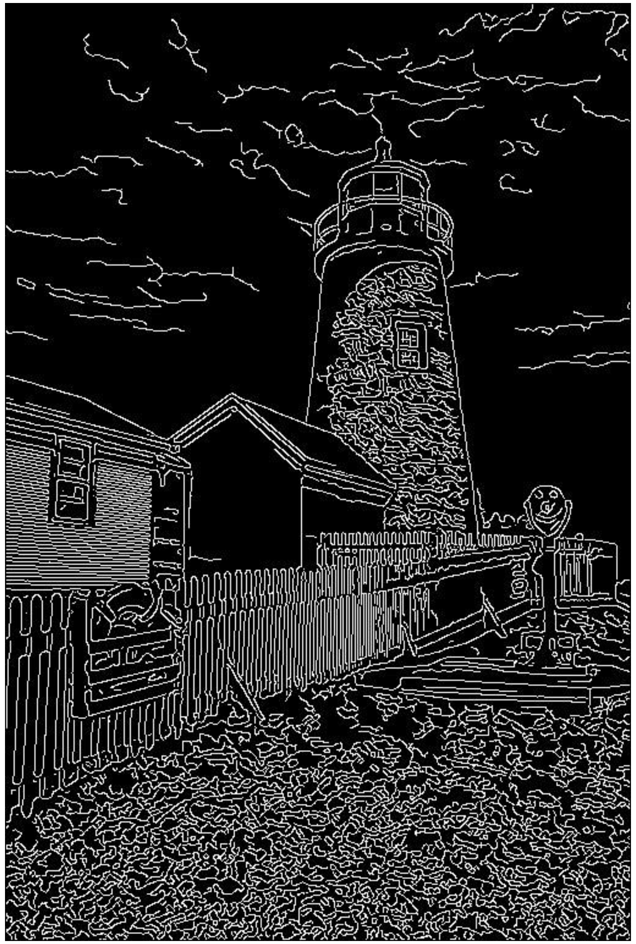
Thinning and thresholding

- Detection of local maxima (i.e. suppression of non-maxima) along the gradient (across edges)
- Thresholding

Binary image with isolated edges (single pixels at discrete locations along edge contours)



Edge enhanced image (Sobel)

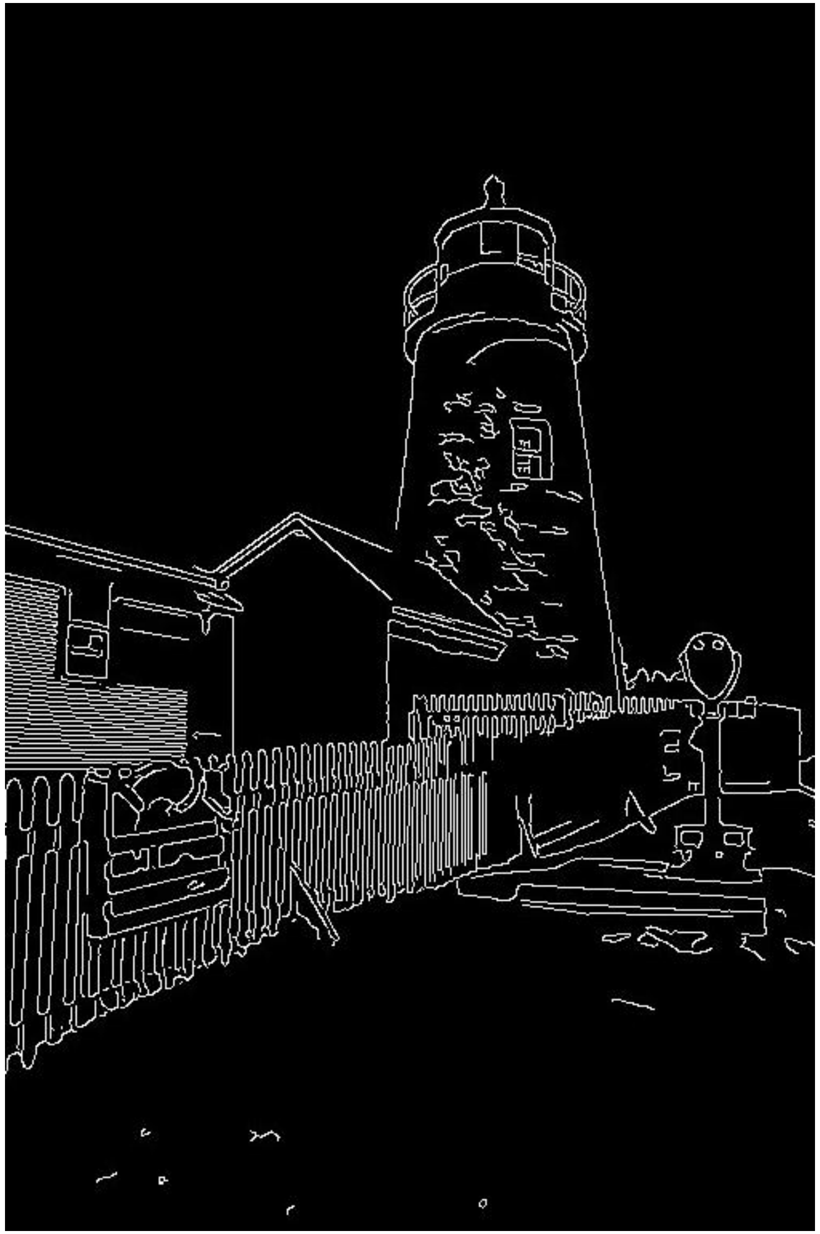


Edge image (Canny)

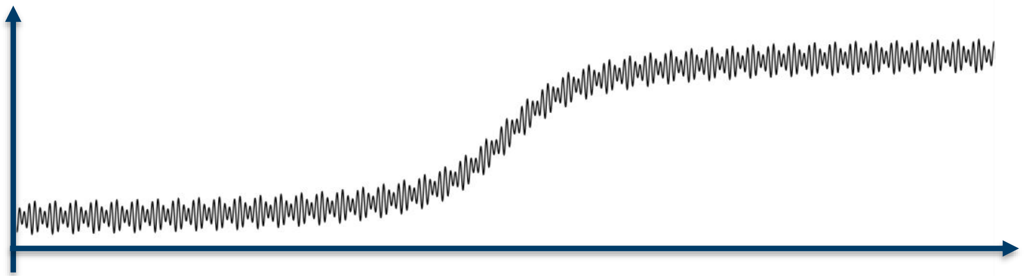
Canny edge detector

- Calculates a gradient image using the derivative of a Gaussian filter (i.e. Sobel operator)
- Detects local maxima of the gradient
- Thresholding using two thresholds:
  - **High** threshold for detection of strong edges
  - **Low** threshold for detection of weak edges
- Only weak edges connected to strong edges are retained in the output image

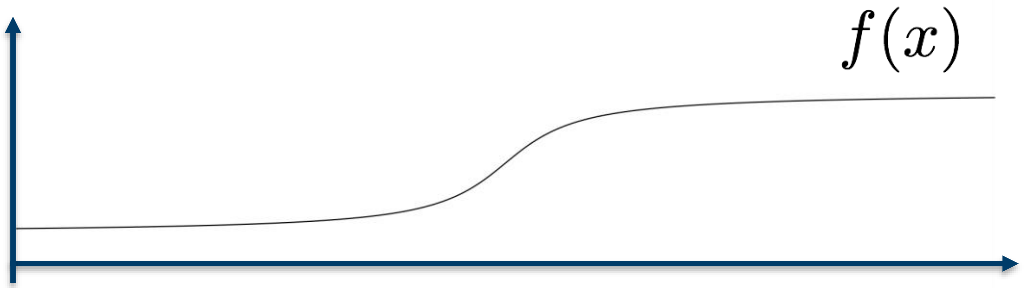
- This method is less likely to be fooled by noise than other methods, and
- More likely to detect true weak edges



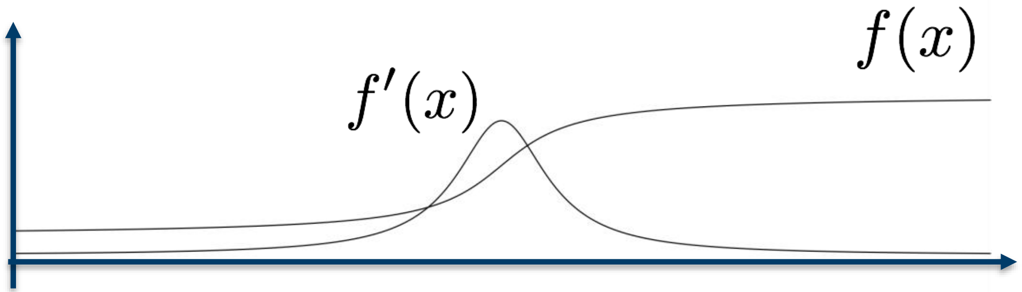
First and second derivatives



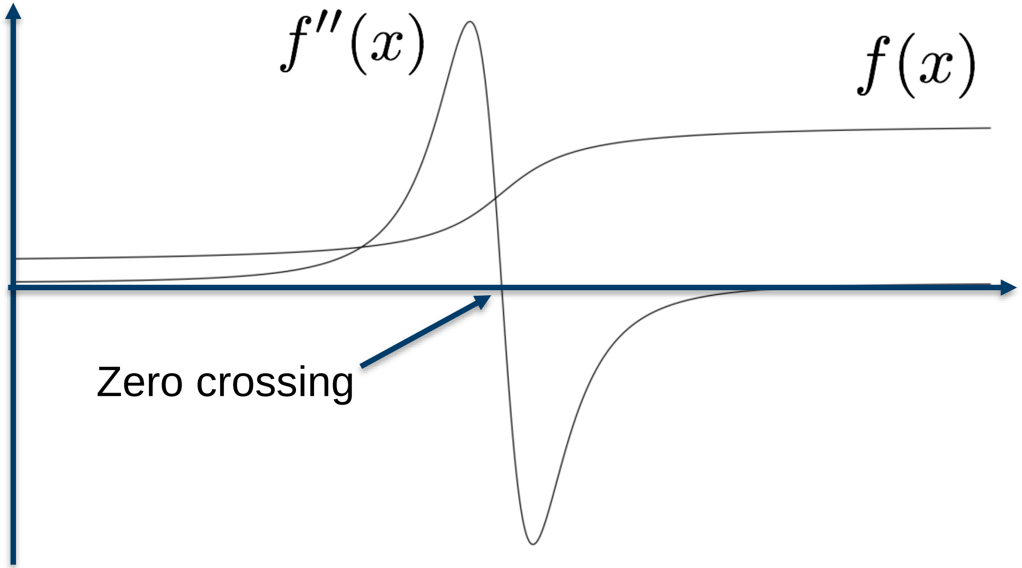
Noisy image function



Low-pass filtered image function



First derivative (Gradient)



Second derivative (Laplacian)

Laplacian operator

Gradient (in two dimensions):

$$\nabla = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

Laplacian:

$$\nabla \cdot \nabla = \nabla^2 = \frac{\partial^2}{\partial^2 x} + \frac{\partial^2}{\partial^2 y}$$

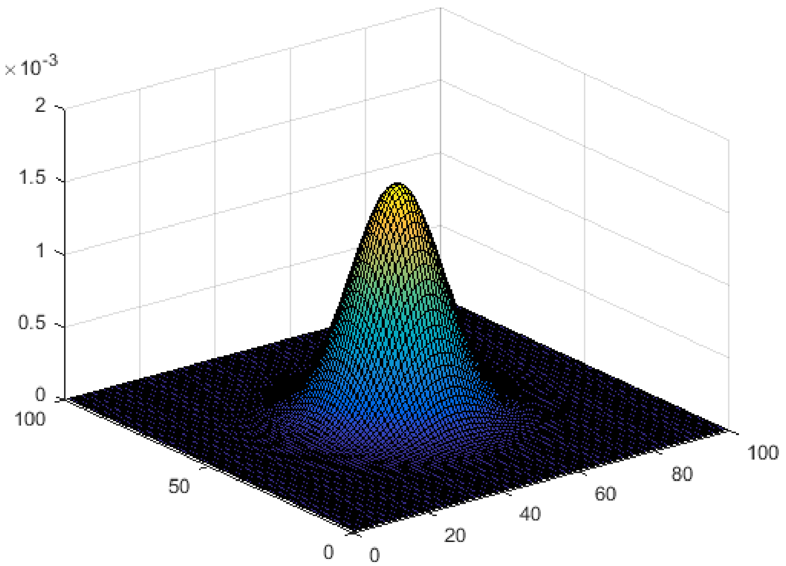
Discrete approximations (3 x 3 kernels):

$$\frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

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

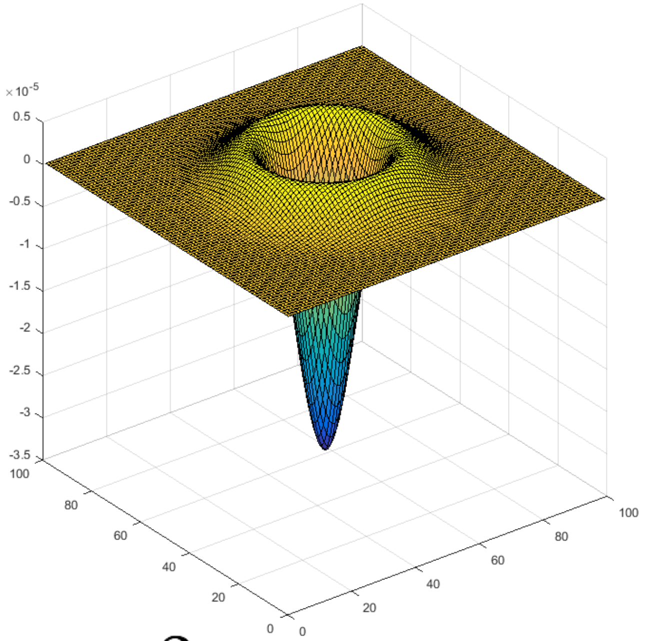
Laplacian of Gaussian (LoG)

Gaussian



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\left(\frac{u^2+v^2}{2\sigma^2}\right)}$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

Edge pixels at zero-crossings in the LoG image!

Laplacian of Gaussian - example

$$\nabla^2 h_\sigma(u,v)$$

Laplace

0	1	0
1	-4	1
0	1	0

*

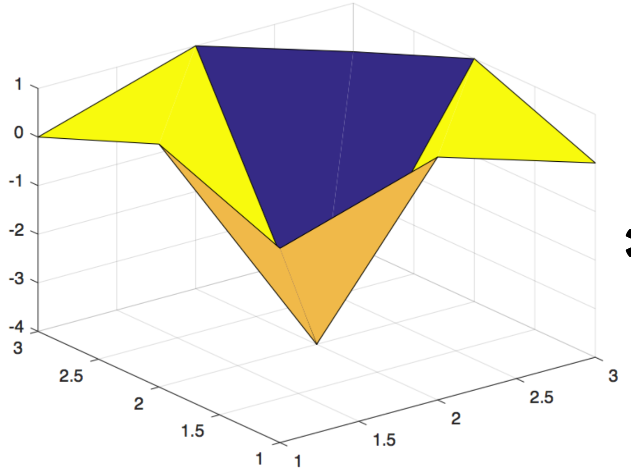
Gauss

0.0113	0.0838	0.0113
0.0838	0.6193	0.0838
0.0113	0.0838	0.0113

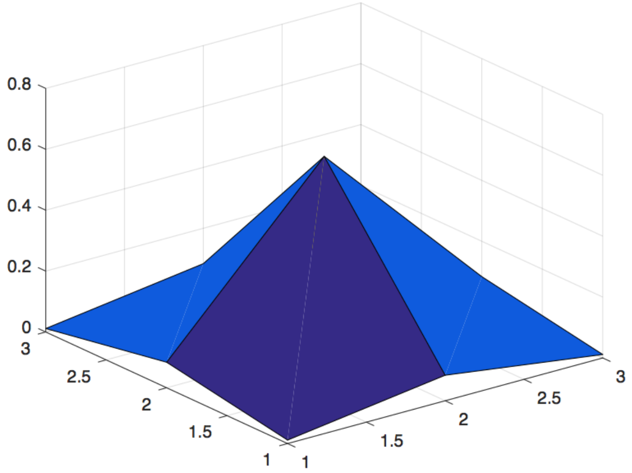
=

LoG

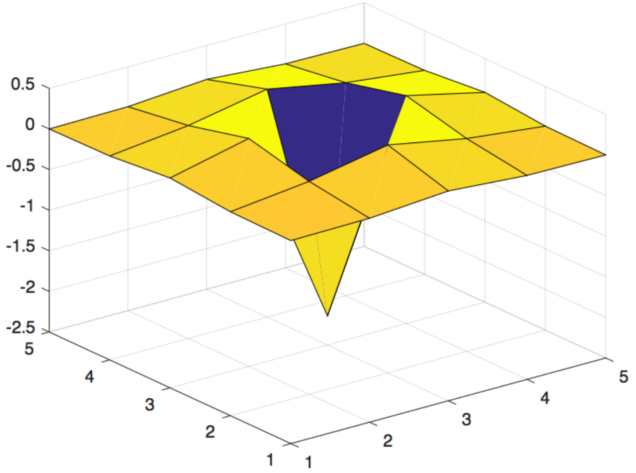
0.0000	0.0113	0.0838	0.0113	0.0000
0.0113	0.1223	0.3068	0.1223	0.0113
0.0838	0.3068	-2.1421	0.3068	0.0838
0.0113	0.1223	0.3068	0.1223	0.0113
0.0000	0.0113	0.0838	0.0113	0.0000



*



=

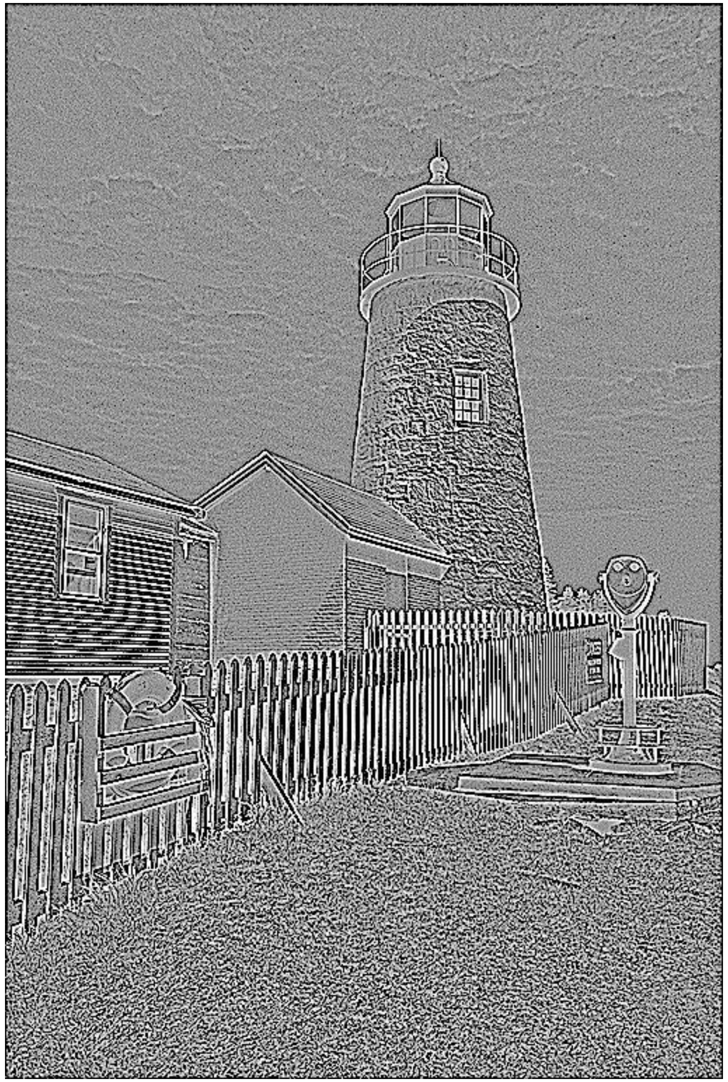


Examples - Laplacian and LoG

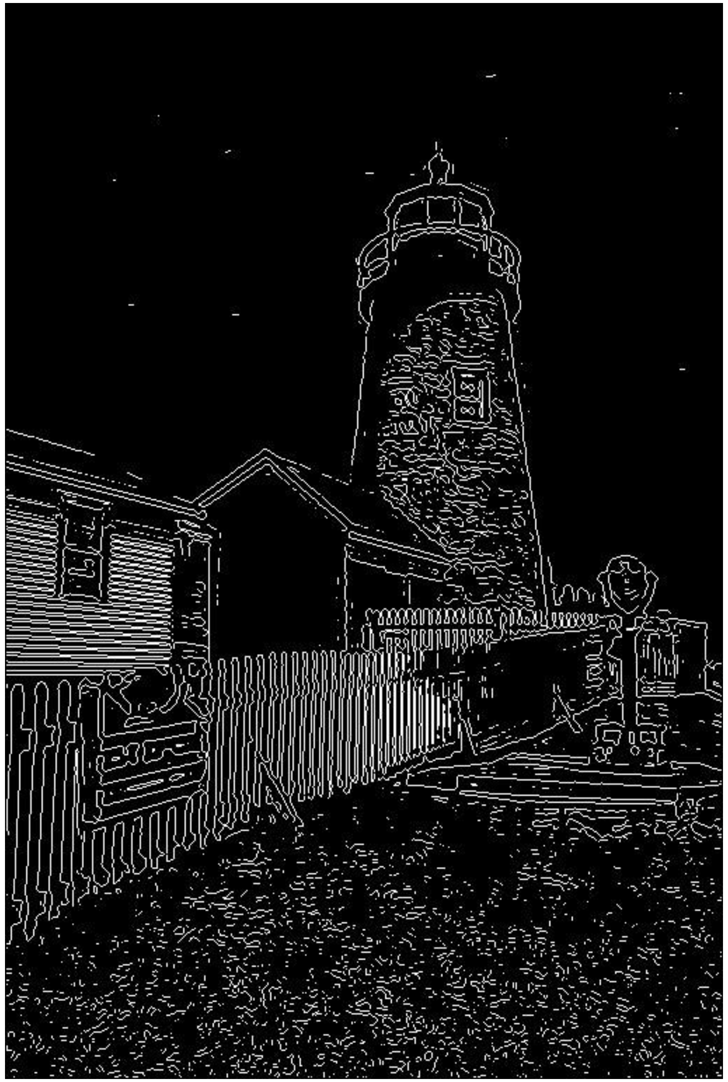
Laplace

Laplacian of Gaussian

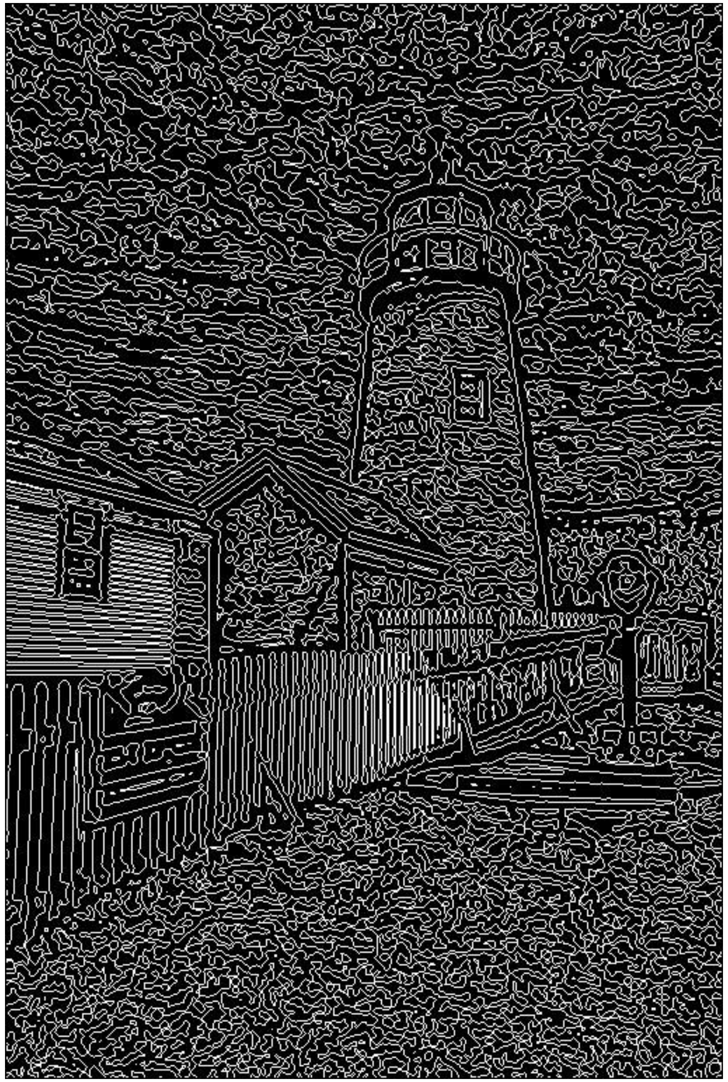
Edge detection - Laplacian of Gaussian (LoG)



LoG (gray level)

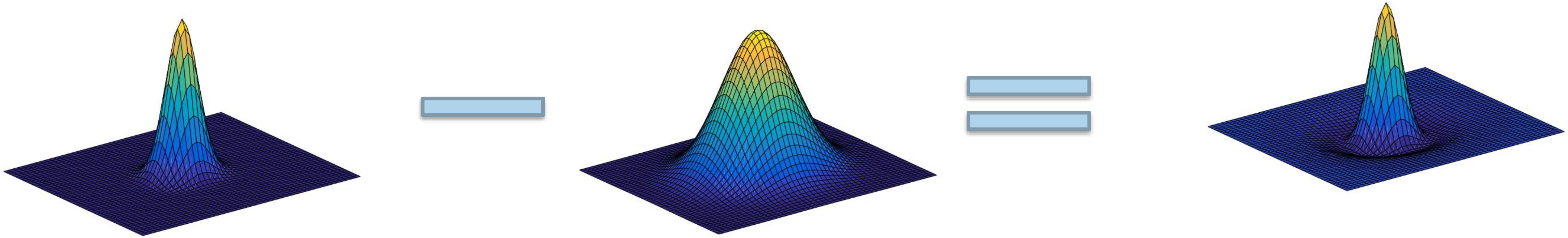


Thresholded zero crossing (binary)



All zero crossings (binary)

Difference of Gaussians (DoG)



Small variance

Large variance

DoG (approximation to LoG)

Difference of Gaussians - approximation to LoG

Another example

RGB original

Gray level

Laplace and LoG images

Laplace

LoG

DoG images

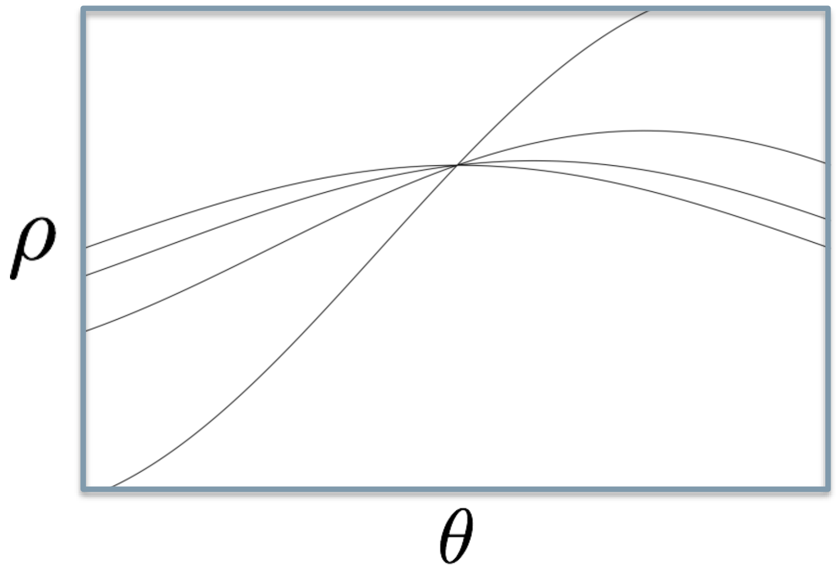
3 x 3 Gaussian kernel

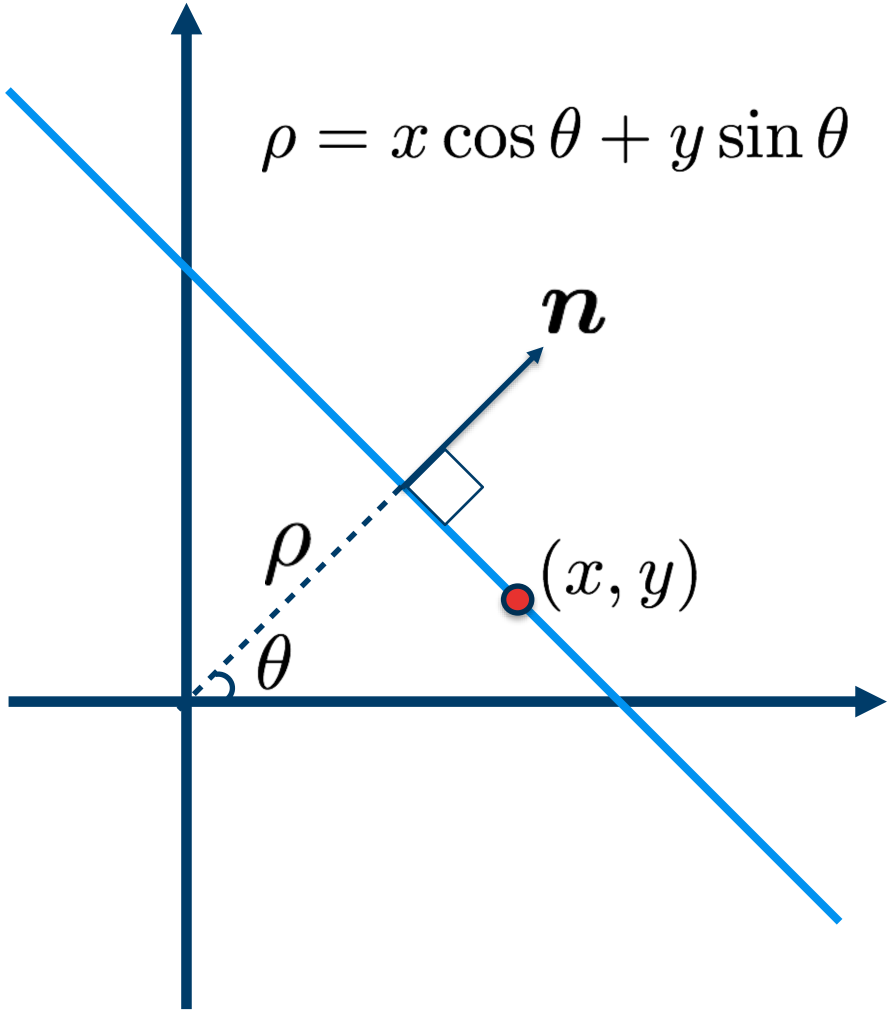
7 x 7 Gaussian kernel

Line detection - Hough transform

The set of all lines going through a given point corresponds to a sinusoidal curve in the (ρ, θ) plane.

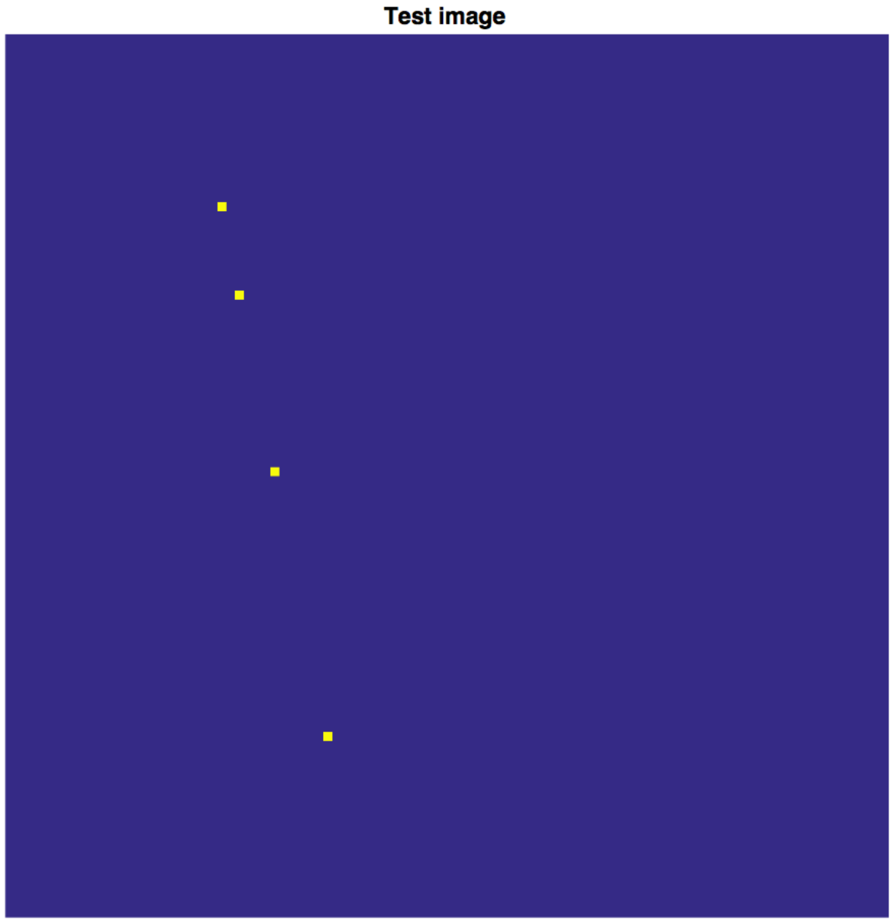
Two or more points on a straight line will give rise to sinusoids intersecting at the point (ρ, θ) for that line.

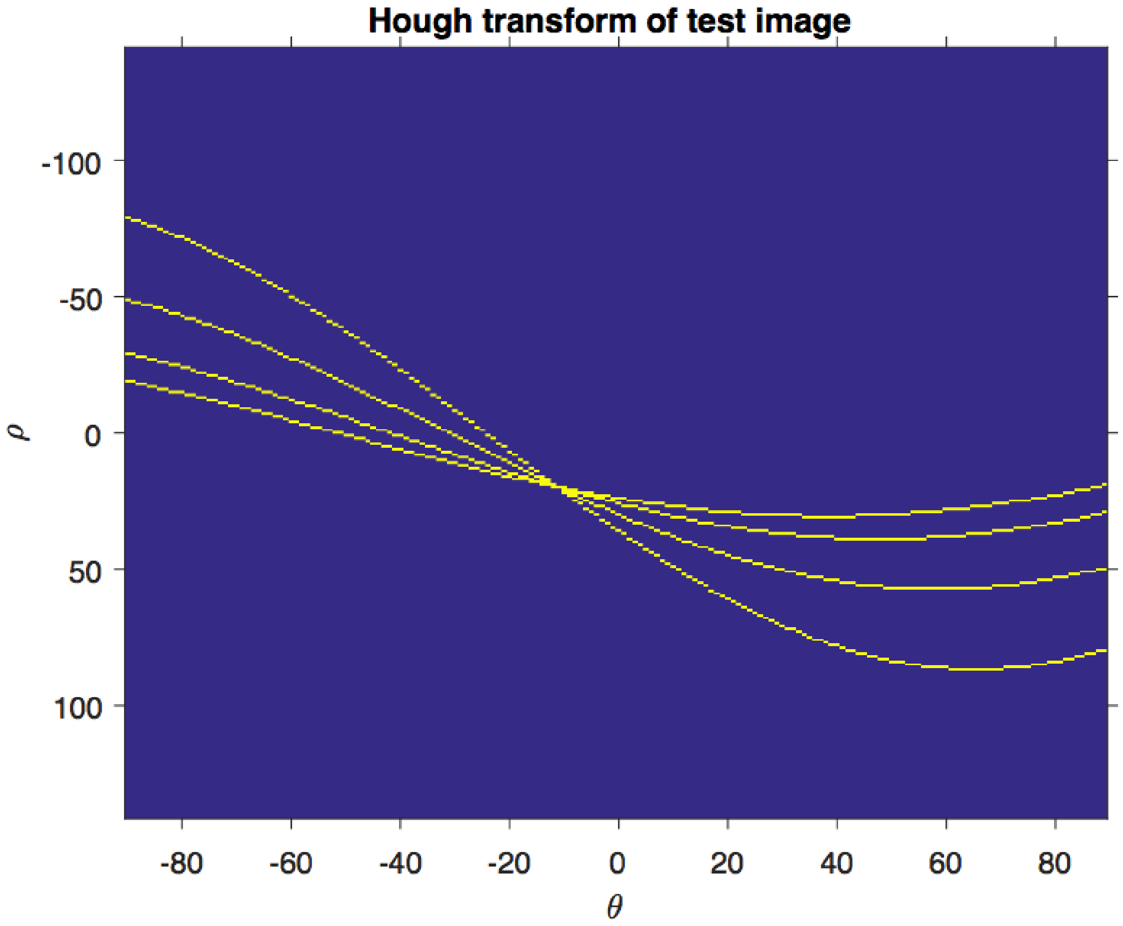




The Hough transform can be generalized to other shapes.

Example





Accumulator

Hough transform

1. Clear the accumulator array

2. For each detected edgel (edge pixel) at location (x, y) and each orientation $\theta = \tan^{-1}(n_y/n_x)$ compute the value of:

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

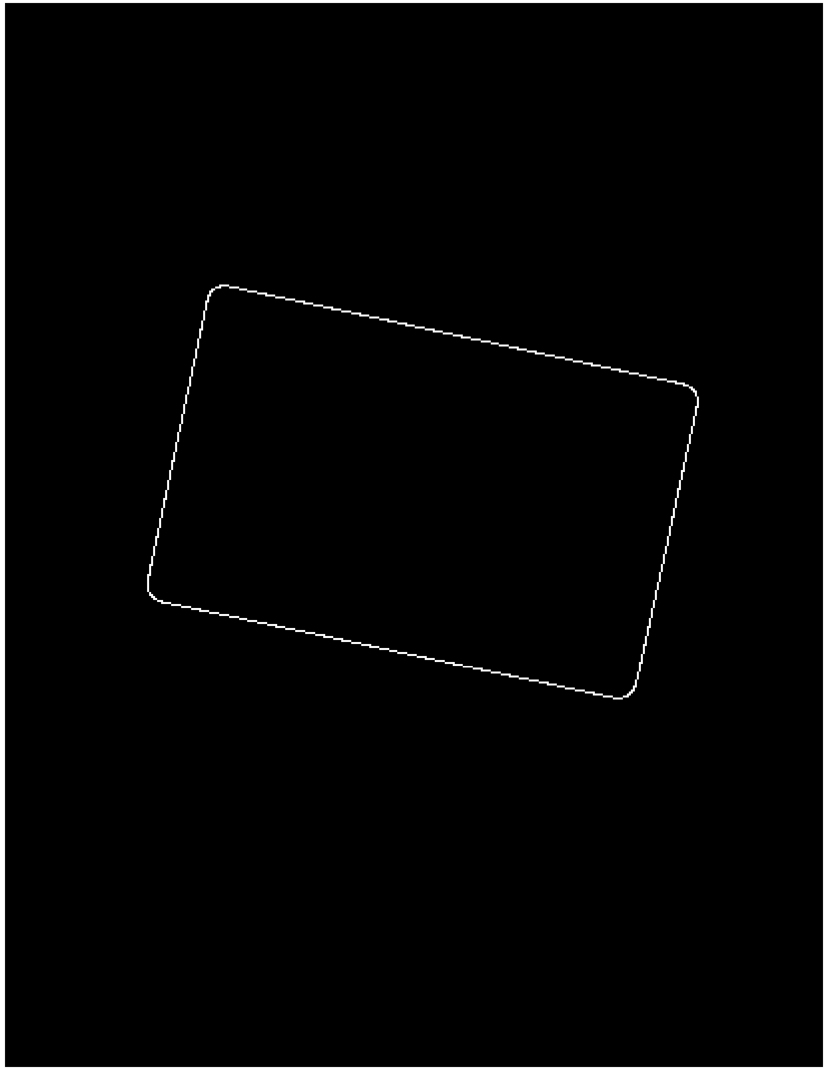
   and increment the accumulator bin corresponding to (ρ, θ)

3. Find the peaks (local maxima) in the accumulator corresponding to lines

4. Optional post-processing to fit the lines to the constituent edgels.

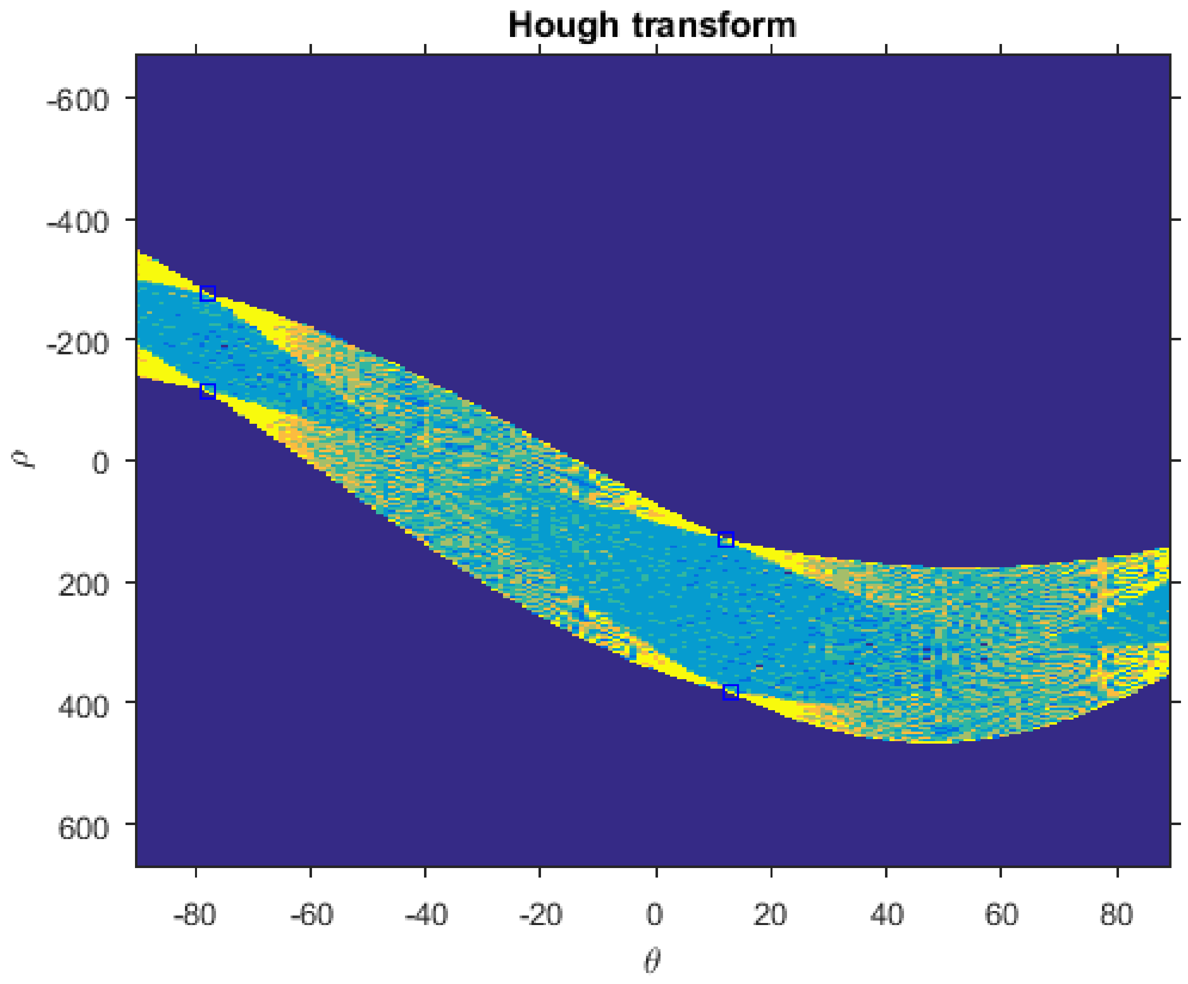
Example 1

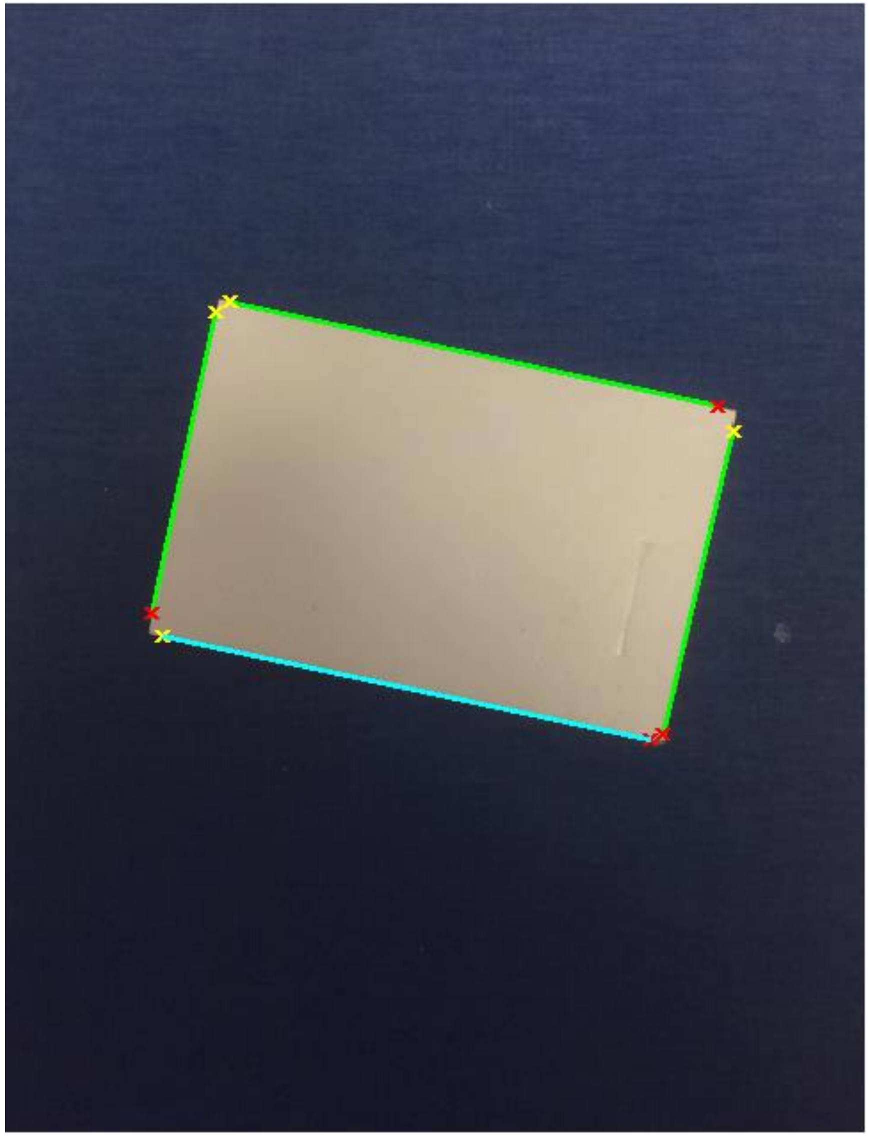
Original



Edge image (Canny)

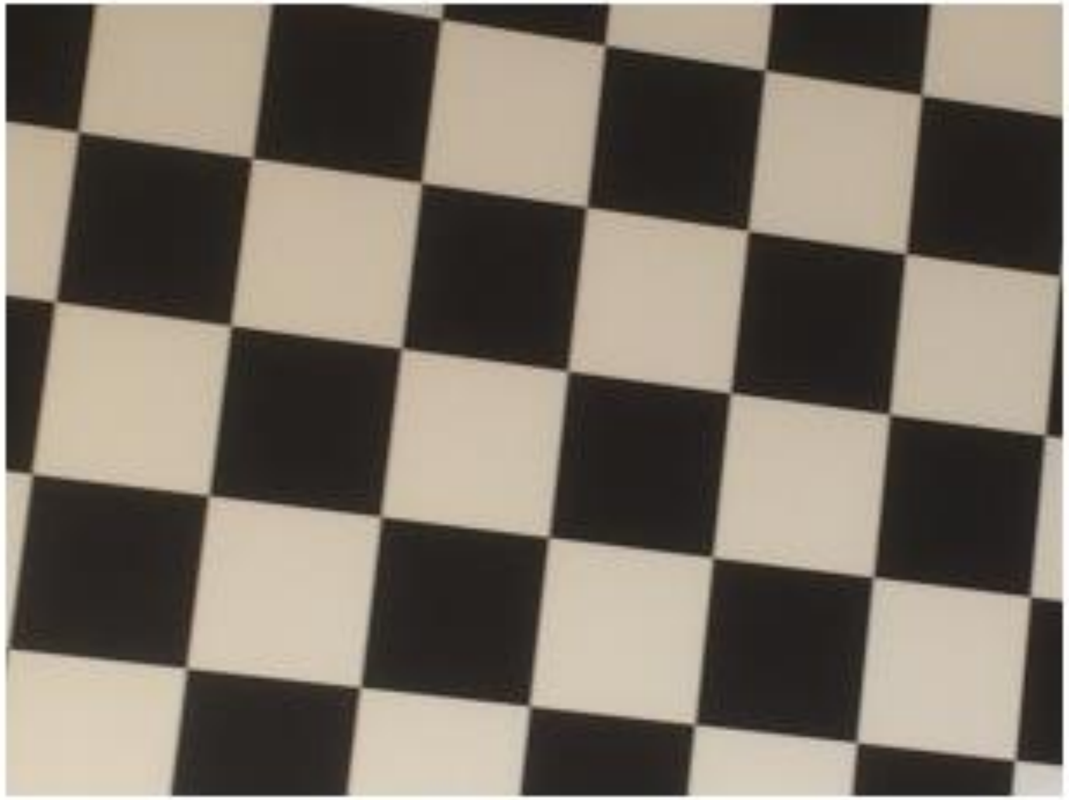
Example 1 (2)



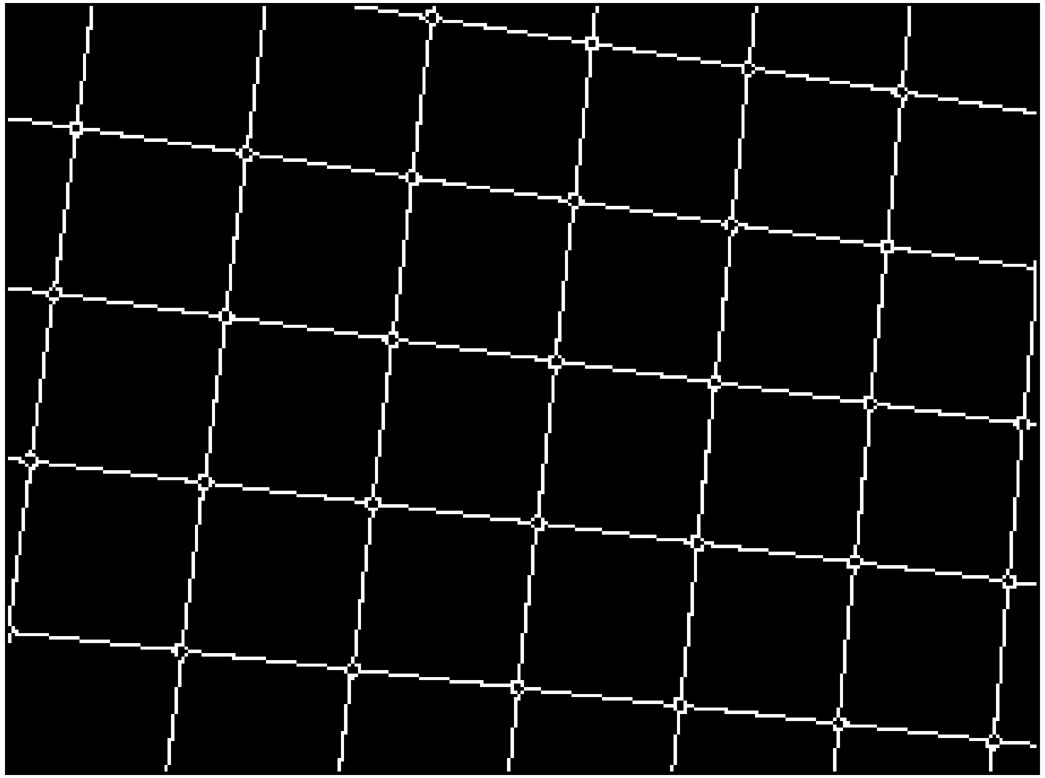


Detected lines

Example 2

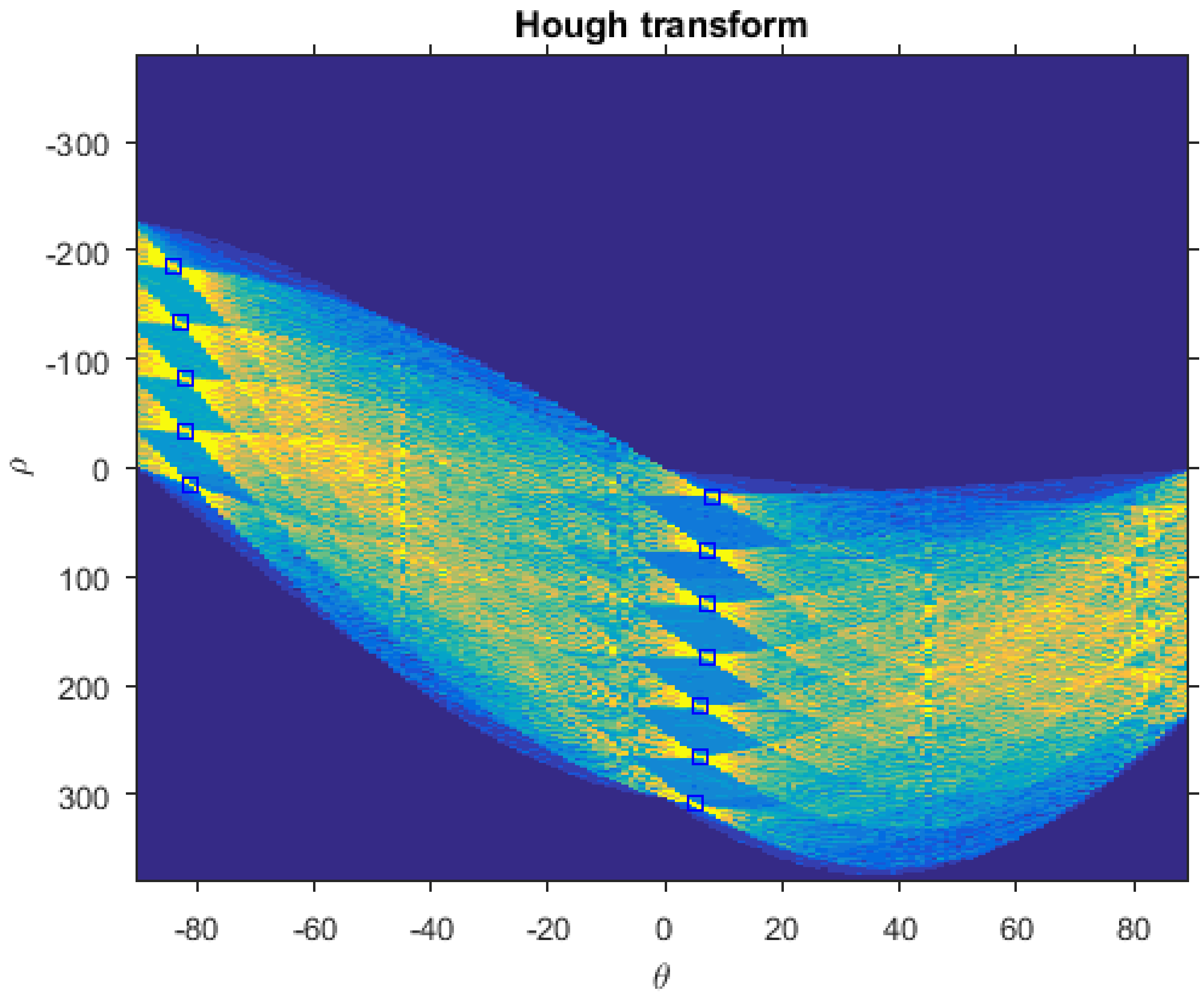


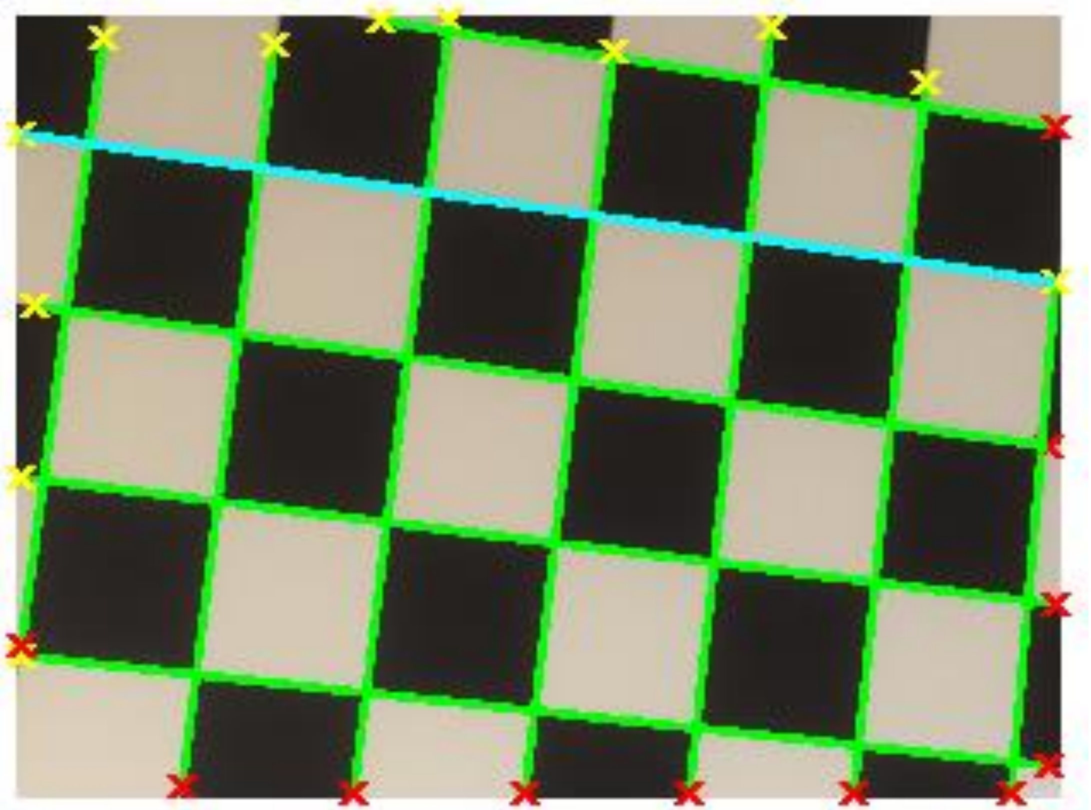
Original



Edge image (Canny)

Example 2 (2)





Detected lines

Example 3

Original

Edge image (Canny)

Example 3 - some results

Line detection - complicated scene

Summary

Line features:

- Edge detectors
- Line detection with the Hough transform

More information: Szeliski 4.2 - 4.3.