CE6902 Computer Vision

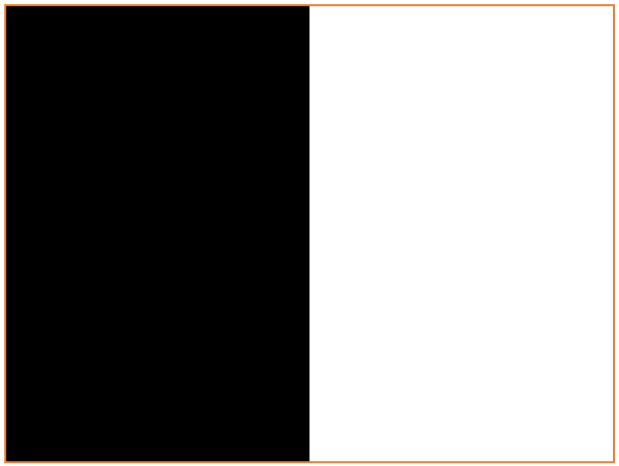
Image Features – Edges

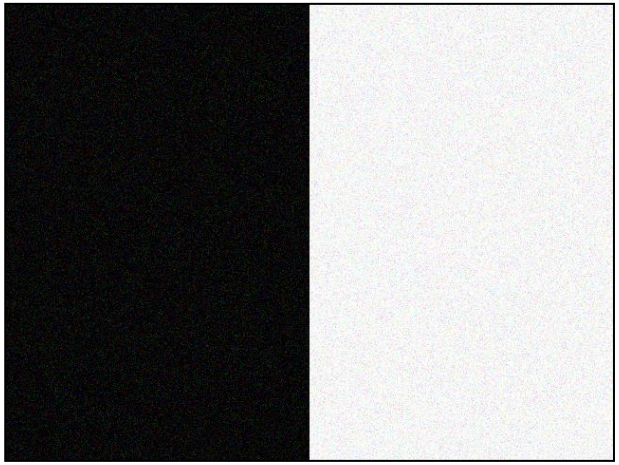
Contents and Learning Objectives

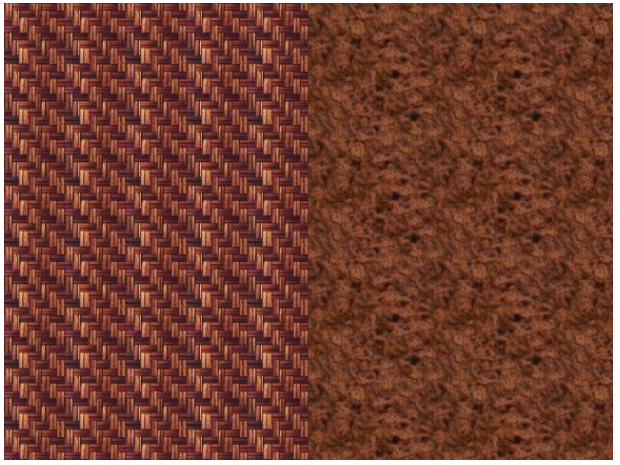
1. Introduction
2. Edge Detection
3. Line Detection
4. Applications

1. Introduction: What is an edge?

The definition of image edges is often ambiguous.

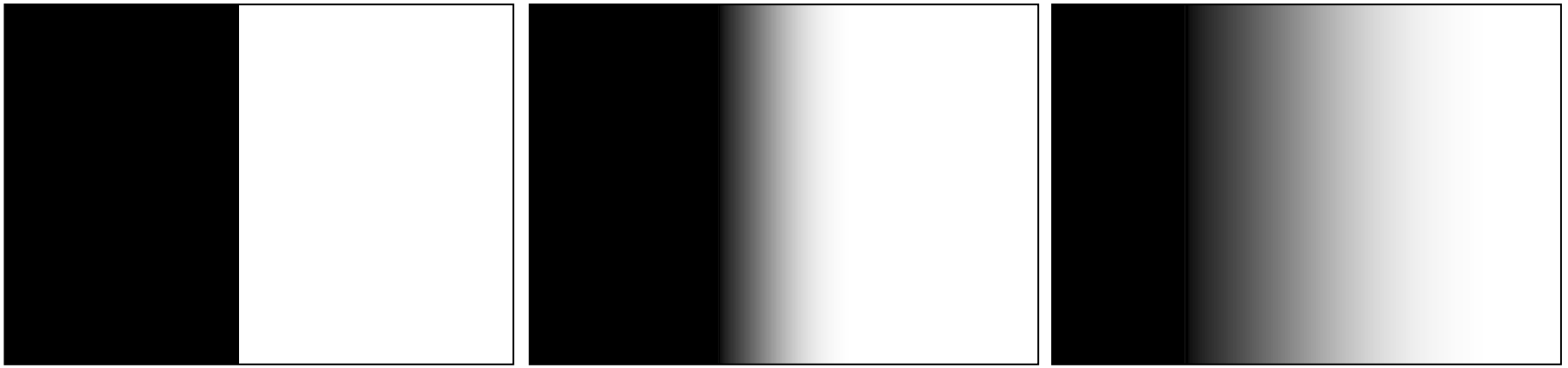






1. Introduction: What is an edge?

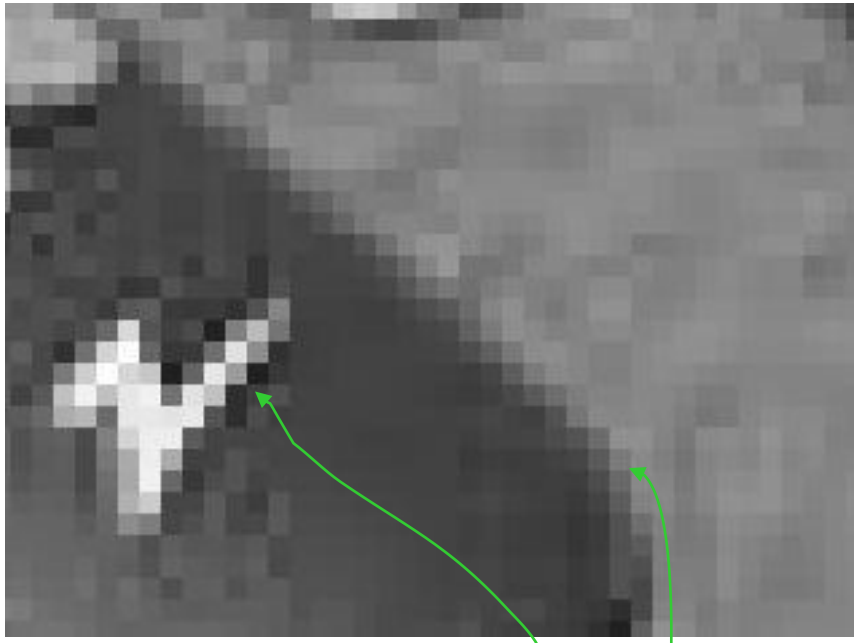
- A *large change* in image brightness of a *short spatial distance*
- Edge strength: $\frac{I(x,y)-I(x+dx,y)}{dx}$



The above definition is still quite general, allowing for different theories, software implementation, etc.

1. Introduction: How is edge formed?

Edges are typically caused by four physical phenomena:

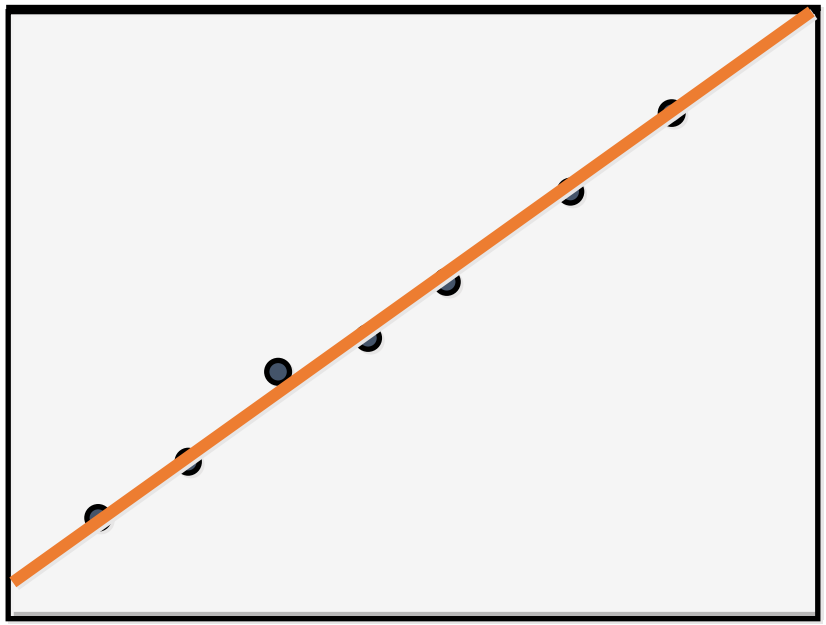


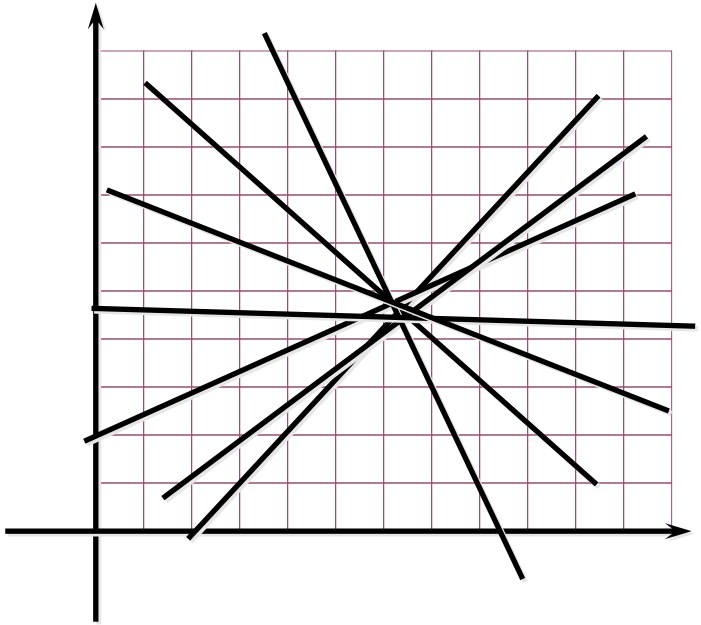
Edges caused by

- surface color/intensity
- surface normal
- depth changes
- lighting (specularities)

1. Introduction: What is line detection?

- Edge is local, while line is non-local
- Line detection usually performed on the output of an edge detector
- Lines can be detected by Hough Transform

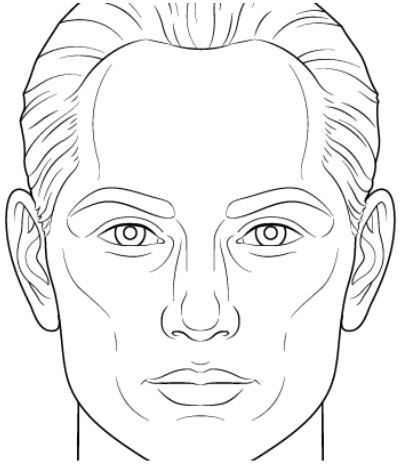
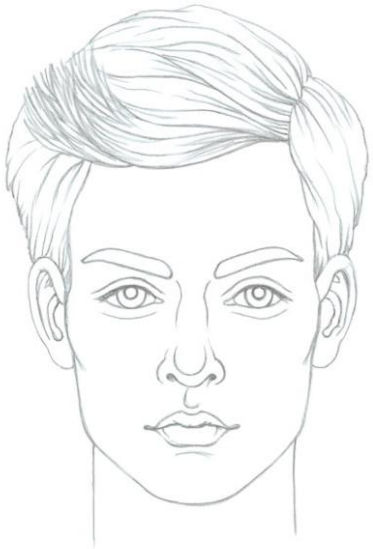


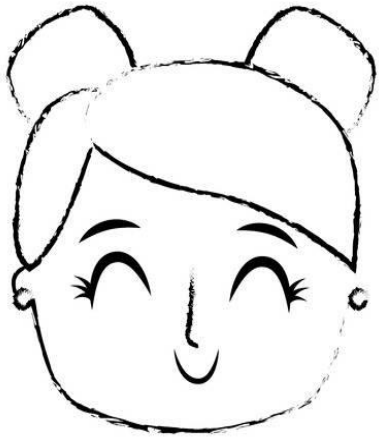


1. Introduction: Why detect edges and lines?

Edge is a very important visual feature in image classification, object detection, object recognition, etc.

Line is a very basic geometric shape that defines the structure of our visual world.

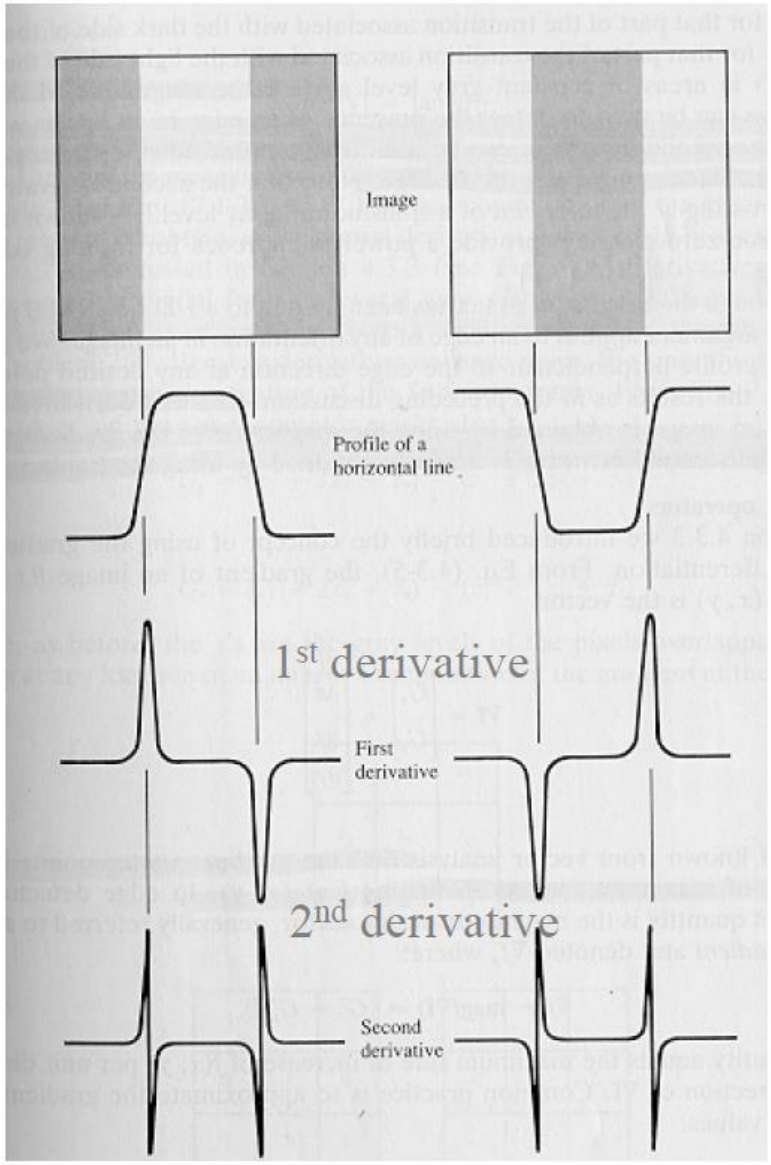


2. Edge Detection

Points that lie on an image edge are usually detected by two different approaches:

(1) Detecting the local *maxima* or *minima* of the **first derivative**.

(2) Detecting the *zero-crossings* of the **second derivative**.

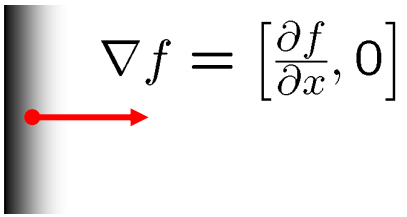


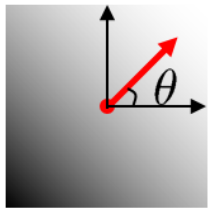
2. Edge Detection using the 1st Derivatives

The *image gradient* is defined by:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity





The *gradient direction* and *gradient magnitude (edge strength)* are given by:

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

How to differentiate a digital image f[x,y]?

$$\frac{\partial f}{\partial x}[x, y] \approx f[x+1, y] - f[x, y]$$

2. Edge Detection using the 1st Derivatives

2D convolution consists of filtering an image *A* using a filter or mask *B*, which is a small image whose pixel values are called weights.

Filter or mask

B

$B_{1,1}$	$B_{1,2}$
$B_{2,1}$	$B_{2,2}$

2×2

$*$

Input image

A

$A_{1,1}$	$A_{1,2}$	$A_{1,3}$	$A_{1,4}$
$A_{2,1}$	$A_{2,2}$	$A_{2,3}$	$A_{2,4}$
$A_{3,1}$	$A_{3,2}$	$A_{3,3}$	$A_{3,4}$
$A_{4,1}$	$A_{4,2}$	$A_{4,3}$	$A_{4,4}$

4×4

$=$

Convolved Image

C

$C_{1,1}$	$C_{1,2}$	$C_{1,3}$
$C_{2,1}$	$C_{2,2}$	$C_{2,3}$
$C_{3,1}$	$C_{3,2}$	$C_{3,3}$

3×3

$A_{1,1} \times B_{1,1}$	$A_{1,2} \times B_{1,2}$
$A_{2,1} \times B_{2,1}$	$A_{2,2} \times B_{2,2}$

$$C_{1,1} = A_{1,1} \times B_{1,1} + A_{1,2} \times B_{1,2} + A_{2,1} \times B_{2,1} + A_{2,2} \times B_{2,2}$$

2. Edge Detection using the 1st Derivatives

2D convolution consists of filtering an image *A* using a filter or mask *B*, which is a small image whose pixel values are called weights.

Filter or mask | Input image | Convolved Image

B

-1	2
-1	2

2×2

*

A

4	4	7	9
4	3	8	9
3	5	9	9
3	6	10	9

4×4

=

C

6	23	21
9	26	19
16	27	17

3×3

2. Edge Detection using the 1st Derivatives

Sobel Operator uses two masks to compute the image gradient.

Column Mask M_c

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

Row Mask M_r

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

For an image *A*, the gradient in *horizontal* and *vertical* directions can be derived by *convolving* *A* with M_c and M_r: $G_x = A * M_c$ and $G_y = A * M_r$

The overall image gradient can thus be derived by

Magnitude $G = \sqrt[2]{G_x^2 + G_y^2}$

Direction $G = \sqrt[2]{G_y^2 / G_x^2}$

2. Edge Detection using the 1st Derivatives

Prewitt Operator uses two masks to compute the image gradient.

Column Mask M_c

$$\begin{pmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

Row Mask M_r

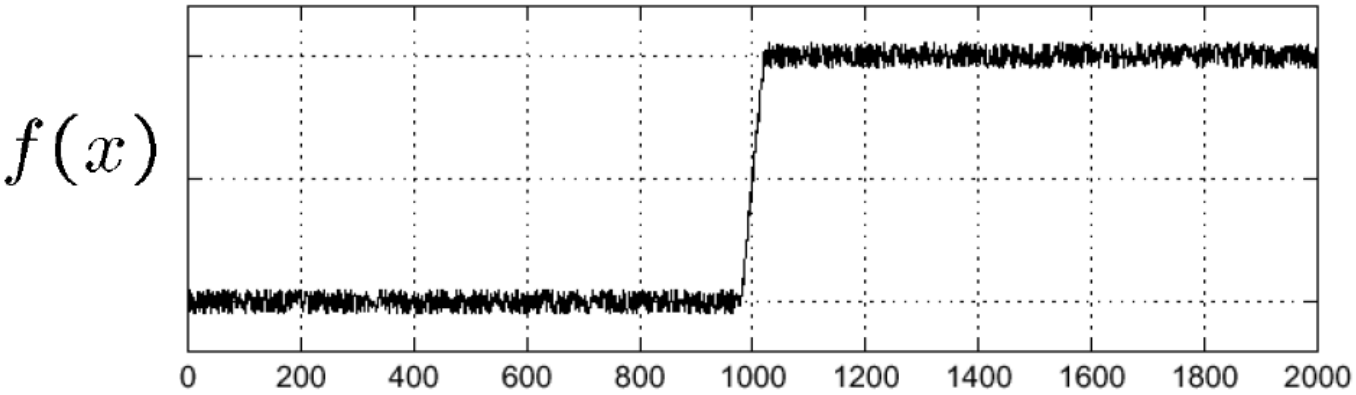
$$\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix}$$

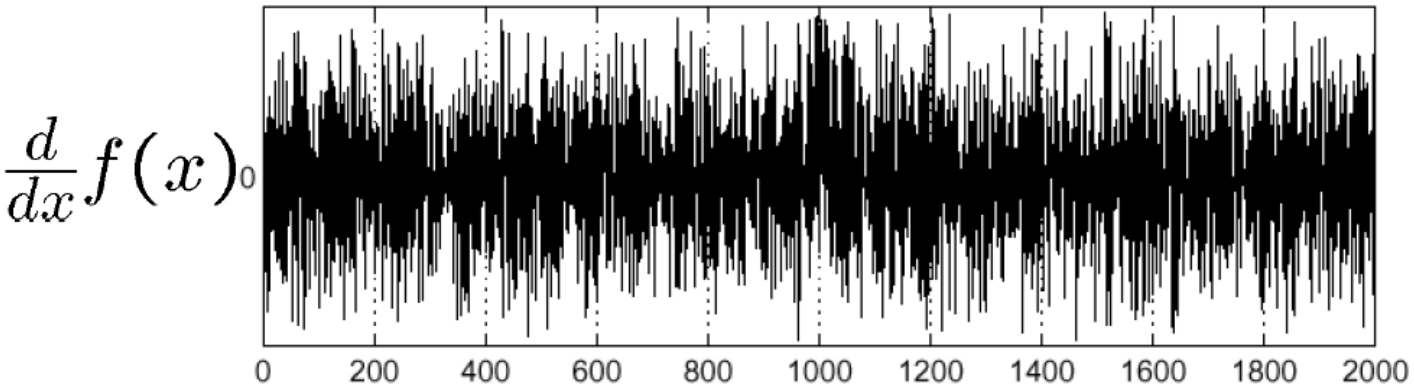
Given an image *A*, the image gradient in *horizontal* and *vertical* directions and the overall image gradient can be computed in the similar way as the Sobel operator.

As a comparison, the Sobel operator gives *greater weight* to center pixels.

2. Edge Detection using the 1st Derivatives

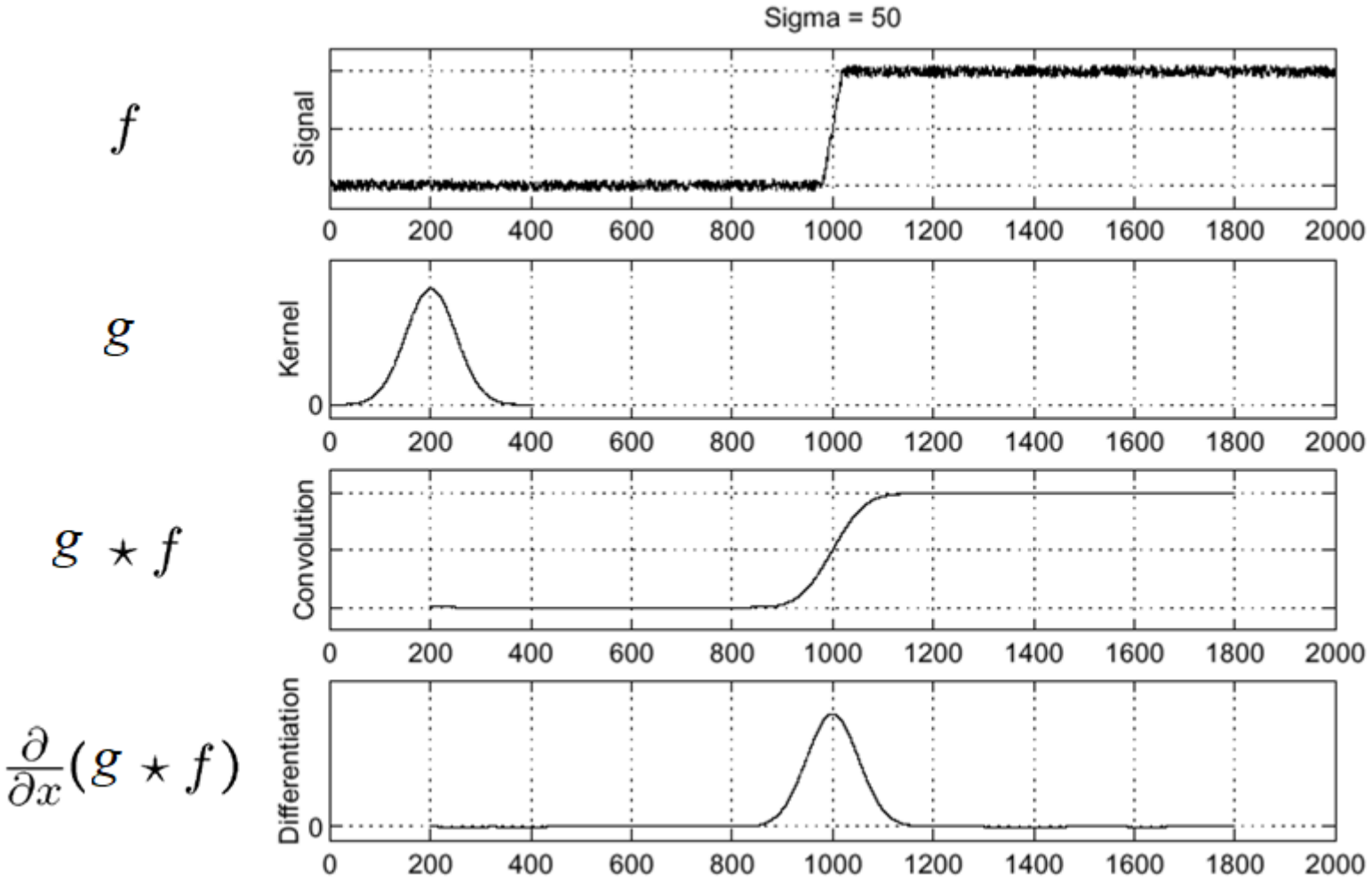
Direct applying 1st derivative such as the Sobel and Prewitt operators is heavily affected by noises as image noises make pixels very different from their neighboring pixels.





2. Edge Detection using the 1st Derivatives

Image noises can be suppressed by first performing image smoothing, and this is often performed by Gaussian smoothing.



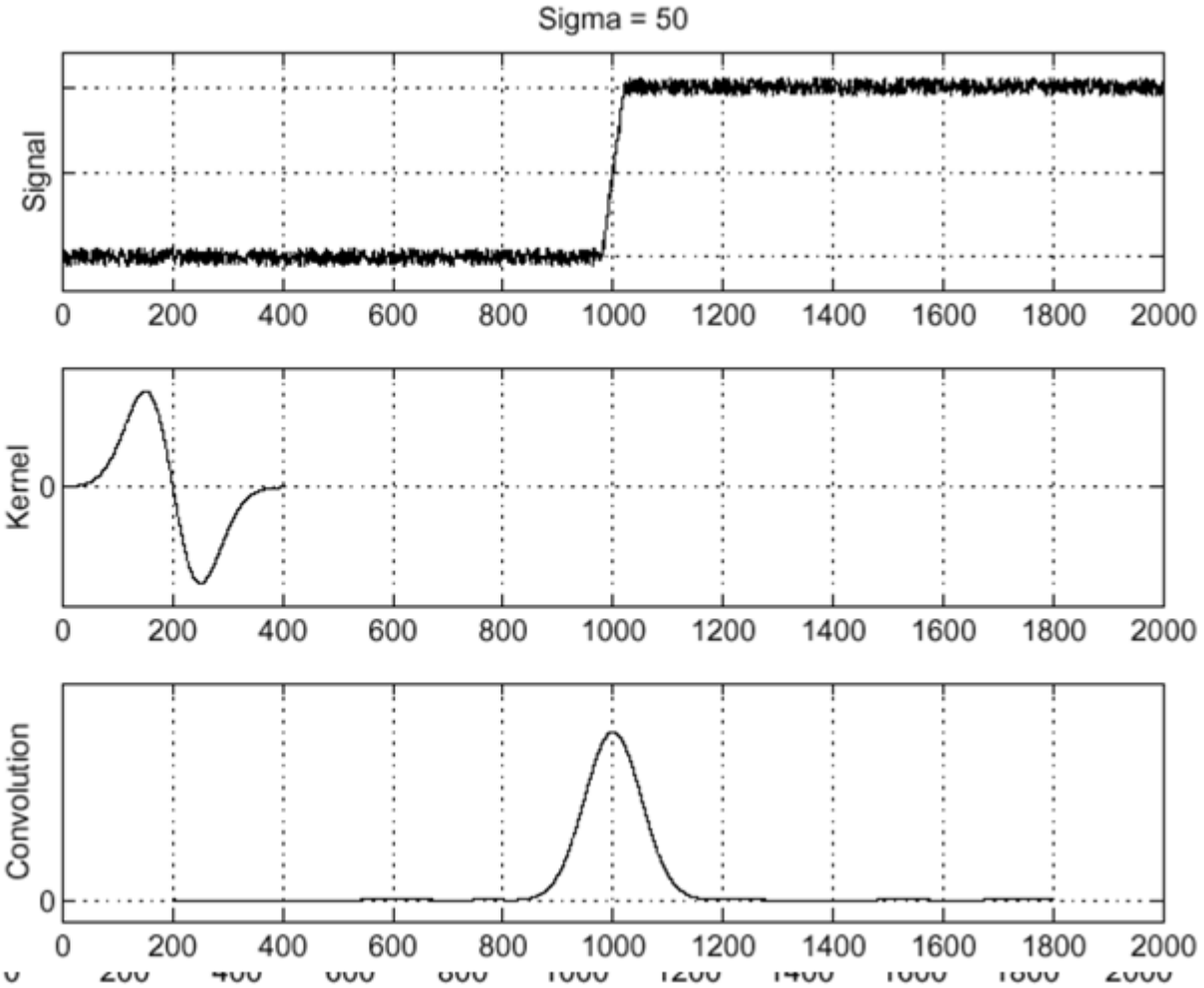
2. Edge Detection using the 1st Derivatives

The operation can actually be simplified as $\frac{\partial}{\partial x}(h \otimes f) = \left(\frac{\partial}{\partial x} h\right) \otimes f$.

f

$\frac{\partial}{\partial x} g$

$(\frac{\partial}{\partial x} g) \star f$

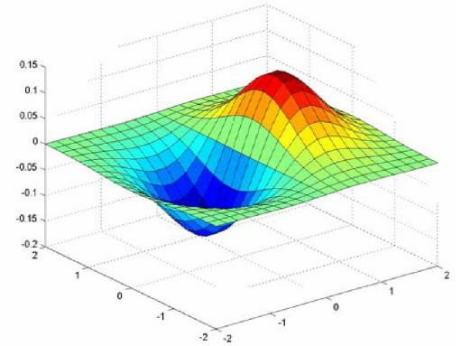


2. Edge Detection using the 1st Derivatives

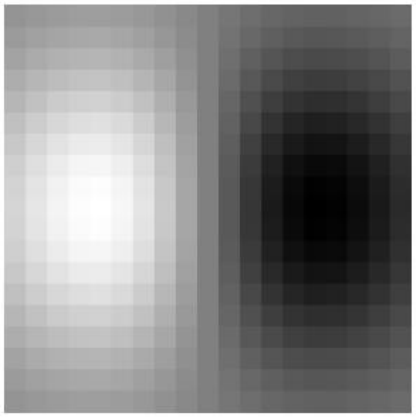
With an image *I*, a Gaussian filter *g*, and a convolution mask *h*, the image gradient can be computed as follows.

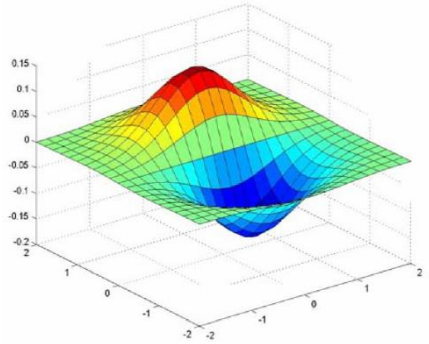
$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

$$\begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix} \otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$$

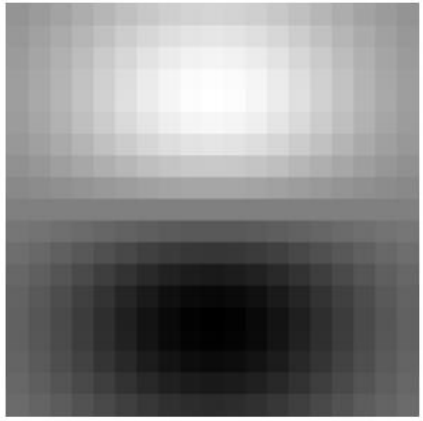


***x*-direction**





***y*-direction**



2. Edge Detection using the 1st Derivatives

With the image gradient derived by using the Sobel mask, Prewitt mask, or the 1st derivative of Gaussian, edge pixels can be detected by applying a global threshold to the gradient images.

On the other hand, a good edge detector should satisfy three *criteria*:

(1) Good detection

- Minimize the probability of false positives (i.e., spurious edges).
- Minimize the probability of false negatives (i.e., missing real edges).

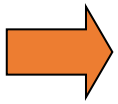
(2) Good localization – Detected edges must be as close as possible to the true edges.

(3) Single response – Minimize the number of local maxima around the true edge.

2. Edge Detection using the 1st Derivatives

Canny edge detector is widely used which considers *good detection*, *good localization*, and *single response*. It detects edges by three steps:

1. *Gaussian edge filtering*: It aims to suppress noises and determine image gradient by the 1st derivative of Gaussian as described.

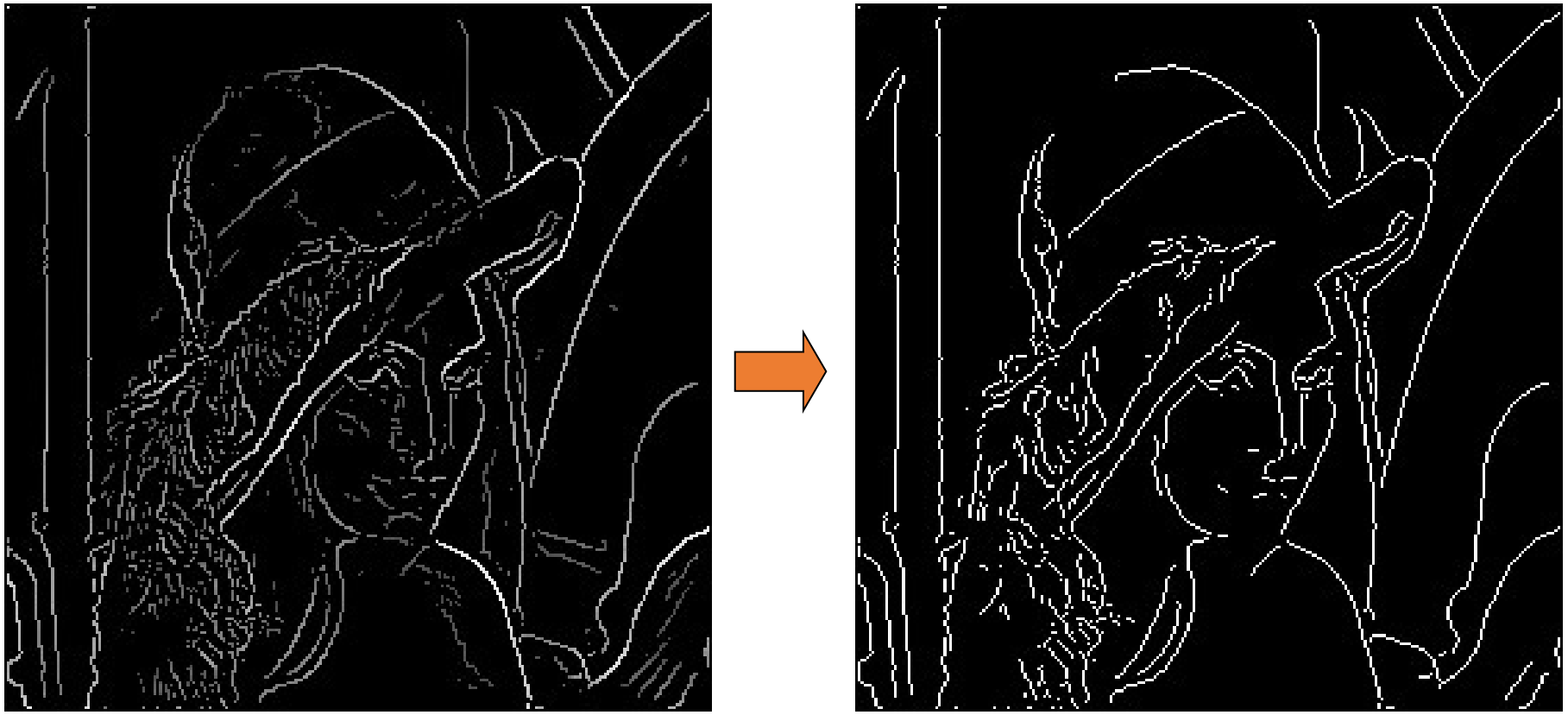


2. Edge Detection using the 1st Derivatives

2. *Non-maximal suppression*: It aims to reduce the broad filtered edges to single-pixel-wide by setting Pixels in the gradient magnitude map to zero if they are not the local maxima.

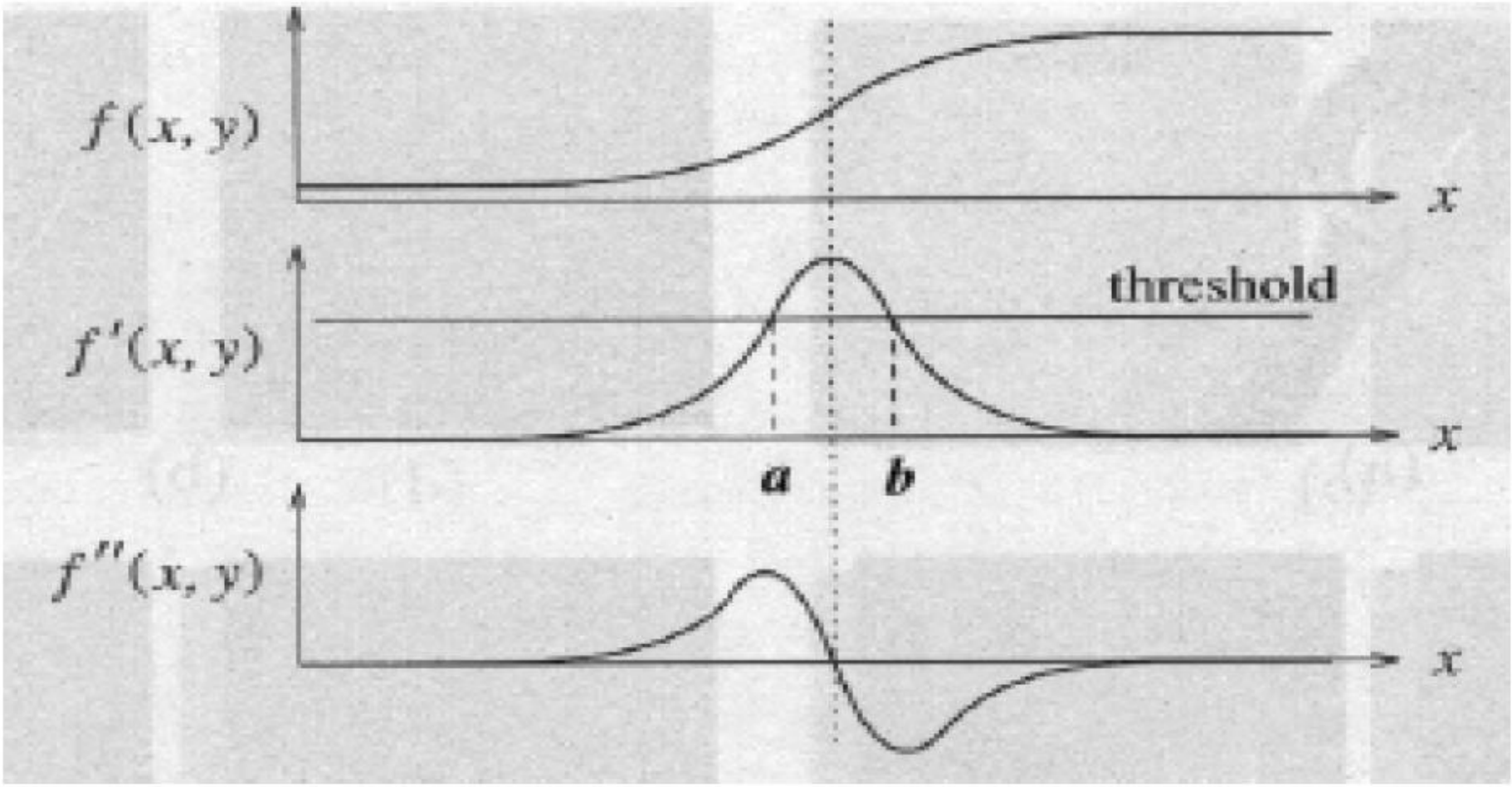
2. Edge Detection using the 1st Derivatives

3. Hysteresis Thresholding: It uses a high threshold t_h and a low threshold t_L. For pixels whose gradient lies between t_L and t_h, it is decided as an edge pixel if the pixel's neighbor pixels perpendicular to the edge gradient have been decided as edge pixels. It helps remove tiny noisy edges.



2. Edge Detection using the 2nd Derivatives

Edge points can also be detected by finding the zero-crossings of the second derivative.



2. Edge Detection using the 2nd Derivatives

So finding the maxima/minima in gradient magnitude can be approximated by finding the point that satisfies:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} = 0$$

Note $\nabla^2 f(x,y)$ is actually *Laplacian* operator.

May not always find discrete pixels where the second derivative is zero – look for zero-crossing instead.

{+,-}, {+,0,-}, {-,+}, {-,0,+}

Slope (or strength) of zero-crossing {a, -b} is: |a+b|. To detect "strong" zero-crossing, threshold the slope.

2. Edge Detection using the 2nd Derivatives

Laplacian operator can be approximated by a convolution mask

$$\frac{\partial^2 f}{\partial y^2} = f(i,j+1) - 2f(i,j) + f(i,j-1)$$

$$\frac{\partial^2 f}{\partial x^2} = f(i+1,j) - 2f(i,j) + f(i-1,j)$$

$$\nabla^2 f = f(i,j+1) + f(i,+1\,j) - 4f(i,j) + f(i,j-1) + f(i-1,j)$$

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Laplacian operator has different variations like:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{pmatrix}$$

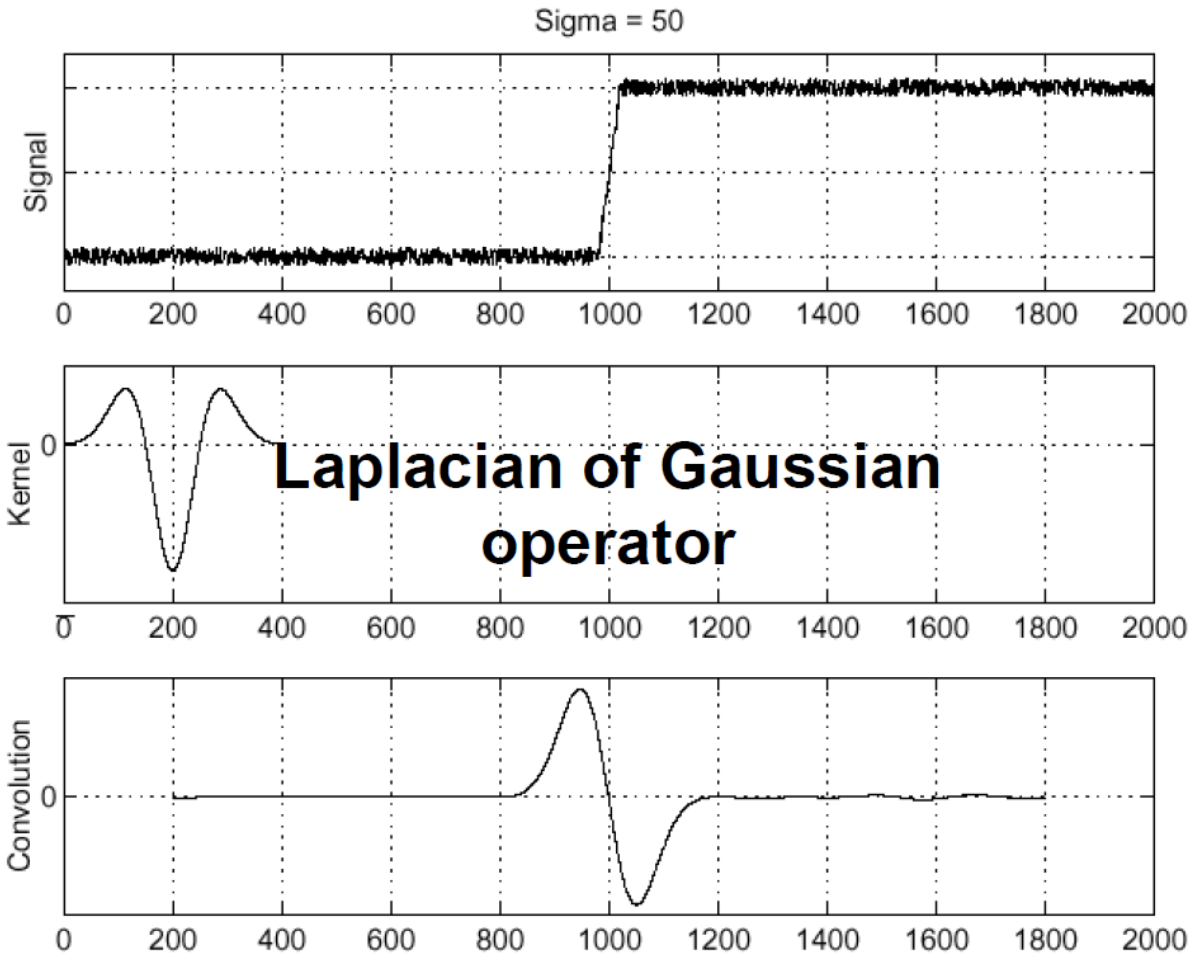
2. Edge Detection using the 2nd Derivatives

Similar to edge detection with 1st derivative, edge detection with 2nd derivative can be performed with Gaussian to suppress noises.

f

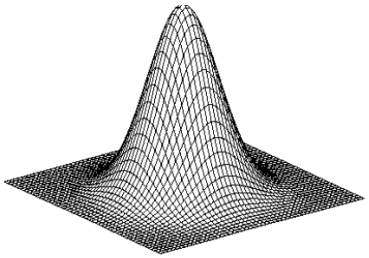
$\frac{\partial^2}{\partial x^2} h$

$(\frac{\partial^2}{\partial x^2} h) \star f$



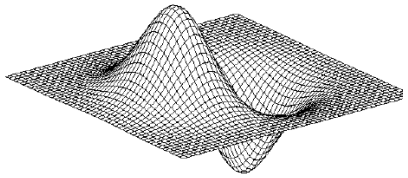
2. Edge Detection using the 2nd Derivatives

Laplacian of Gaussian smooths images first to suppresses noises before performing convolution. It is often called LoG edge detector.



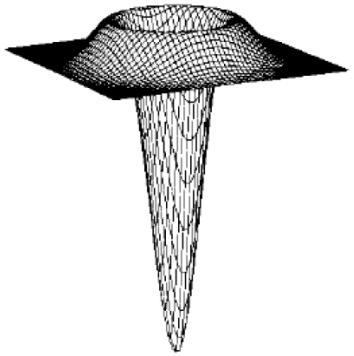
Gaussian

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



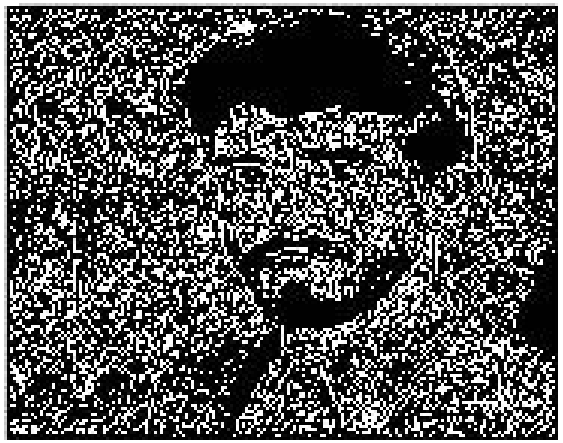
derivative of Gaussian

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



Laplacian of Gaussian

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



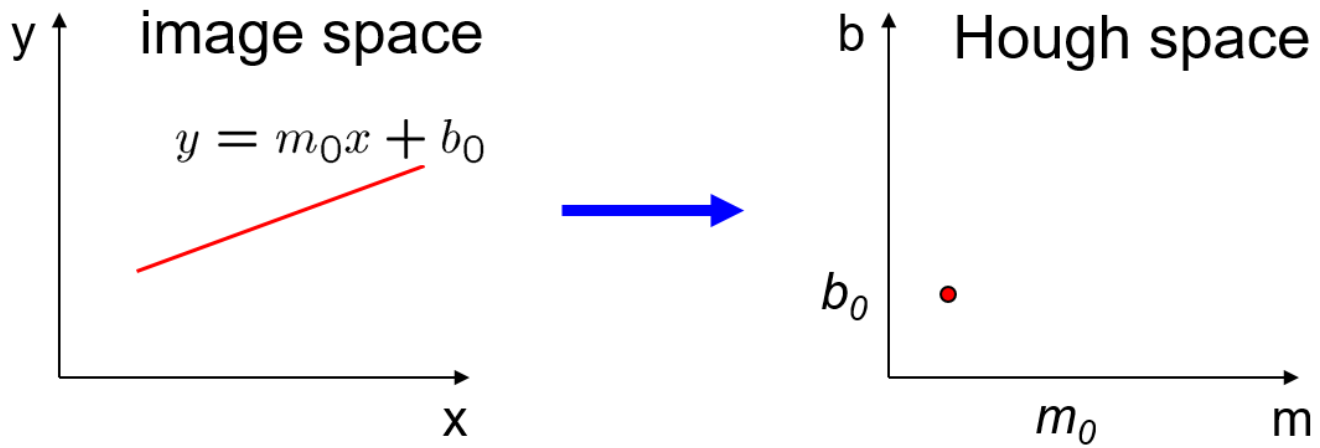
3. Line Detection

Edge detection often gives disconnected edge segments due to noises, occlusions, etc. Lines can be detected by applying *Hough Transformation* to the edge pixels detected by edge detectors.

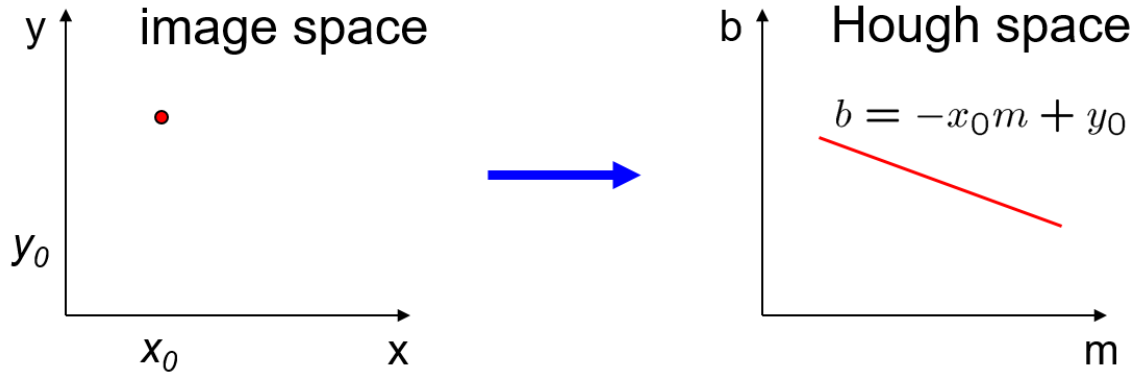
3. Line Detection

Connection between image (x, y) and Hough (m, b) spaces:

A line in the image space corresponds to a point in Hough space.

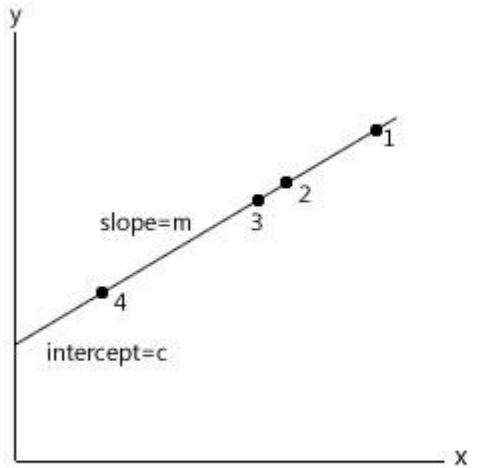


A point in the image space corresponds to a line in Hough space.

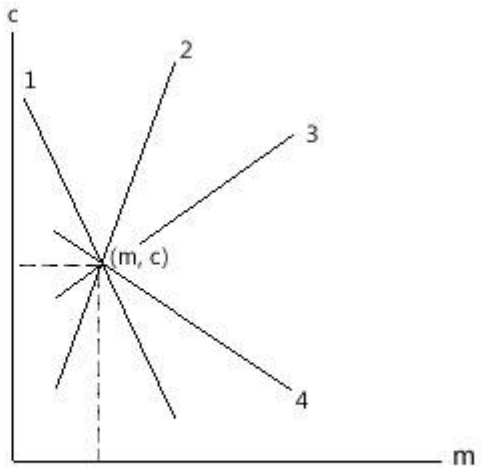


3. Line Detection

Each edge pixel gives a line in the *mc* place. Lines (in *mc*) of edge pixels on the same line (in image) intersects giving the parameters of the line (in image).



xy space



mc space

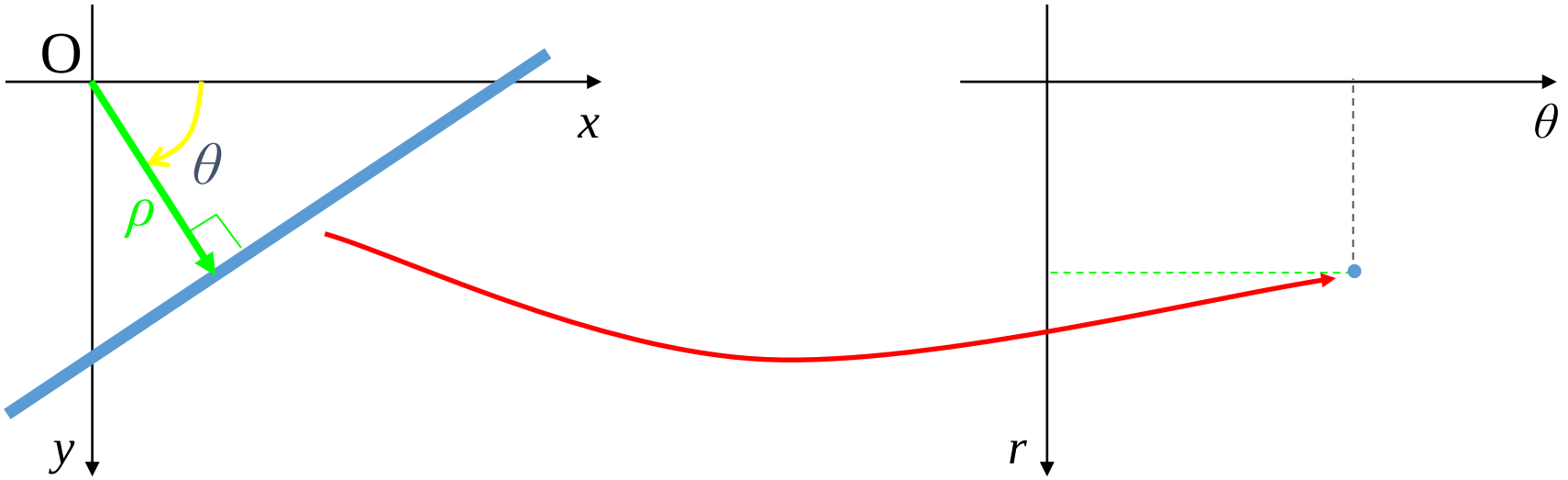
The above Hough transform has an flaw. The value of m (slope) tends to be infinity for vertical lines. So need infinite memory to store the mc space.

3. Line Detection

A normal-form parametrization instead of slope-intercept form of lines is widely adopted in Hough Transformation.

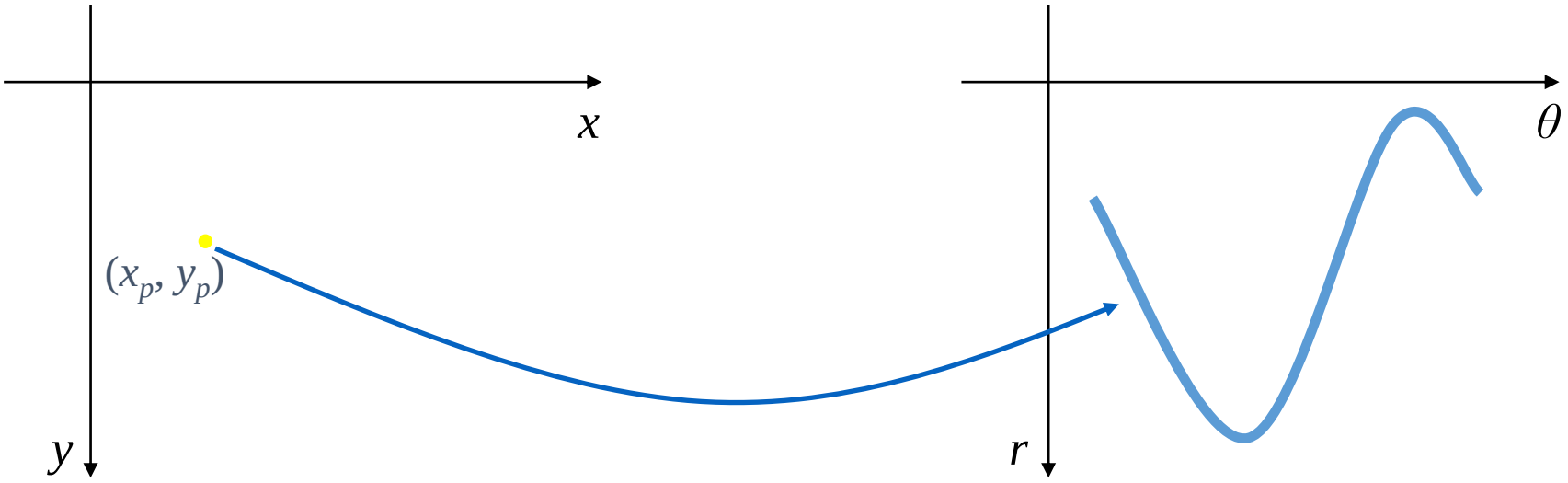
$$r = x \cdot \cos\theta + y \cdot \sin\theta$$

In the normal-form space, each line in the image space corresponds to a point in the normal-form space.



3. Line Detection

Similarly, every line passing through a point (x_p, y_p) in the image space produces a point $(r\ \vartheta)$ in the normal-form space. The point (x_p, y_p) thus corresponds to a sine-like curve in the normal-form space.



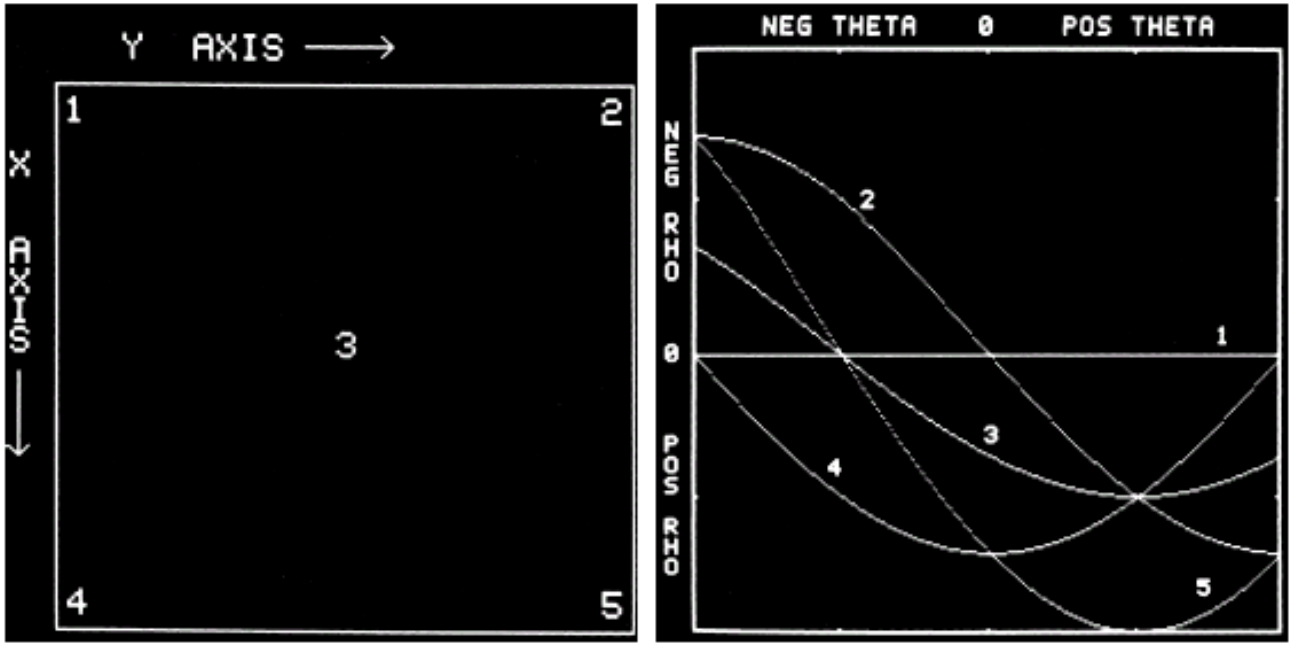
$$r = x \cdot \cos\theta + y \cdot \sin\theta$$

3. Line Detection

- Take all edge points in the image and transform them into sinusoidal curves in parameter space
- Edge points having a common point in parameter space lie on a common straight line

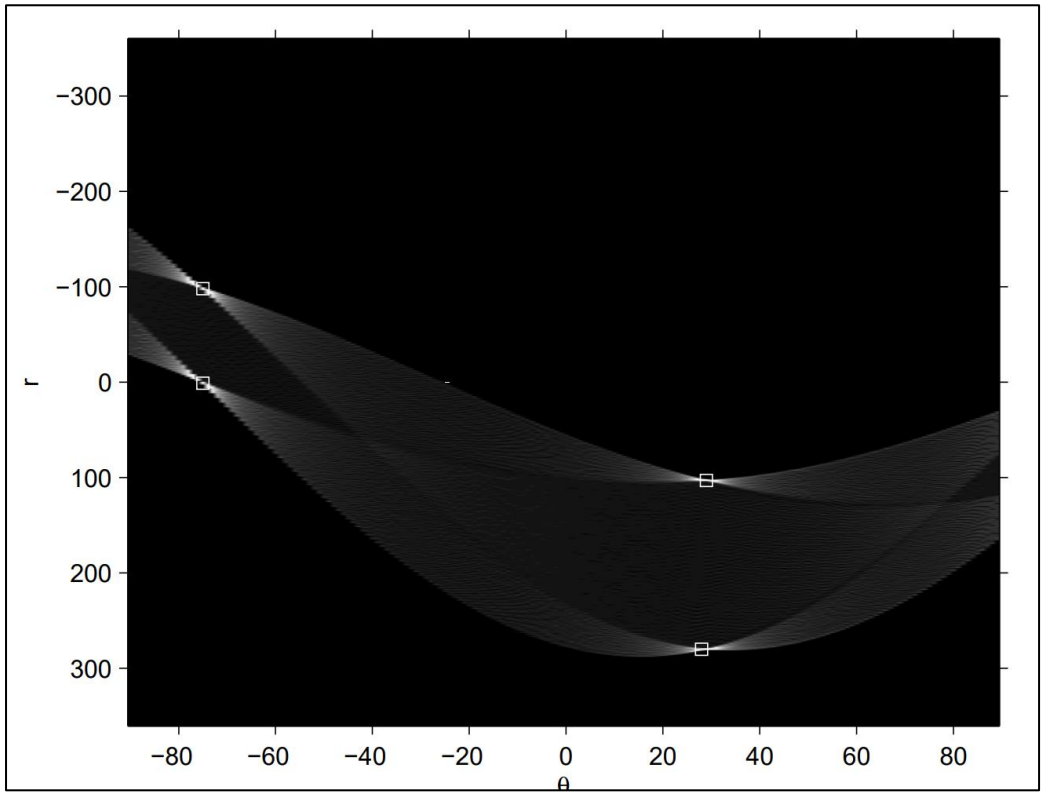
a b
c d

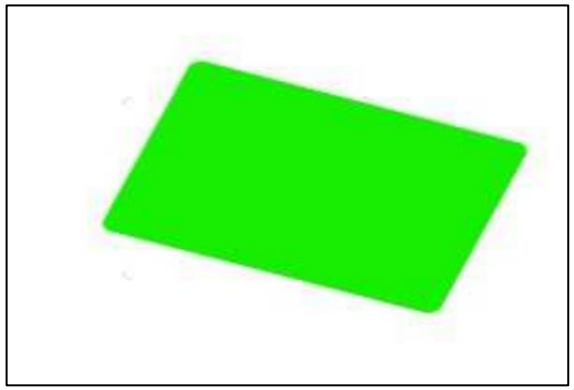
FIGURE 10.20
Illustration of the Hough transform. (Courtesy of Mr. D. R. Cate, Texas Instruments, Inc.)

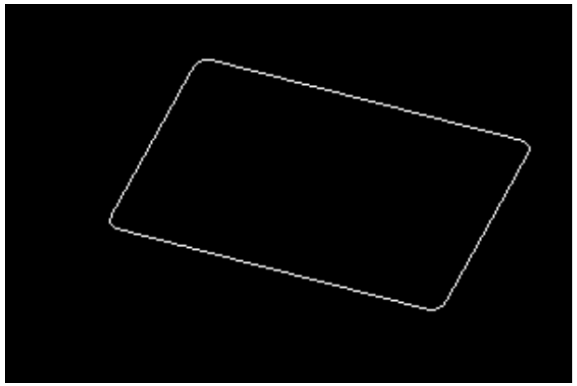


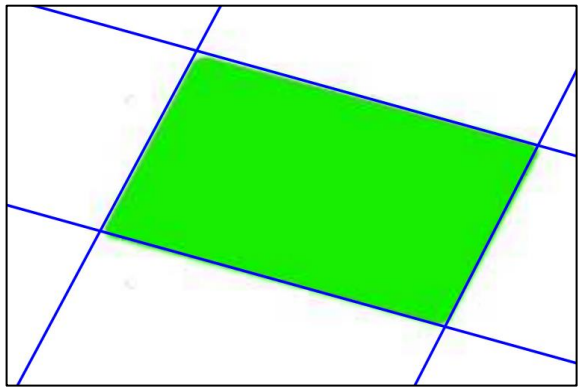
3. Line Detection

Another example: Given a image with a parallelogram, edge pixels can be detected in the image space. Every edge pixel maps to a sine-like curve in the r-ϑ space. The intersect peak gives the parameters of the lines in the image space.





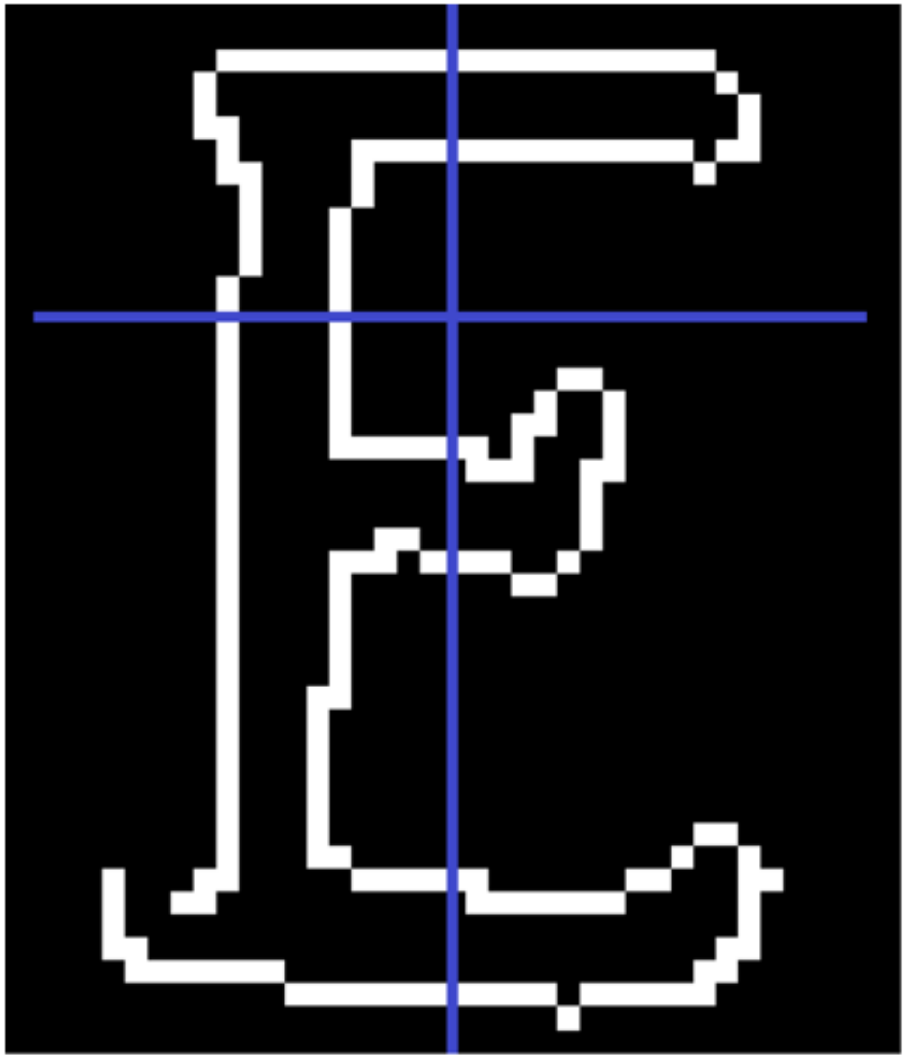


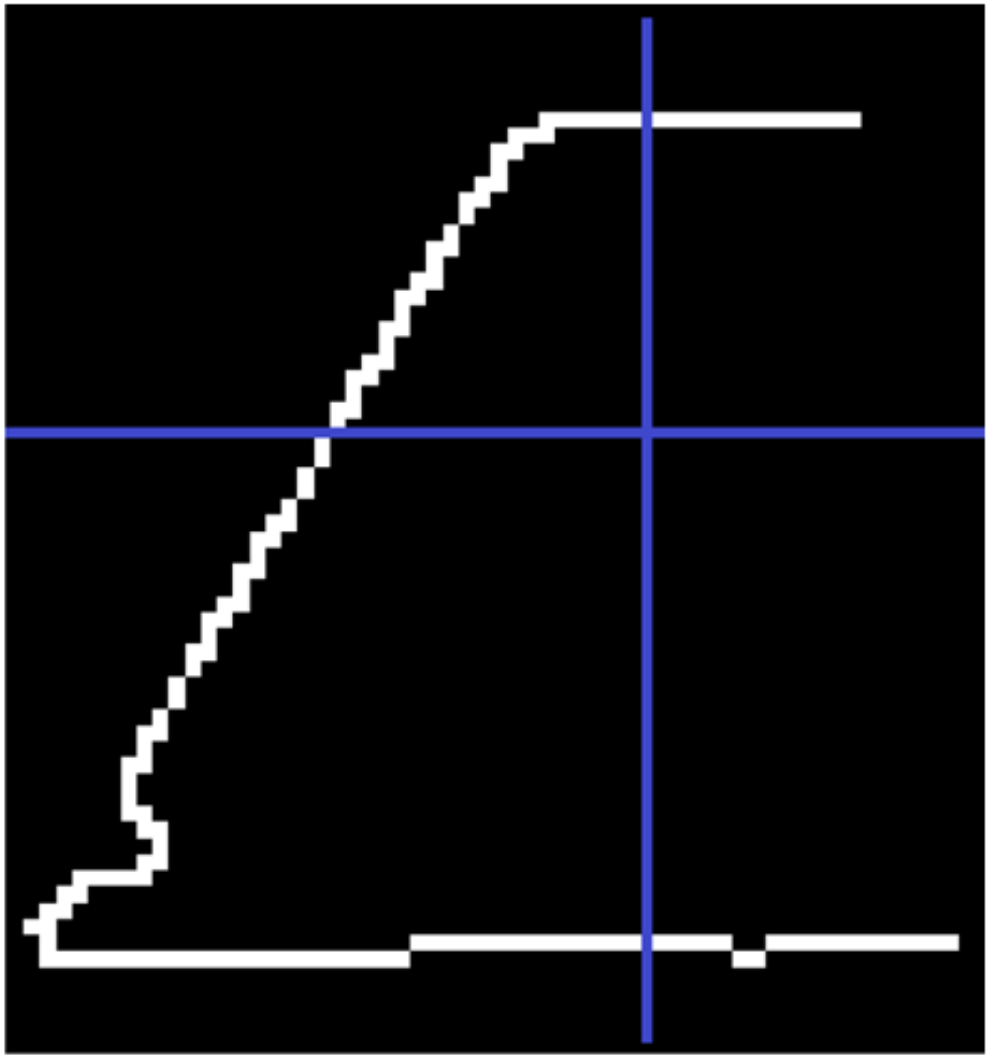


4. Applications – Scene Text Detection

Shijian Lu · Tao Chen · Shangxuan Tian, Joo-Hwee Lim · Chew-Lim Tan, Scene text extraction based on edges and support vector regression, IJDAR, 2016.

4. Applications – Scene Text Detection



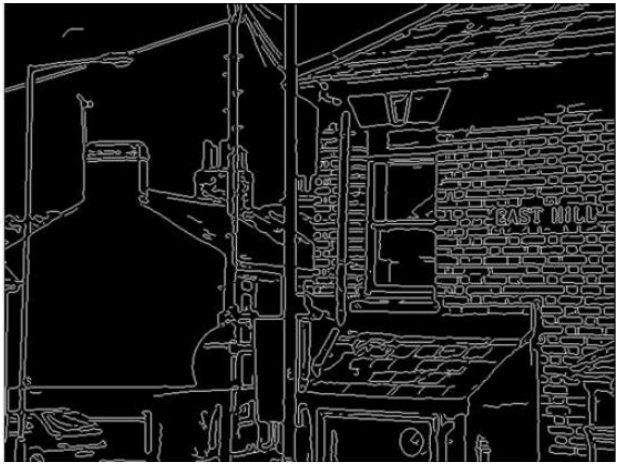


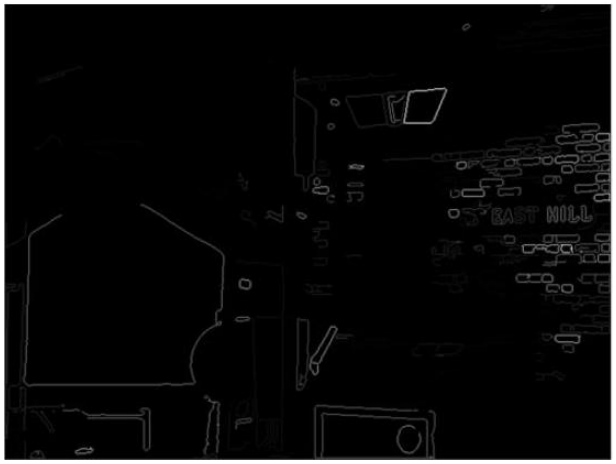
$$F_2 = R^{\left(\sum_{i=1}^{H} f(cn_i) + \sum_{j=1}^{W} f(cn_j)\right) / (W+H)}$$

$$F_3 = \frac{\sum_{i=1}^{H} g(cn_i) + \sum_{j=1}^{W} g(cn_j)}{W+H}$$

$$S_{i,j} = \prod F_{i,j,k}$$

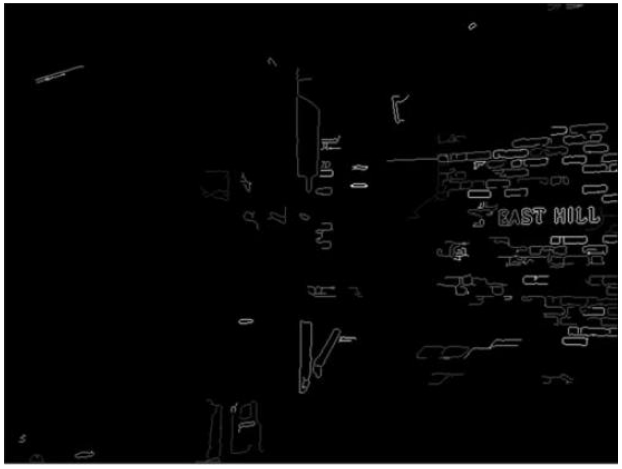
4. Applications – Scene Text Detection

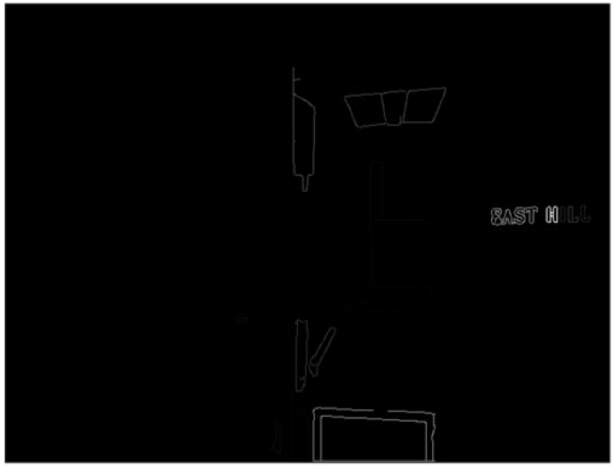




Canny edge detection

$$F_1 = \frac{\mu(G_e)}{\sigma(G_e) + \xi}$$



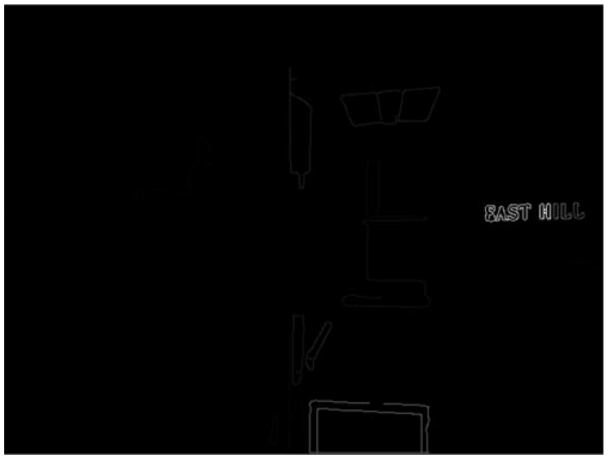


$$F_2 = R^{\left(\sum_{i=1}^{H} f(cn_i) + \sum_{j=1}^{W} f(cn_j)\right) / (W+H)}$$

$$F_3 = \frac{\sum_{i=1}^{H} g(cn_i) + \sum_{j=1}^{W} g(cn_j)}{W + H}$$

$$S_{i,j} = \prod F_{i,j,k}$$

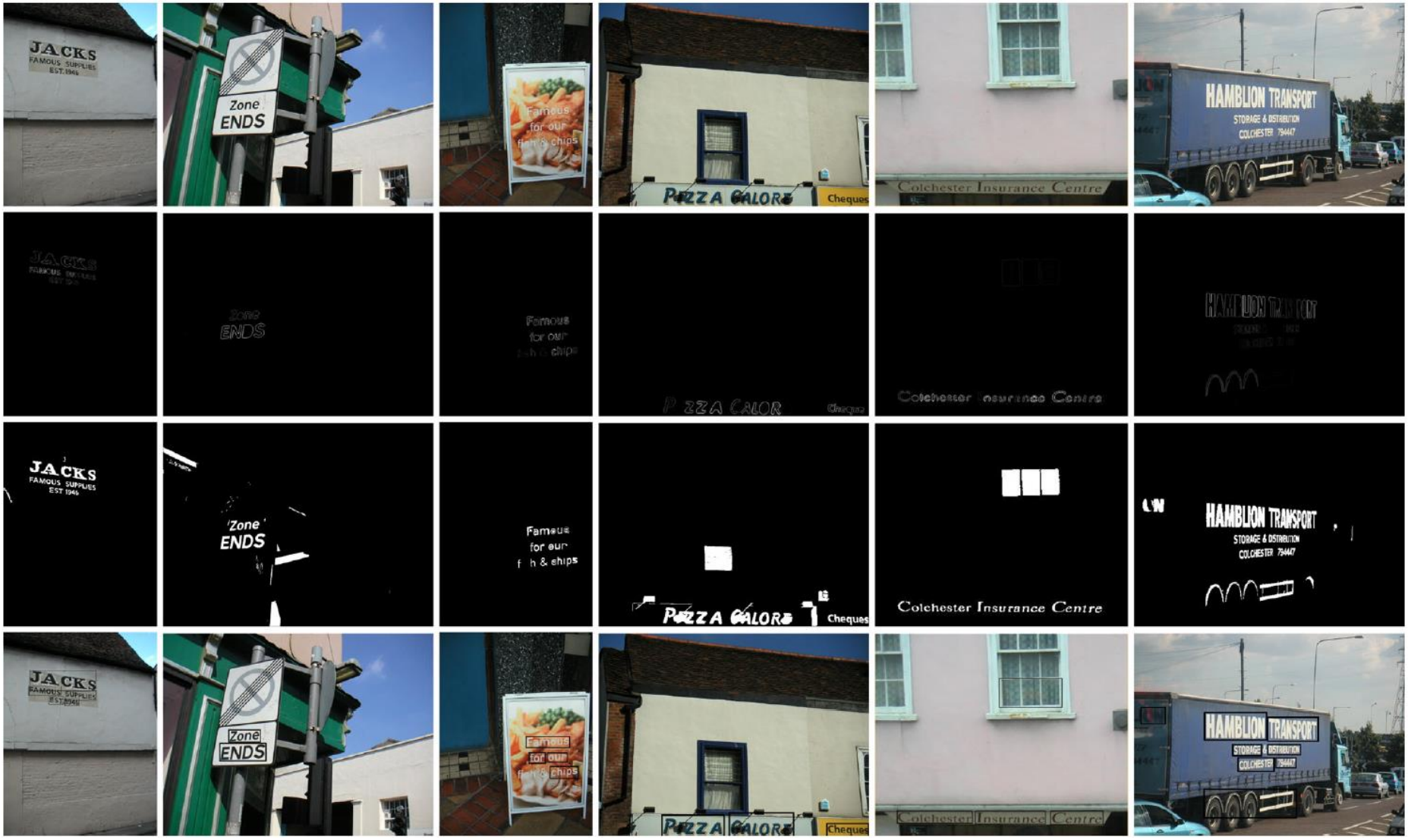
2. Edge Detection Applications



$$M = \frac{1}{C \times S} \sum_{i=1}^{S} \sum_{j=1}^{C} S_{i,j}$$

$$M = \frac{1}{C \times S} \sum_{i=1}^{S} \sum_{j=1}^{C} S_{i,j}$$

4. Applications – Scene Text Detection



4. Applications – Autonomous Driving

Line detection is very useful in various image/video analytics tasks.

Summary

1. Why detect edges and lines
2. How to detect edges
3. How to detect lines
4. Any applications