Robot Vision

TTK4255

Lecture 06 – Image Processing

Annette Stahl

(Annette.Stahl@ntnu.no)

Department of Engineering Cybernetics – ITK

NTNU, Trondheim

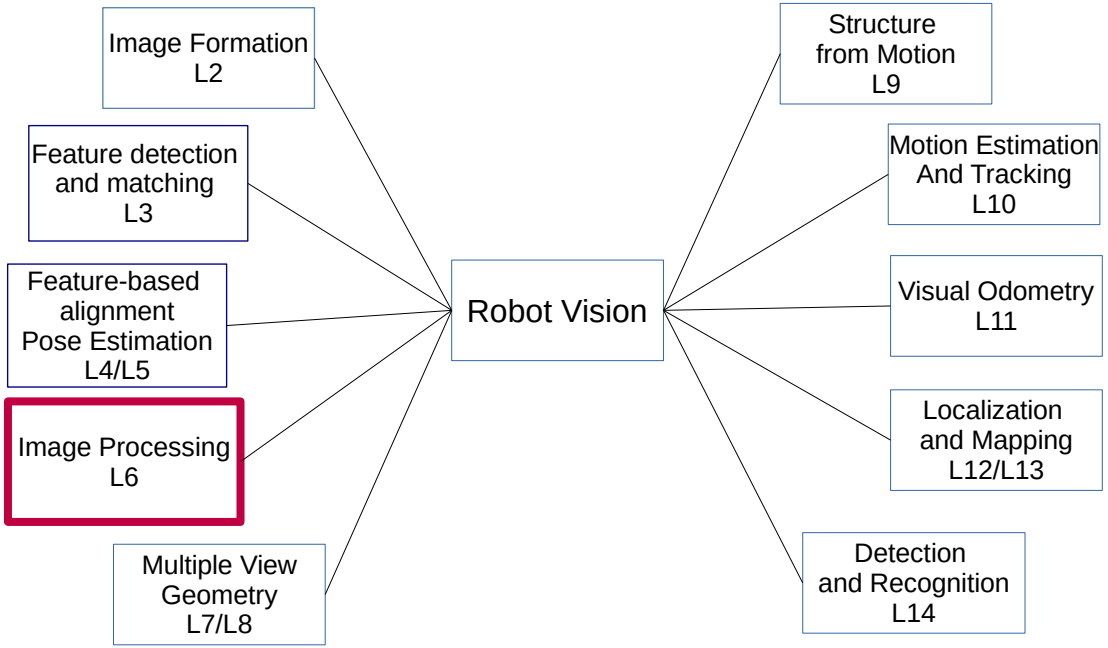
Spring Semester

10. Februar 2020

Lecture 06 – Image Processing

Annette Stahl (Annette.Stahl@ntnu.no)

Simen Haugo (Simen.Haugo@ntnu.no)



Outline of the fifth lecture:

- Image Enhancement
- Thresholding
- Convolution and Correlation
- Filtering
- Edge Detection
- Line Detection
- Image Pyramids

Recap L05

Non-linear Pose Estimation

- Gauss-Newton
- Levenberg-Marquardt

Image Enhancement

Aims:

- Make the image visually more appealing
- Removal of Noise
- Highlight details of interest

Two main categories of techniques

Spatial domain:

- Direct manipulation of pixels, with or without considering neighbor-pixels.

Transform domain:

- Involves the transformation of the image into a different representation. (→ Fourier transform, Wavelet transform)

Note: The spatial domain refers simply to the plane containing the pixels of an image.

We start with operations in the spatial domain!

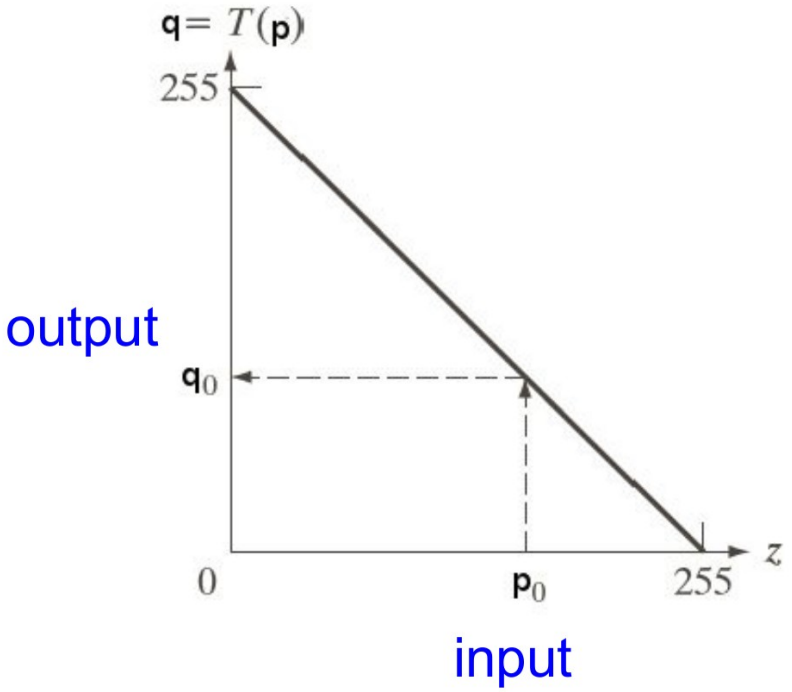
Intensity Transformation

- The intensity transformation function is a mapping from an input gray-value p_0 to its corresponding output value q_0

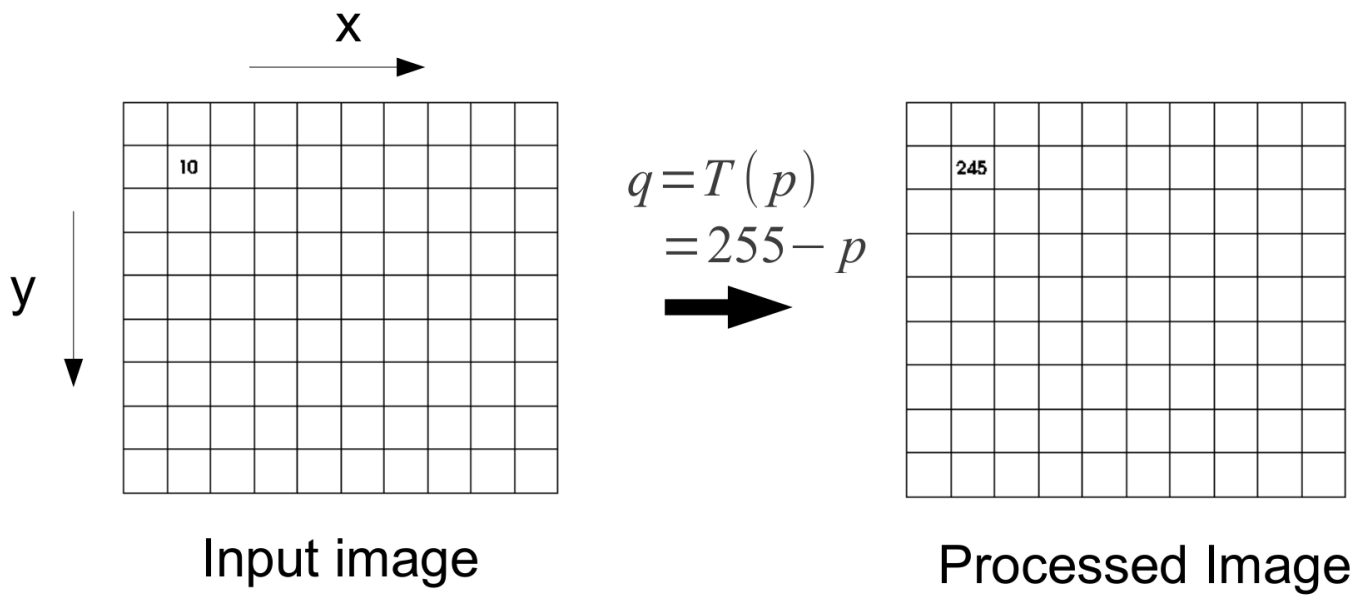
Example:
Negative Image

$$q = T(p) = 255 - p$$

Mapping-function T(p):



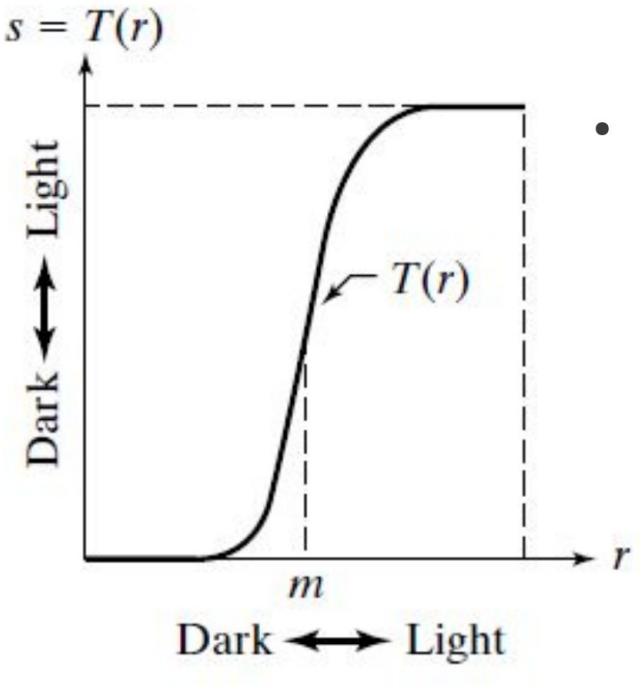
Intensity Transformation



$$g(x, y) = T[f(x, y)]$$

- Scan through all pixels (x,y) in the original image f
  - Compute the transformation $q = T(p)$ for current pixel
  - Set the resulting pixel-value q in result image g at (x,y)

Intensity Transformation



- Such a sigmoid shaped function results in a contrast stretching by
  - darkening levels below m
  - brightening levels above m

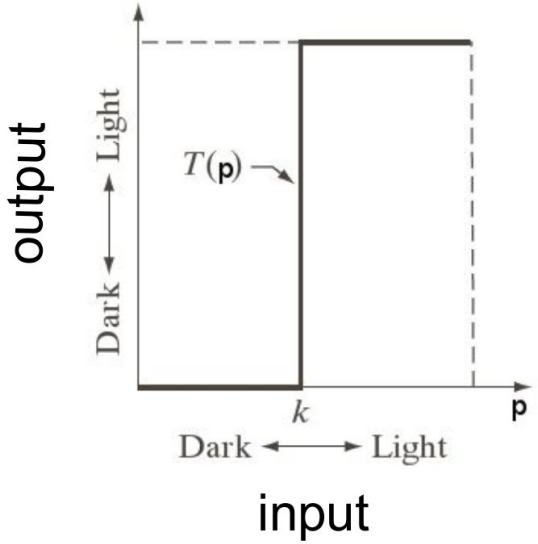
(Sigmoid shaped = shaped like a “S”)

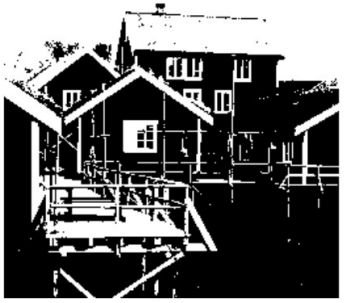
Binary Image

- The following intensity transformation

$$T(p,k)=\begin{cases} 0 & for\ p<k \\ 255 & for\ p\geq k \end{cases}$$

sets a threshold k and results (depending on input p) in in a black (=0) or white (=255) gray-value.





→ Extreme case to higher contrast!

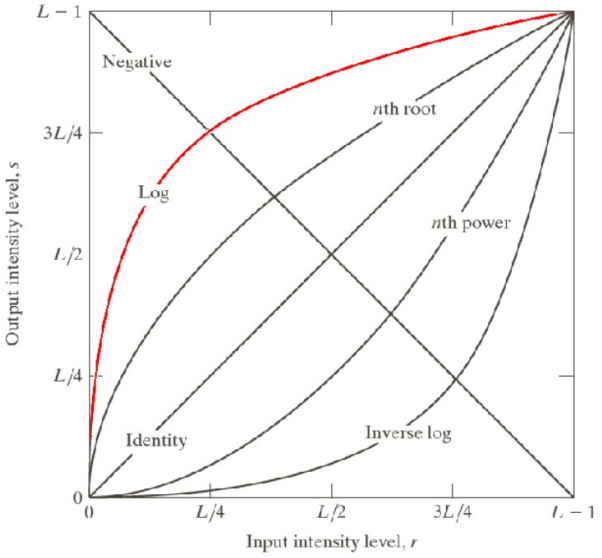
Logarithmic Transformations

General form: $q = T(p) = c \log(1 + p)$

$c = const. \qquad p \geq 0$

- Very low intensities widened
- Bright intensities squeezed

→ Compresses the dynamic range of “images” with large variations in pixel values.



Note: The **inverse log** widens the bright intensities and squeezes the darker intensities.

Gamma Transformation

Basic form:

$$q = c\, p^{\gamma}$$

with

$c > 0$

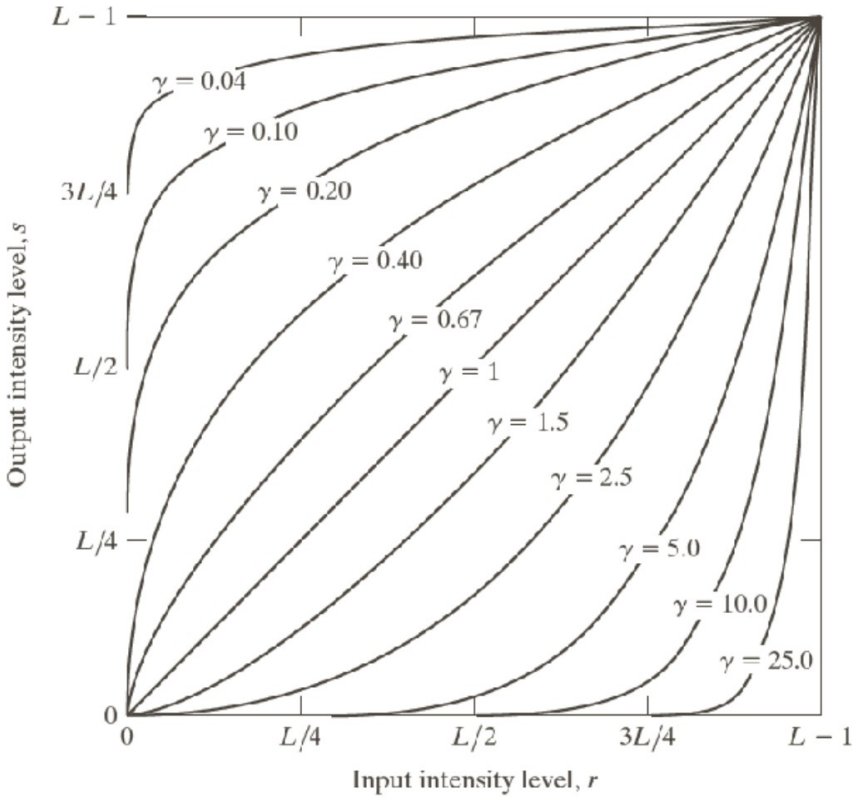
$\gamma > 0$

For $\gamma < 1$ (→ brighter)

- Low intensities widened
- Bright intensities squeezed

For $\gamma > 1$ (→ darker)

- Bright intensities widened
- Low intensities squeezed

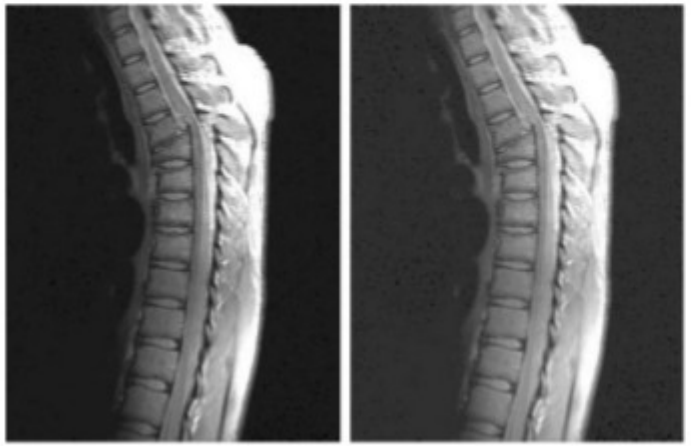


Only one parameter involved!

Gamma correction

- MRI

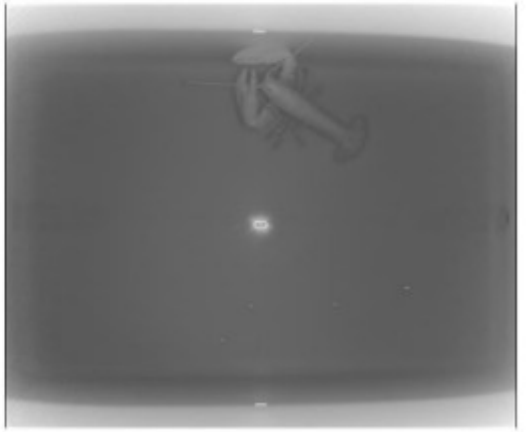
original image $\gamma = 0.6$



$\gamma = 0.4$ $\gamma = 0.3$

- Infrared image

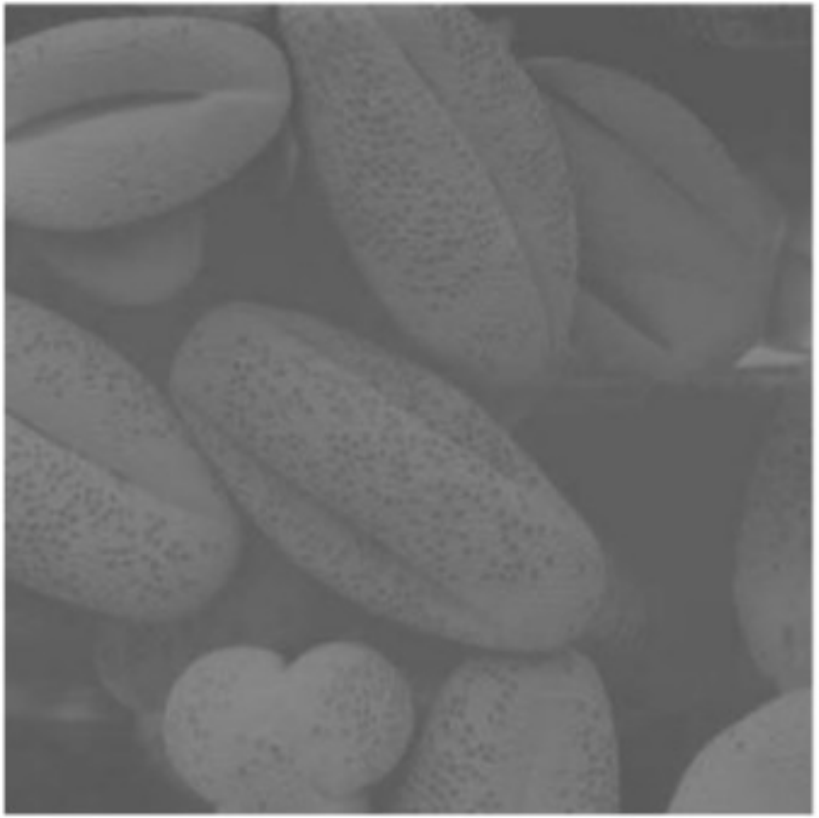
original image



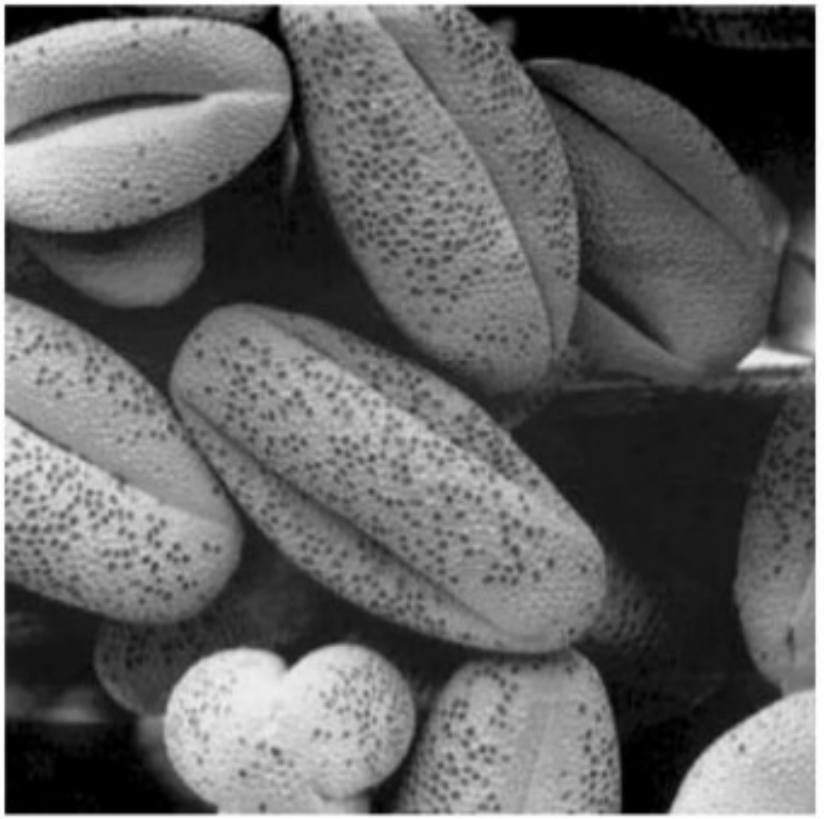
$\gamma = 0.32$

Contrast Stretching

Contrast stretching performed using a piecewise linear transform (user-defined).



low-contrast image.



result of contrast stretching

Histogram

- The **histogram** h(k) of an image provides the information about the distribution of intensity-levels k

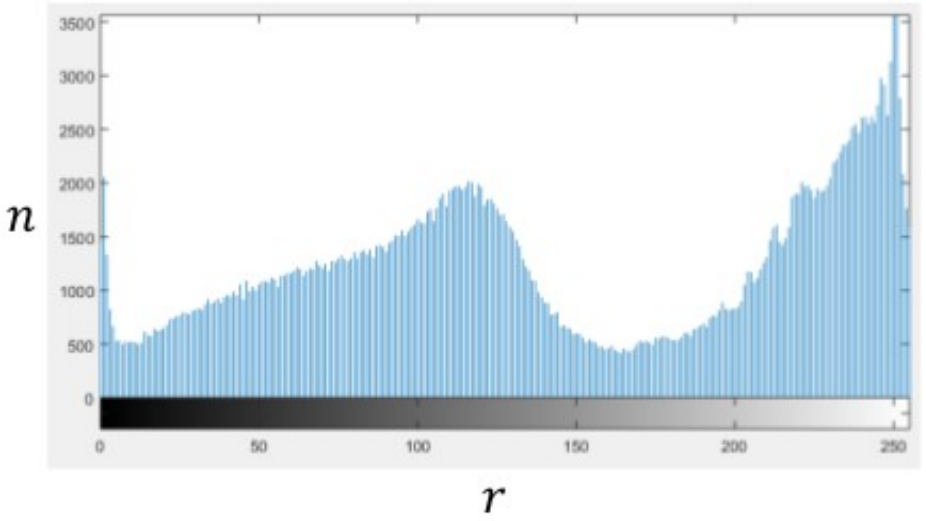
$$h(k) = \sum_{(ij)\in\Omega} \delta(I_{ij} - k) \qquad \delta(x) = \begin{cases} 1 & for\ x = 0 \\ 0 & otherwise \end{cases}$$

→ Count the numbers of pixel with gray-level k

Ω : considered region

- **global**: Histogram for the whole image
  local: Histogram over a smaller area

Image Histogram



- histogram:

$$h(\mathrm{r}) = \mathrm{n}$$

$r \in [0, \mathrm{L}-1]$ intensity value in image

n is the number of pixels in the image with intensity r

- normalized histogram:

$$p(\mathrm{r}) = \frac{n}{MN}$$

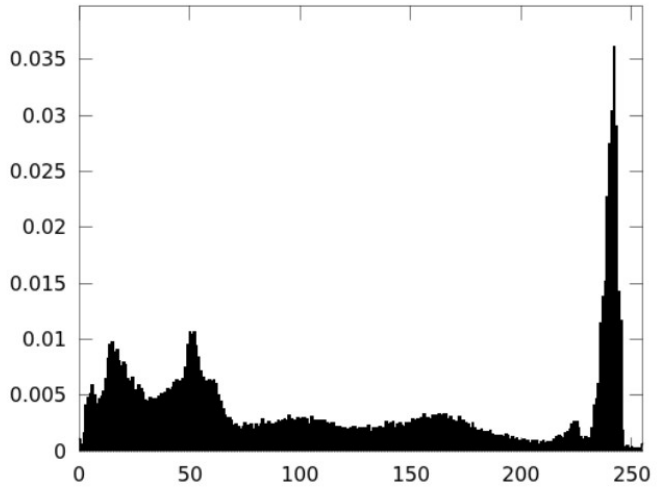
M and *N* are the row and column dimensions of the image

Normalized Histogram

- It is also common to normalize the histogram
  (→ dividing by the total number of considered pixels)

$$p(k) = \frac{h(k)}{|\Omega|} = \frac{h(k)}{MN}$$ ($|\Omega| = NM$ for a NxM image!)

→ That can be seen as an estimate for the probability of the occurrence of intensity level k in the concerned image.

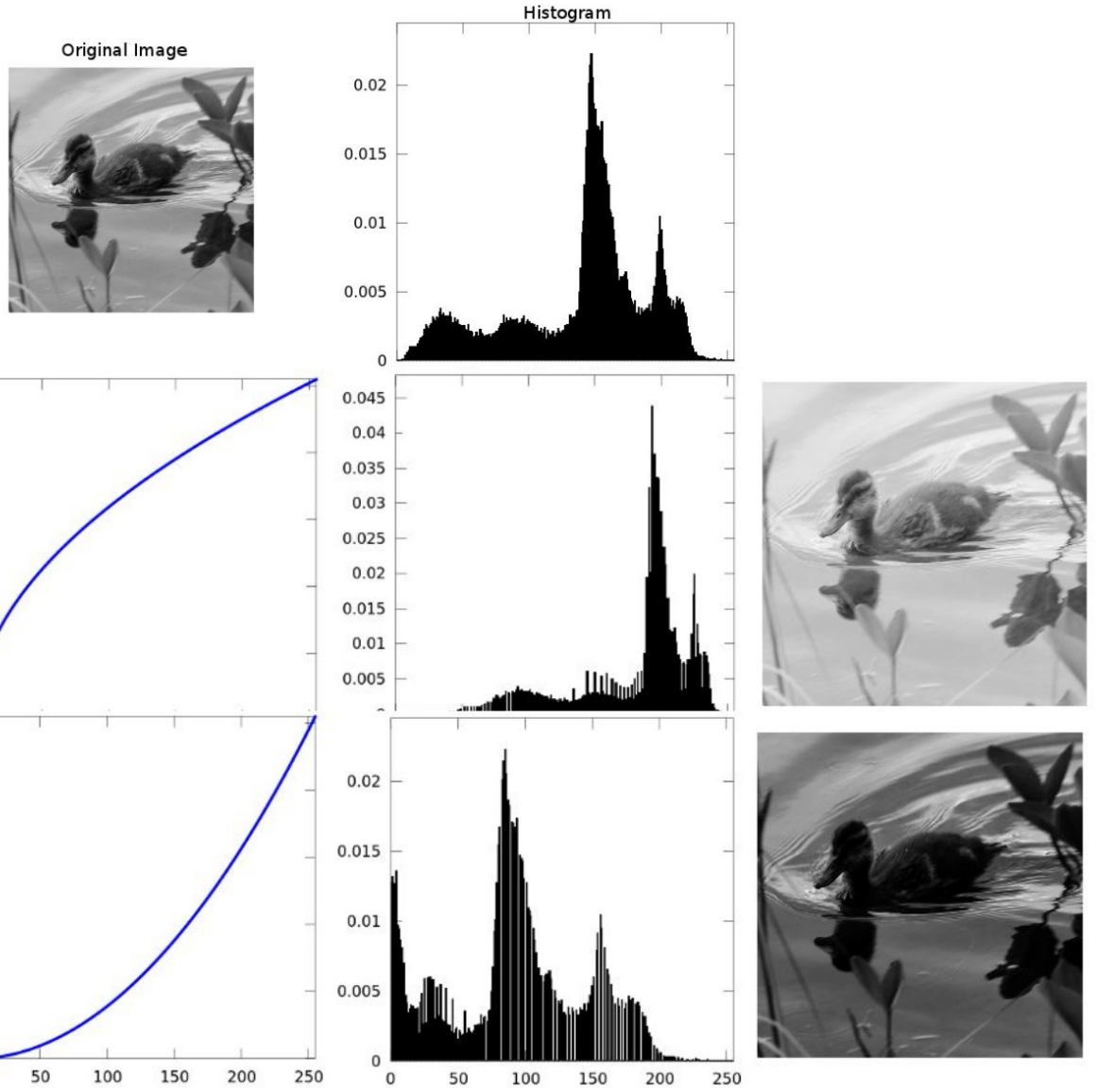


Histogram

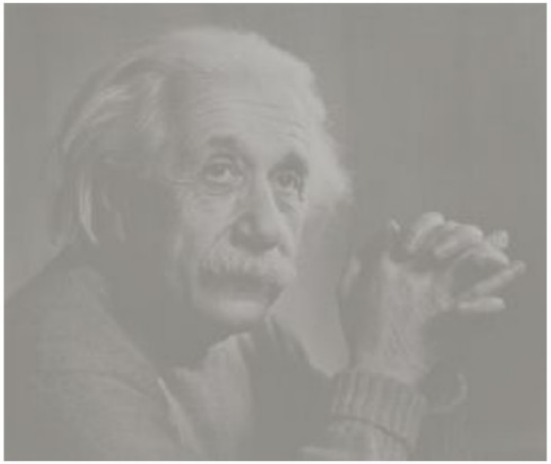
Gamma-transform

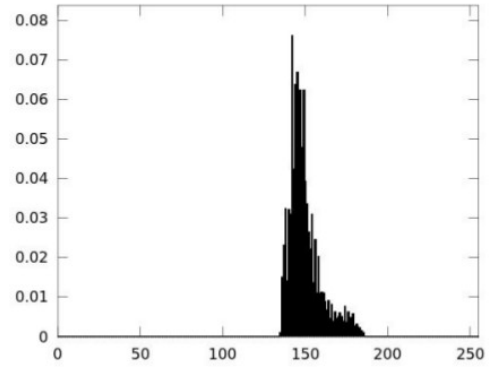
$\gamma < 1$

$\gamma > 1$

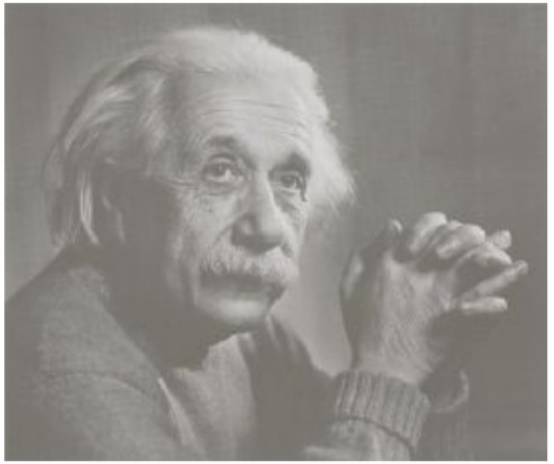


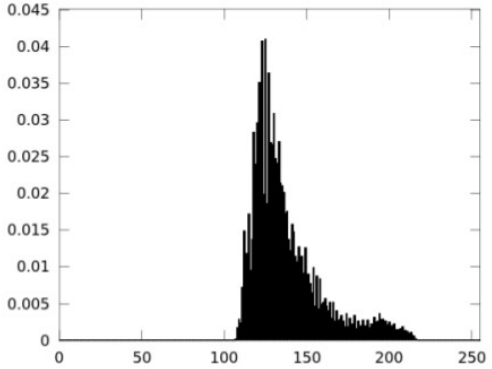
Histogram



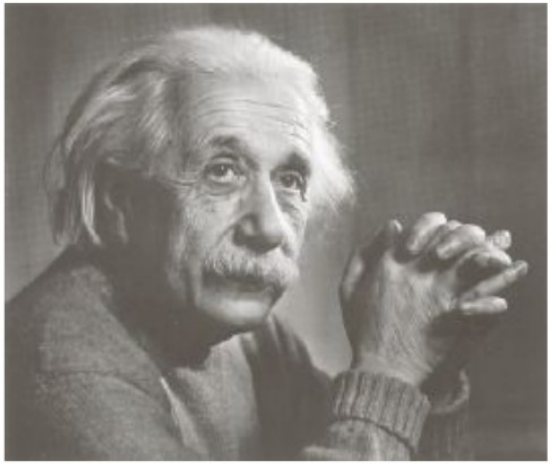


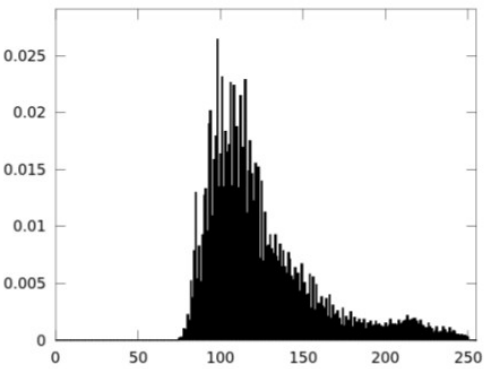
low-contrast





too bright

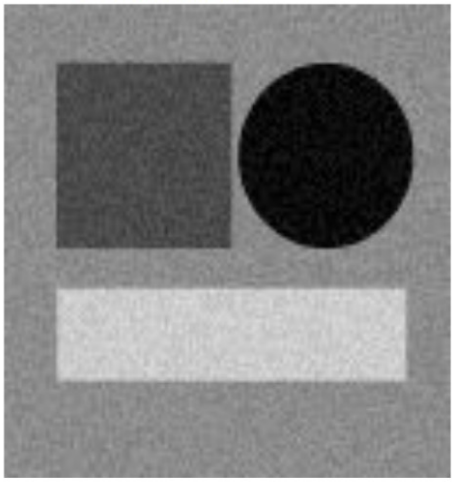


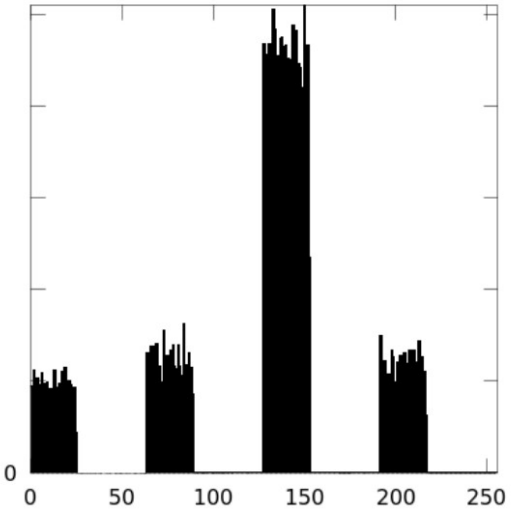


nearly OK

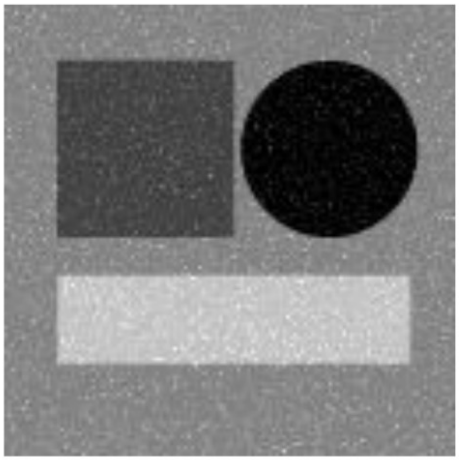
Noise - Histogram

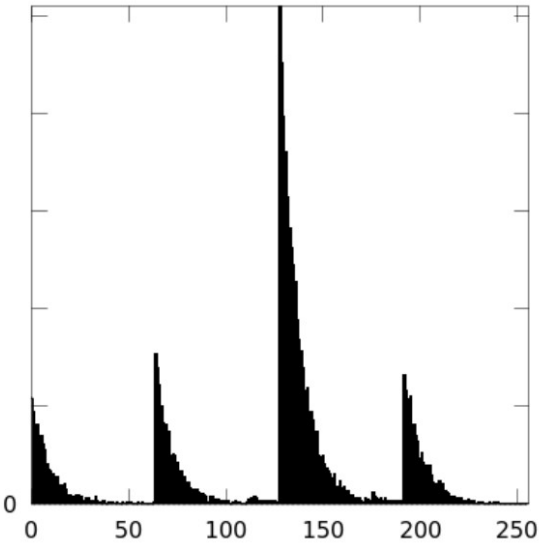
Uniform Noise





Exponential Noise





The histogram allows us to identify the type of noise that is present!

Note: Most often a different type of noise is observable!

→ Gaussian

Noise Models

Several Noise-Models:

- Normal (Gaussian) noise:

$$p(x,\sigma)=\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{-x^2}{2\sigma}}$$

  - Main type of noise (Amplifier noise) from an image sensor.
  - Independent at each pixel and independent of the signal intensity

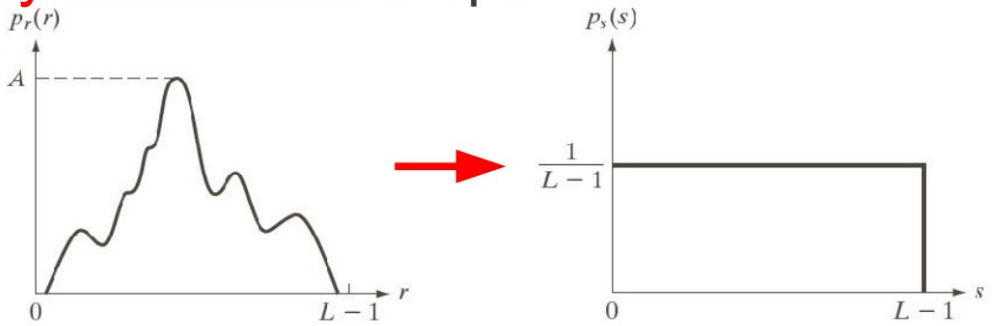
- Uniform noise:
  - Equally distributed over a given range [a,b]. $p(x)=\frac{c}{b-a}$

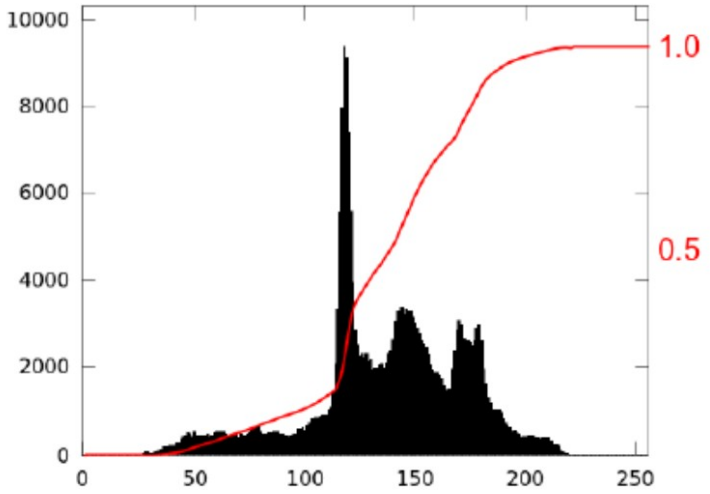
- Salt-and-pepper noise:
  - Caused by transmission errors, analog-to-digital converter errors, dead pixels.

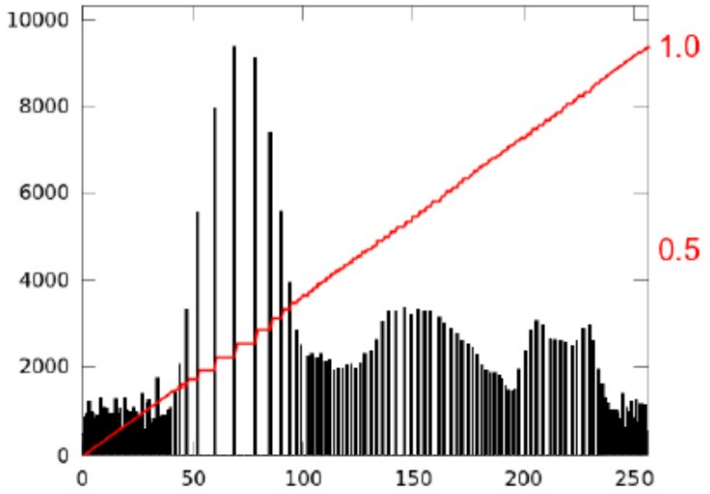
More noise: Exponential, Poisson, Speckle noise, Rayleight, Gamma ...

Histogram Equalization

- For a continuous density function (probability function) the cumulative probability distribution maps to a uniform distribution







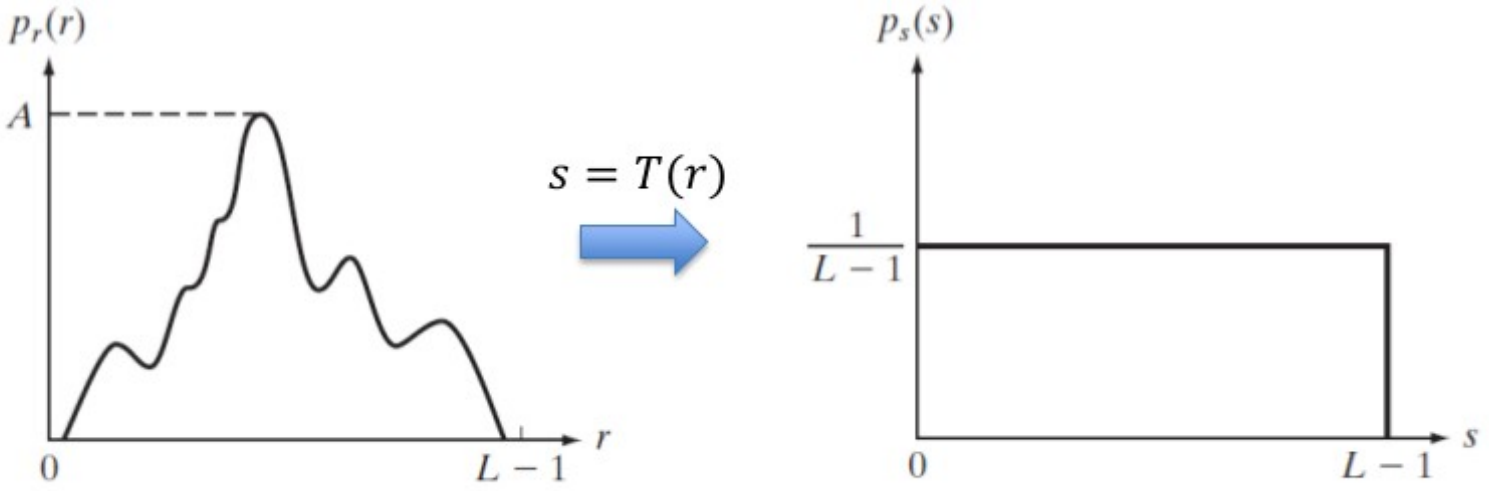
For discrete distributions (histograms) the resulting histogram is **not** flat, but more evenly distributed.

Histogram Equalization

Problem:

suppose the intensity values is continuous, $p(r)$ is continues PDF

want to find $s = T(r)$ that



Histogram Equalization

- For discrete values

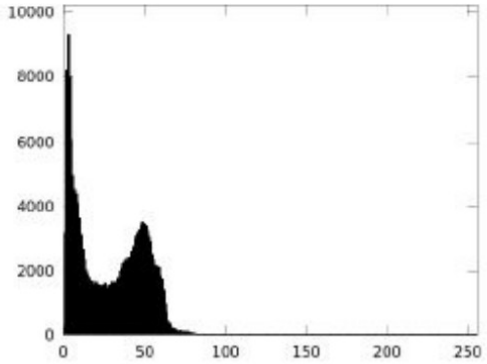
$$p_r(r_k) = \frac{n_k}{MN} \qquad k = 0, 1, 2, \ldots, L-1$$

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j)$$

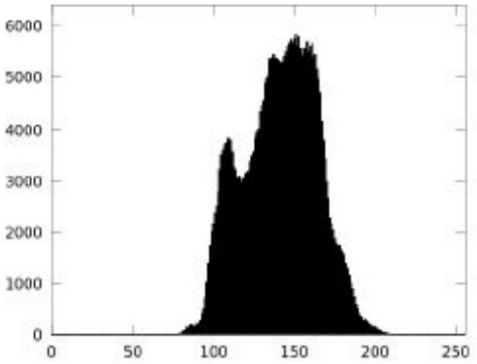
$$= \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j \qquad k = 0, 1, 2, \ldots, L-1$$

Histogram Equalization

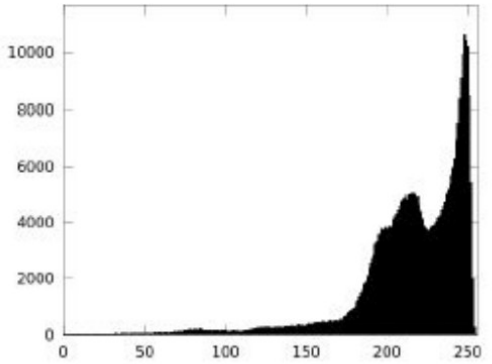
underexposed



low contrast

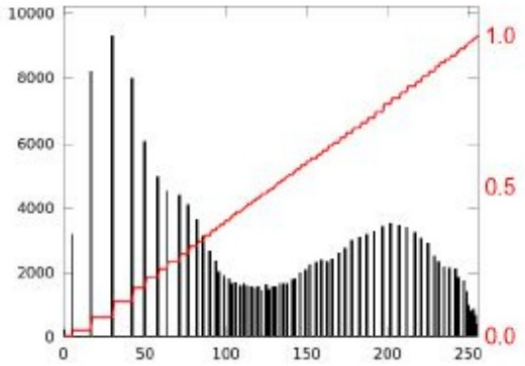


overexposed

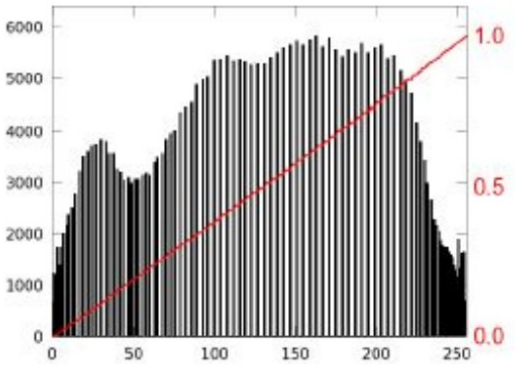


Histogram Equalization

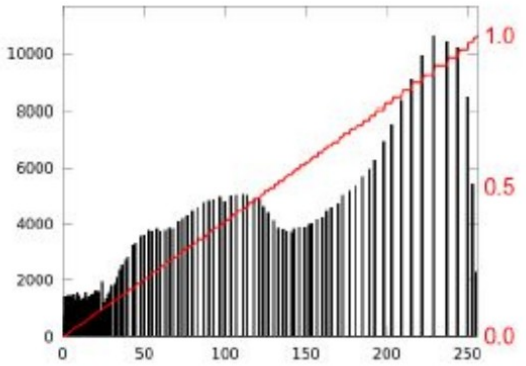
underexposed



low contrast



overexposed

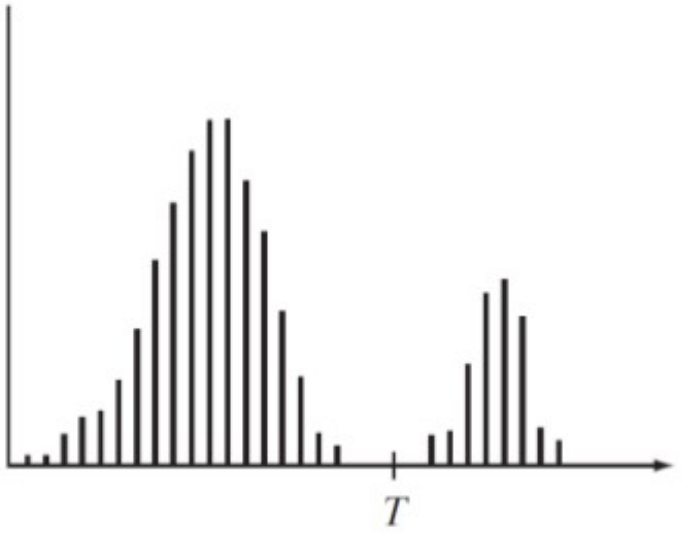


Thresholding

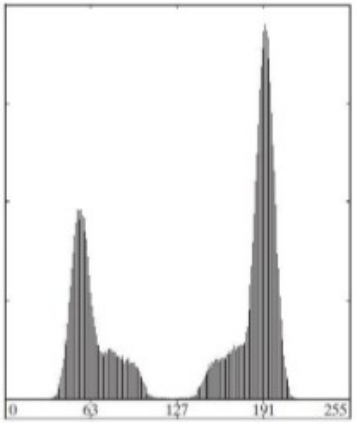
Turn gray image into binary image

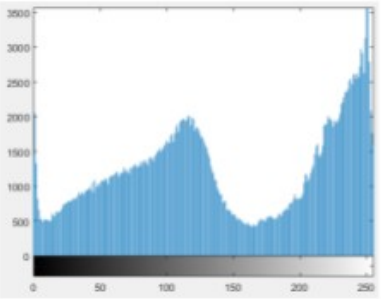
Application: segment out interesting region or objects

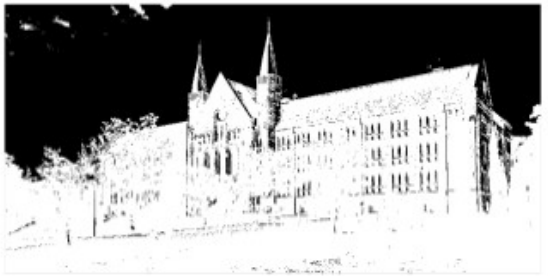
$$g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$



Thresholding







Otsu Method

Basic idea of Otsu's method

- Divide the histogram by a threshold k
- Find the threshold with **minimum weighted spread within both parts** (within-class variance)
  - Variance is weighted by the fraction of pixels in the class

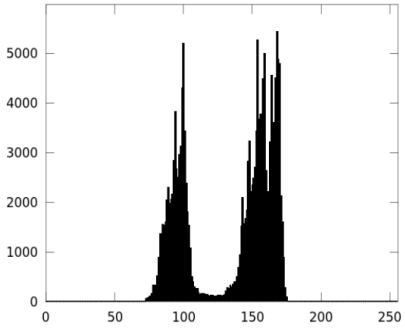
→ Same as **maximizing** the **between-class variance**

"Between-class variance"

"mean of the classes, global image mean"

$$\sigma_B^2(k) = P_1(k)[m_1(k) - m_G]^2 + P_2(k)[m_2(k) - m_G]^2$$

"fraction of pixels in the classes"



Otsu Method

Maximize Between-Class-Variance

$$\sigma_B^2(k) = P_1(k)[m_1(k) - m_G]^2 + P_2(k)[m_2(k) - m_G]^2$$

Normalized Histogram: $p_q = \frac{n_q}{n} \quad q = 0, 1, \ldots, L-1$

Fractions of pixels in classes: $P_1(k) = \sum_{i=0}^{k} p_i \qquad P_2(k) = 1 - P_1(k)$

Mean values: $m_G = \sum_{i=0}^{L-1} i\, p_i \qquad m_1(k) = \sum_{i=0}^{k} i\, p_i \qquad m_2(k) = \sum_{i=k+1}^{L-1} i\, p_i$

→ Optimum threshold:
Select the threshold with the largest value of $\sigma_B^2(k^*)$

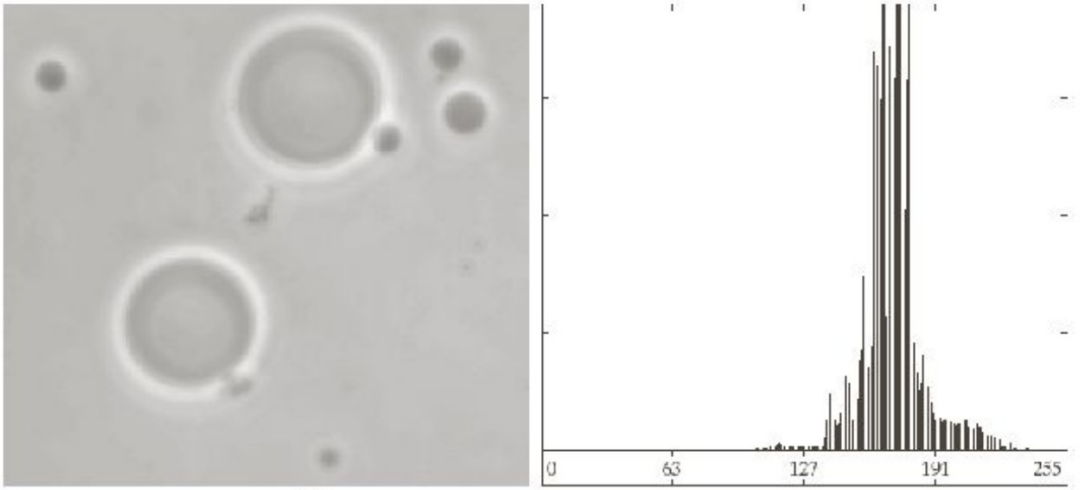
No parameters have to be adjusted !

Octave - Otsu Method

```
myimg=imread('myimage.png');

%Basic Global Thresholding
%(Source: Book from Gonzalez,Woods,Eddins)
T=mean2(myimg); done=false;
while ~done
  g=myimg>T;
  Tnext= 0.5*(mean(myimg(g))+mean(myimg(~g)));
  done= abs(T-Tnext)<0.5;
  T=Tnext;
end

tmpimg = im2bw(myimg,T/255);
img = uint8(tmpimg); figure(1); imshow(img);

%Build-in Otsu-thresholding
otsuT=graythresh(myimg)
tmpimg = im2bw(myimg,otsuT);
img = uint8(tmpimg); figure(2); imshow(img);
```

Otzu Method

Original



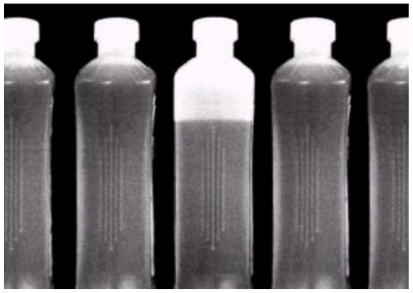
Basic global thresholding

Otsu's Method

Dual Threshold

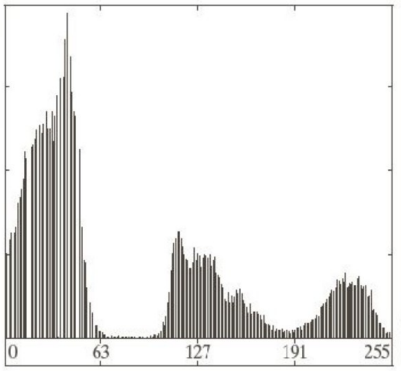
Using a Single Threshold

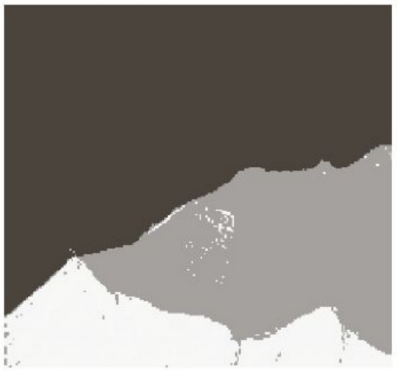
Is sometimes not the best choice ...



→ There is an extension for more thresholds in Otsu's method. (→ reasonable for up to 2 thresholds, 2d search)

Otsu dual Thresholds:



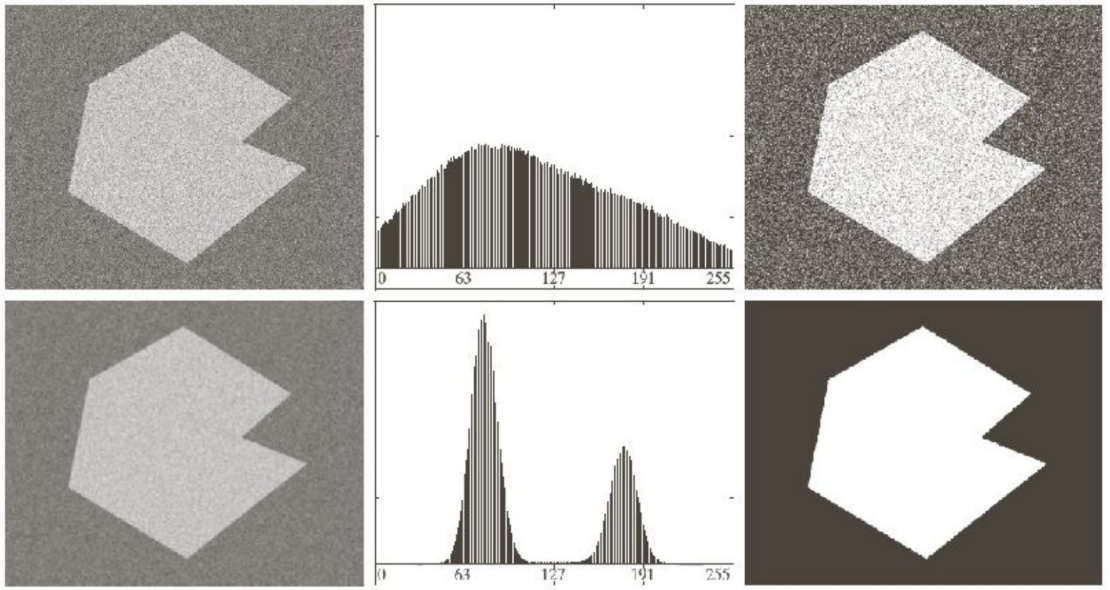


Threshold - Noise

Ways to Improve Global Thresholding

Noise can turn simple thresholding problems into hard ones

Noisy image

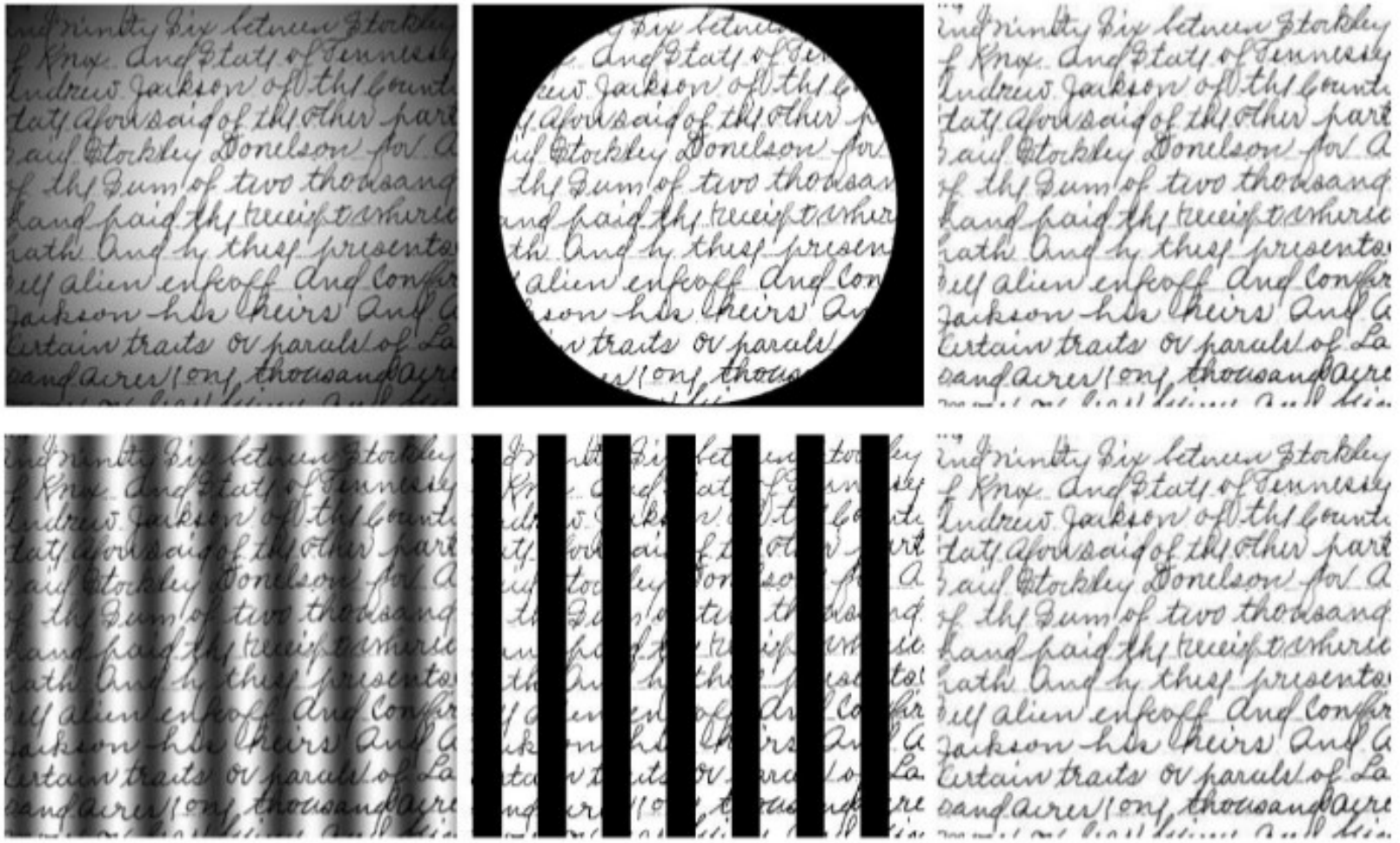


Otsu's Method

Smoothed by an averaging mask

→ Smoothing can help to increase the performance of thresholding as segmentation method.

Moving Average Method



input image | Otsu's method | Moving average n=20,b=0.5

Moving Average Method

- Local threshold
- Carried out line by line

The moving average (mean intensity) at this new point is given by

$$m(k+1) = \frac{1}{n}\sum_{i=k+2-n}^{k+1} z_i = m(k) + \frac{1}{n}(z_{k+1} - z_{k-n})$$

Threshold T_{xy}

$$T_{xy} = bm_{xy}$$

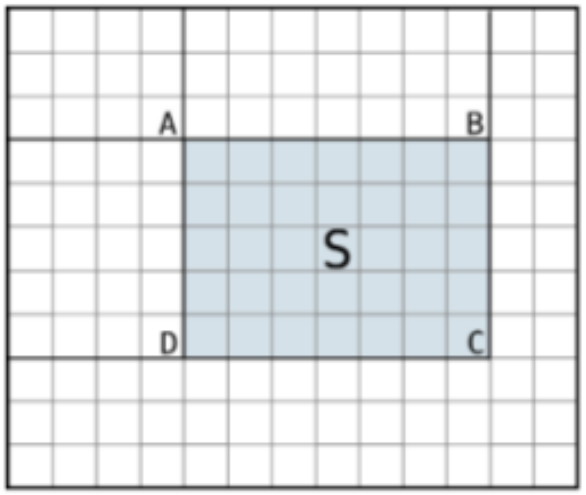
where b is constant and is the moving average m_{xy} from $m(k+1)$

Adapting Thresholding

- Threshold is based on the average of neighbor area

$$T_{xy} = \frac{1-t}{s^2} \sum_{i=x-s/2}^{x+s/2} \sum_{j=y-s/2}^{y+s/2} I(\mathrm{i}, \mathrm{j})$$

- The summation operation is optimized by integral image

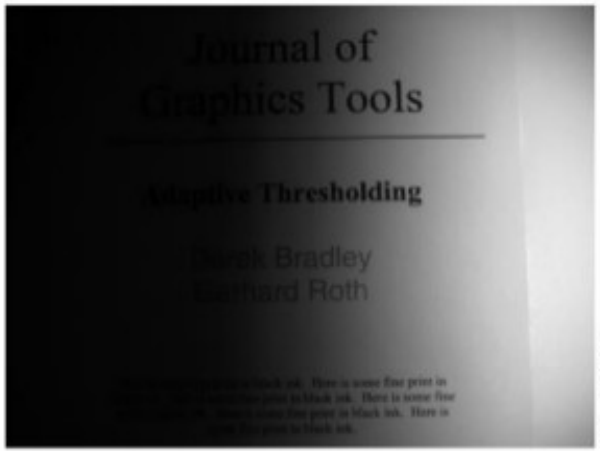


$$Int(\mathrm{x}, \mathrm{y}) = \sum_{i \le x, j \le y} I(i, \mathrm{j})$$

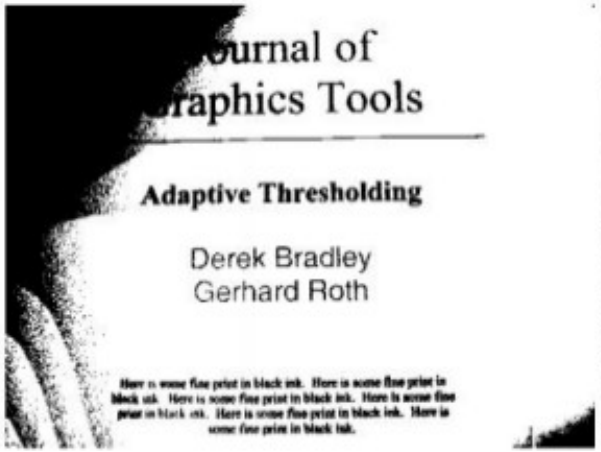
$$S = Int(\mathrm{C}) + \mathrm{Int}(\mathrm{A}) - \mathrm{Int}(\mathrm{B}) - \mathrm{Int}(\mathrm{D})$$

- Bradley, D., G. Roth, "Adapting Thresholding Using the Integral Image," Journal of Graphics Tools. Vol. 12, No. 2, 2007, pp.13-21..

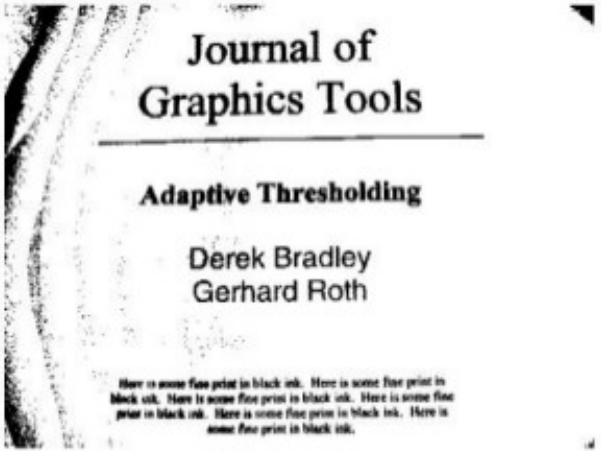
Adapting Thresholding



input image

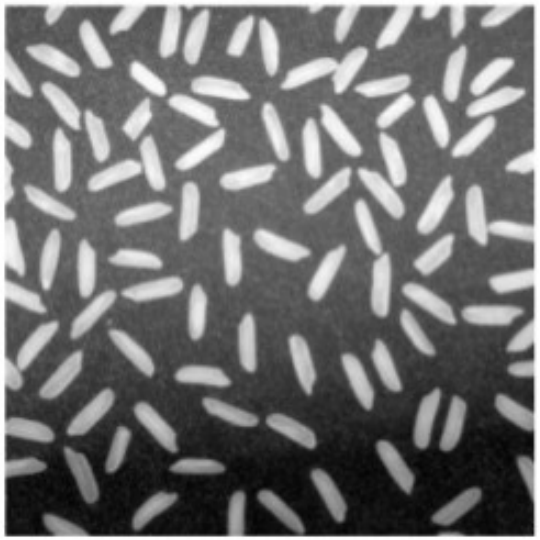


Moving average



Adapting Thresholding
Using the Integral Image

Adapting Thresholding



original image

Adapting Thresholding

t=0.4

Otsu's method

Filtering - Convolution

Convolution is one of the most often used operations in image processing.

Used for

- smoothing/noise reduction
- edge detection/sharpening
- finding locations of interest

Convolution in 1D

Continuous Convolution $I(x), h(x), x \in \mathbb{R}$

$$(I * h)(t) := \int_{-\infty}^{\infty} I(\tau) h(t - \tau) d\tau$$

Discrete version of the 1D convolution (common notations)

$$(I * h)(i) := \sum_{k \in D} I(k) h(i - k)$$

Partial Derivatives

$$\frac{\partial}{\partial x} f(x,y) = \lim_{h \leftarrow 0} \frac{f(x+h,y) - f(x,y)}{h}$$

Discretization

$$\frac{\partial}{\partial x} f(x,y) \approx \frac{f(x+\Delta x,y) - f(x,y)}{\Delta x}$$

$$= (f(i+1,j) - f(i,j))$$

x-direction $\frac{\partial}{\partial x}$

-1	1

y-direction $\frac{\partial}{\partial y}$

-1
1

Partial Derivative 1D Filter Mask

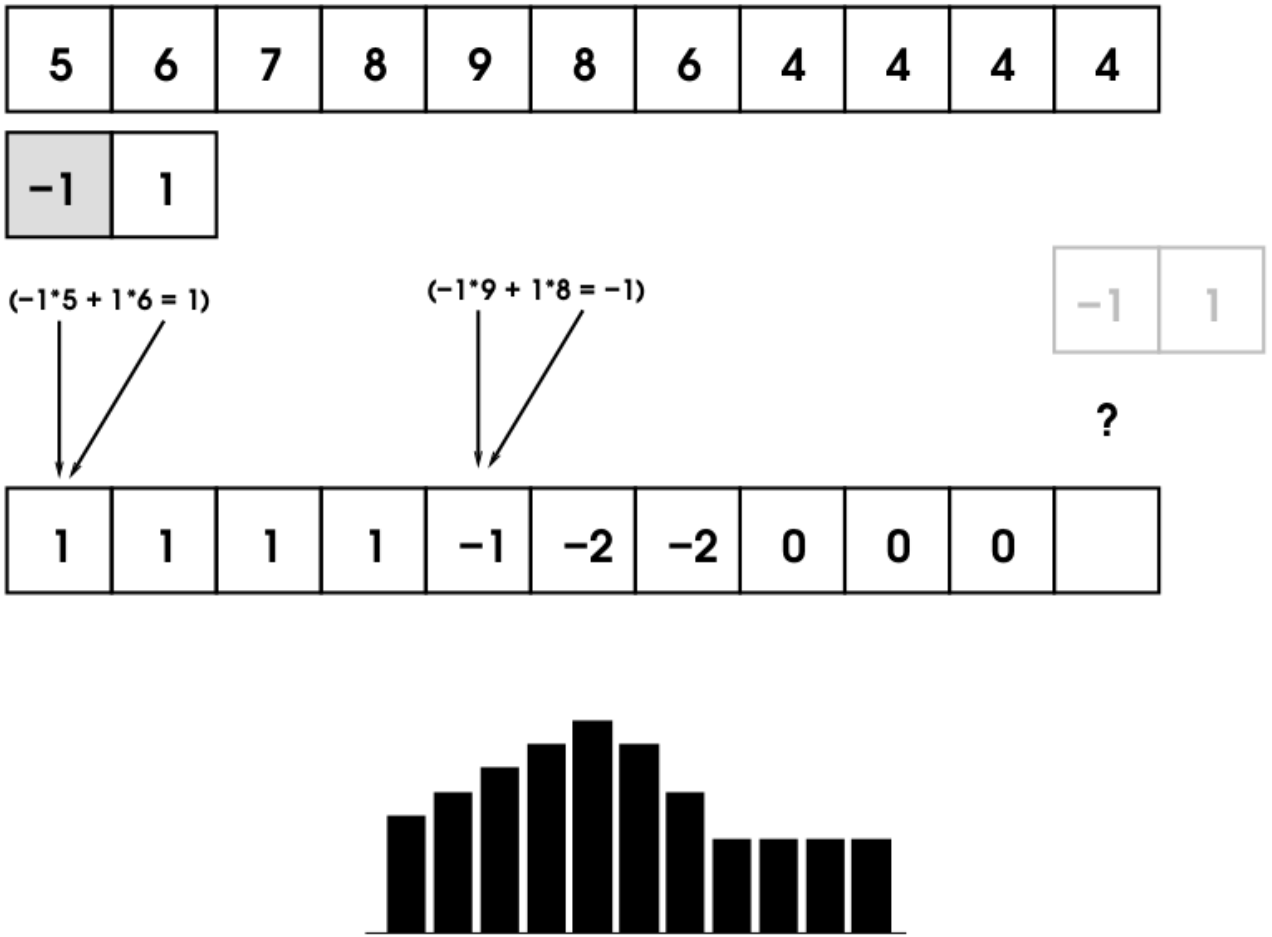
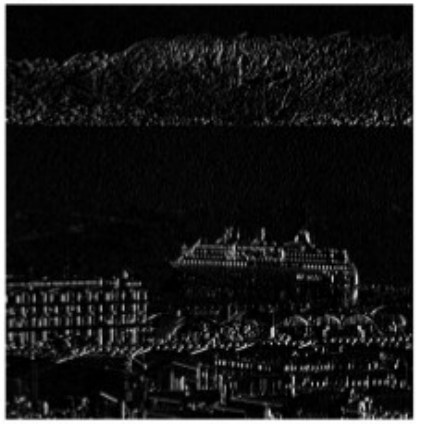


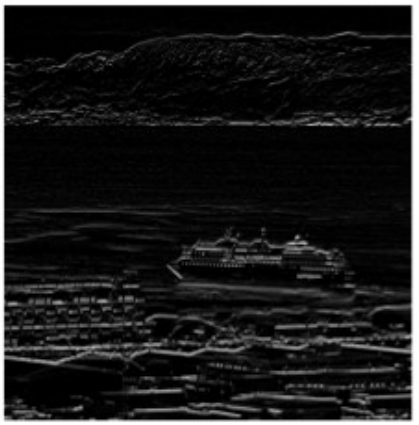
Image Derivative

$\Rightarrow$

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



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



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

$\nabla I(x,y) = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$

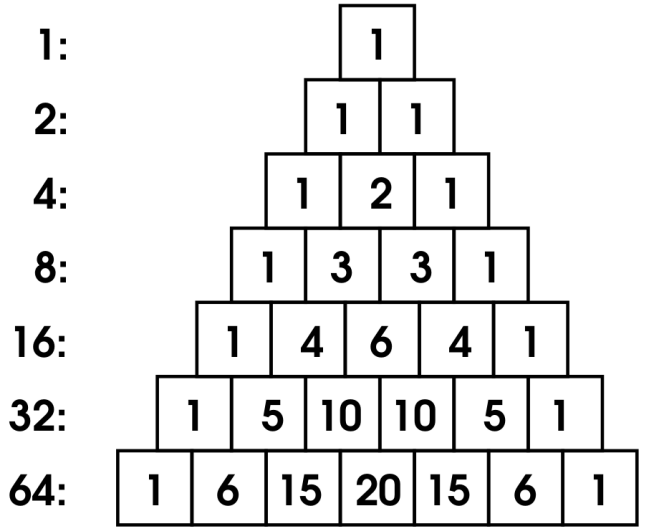
$|\nabla I|$

Pascal Triangle

Reduce the high-frequency contribution of the image

→ low-pass filter (blurring of the image)

The binomial coefficients provide a good discrete approximation for Pascal's triangle a Gaussian filter:



Binomial-mask: 1/64 [1 6 15 20 15 6 1]

(Sum of elements normalized to 1.0!)

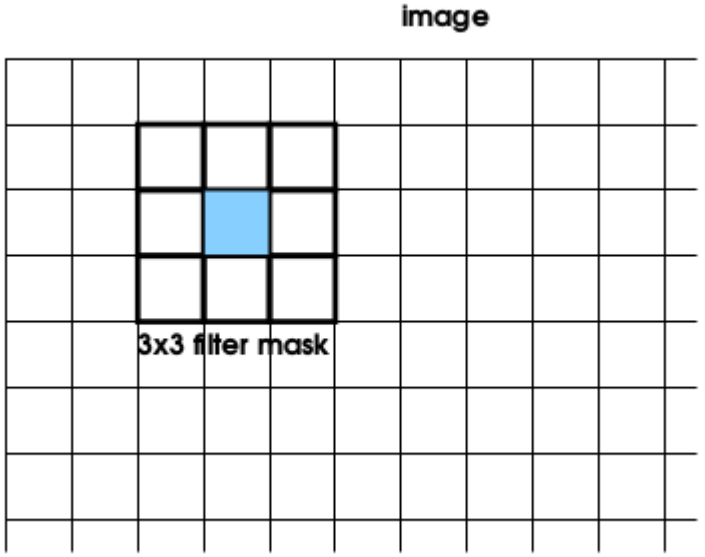
2D Convolution

Extend the 1D convolution to a 2D convolution

$$g(i,j) = (I * h)(i,j) = \sum_{k,l \in D} I(k,l)h(i-k, j-l)$$

$$= \sum_{k,l \in D} I(i-k, j-l)h(k,l)$$

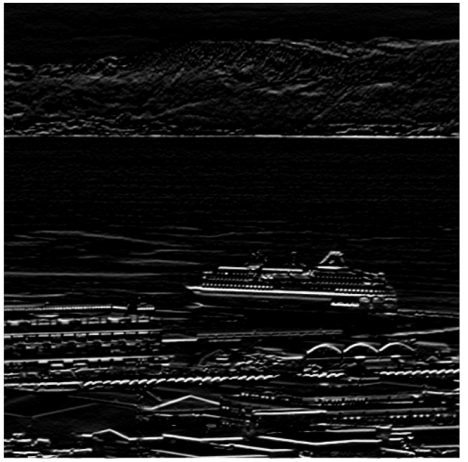
Image I and filter h



Filter Example: Sobel

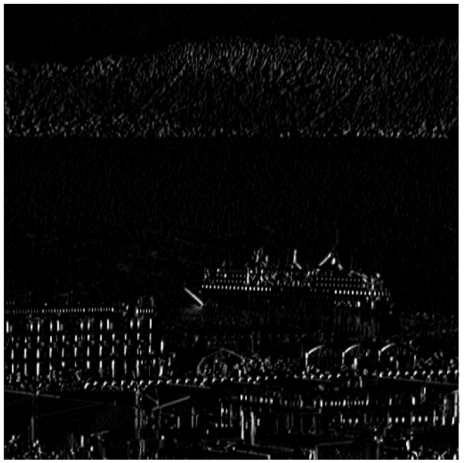
$$\Rightarrow \quad \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

rotate $0°$

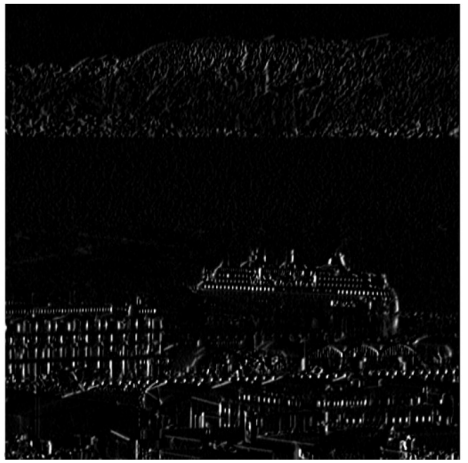


rotate $180°$

rotate $90°$



rotate $270°$



Cut-off at zero.

OCTAVE/MATLAB example

Load in an image and apply the Sobel filter in different orientations

```
myimg=imread('boat.png');
figure(1);
imshow(myimg);
filtermatrix=fspecial("sobel");
% sobel = [1 2 1; 0 0 0; -1 -2 -1]; % alternatively
filterresult = filter2(filtermatrix, myimg);
filterresult = uint8(filterresult);  %!!! convert 8-bit unsigned
figure(2);
imshow(filterresult);
```

Try different orientations of the filter:

```
n=3;
filter=rot90(fspecial(),n); %rotate filter-matrix n*90 degrees
```

Smoothing: Box Filter

original

=>

box-filtered

3x3 mask

$$\frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Smoothing: Gaussian

Gaussian filter mask with variance 1

original

=>

Gauss filtered

Image boundary effects - Padding

How to extend the boundary?

Padding types:

- Zero/mean/constant
- Symmetric/reflected
- Replicate/clamp
- Periodic/circular/wrap
- Mirror
- Extend

zero-padding

symmetric

replicate

periodic

Linear Filters

Linear filter operations calculate the output image pixel $g(i,j)$ as a linear combination of brightnesses $I(i,j)$ in the neighborhood D weighted by the coefficients $h(k,l)$ of the convolution mask.

$$g(i,j) = (I * h)(i,j) = \sum_{k,l} I(k,l)h(i-k, j-l)$$

(Equivalent to the discrete convolution: convolution = linear filtering)

Note: Other filters are called **non-linear.**

Linear Filtering

Linear Filter: Ouput pixel is determined as wighted sum of input pixel values

$$g(i,j) = \sum_{k,l} I(i+k, j+l)h(k,l)$$

The entries in the weight **kernel** or **mask** h(k,l) are called filter coefficients

Correlation operator can be noted as

$$g = I \otimes h$$

Convolution operator:

$$g = I * h$$

Reversed sign

$$g(i,j) = \sum_{k,l} I(i-k, j-l)h(k,l) = \sum_{k,l} I(k,l)h(i-k, j-l)$$

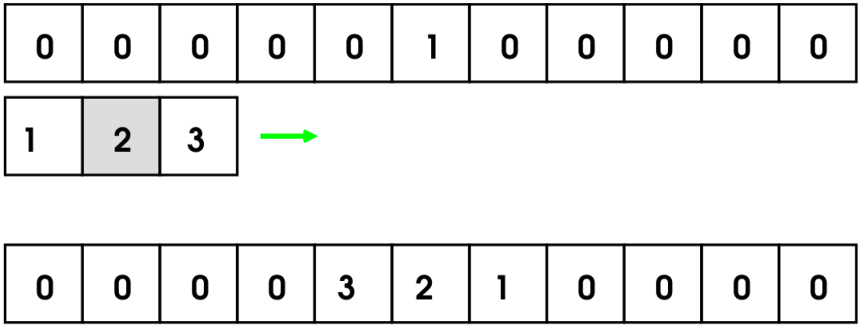
With h as the **impulse response function.**

Linear Filtering

In computer vision and image processing literature convolving a ” mask with an image“ is often used for the sliding process and does not necessarily differentiate between correlation and convolution.

→ Expect some imprecise terminology

(Filter mask = unit pulse response)



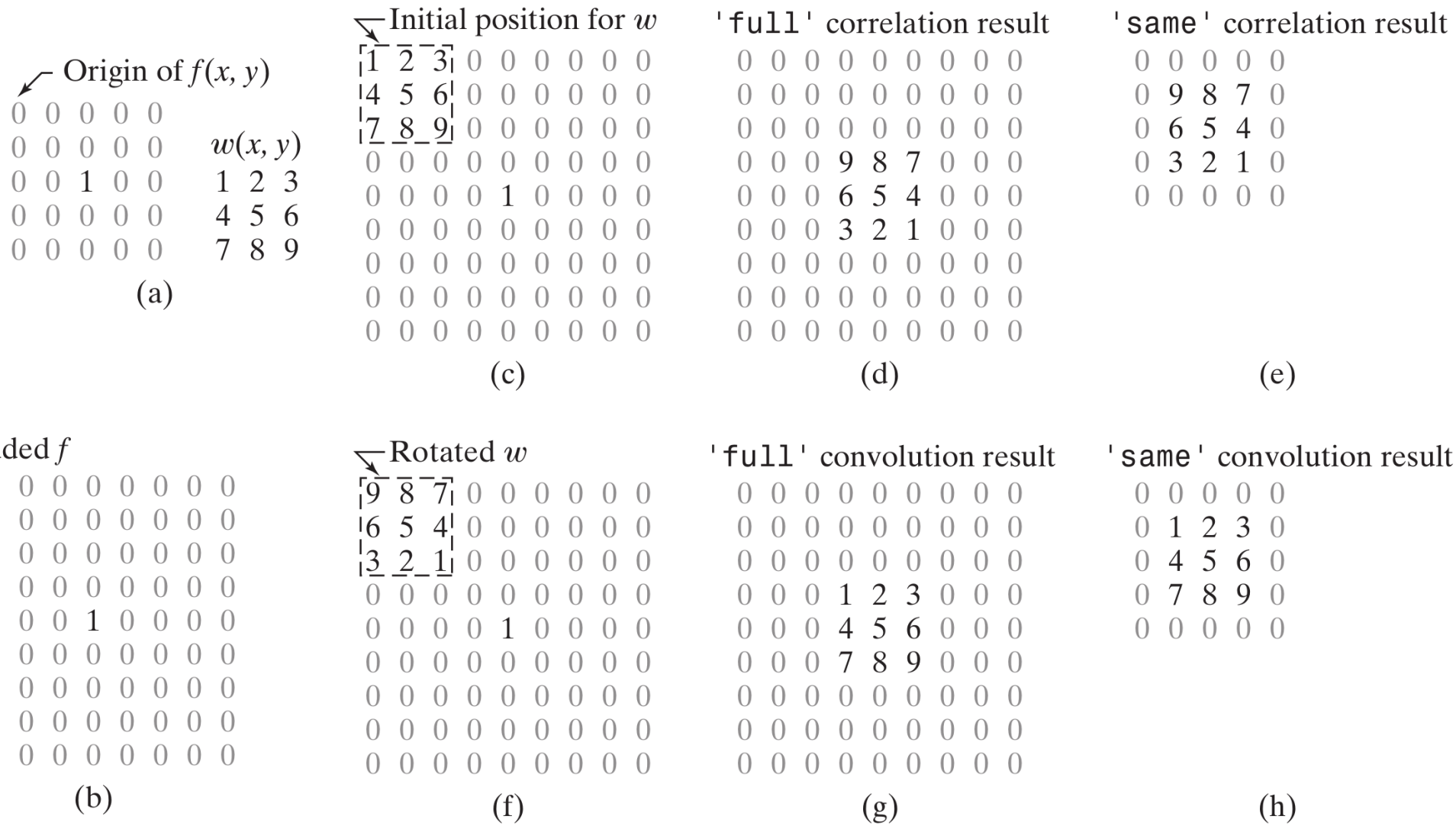
The 1D sliding example shows a correlation. A convolution has the elements in the mask reversed (2D case: reversed elements in x- and y- direction).

Note: When the filter is symmetric: correlation = convolution!

Convolving a function with a unit impulse yields a copy of the function at the location of the impulse.

Kernel Function - Convolution/Correlation

Kernal function h convolved with an impulse signal (an image that is 0 everywhere except at the origin) reproduces itself, whereas correlation produces the reflected signal.



Convolution

Commutativity

$$f * g = g * f$$

Associativitiy

$$f * (g * h) = (f * g) * h = f * g * h$$

Distributivity

$$f * (g + h) = (f * g) + (f * h)$$

Associativity with scalar multiplication

$$a(f * g) = (af) * g = f * (ag)$$

Differentiation

$$\frac{d}{dx}(f * g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

Superposition - Linear Shift Invariant

Correlation and Convolution are **linear shift invariant (LSI) operators**, which obey the

Superposition (addition) principle

$$h \circ (I_0 + I_1) = h \circ I_0 + h \circ I_1$$

and

Shift invariance principle

$$g(i,j) = I(i+k, j+l) \Leftrightarrow (h \circ g)(i,j) = (h \circ g)(i,j)(i+k, j+l)$$

Separable Filter

Filters $h = h_x * h_y$ are separable.

=> efficient implementation

Exploit associativity of the convolution and perform two separate 1D filtering:

$$h * I = (h_x * h_y) * I = h_x * (h_y * I)$$

Examples:

$\frac{1}{K^2}$

1	1	$\cdots$	1
1	1	$\cdots$	1
$\vdots$	$\vdots$	1	$\vdots$
1	1	$\cdots$	1

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$\frac{1}{256}$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

$\frac{1}{8}$

−1	0	1
−2	0	2
−1	0	1

$\frac{1}{4}$

1	−2	1
−2	4	−2
1	−2	1

$\frac{1}{K}$ | 1 | 1 | $\cdots$ | 1 |

$\frac{1}{4}$ | 1 | 2 | 1 |

$\frac{1}{16}$ | 1 | 4 | 6 | 4 | 1 |

$\frac{1}{2}$ | −1 | 0 | 1 |

$\frac{1}{2}$ | 1 | −2 | 1 |

(a) box, $K = 5$ (b) bilinear (c) "Gaussian" (d) Sobel (e) corner

Band-pass filters

Simple example of **band-pass** filters: **Sobel** and **corner operators**

Smoothing the image with a symmetric **Gaussian kernel (filter)**

$$G_\sigma(i,j) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{(i^2+j^2)}{2\sigma^2}\right)$$

and taking the first or second derivative creates a standard **band-pass filter**, where both low and high frequencies are filtered out.

Undirected **second derivative** of an image – **Laplacian operator**

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

High- /Low-pass Filter

High-pass filter (derivative filter)

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

Low-pass filter (smoothing filter)

$$* \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} =$$

Non-linear Filtering: Median

The central value of an ordered set is known as median:

Idea of the filter: replace the current pixel brightness with the median of the brightness in its neighborhood.

Properties:

- acts like a low-pass filter
- eliminates impulse noise quite well
- preserves edges (does not blur them much)

Generalizations:

- rank filtering orders the neighborhood into a sequence
- order statistics orders and weights the values of the sequence

=> the median filter is a special case of both

Non-linear Filtering: Median

20% salt/pepper noise

=>

5x5 median filtered

Disadvantages:

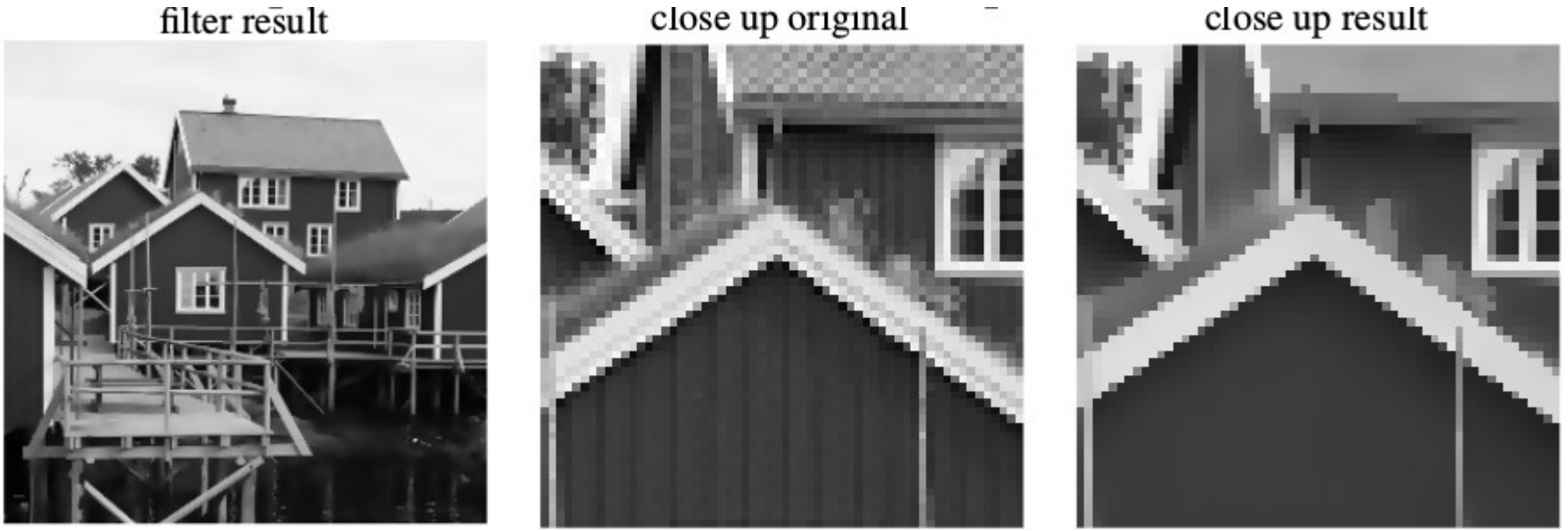
- damaging thin lines and sharp corners
- sorting needed

These issues can be reduced by adapting the neighborhood and recognizing that the middle (mn − 2m) pixels of a m × n window are unchanged by a shift.

Non-linear Filtering: Anisotropic Diffusion

Main idea:

- **Aim:** reduce image noise without removing significant parts of the image content
- stop smoothing across step-edges
- adapt the diffusion coefficient (which controls the rate of diffusion) to image content



Note: isotropic (independent of the direction) diffusion is equivalent to Gaussian blurring.

References:
P. Perona and J. Malik (1990). Scale-space and edge detection using anisotropic diffusion.
Joachim Weickert (1997). A Review of Nonlinear Diffusion Filtering)

Edges

- Edges seem to encode a lot of semantic and shape information which help us to extract information and recognize objects.
- Edges help us to recover information about the geometry of objects and the observers viewpoint

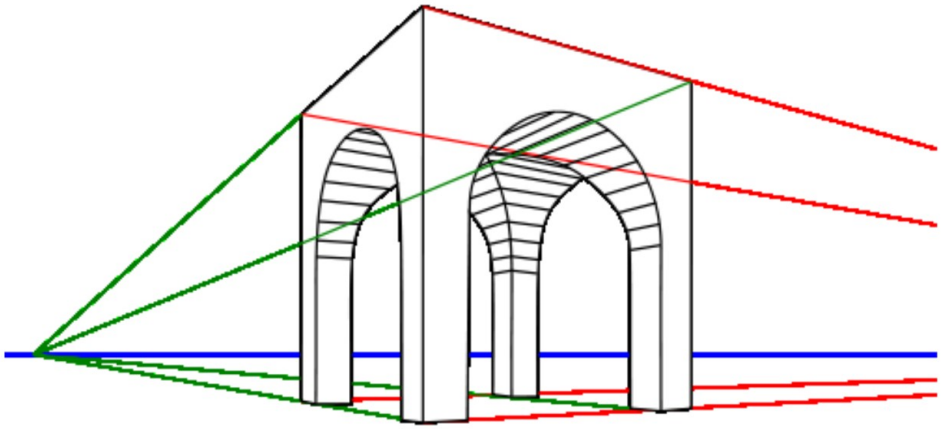


Image: Wikipedia/Wolfram Gothe

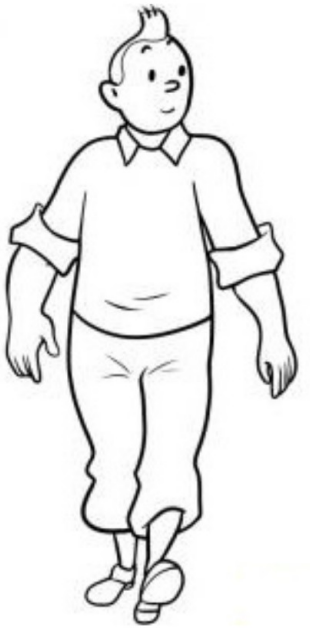


Image: www.dragoart.com

Edge Detectors

Aim:

- Identify sudden changes (discontinuities) in an image.
- Find the edges that are considered to be relevant by humans
- Extract information that could help to recognize objects

Origins of Edges in Images

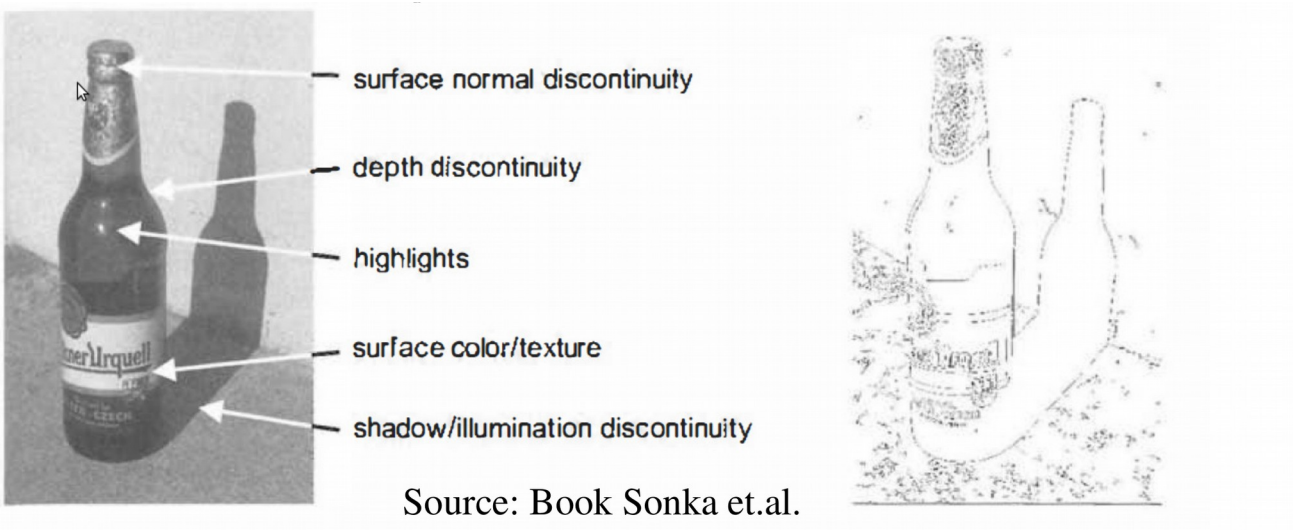
Edges in images occur due to several physical phenomena:

Edges which are less influenced by lighting conditions:

- object boundaries/depth discontinuities (shape)
- surface texture (texture)

Edges which are often considered as disturbances (as they depend highly on lighting conditions):

- reflections/highlights
- surface normal discontinuity
- shadow/illumination discontinuity



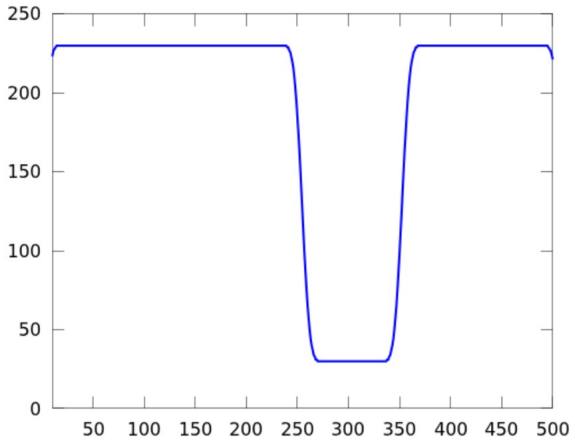
Source: Book Sonka et.al.

1D Edge

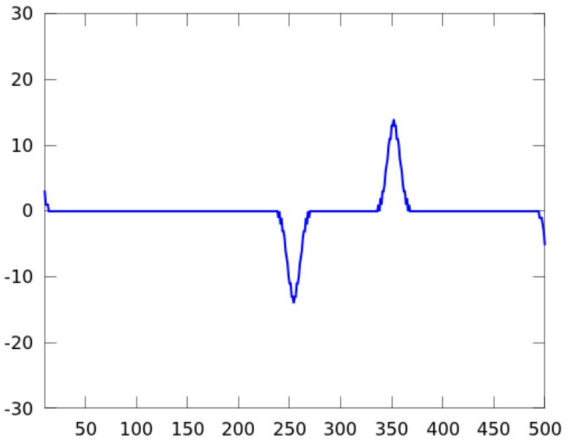
Edges are located at places where the image intensity function abruptly changes.

image

intensity function



first derivative



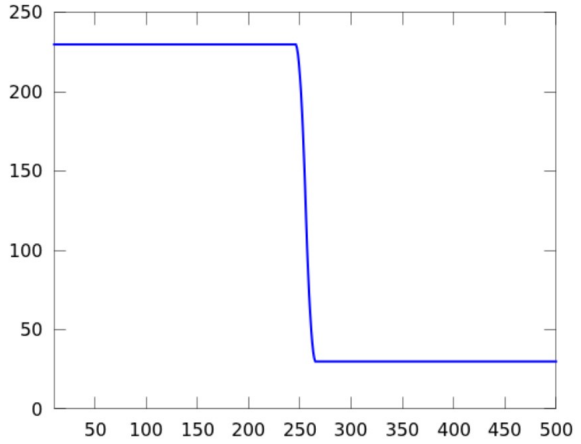
The edge location corresponds to the maximum or minimum of the first derivative.

Undisturbed Edge Detection

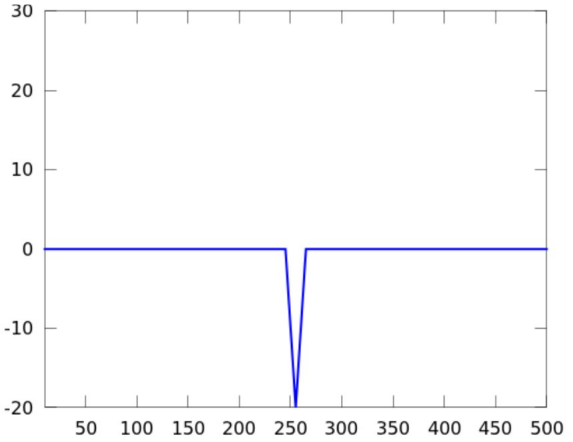
Consider a single row (1D-Scan-line) of the shown image.

Image

1D scan-line $f(x)$



1D derivative $\frac{\partial}{\partial x} f(x)$

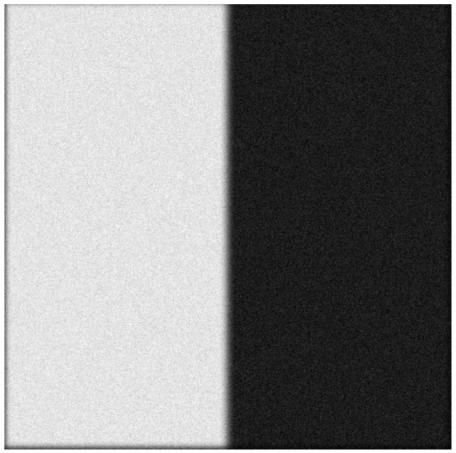


The derivative allows a clear detection and localization of the step-edge within the image function (i.e. to find edges we could look for peaks)

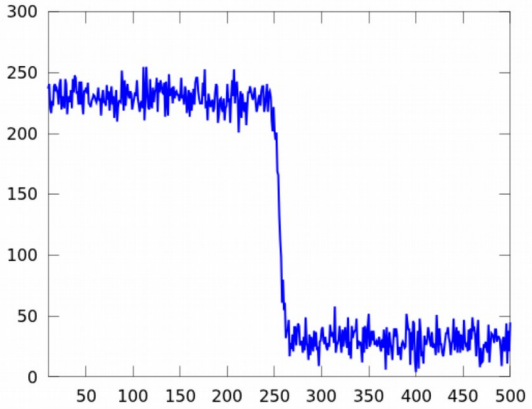
Influence of Noise

Consider now a single row (scan-line) of the same image with some noise added.

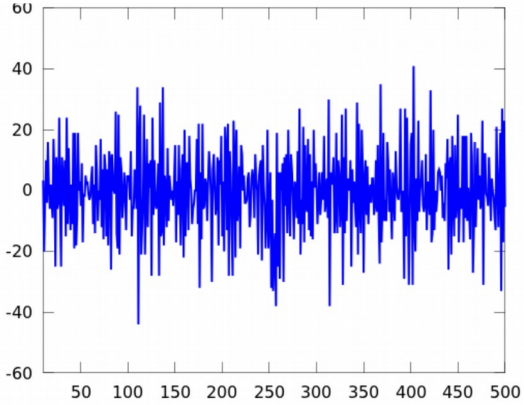
Added noise



1D signal $f(x)$



1D derivative $\frac{\partial}{\partial x} f(x)$



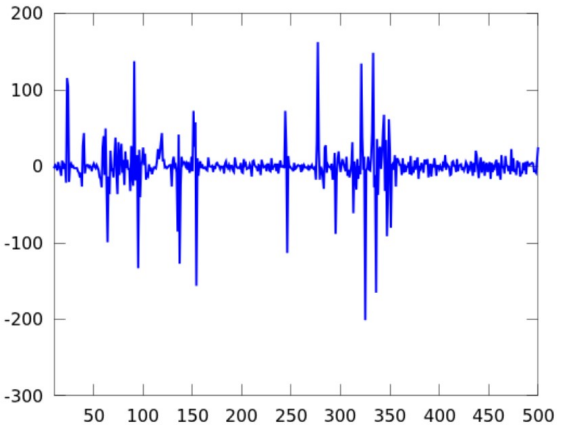
Would you recognize or could you localize the edge from the derivative of the image function ?

Influence of Noise To a Real Image

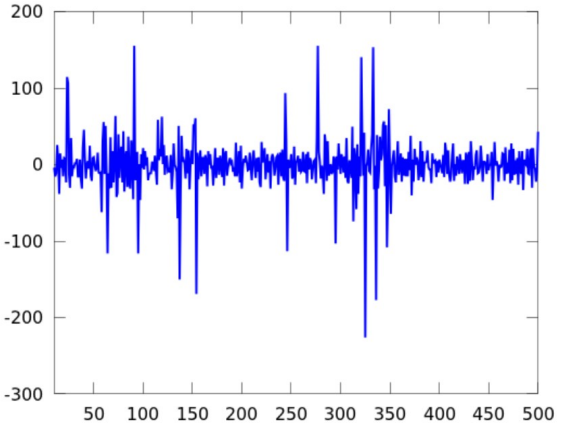
image

added noise

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



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



Influence of Noise

Observation:

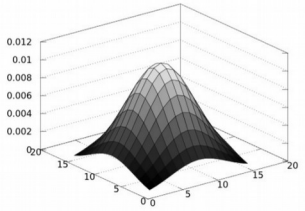
- Derivative/Difference filters respond highly to noise. (Generally, a higher noise level corresponds to a stronger response.)
- Applying the derivative filter to a noisy signal makes it difficult to recognize and localize an edge in the resulting signal.

Do we have a chance to recognize the edge ?

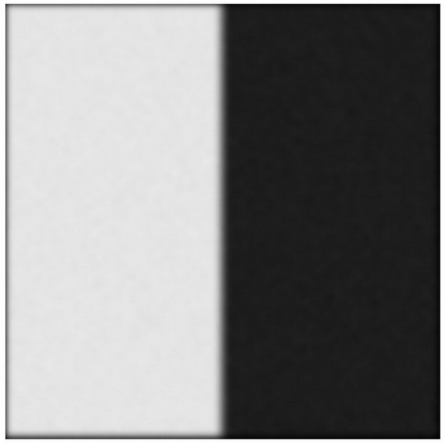
Reduce Influence of Noise

Smooth the image!

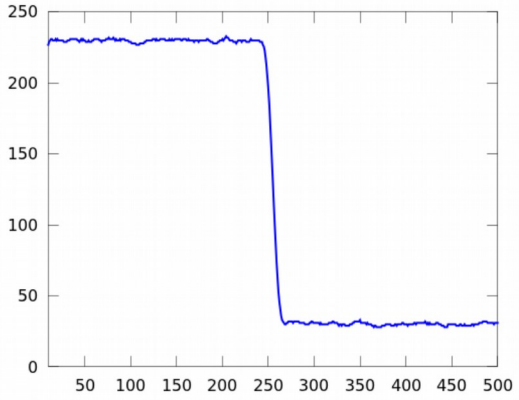
For example with a Gaussian filter g:



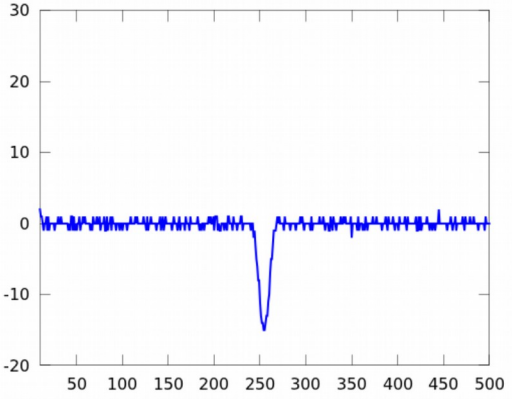
Smooth image



Signal $(f * g)$



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



Convolution and Differentiation

Our aim is to compute the partial derivative $\frac{\partial}{\partial x}$ of an image function f that is smoothed by a Gaussian filter g. This process can be represented by the following:

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

Recall how the differentiation applies to a convolution:

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

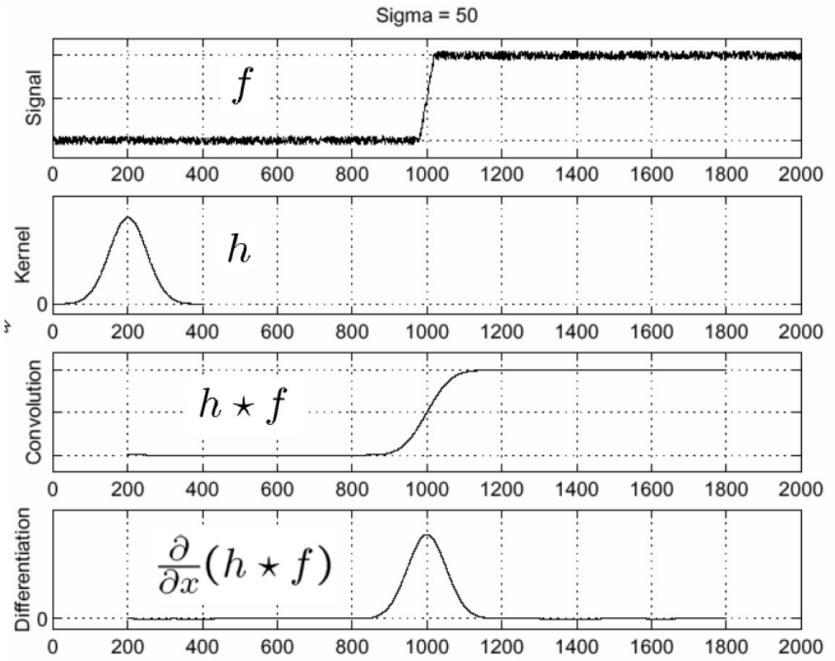
Note: The term $\frac{\partial g}{\partial x}$ (derivative of the Gaussian) can be pre-computed independently of the image content.

The reformulation above is valid because the differentiation itself is a convolution and then the associativity $[f * (g * h) = (f * g) * h = f * g * h]$ of convolutions can be exploited.

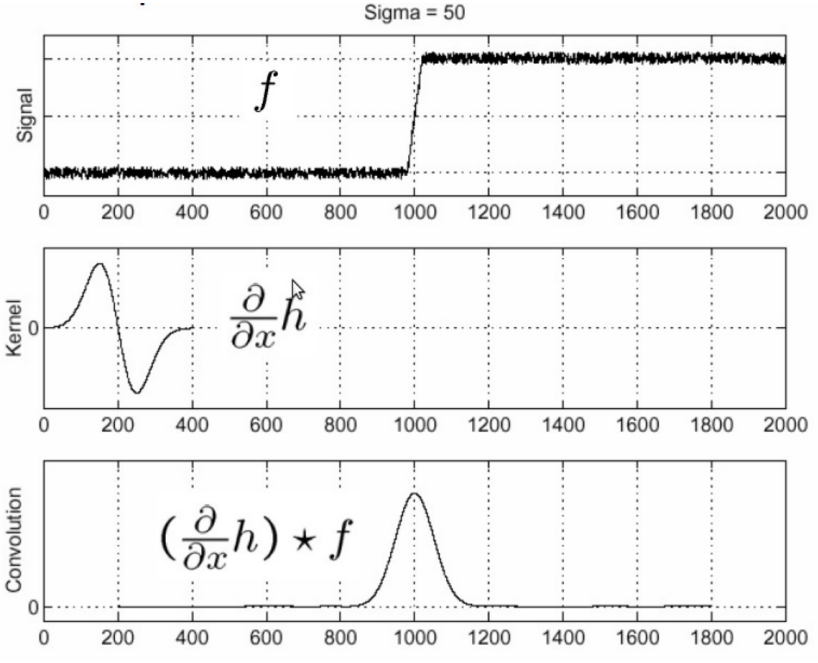
Combined Derivative and Smoothing

This property allows us to combine the smoothing and the computation of the derivative into a single operation: Filter the image f with the derivative of the Gaussian

Smooth then take derivative $\frac{\partial}{\partial x}(f * g)$



Filter with $\frac{\partial g}{\partial x} * f$

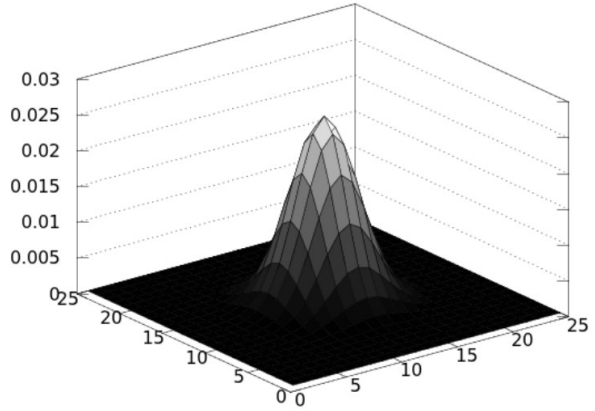


Graphs: S. Seitz (The Gaussian is denoted as h in the graphs)

The Derivative of Gaussian

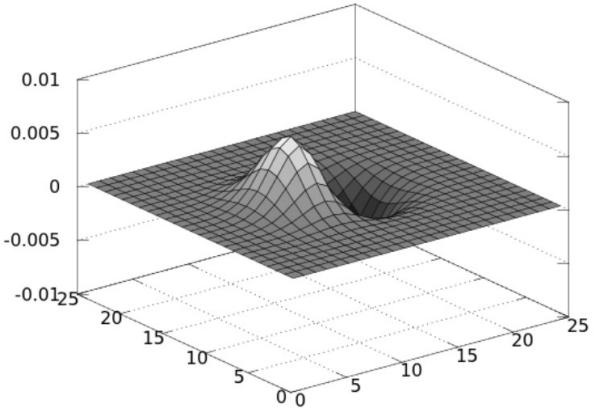
Recall: An approximation of the first derivative in x-direction:

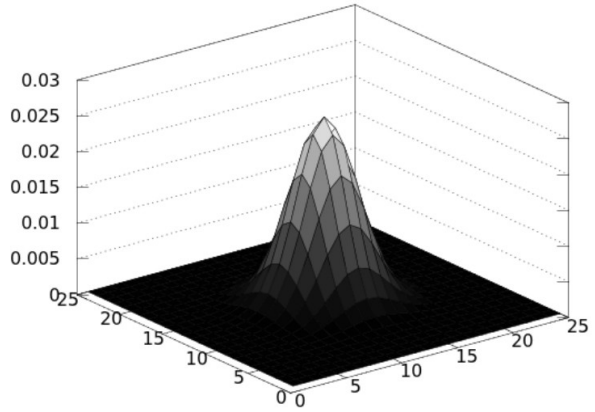
Gaussian



$$*(\ -1\ \ 1) =$$

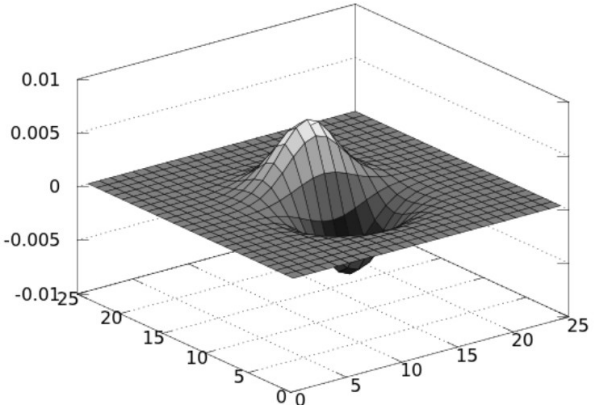
derivative





$$* \begin{pmatrix} -1 \\ 1 \end{pmatrix} =$$

$$\frac{d}{dy} g$$

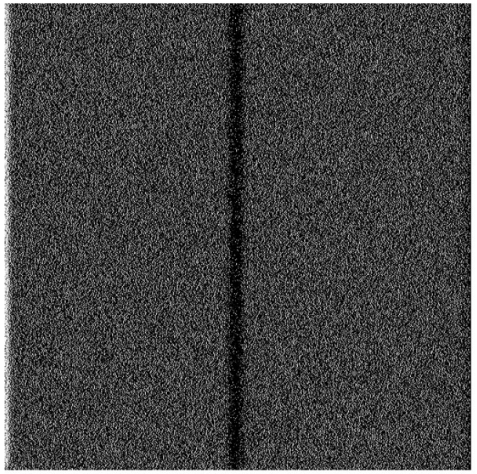


Edges, Noise, Derivative

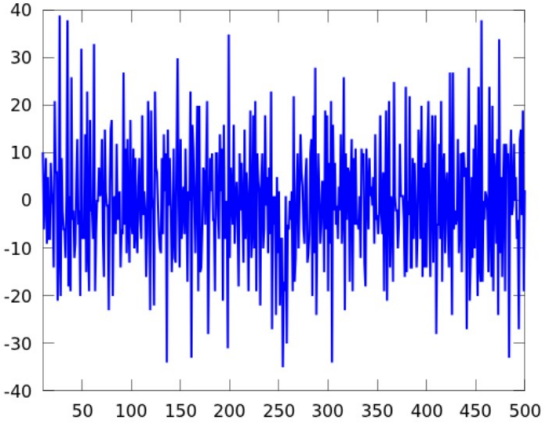
At some point we asked if you could recognize or localize the edge from the derivative of the noisy image function? (Here the center image)

Due to $\frac{\partial}{\partial x}(f * g) = \frac{\partial f}{\partial x} * g = f * \frac{\partial g}{\partial x}$ we can apply the smoothing after computing which allows the localization of the derivative and still get the desired $\frac{\partial}{\partial x}(f * g)$ which allows the location of the edge!

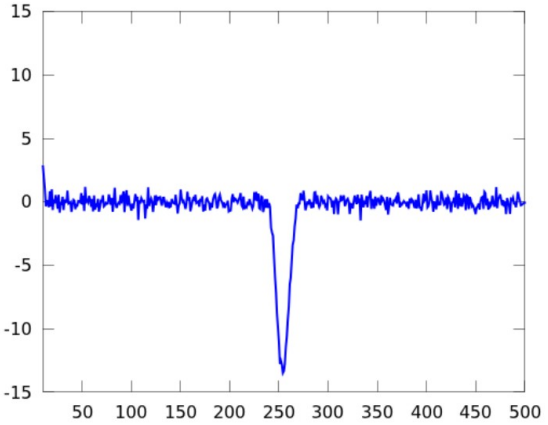
derivative $\frac{\partial f}{\partial x}$



1D derivative $\frac{\partial}{\partial x} f(x)$



smooth $g * \frac{\partial f}{\partial x}$



Operators approximating derivatives

Beside the simplest approximation of the first derivative (x-direction) there are more operators approximating first derivatives

Roberts operator:

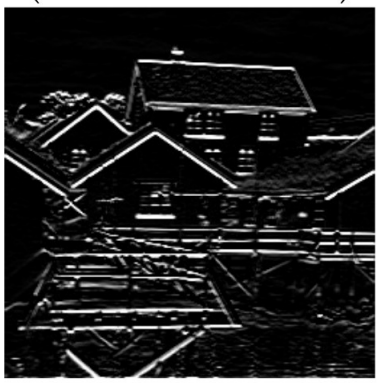
$$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

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

Derivative Filters

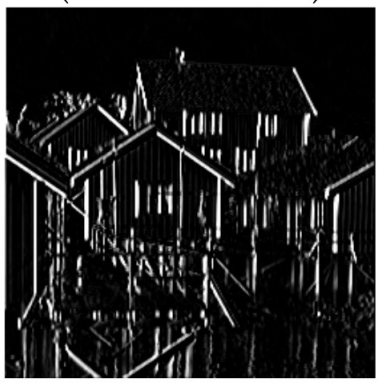
Prewitt operator:

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



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

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

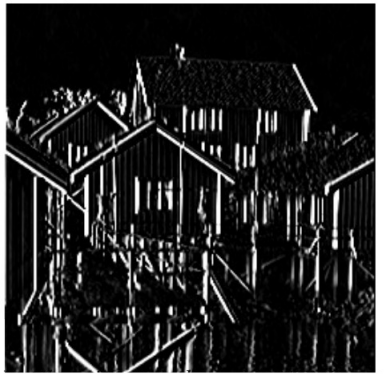


Sobel operator:

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

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

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



Derivative Filters

Robinson operator:

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

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

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

Kirsch operator:

$$\begin{pmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{pmatrix}$$

$$\begin{pmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{pmatrix}$$

$$\begin{pmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{pmatrix}$$

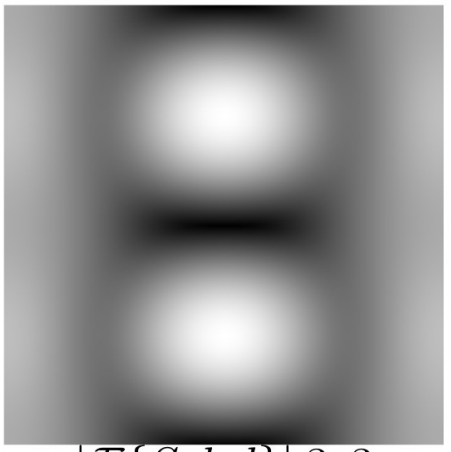
High-Pass

Fourier transforms of two selected high-pass filter

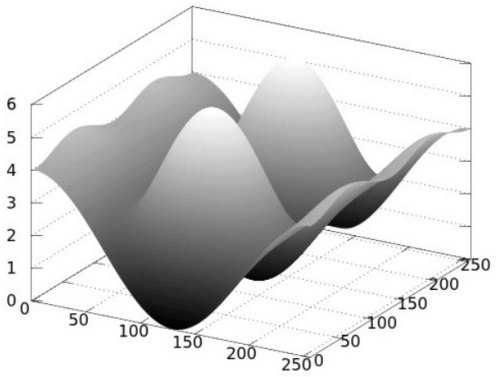
Robinson

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

$|\mathcal{F}\{Robinson\}|$ 3x3



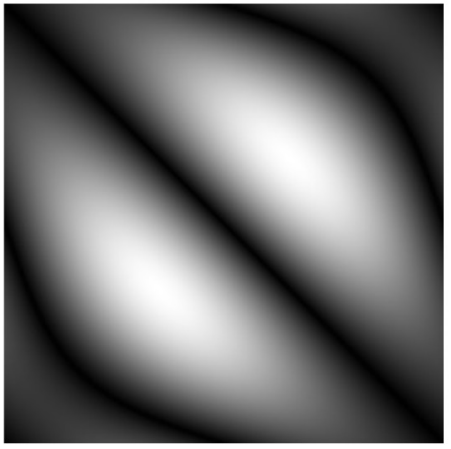
3D-View



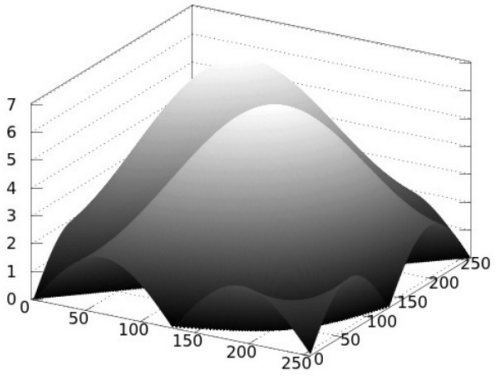
Sobel

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

$|\mathcal{F}\{Sobel\}|$ 3x3

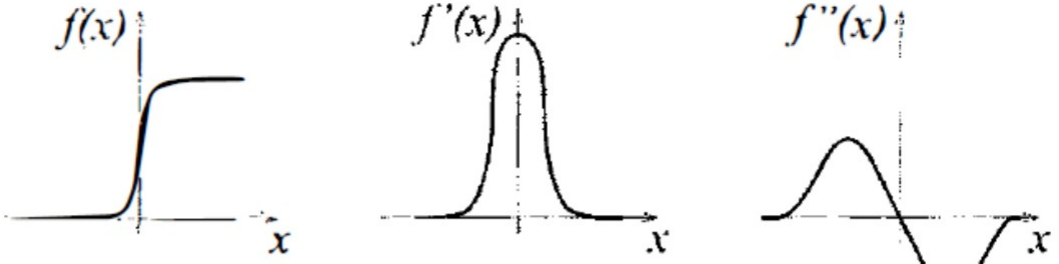


3D-View



Edges as Zero-Crossing

The **first derivative** is expected to have an extrema at the edge-location:



Source: Book Sonka et.al.

The **second derivative** is zero in that point.

That means an edge can be determined quite accurately by finding the zero-crossing of the second derivative.

Laplacian/ Laplace operator / Laplace-filter:

$$\Delta = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \approx \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

(This is a linear and rotationally symmetric filter, that is not separable)

The Laplacian of Gaussian

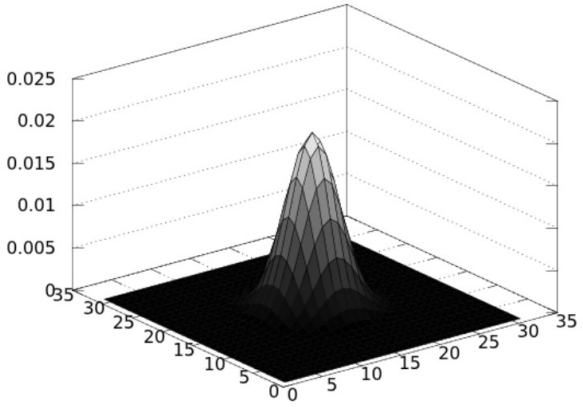
Noise problem → smooth the image

Apply the Laplace-filter to an Image f that is Gaussian-smoothed with G:

$$\nabla^2(f * G) = (f * (\nabla^2 G))$$

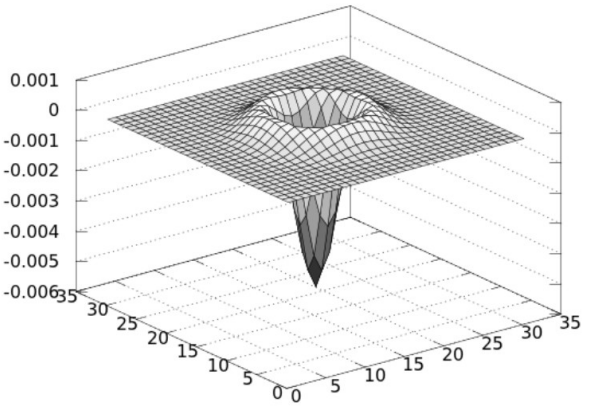
The Laplacian of Gaussian (LoG) is independent of the image content and can be pre-computed before applying it to the image

Gaussian



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

LoG



-LoG = Mexican Hat

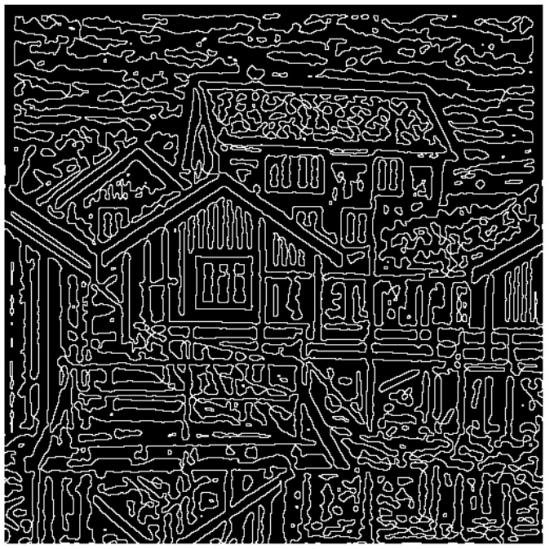
The Laplacian of Gaussian

After image convolution with the LoG, the locations in the convolved image where the zero level is crossed corresponds to the positions of edges.

LoG (equalized)

white: > 0, black: <= 0

zero-crossings



→ closed contours

Simple zero crossing detector: moving 2x2 window, label as edge if + and – values occur simultaneously

Difference of Gaussians

In practice the LoG is often approximated:

- The difference of two Gaussian averaging masks with different standard deviation σ is widely used as an approximation for the LoG.
- This method is called **difference of Gaussians** and referred to as **DoG**
- The DoG filter is separable and can therefore save considerable computation time in two or more dimensions

Properties: Low/High-pass-Filtering

Low-pass-Filtering/Smoothing filters:

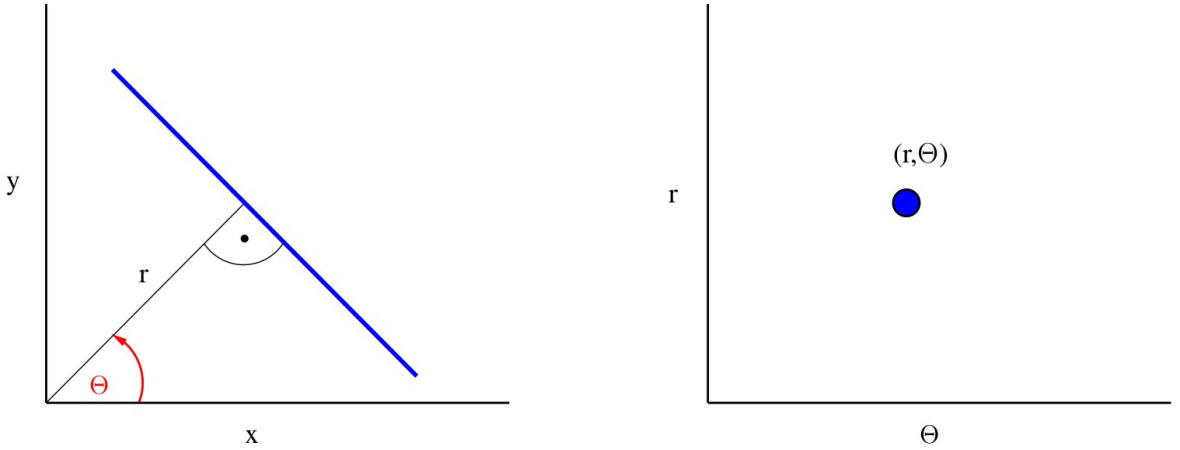
- Often isotropic (directions treated the same way)
- Preservation of mean
- Keep low-frequency components
- Reduce noise
- Kernel sum = 1 (Filter mask elements sum up to 1.0)

High-pass-Filtering/Derivative filters:

- Direction-dependent
- Enhance oriented features
- Keep high-frequency components
- Enhances noise
- Kernel sum = 0 (Filter mask elements sum up to 0.0)

The Hough Transform

A line can be represented by a single point in the parameter space:



Each point gives rise to “many“ $(= \infty)$ lines that cross this point → curve in parameter space:

image space

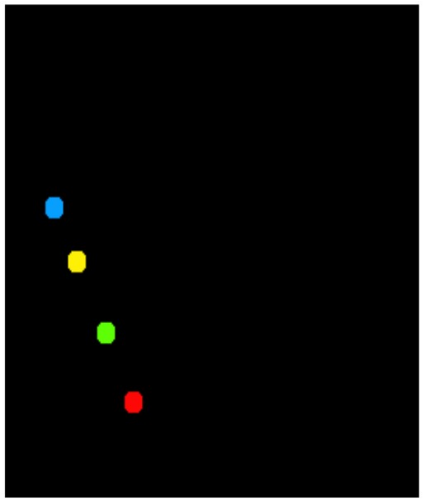
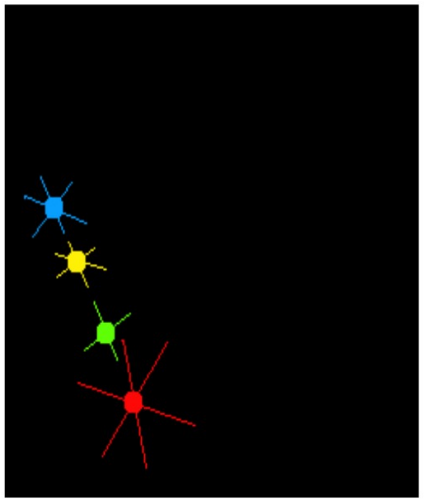
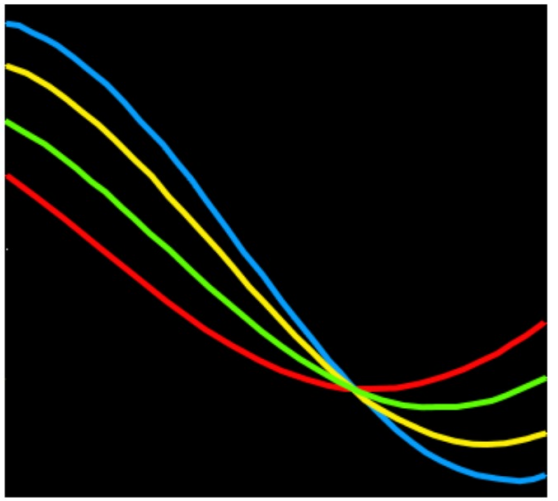


image space



parameter space

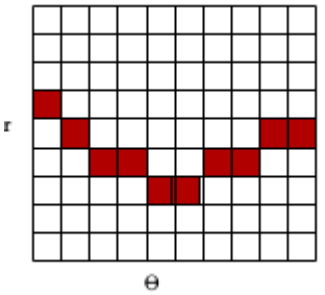


The Hough Transform

Parametric curve equation $f(x, a) = 0$ for a line:

$$f(x, a) = f(\underbrace{x, y}_{x}, \underbrace{\theta, r}_{a}) = x\cos(\theta) + y\sin(\theta) - r = 0$$

The two parameters $a = (\theta, r)$ give rise to a 2d-parameter space that is quantized into a 2D-accumulator array $A(a)$.



For a given point (x_i, y_i) the accumulator cells $A = (\theta_d, r_d)$ fulfilling the equation

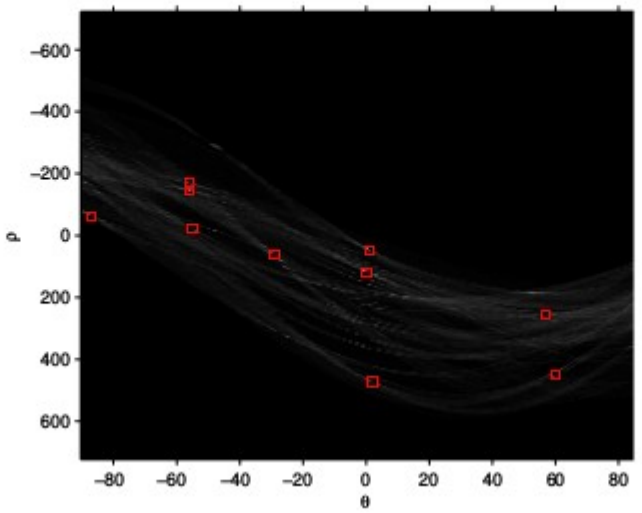
$$r_d(\theta_d) \approx x_i \cos(\theta_d) + y_i \sin(\theta_d)$$

are increased for all discretised angles θ_d

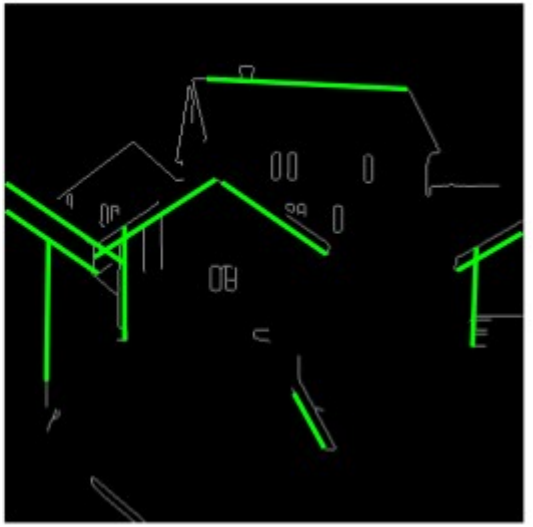
The Hough Transform

canny edges

parameter space
(accumulator)



Houghline-segments



- The shown 10 highest local maxima correspond to the shown line-segments in the edge-image on the right side.
- The line-segments are obtained by selecting the parts of the lines with a strong edge-support.

The Hough Transform

Main Idea:

Determine for all points (=edge pixels) in the image space all possible line-parameters for the lines that can go through that point. Line parameters that occur most often are likely to correspond to important lines in the image.

Steps of the Hough transform:

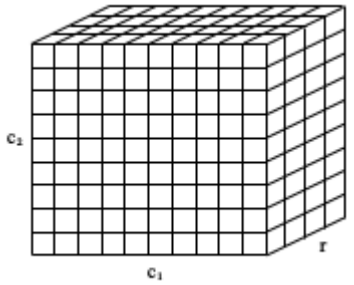
- Apply an edge image thresholding (→ line pixel candidates)
- discretization/quantization of the parameter-space into an accumulator array
- for each edge pixel determine all parameters of lines that might go through that point
- increase the corresponding values in the accumulator array.
- For true lines the appropriate accumulator cell will be increased many times
- line detection is transformed into the detection of maxima in the accumulator array.

The Hough Transform

Parametric curve equation $f(x, a) = 0$ for a circle:

$$f(x, a) = f(\underbrace{x_1, x_2}_{x}, \underbrace{c_1, c_2, r}_{a}) = (x_1 c_1)^2 + (x_2 c_2)^2 r_2 = 0$$

The three parameters $a = (c_1, c_2, r)$ give rise to a 3d-parameter space that is quantized into a 3D-accumulator array $A(a)$.



For a given point (x_1^i, x_2^i) the accumulator cells $A = (c_1, c_2, r_d)$ fulfilling the equation

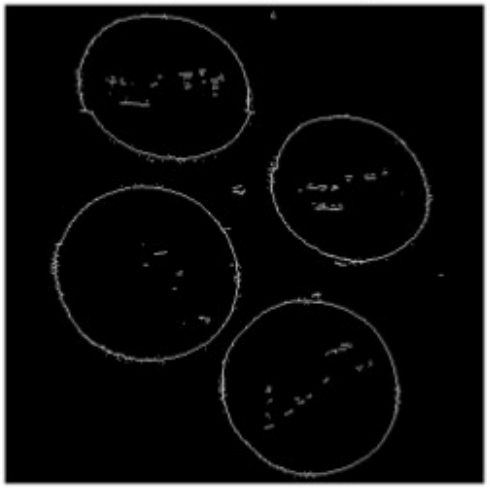
$$f(x, a) = (x_1^i - c_1)^2 + (x_2^i - c_2)^2 - r_d^2 \approx 0$$

are increased.

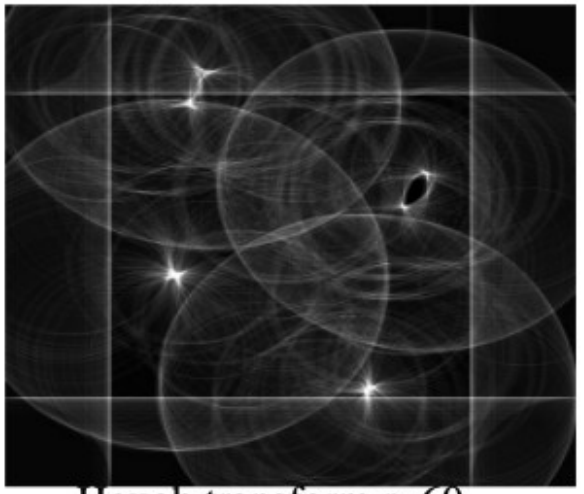
→ Note: the search for a particular radius results in a 2d-accumulator.

The Hough Transform for circles

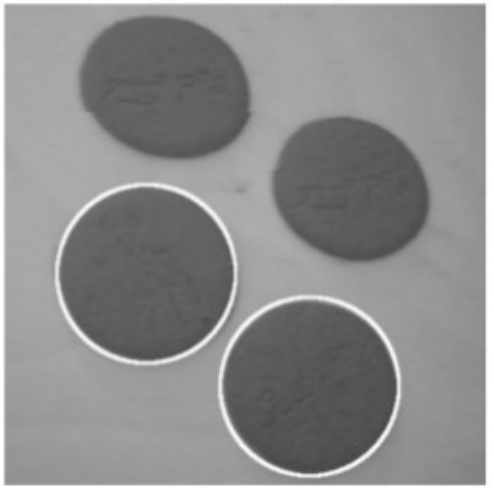
binary edge image



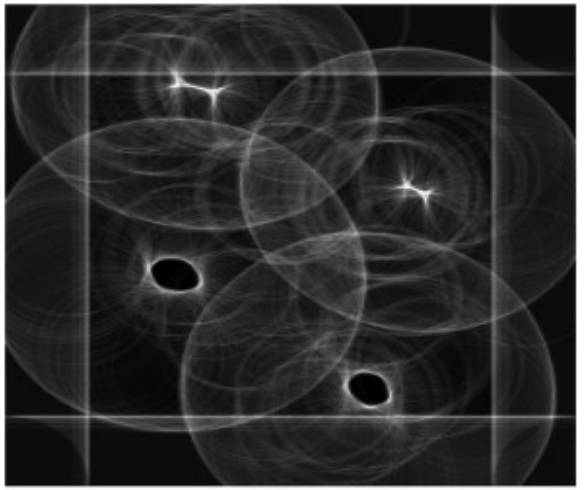
Hough transform r=76



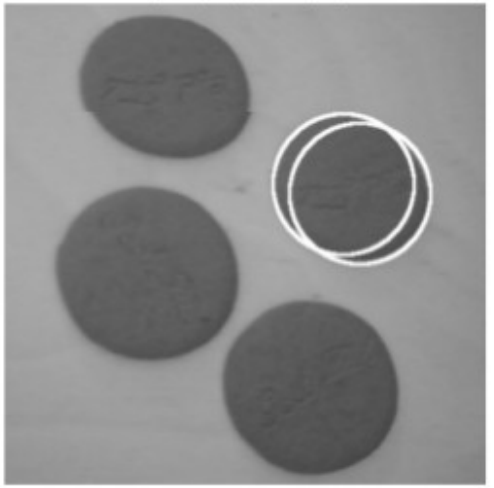
circles r=76



Hough transform r=60



circles r=60



The Generalized Hough Transform

The Generalized Hough Transform can be used to find arbitrary non-parametric curves (i.e. no simple analytic equation description of the shape) in an image.

→ A Look-up table is used as transform mechanism.

Steps of the Generalized Hough Transform:

- determine a reference table (R-table)
- increment accumulator by using the look-up R-table
- detection (as before, find local maxima in the accumulator array)

Determine the Reference Table

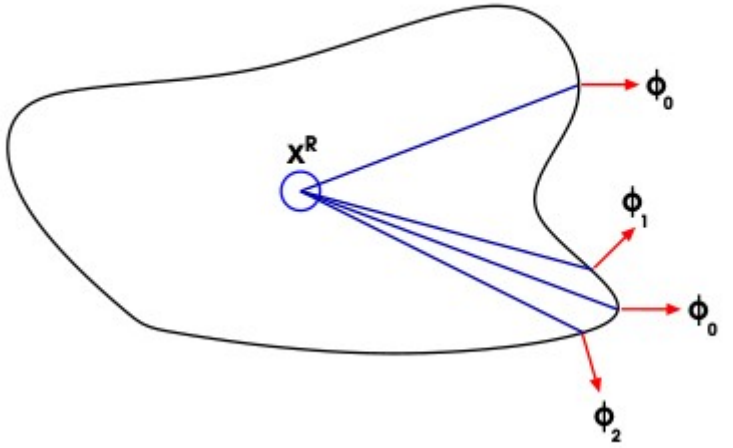
Generalized Hough Transform for non-parametric curves:

Determine R-Table:

- Select an arbitrary reference point x^R
- For each boundary pixel x_i determine distance and direction (r_i, α_i) to the reference point x^R
- Store (r_i, α_i) with respect to the border direction $\Phi \rightarrow$ R-Table

ϕ_1	$(r_1^1, \alpha_1^1), (r_1^2, \alpha_1^2), \ldots, (r_1^{n_1}, \alpha_1^{n_1})$
ϕ_2	$(r_2^1, \alpha_2^1), (r_2^2, \alpha_2^2), \ldots, (r_2^{n_2}, \alpha_2^{n_2})$
ϕ_3	$(r_3^1, \alpha_3^1), (r_3^2, \alpha_3^2), \ldots, (r_3^{n_3}, \alpha_3^{n_3})$
...	...
ϕ_k	$(r_k^1, \alpha_k^1), (r_k^2, \alpha_k^2), \ldots, (r_k^{n_k}, \alpha_k^{n_k})$

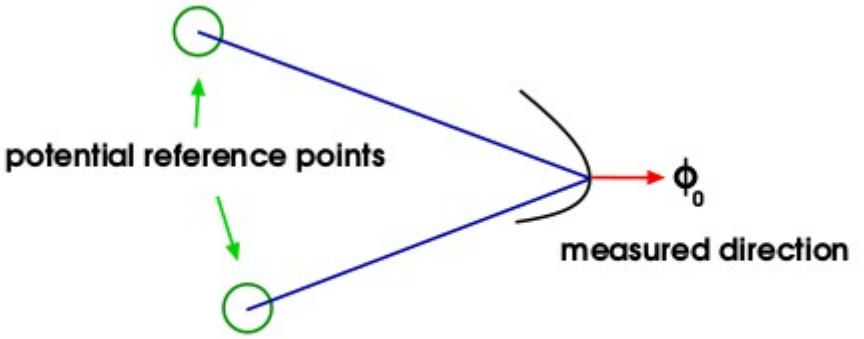
Table 6.1: R-table



Increment Accumulator

In order to detect candidates for reference points of the object the accumulator is increased based on the detected border pixels and their measured direction:

- For edge points with a measured border direction Φ determine (using the R-Table) the relative position of
- increase the accumulator cells for all potential reference points



The maxima of the accumulator array correspond to good candidates for reference points of the shape.

→ Note: In order to cope with arbitrary rotations and with a certain range of scale, the size of the accumulator dimension increases by two

Hough Transform

Crucial tasks of the Hough transform :

Discretization of the parameter space is an important step within the approach.

→ a smaller cell-size (higher sampling) increases the computation time

→ a too large cell-size might not be accurate enough

Find local maxima in the accumulator array is a non-trivial problem

often a single line results in more than one local maximum

→ smoothing the parameter space might lead to an appropriate single local maximum

Hough Transform

- not sensitive to missing (line) parts

→ missing (line) parts only lower the contribution to an accumulator cell

- not sensitive to noise

→ noise leads to a spread of points in the parameter space

- not sensitive to structures co-existing in the image

→ other structures contribute only “randomly“ to accumulator cells and appear as background noise

- can detect multiple locations which represent a good fit
- can cope with partial occlusions of the objects
- only suitable for structures represented by a few parameters

→ accumulator size grows largely with the number of parameters

Literature

Szeliski 3.2, 4.2, 4.3.2.

Gonzalez 3.1-3.6

Corke 12.5, 13.2