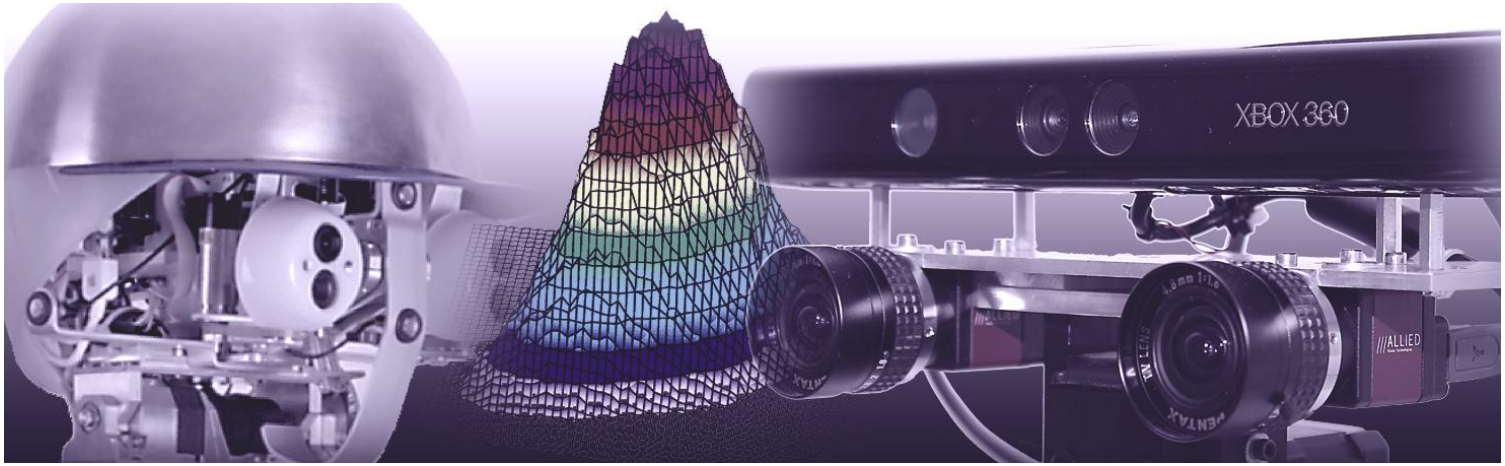
Karlsruhe Institute of Technology

Robotics III: Sensors

Chapter 8: Fundamentals of Image Processing

Tamim Asfour

Department of Informatics, Institute for Anthropomatics and Robotics (IAR)
High Performance Humanoid Technologies Lab (H^2T)



http://www.humanoids.kit.edu

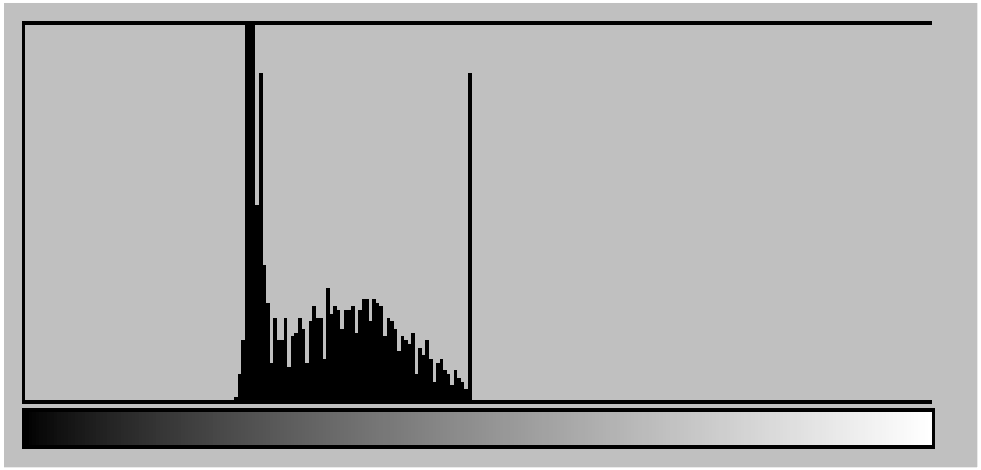
http://h2t.anthropomatik.kit.edu

Content

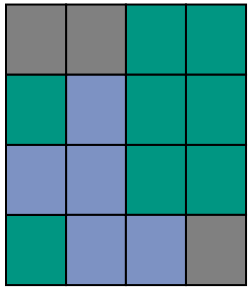
- Histograms
- Filters
  - Smoothing filter
  - Edge filter
- Segmentation
  - Color segmentation
  - Hough-Transformation
    - Lines
    - Curves
  - Generalized Hough-Transformation

Histograms

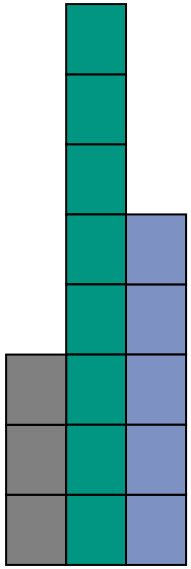
- Histogram function: indicates the frequency of a selection characteristic (pixels, features, etc.)
- Common features: colors or gray values



Histograms II

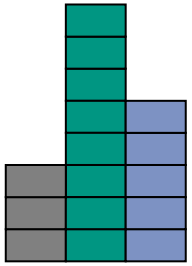


Image



(3, 8, 5)

Histogram



(0.18, 0.5, 0.32)

Normalized Histogram

$$h(r_k) = n_k$$

$$k = 0, 1, 2, \cdots, L-1$$

L is the number of possible intensity level.

$$L = 2^{BitNumber}$$

$$p(r_k) = \frac{n_k}{MN}$$

$$\sum_{k=0}^{L-1} p(r_k) = 1$$

Normalized histogram is a probability density function and indicates the probability of occurrence of intensity level in an image!

Histograms III

- ***Histogram equalization:*** It is a method that improves the contrast in an image, in order to stretch out the intensity range, no information gain
- Automatic contrast and brightness adjustment by histogram analysis

- The cumulative distribution function (CDF) must be computed as:

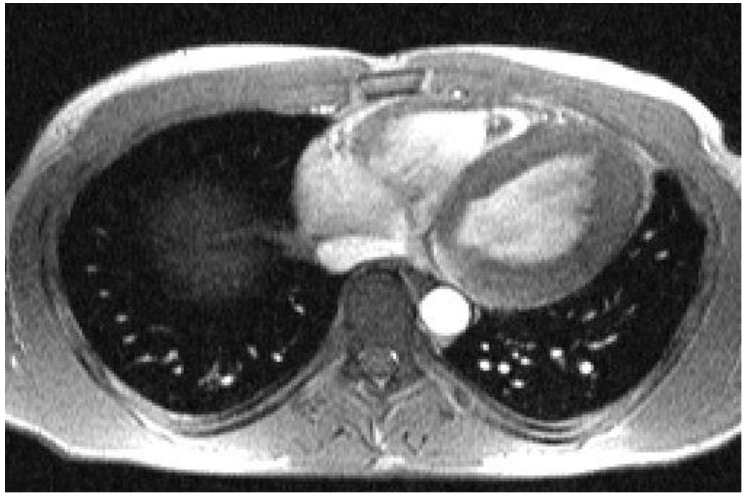
$$H'(i) = \sum_{0 \le j < i} H(j)$$

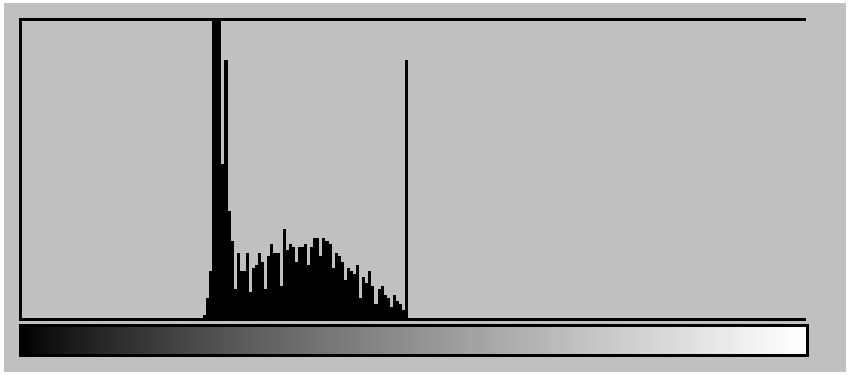
- The normalized CDF is used as a mapping function;

$$equalized\,(x, y) = H'\,(src(x, y))$$

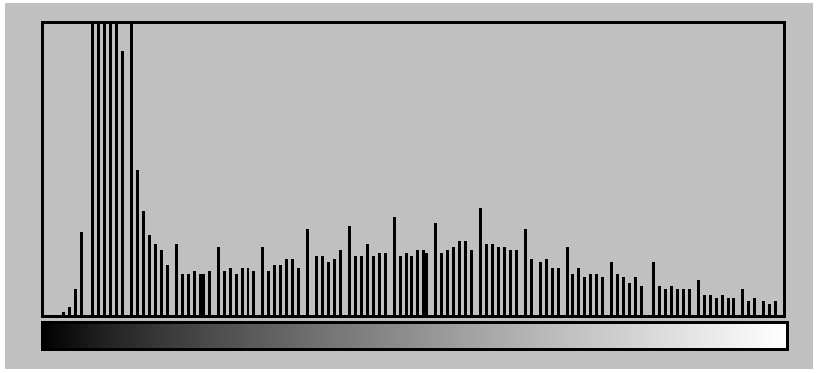
Histograms IV

Equalization





H_{Min} H_{Max}



H_{Min} H_{Max}

Filters I

- Digital filters are used in image processing for the preprocessing of images

- Typical applications
  - Image enhancement
    - Elimination of outliers (impulse noise)
    - Elimination of Gaussian noise
  - Extraction of edge information
    - First derivative of the image function
    - Second derivative of the image function

Filters II

- The application of a digital filter can be formulated in
  - Local area (Spatial Filtering)
    - Operates directly on the image data
    - Filtering the image by **convolution** with a corresponding filter matrix (**kernel**)
  - Frequency domain
    - Image is transformed into the frequency domain by the Fourier transform
    - Filtering the image by **point-by-point multiplication** with a corresponding filter matrix
    - After filtering , the resulting image is calculated by back transformation from the frequency domain

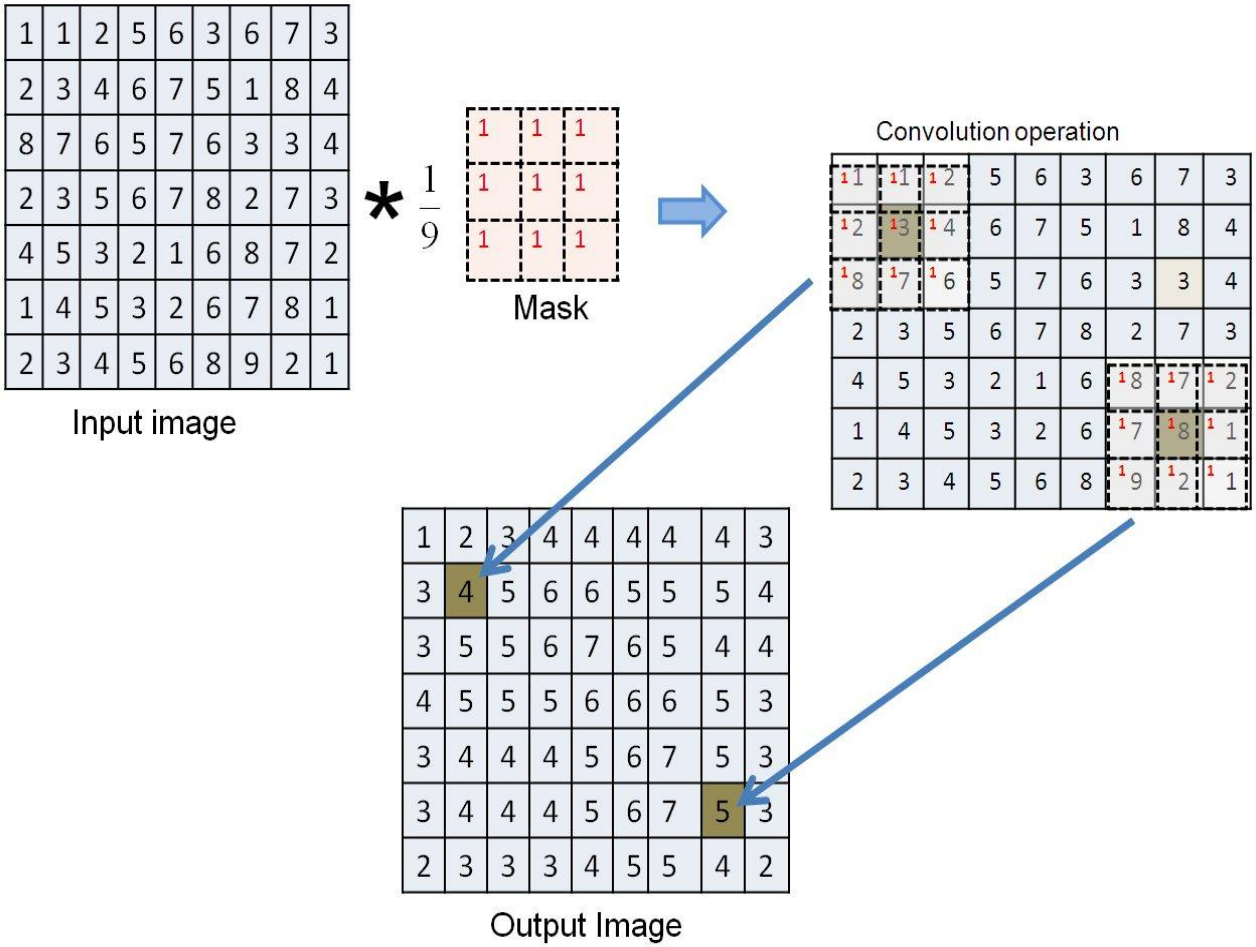
Filters III

- Below: Filtering in the local area (Spatial Filtering)

- As a rule, square filter matrices with odd edge length n = 2k + 1 are used
- Filter matrix (kernel) F determines the type of the filter
- Convolution of F and image I occurs as follows :

$$I'(x, y) = F(x, y) * I(x, y) := \sum_{i=-k}^{k} \sum_{j=-k}^{k} F(i, j) \cdot I(x + i, y + j)$$

Filters IV

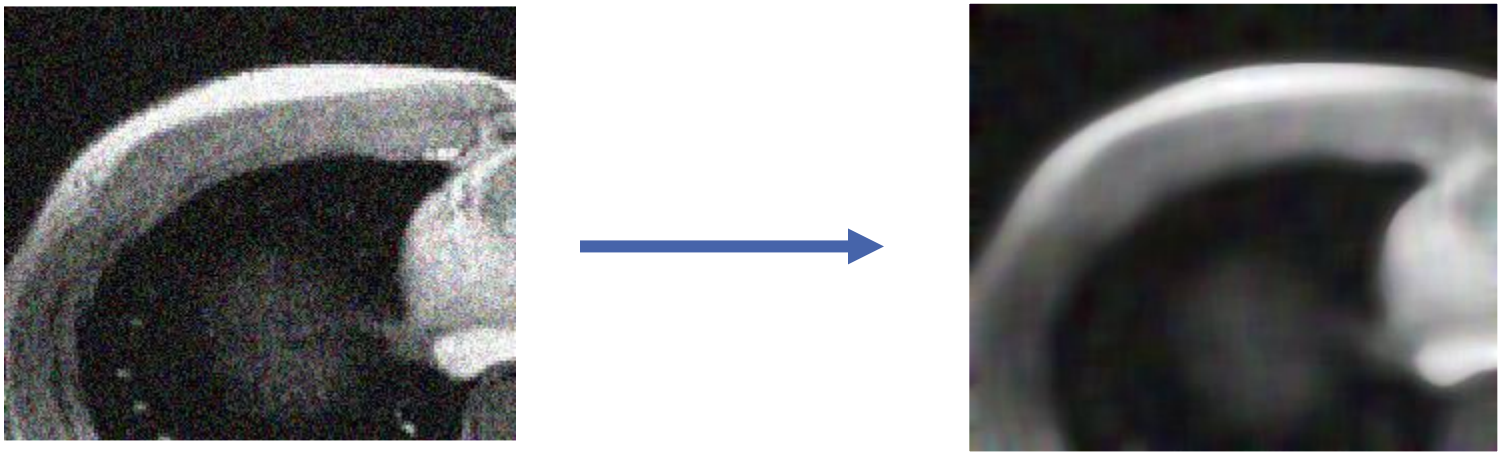

1 1 2 5 6 3 6 7 3
2 3 4 6 7 5 1 8 4
8 7 6 5 7 6 3 3 4
2 3 5 6 7 8 2 7 3
4 5 3 2 1 6 8 7 2
1 4 5 3 2 6 7 8 1
2 3 4 5 6 8 9 2 1
Input image
* 1/9
1 1 1
1 1 1
1 1 1
Mask
Convolution operation
1 2 3 4 4 4 4 4 3
3 4 5 6 6 5 5 5 4
3 5 5 6 7 6 5 4 4
4 5 5 5 6 6 6 5 3
3 4 4 4 5 6 7 5 3
3 4 4 4 5 6 7 5 3
2 3 3 3 4 5 5 4 2
Output Image

Filter Operations

- Low pass filter: smoothing, noise elimination
  - Median filter
  - Average filter
  - Gaussian filter
- High-pass filter: edge detection
  - Prewitt
  - Sobel
  - Laplace
  - Laplacian of Gaussian

Median Filter

- Nonlinear filter for noise suppression
- Filter response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter
- Steps:
  - Choose kernel size to define the pixel to be filtered.
  - Sort out all gray values within it.
  - Average gray value from the sorted pixels
  - Select the pixel as a new value.



Median Filter: Example

2	4	3
5	9	2
7	2	4

2,4,3,5,9,2,7,2,4

2,2,2,3,4,4,5,7,9

4 will be chosen out of 9

New:

2	4	3
5	4	2
7	2	4

Median Filter: Example

- Suitable for salt-and-pepper noise
- Not suitable for Gaussian noise
- Preserves edges while removing noise in the image.

Average Filter

- Objective: Suppression with linear filter. Known as a smoothing linear filter. The output is the average of the pixels contained in the neighborhood of the filter mask.

- Example:

$$\frac{1}{9} \cdot \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \rightarrow \text{Average}$$

- Size can vary and the effect increases with size.

$$\sum_{j=-N}^{N} \sum_{i=-N}^{N} p(x+j, y+i) \cdot m(i, j) = \sum \frac{1}{9} \cdot p(x, y)$$

Average Filter II

- Average filters reduce sharp transition (e.g. **random noise**) in intensities, so good for noise reduction.
- Averaging filters are **linear filters** but have the undesirable side effect that they blur **edges**! (Smoothing suppresses edges in the image!)
- **Weighted average filters** are preferred to reduce blurring in the smoothing process.
- Good results with Gaussian noise.
- Not robust against outliers.

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

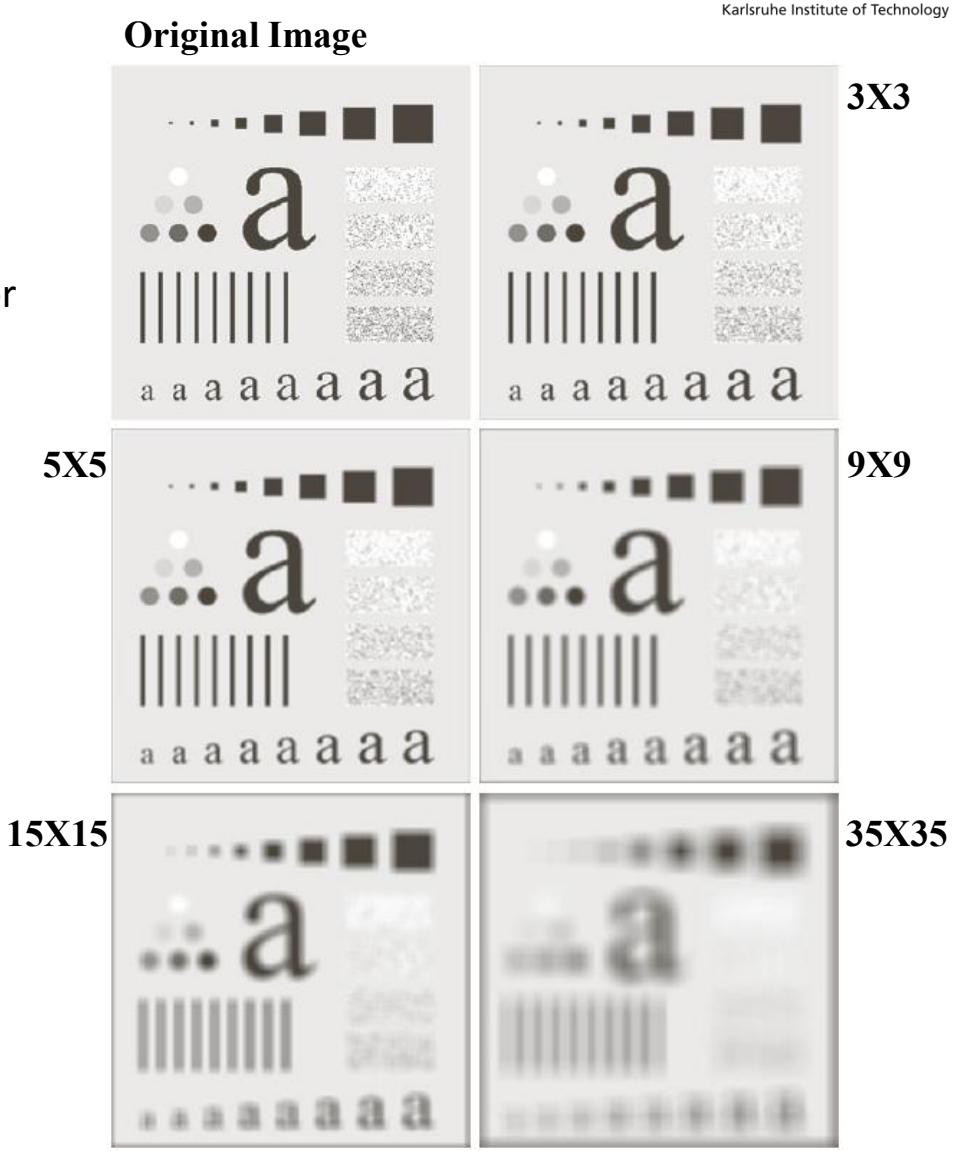
Weighted Average

Average Filter III

- The smoothing effect increases w.r.t. filter size!
- Aggressive blurring as observed in results for 15x15 and 35X35 is generally used to eliminate small objects in the image.
- The black border in the result for 35X35 is due to padding the border of the original image with 0.

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Averaging filter (Box filter)



Gaussian Filter

- Aim: Noise suppression with linear filter
- Defined by two-dimensional Gaussian function
  (Left: local area, right: frequency range):

$$f(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \longleftrightarrow \quad F(u,v) = e^{-\frac{u^2+v^2}{2}\sigma^2}$$

- Approximation of *f(x)* (also for the application in the local area) by a 3×3-Filter for $\sigma = 0{,}85$:

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

Gaussian Filter II

- The strength of the smoothing can be varied with the variance parameter σ:

  The larger the variance σ, the stronger the smoothing.

- A Gaussian filter with a larger filter mask achieves a better approximation of the Gauss function, but does not lead to a differing smoothing behavior.

- In the local area, a rule of thumb is used for a sufficient approximation:
  $n = |2\sigma| \cdot 2 + 1$ (n is the size!)

Gaussian Filter Example

Original Image

Gaussian Filter Example

Result after application of the Gaussian filter with $\sigma = 1$

Gaussian Filter Example

Result after application of the Gaussian filter with $\sigma^2 = 2$

Gaussian Filter Example

Result after application of the Gaussian filter with $\sigma^2 = 4$

Gaussian Filter Example

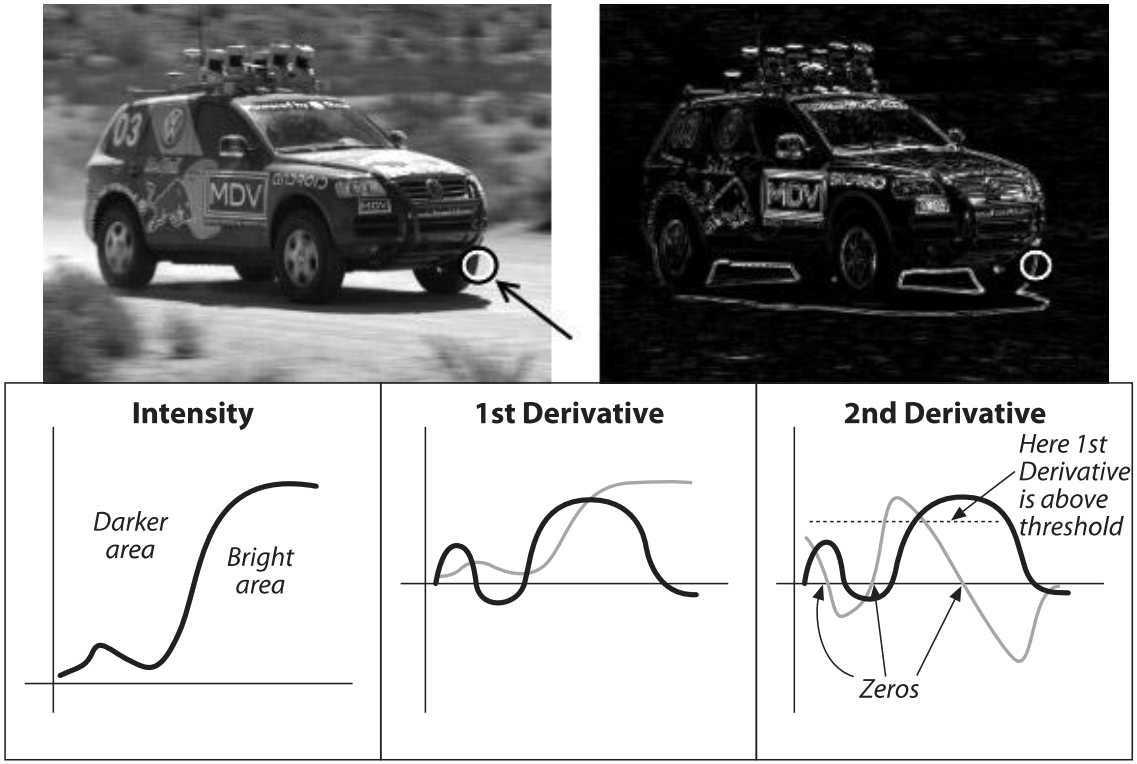
Result after application of the Gaussian filter with $\sigma^2 = 8$

Gaussian Filter Example

Result after application of the Gaussian filter with $\sigma^2 = 16$

Edge Detection

- Edges correspond to discontinuities in the image function.
  - A peak in the first derivative of the image function.
  - Zero crossing in the second derivative of the image function.



[Bradski 08]

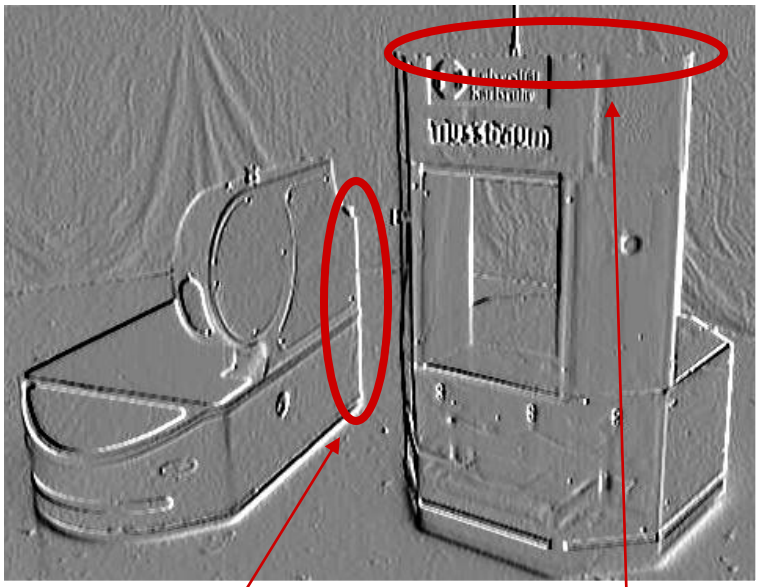
Filtering – Prewitt

Prewitt-X Filter

$$P_x = \frac{\partial g(x,y)}{\partial x}$$

- Detection of vertical edges, i.e. calculating gradients in horizontal (x) direction

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



Good vertical edge detection

Bad horizontal

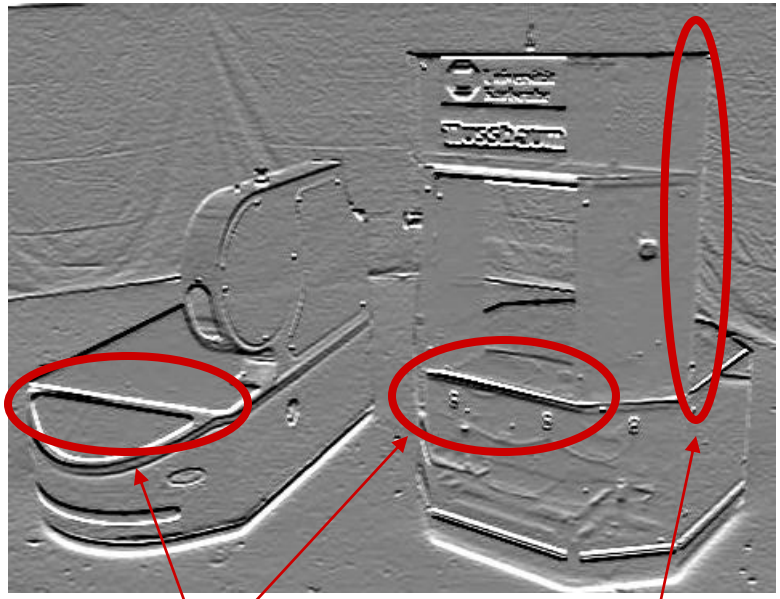
Filtering – Prewitt II

Prewitt-Y Filter

$$P_y = \frac{\partial g(x,y)}{\partial y}$$

- Detection of horizontal edges, i.e. calculating gradients in vertical (y) direction

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



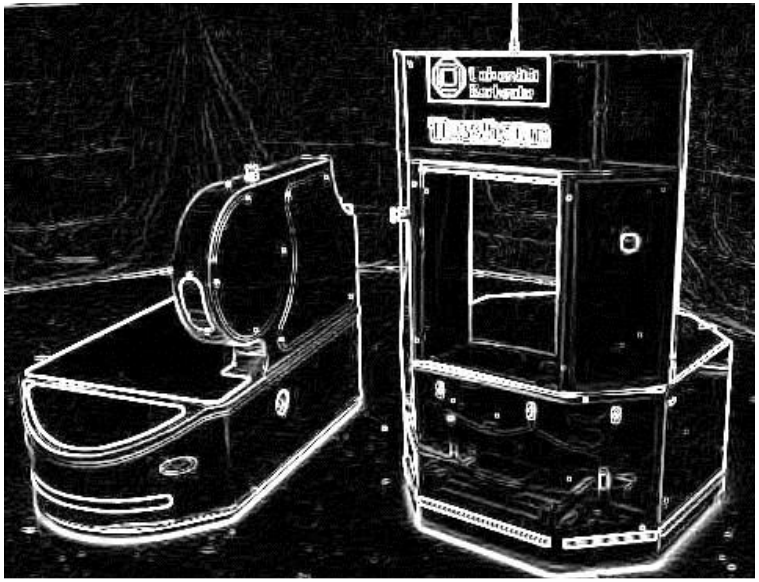
edge detection

Filtering – Prewitt III

Prewitt-Operator

- Combination of Prewitt filters to determine the gradient magnitude M

$$M \approx \sqrt{P_x^2 + P_y^2}$$



- Then: Threshold filtering for separating the edge pixels from image.

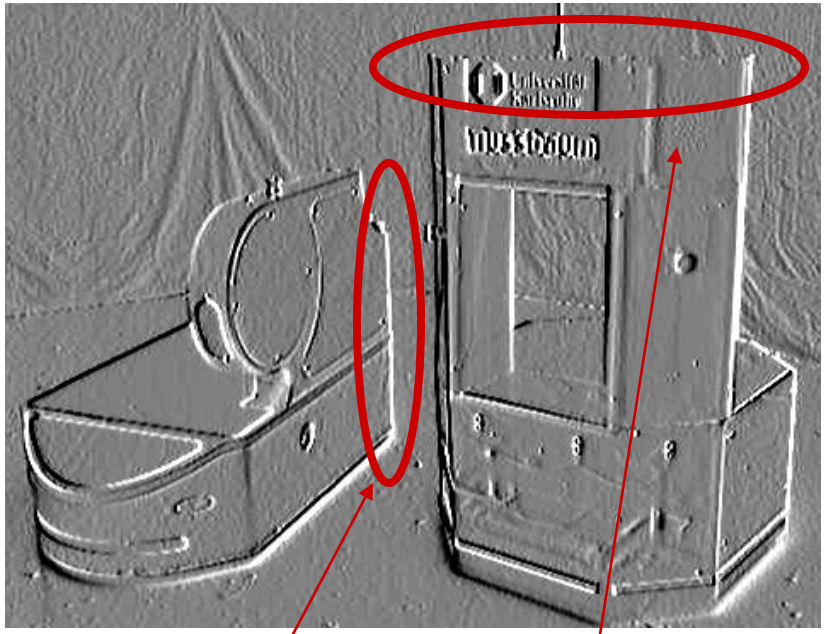
Filtering – Sobel

Sobel-X Filter

$$S_x = \frac{\partial g(x,y)}{\partial x}$$

- Detection of vertical edges, i.e. calculating gradients in horizontal (x) direction

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



Good vertical edge detection

Bad horizontal

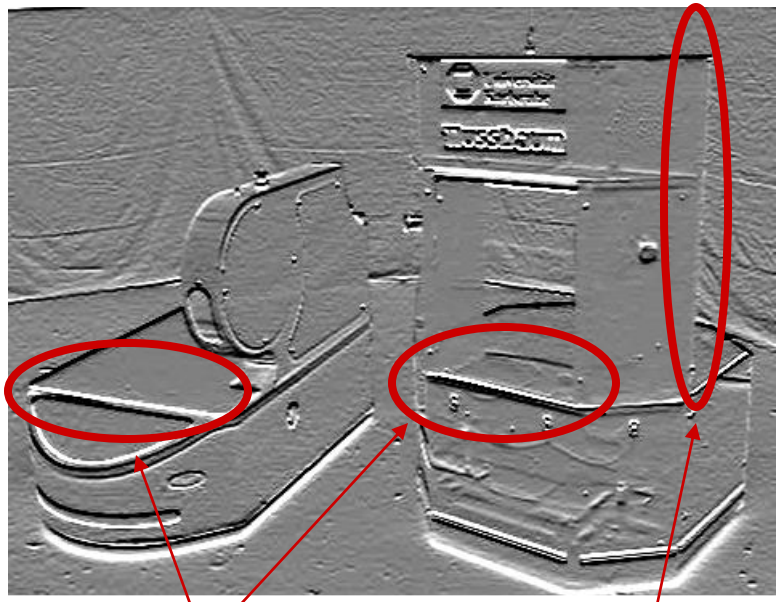
Filtering – Sobel II

Sobel-Y Filter

$$S_y = \frac{\partial g(x,y)}{\partial y}$$

- Detection of horizontal edges, i.e. calculating gradients in vertical (y) direction

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



Good horizontal edge detection

Bad vertical

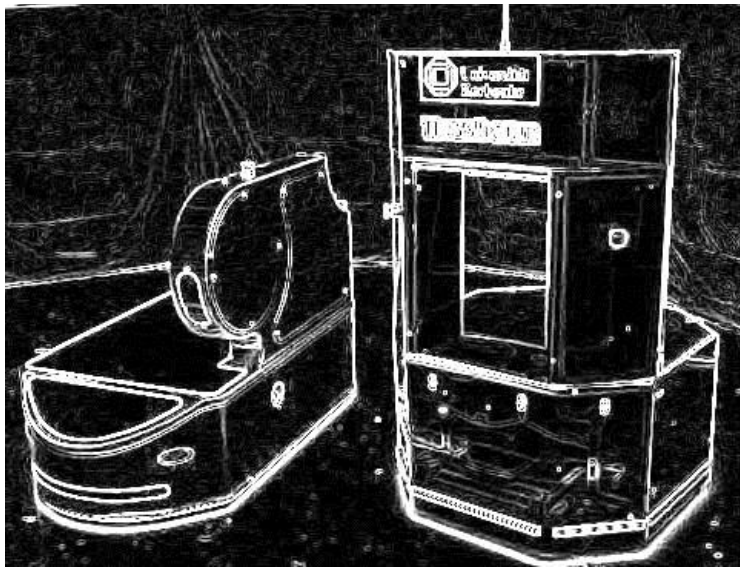
Filtering – Sobel III

Sobel-Operator

- Combination of the Sobel filters to determine the gradient magnitude M

$$M \approx \sqrt{S_x^2 + S_y^2}$$

- Then: Threshold filtering for separating the edge pixels from image.



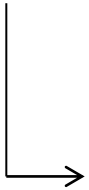
Filtering – Laplace

- **Laplacian** of an image is a linear and rotation invariant second-order derivative operator.

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

$$\begin{cases} \dfrac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \\ \dfrac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \end{cases}$$

$$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$



$f(x-1,y-1)$	$f(x-1,y)$	$f(x-1,y+1)$
$f(x,y-1)$	$f(x,y)$	$f(x,y+1)$
$f(x+1,y-1)$	$f(x+1,y)$	$f(x+1,y+1)$

0	1	0
1	−4	1
0	1	0

Filter Mask
(invariant to rotations in 90°)

1	1	1
1	−8	1
1	1	1

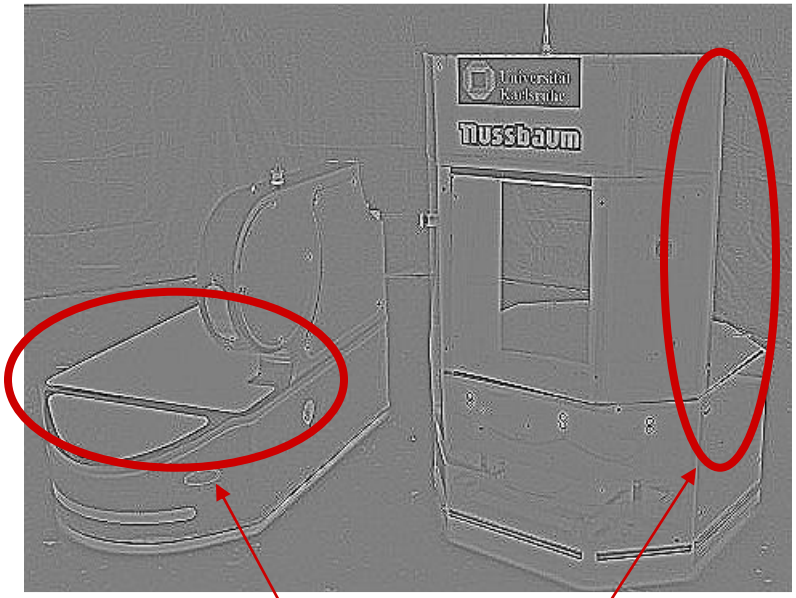
Extended Mask
(invariant to rotations in 45°)

Note that the coefficients of the mask **sum to zero**, indicating that the mask response will be zero in the areas of **constant intensity**.

Filtering – Laplace II

Laplace-Operator:

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



$$\nabla^2 \approx \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

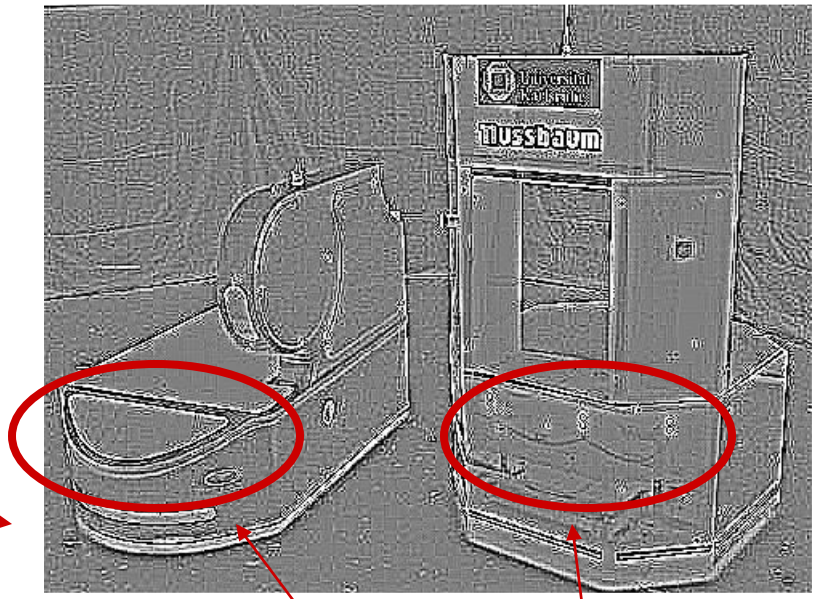
Zero-crossings defines edges

Edges are thinner than in Prewitt or Sobel.

Filtering – Laplace III

Variation of the Laplace-Operator:

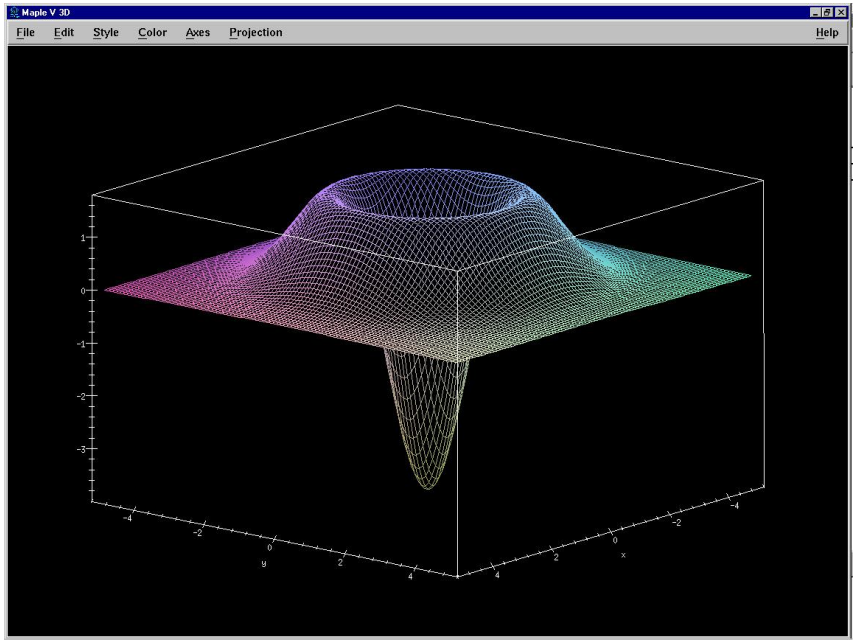
$$\nabla^2 \approx \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$



Stronger, but more jamming edges

Filtering – LoG

- The Laplace operator is very sensitive to noise.
- Essentially better results are obtained by smoothing the image with a Gauss filter and then using the Laplace operator (Laplacian of Gaussian, LoG):



$$LoG(g(x,y)) = \nabla^2(f(x,y) * g(x,y))$$

F denotes the filter function of a Gaussian filter

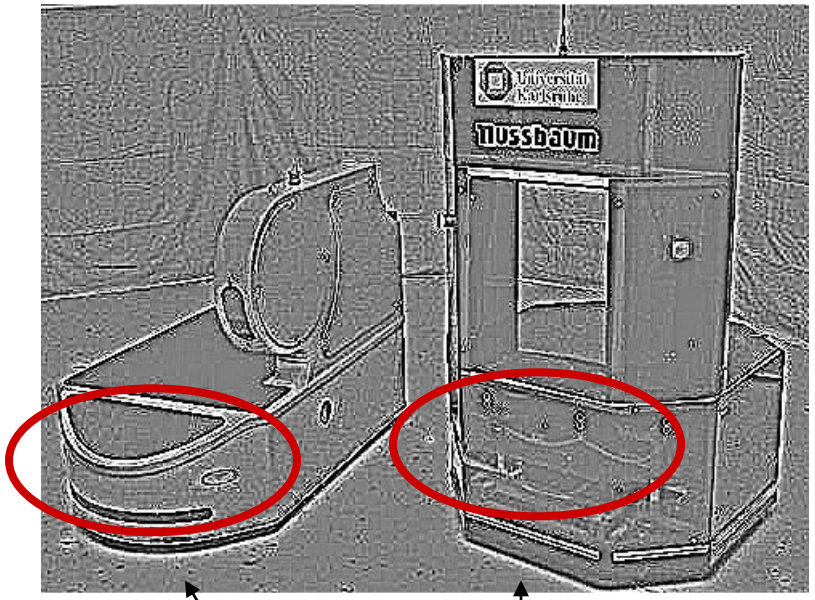
Filtering – LoG

- **Key features** of the **LoG** operator:
  - The **Gaussian** part of the operator is a **low pass** filter that **blurs** (smoothing) the image, thus reducing the intensity of structures (e.g. noise) at scales much smaller than sigma, σ.
  - Unlike the averaging filter, the Gaussian function is less likely to introduce **artifacts** such as “staircase‘” like effects that are not present in the original image.
  - The **Laplacian** (the second derivative) is **invariant to rotation** which responds equally to changes in intensity in any mask direction. This way we can avoid using multiple masks to calculate the strongest response at any point in the image.
  - **Zero-crossings** of the Laplacian corresponds to **edges**.

Approximation

Convolution with Matrix

$$\nabla^2 F(x,y) = \begin{pmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{pmatrix}$$



Stronger edges,

Less noise

Canny Edge Detector I

- John F. Canny from 1986
- The goal was to find the "optimal" edge detector:
  - Good detection of correct contour points
  - Less false detection
  - Good localization of the contour
  - Minimum response ("thin lines")
- The Canny operator combines the beneficial properties of Laplace with those of the Sobel Operator without adapting their weaknesses
- Canny edge detector calculates binary response

  (usually 0: no edge, 255: edge)
- Algorithm consists of several steps

Canny Edge Detector II

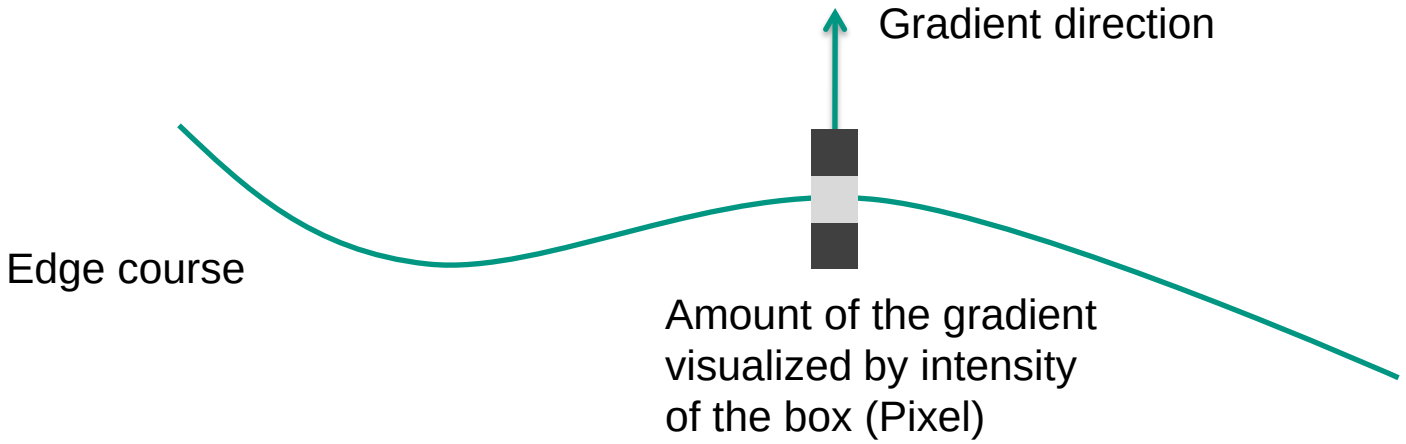
- Steps of the algorithm
  1. Noise Reduction: Gaussian filter
  2. Calculation of the gradients in the horizontal and vertical directions with Prewitt_X / Prewitt_Y oder Sobel_X / Sobel_Y
     - Calculation of gradient strength: $|| g ||$
     - Calculation of gradient direction: $\phi = \text{atan}(g_y / g_x)$
     - Classification of direction according to neighborhood relations to surrounding pixels:
       1 : [-67.5°, -22.5°), 2 : [-22.5°, 22.5°),
       3 : [22.5°, 67.5°), 4 : [-90°, -67.5°) oder [67.5°, 90°]
  3. Non-Maximum Suppression
  4. Hysteresis thresholding

Canny Edge Detector III

- Non-Maximum Suppression

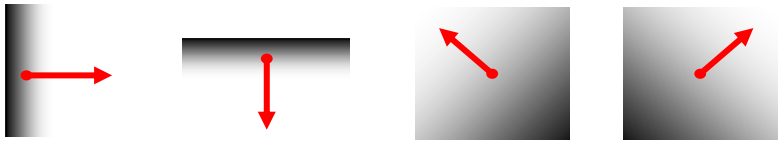
  For a potential edge pixel:

  - Gradient Strength || g || must correspond to the local maximum, considering both direct neighbors along the direction α.
    - Determination of direction based on division into quadrants.
  - Objective: Thinning of edge pulls with width> 1 pixel.



Canny Edge Detector III - Non-Maximum Suppression

We define **four** edge orientations!



Vertical **Horizontal** **+45** **-45**

We define a range of directions to quantize all possible edge directions into four!

Edge normal:

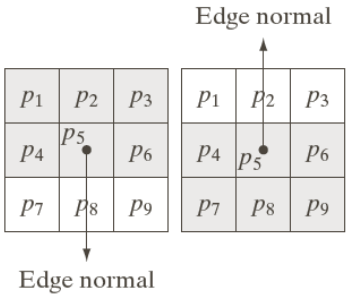
$$a(x,y) = \tan^{-1}\left[\frac{g_y}{g_x}\right]$$

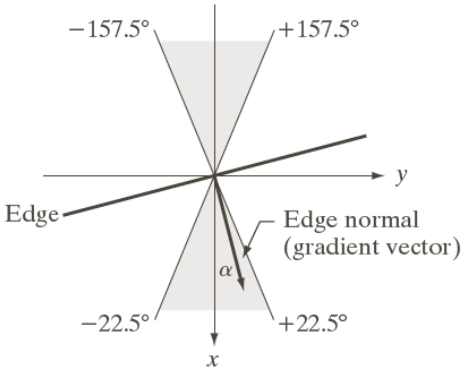
Nonmaxima Suppression:

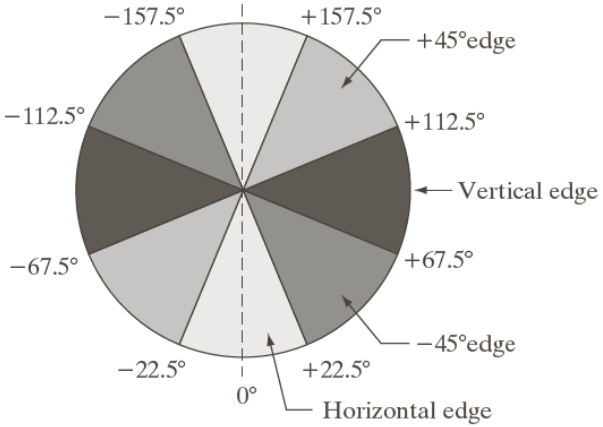
1. Define a 3x3 region centered at every point (x,y) in α(x,y). For instance, this 3x3 region with pixels from *p1* to *p9.*

2. Find the edge direction that is close to α(x,y). (In this example *p5* is horizontal.)

3. If the value at the center of M(x,y) is less than at least one of its two neighbors along the edge normal (gradient) direction (in this case *p2* and *p8*), let M(x,y)=0 (**suppression!**)







Canny Edge Detector IV

- Hysteresis thresholding
  - Use of two thresholds: „low“ / „high“
  - If the magnitude of the gradient for a pixel is above "high", it is accepted as an edge regardless of its neighbors
  - Across the accepted pixel, recursion (recursive) is searched for in the image:
    - Check the eight direct neighbors of the pixel
    - If their gradient strength is "low", they are also accepted edge pixels.

  - Ambient Adaptive Threshold Method!
  - Isolated pixels with gradient strength over "low" are discarded.
  - Pixels in edge line with element beyond "high" are retained.

Canny Edge Detector V

- Example

Input image

Output image

Segmentation

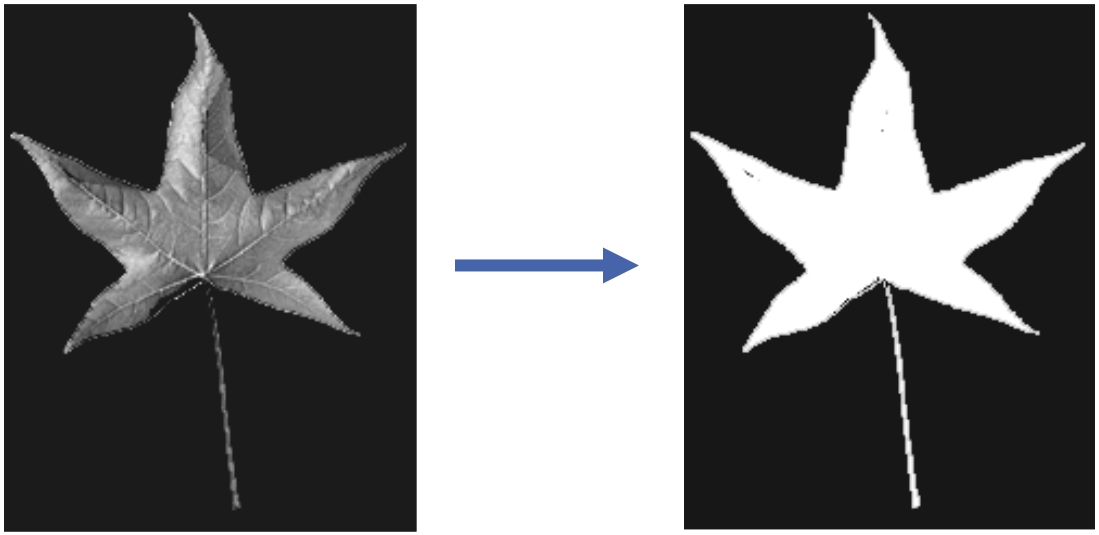
- Segmentation is the division of an image into meaningful elements
- Allowed:
  - Statements about the picture
  - Reduction of the data volume
- Common among others are:
  - Texture, color (area content)
  - Contours, corners (area boundaries)

Segmentation: Threshold filtering

- Threshold filtering for converting a gray value image into a binary image
- Objective:

  Separation of interesting objects from the background

$$\text{Img}'(u,v) = \begin{cases} q & \text{if } \text{Img}(u,v) \geq T \\ 0 & \text{otherwise} \end{cases}$$



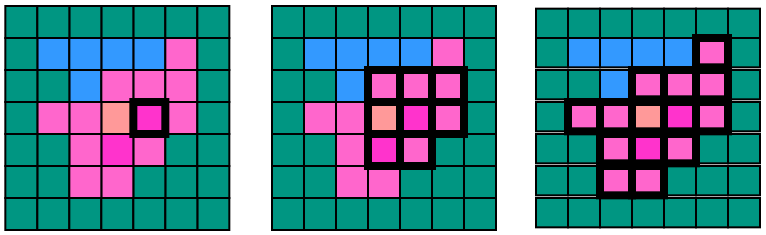
Segmentation: Region Growing

Given: Grayscale image

Output: Related regions

Algorithm in Pseudocode:

1. Select seed point $\boldsymbol{p_0} = (u_0, v_0)$
2. Initialize Region $R = \{\boldsymbol{p_0}\}$, select threshold ε
3. while $\exists\, \boldsymbol{p} \in R, \boldsymbol{q} \notin R$ with $\|\boldsymbol{p}\text{-}\boldsymbol{q}\| \leq 1$ und $|\mathrm{Img}(\boldsymbol{p_0})\text{-}\mathrm{Img}(\boldsymbol{q})| \leq \varepsilon$: $R = R \cup \{\boldsymbol{q}\}$



RGB – Color Model

$$\text{Img} : [0..m] \times [0..n] \rightarrow [0..R] \times [0..G] \times [0..B]$$

$$(u,v) \mapsto \text{Img}(u,v) = (r,g,b)$$

- Additive colors
- 3 color values red, green, blue

  often: $256 \times 256 \times 256$ Nuances =16,8 Mio.

  (8 Bit) „True color“
- Often used by frame grabbers and graphics cards

HSI Color Model

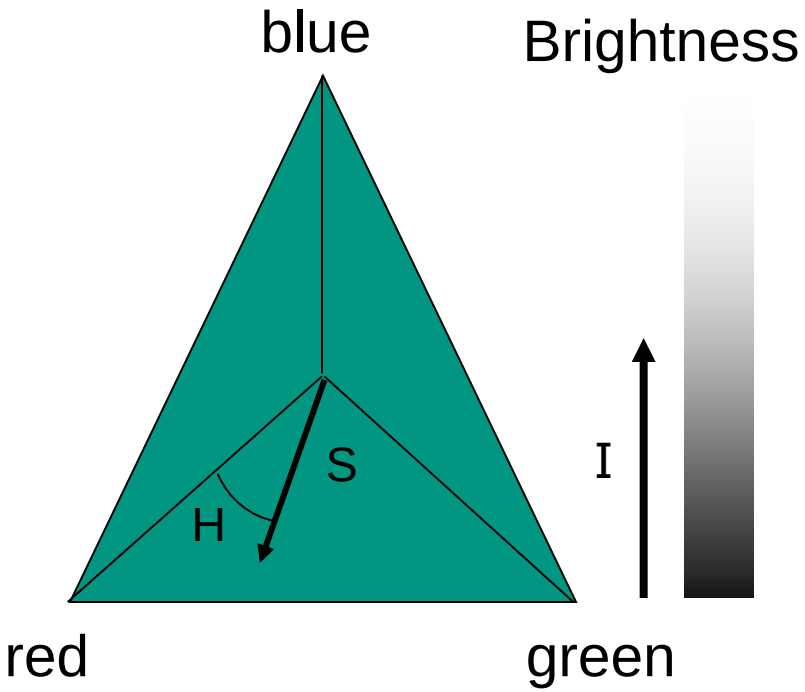
- Hue (color nuance), Saturation (saturation), Intensity (brightness)
- Separates brightness from the color value ⇨ insensitive lighting changes
- Conversion from RGB to HSI (if R = G = B, then H is undefined; if R = G = B = 0, then S is undefined):

$$c = \arccos \frac{2R - G - B}{2\sqrt{(R-G)^2 + (R-B)(G-B)}}$$

$$H = \begin{cases} c & \text{if } B < G \\ 360^\circ - c & \text{otherwise} \end{cases}$$

$$S = 1 - \frac{3}{R + G + B} \min(R, G, B)$$

$$I = \frac{1}{3}(R + G + B)$$

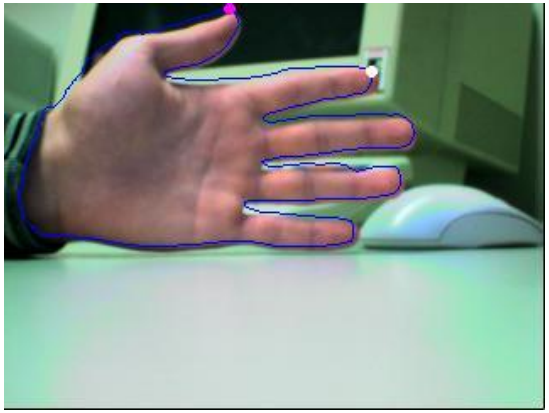


Color Segmentation I

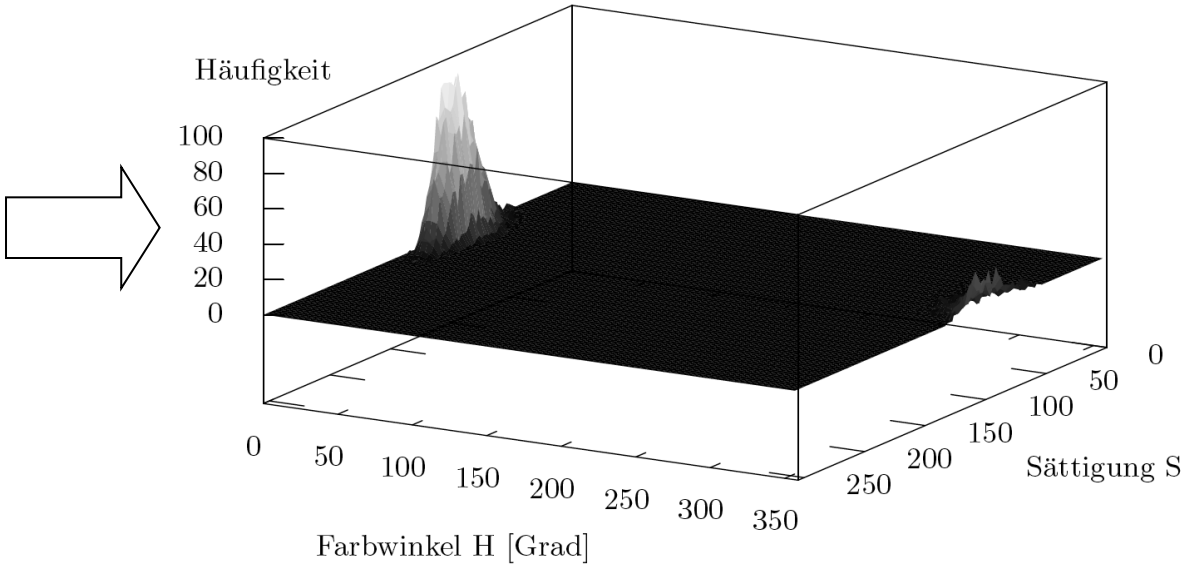
- Objects can often be segmented by their color:
  - Human skin
  - Uniformly colored objects
- Problem:
  - Changing light conditions
  - Reflections, shadows
- Procedure:
  - Histogram-based (e.g. in RGB, HSV or RG, HS)
  - With the help of the Mahalanobis distance (e.g. in RGB)
  - Interval restrictions in the HSV color space

Color Segmentation II

- HS-color histogram
  - Omitting the I-channel results in a 2D histogram
  - Training a classifier on the histogram
  - Example Skin color:



Manually classified training image



Color Segmentation III

- Interval restrictions in the HSV color space :

$$f(H,S,V) = H \geq H_{\min} \text{ AND } H \leq H_{\max} \text{ AND } S \geq S_{\min}$$
$$\text{AND } S \leq S_{\max} \text{ AND } V \geq V_{\min} \text{ AND } V \leq V_{\max}$$

- Based on the Mahalanobis distance :
  - Given: $x_i = (R_i \quad G_i \quad B_i)^T$ are manually classified positively

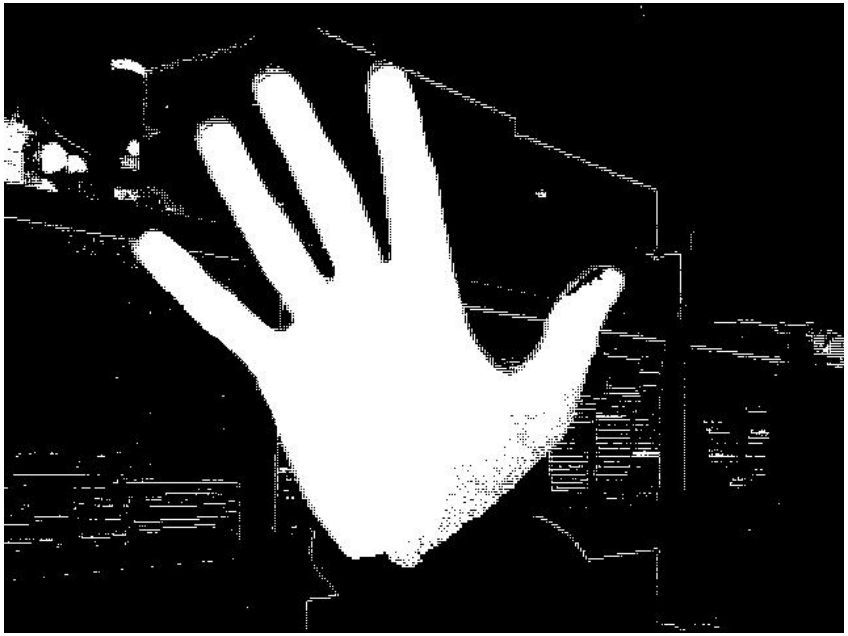
$$C = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$$ covariance matrix

$$p(\boldsymbol{x}) = \exp\left(-\frac{1}{2\sigma^2}\boldsymbol{x}^T C^{-1}\boldsymbol{x}\right)$$ Calculation of color likelihood

Color Segmentation IV

- Example

Input image



Output: Segmented image

Hough Transformation I

- Paul V. C. Hough: Patent „Method and Means for Recognizing Complex Patterns“ in 1962

- Idea:
  - Each pixel is transformed into the so-called Hough space
  - Hits in the Hough space are accumulated in so-called bins
  - The Hough space represents a parametric representation of the desired geometric structure
  - Instances of the desired geometric structure are identified by maxima in the Hough space

Hough Transformation II

- Standard applications
  - Detection of straight lines
  - Detection of circles

- General approach
  - Voting
    - Calculation of image points on edges
      (E.g., by using the Canny edge detector)
    - Transforming each such pixel into the Hough space, discretizing the parameters, and incrementing the corresponding bin
  - Detection
    - Maxima in the Hough area
    - Back transformation of the parameters of the maxima

Hough Transformation III

- Examples of voting formulas

- Line:
  - Hough-Space: r, θ $\quad r = x \cos \theta + y \sin \theta \quad (\theta \in [0, \pi))$

- Circles:
  - Hough-Space: u_m, v_m, r

$$u_m = u - r \cos \theta$$

$$v_m = v \pm r \sin \theta$$

$$\theta \in [0, \pi)$$

Hough Transformation IV

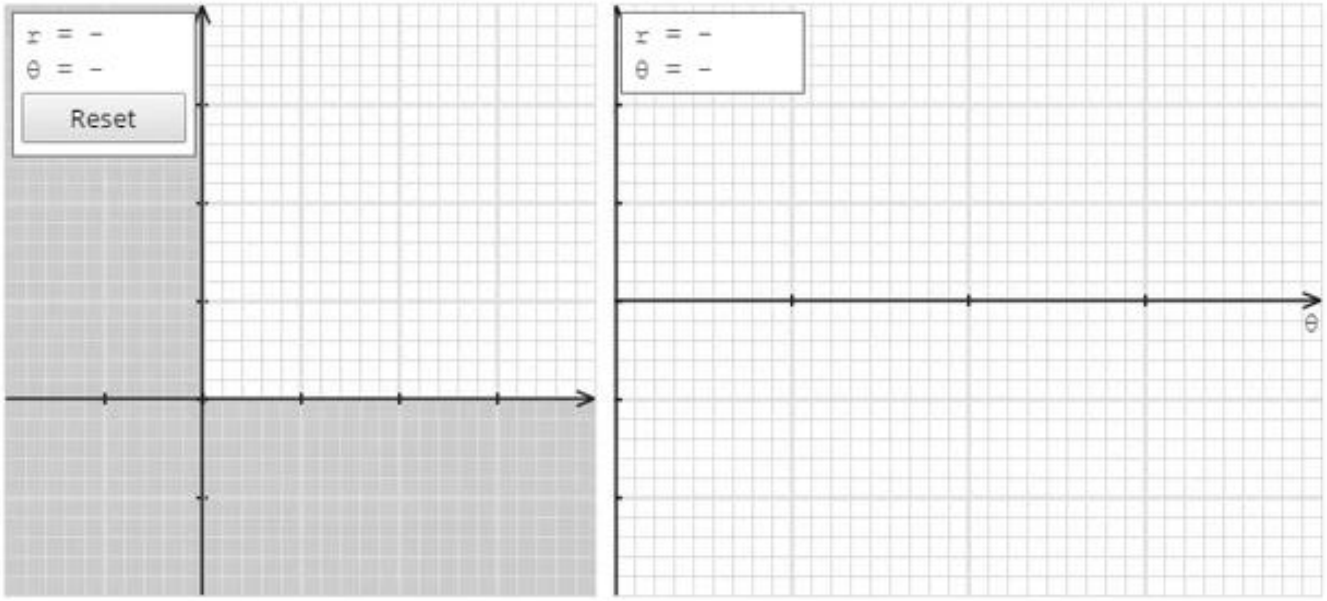


Image Space **Hough Parameter Space**

- The Hough transform represent lines in the parameter space, r and θ. The family of lines that passes through a particular point represents a sinusoidal curve in the parameter space.

Hough Transformation V

Basic Hough transform algorithm:

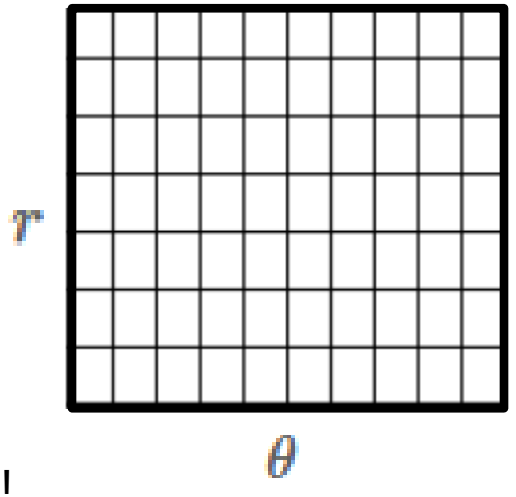
Using the polar parameterization: $r = x \cdot cos(\theta) + y \cdot sin(\theta)$

1. Initialize the parameter space H[r, θ]=0

2. For each edge point I[x,y] in the image

for θ= [θ min to θ max] // some quantization
r=xcos(θ)+ysin(θ)
H[r, θ] += 1

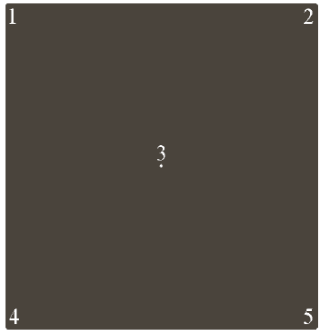
Each point is now a sinusoid in parameter space!



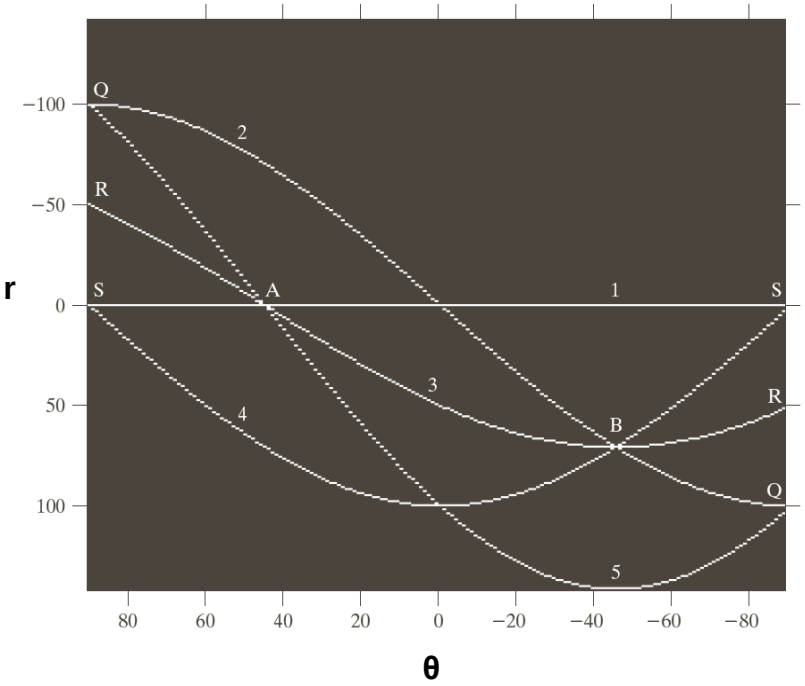
3. Find the value(s) of (r, θ) where H[r, θ] is **maximum.**
The detected line in the image is given by maxima in the parameter space

Hough Transformation VI

Binary image of size 101x101 pixels
Containing 5 points.

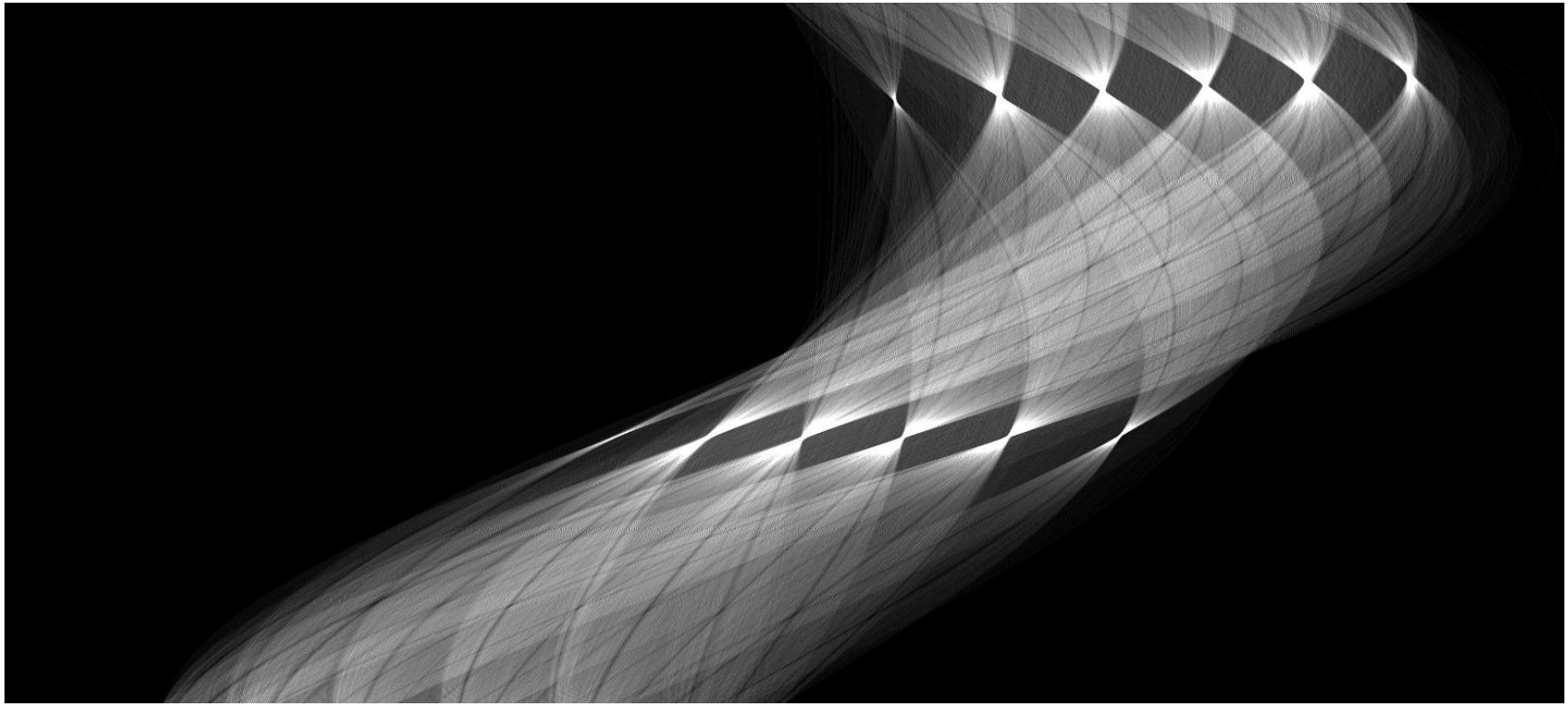


Corresponding parameter space

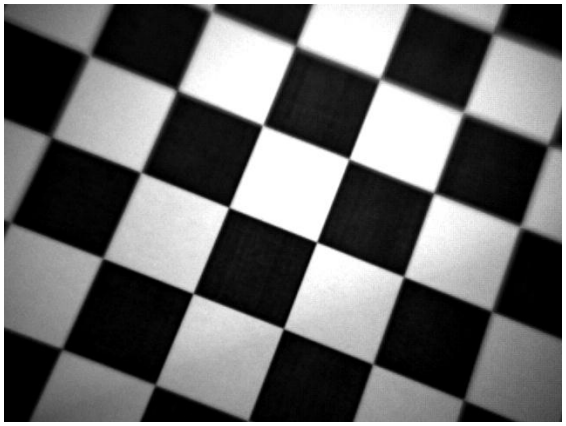
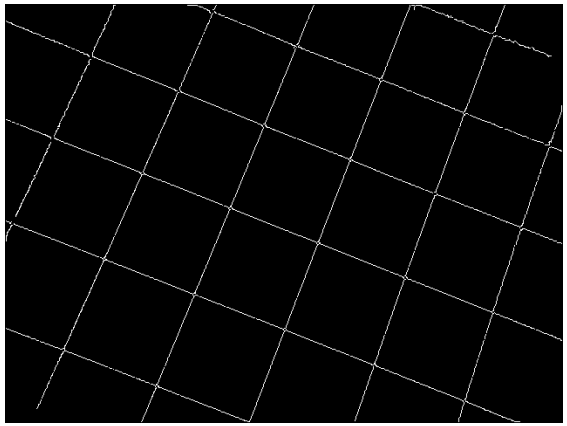
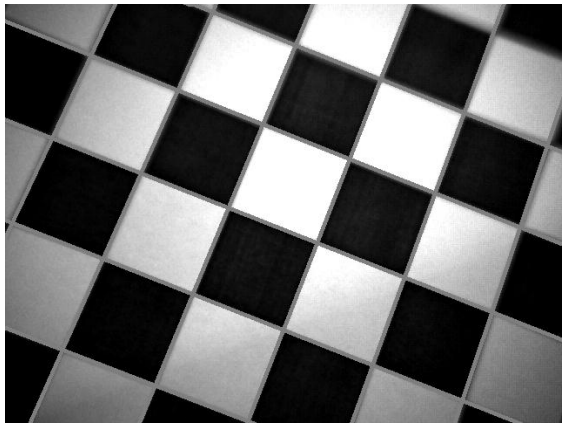
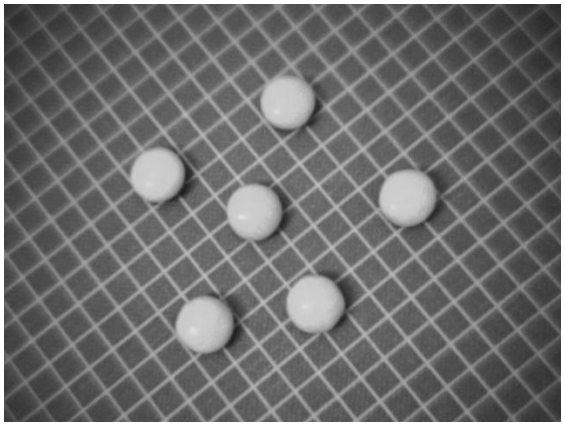
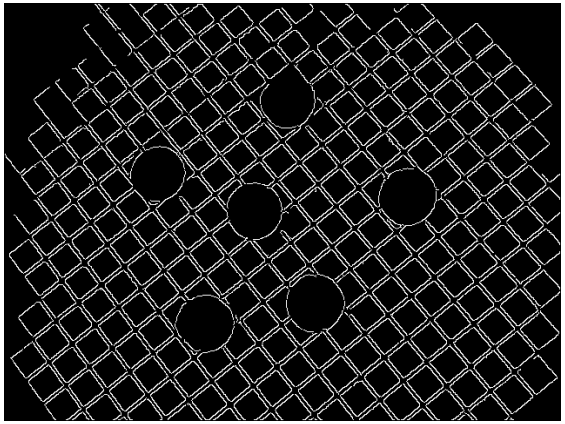
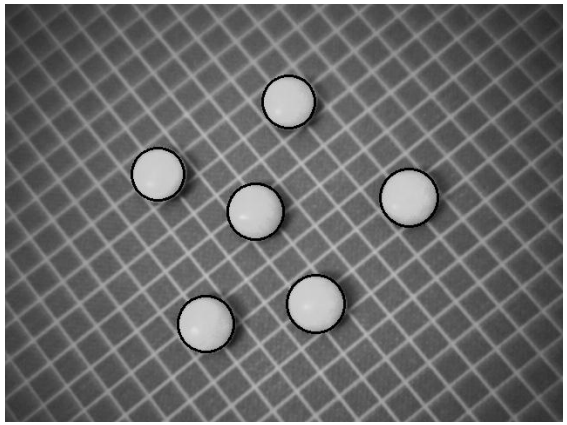


Hough Transformation VII

- Example:
  Visualization of the Hough space for straight after voting



Hough Transformation VIII

Input image

Canny Edges

Detection Results

Generalized Hough Transformation I

- The Hough transform was initially developed to detect analytically defined shapes (e.g., lines, circles, ellipses, etc.)
- In these cases, we have knowledge of the shape and aim to find out its location and orientation in the image.
- The Generalized Hough Transform or GHT, introduced in 1981, is the modification of the Hough Transform using the principle of template matching.
- This modification enables the Hough transform to be used for not only the detection of an object described with an analytic function. Instead, it can also be used to detect an arbitrary object described with its model.
- Generalization of the conventional Hough transformation for the application of any 2D shapes
- Direction of the gradient is also used!
- Wanted: Position of an arbitrarily defined reference point y (rotation known)

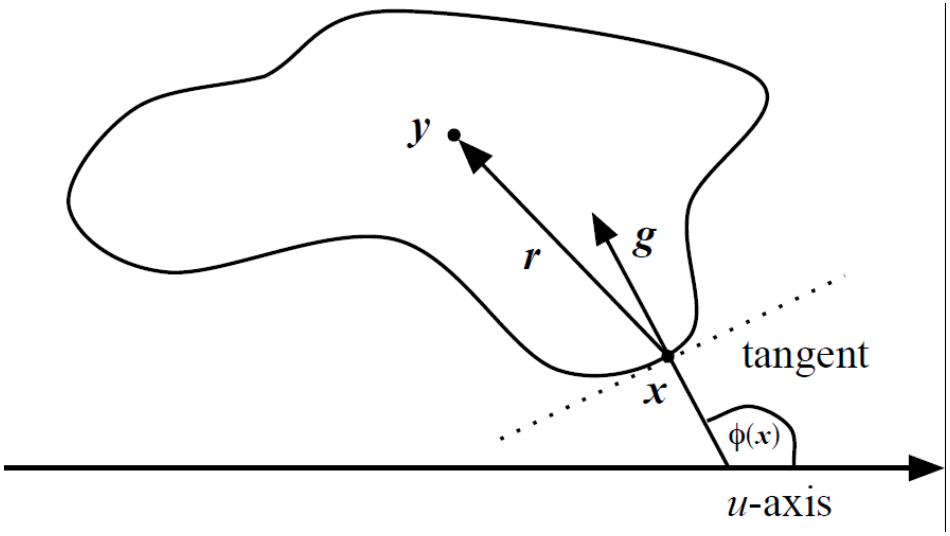
Generalized Hough Transformation II

- Central is the so-called R-Table
- The R-table is indexed by the gradient directions ϕ
- Each entry contains a list of vectors **r** pointing to the reference point (with the corresponding length of the vector)
- If the rotation is known, then only one R-Table is required
- Voting for arbitrary rotations, with several R-Tables (better)
  - Like Voting for constant rotation, however, for each R-Table separately

Generalized Hough Transformation III

- Procedure for calculating the R-Table (Training)
  - For each edge pixel x with Gradient direction $\phi(x)$ apply:

$$R(\phi(x)) = R(\phi(x)) \cup \{ \boldsymbol{y} - \boldsymbol{x} \}$$



  - The following applies (C = amount of edge pixels):

$$R(\phi) = \{ \boldsymbol{r} \mid \exists \boldsymbol{x} \in C : \boldsymbol{y} - \boldsymbol{r} = \boldsymbol{x} \wedge \phi(\boldsymbol{x}) = \phi \}$$

Integrating Vision Toolkit (IVT)

- At the chair Prof. Dillmann developed an image processing library
- Open Source under GPL license
- Project-Homepage: **http://ivt.sourceforge.net**

- Pure ANSI C ++, therefore running under the operating systems Windows, Linux, Mac OS X, etc.
- Own GUI toolkit (implementations for Win32 API, Qt, GTK +, Cocoa)
- Camera interface, generic camera model
- Easy to understand, efficient implementations

Literature

- Histograms
  - Book of Pedram Azad – Chapter 2.5
- Filters
  - Book of Pedram Azad – Chapter 2.6
- Segmentation
  - Book of Pedram Azad – Chapter 2.8
  - Dissertation of Pedram Azad (http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/786884)
    - Chapter 2.2.2.1