Computer Vision

Segmentation – Classic Methods

Technische Hochschule Rosenheim
Winter 2024/25
Prof. Dr. Jochen Schmidt

Segmentation

images: Li, Karpathy, Johnson, CS231n,
lecture 8, Winter 15/16, Stanford

Overview

Technische Hochschule Rosenheim

- Contours
  - Canny edge detection
  - Split & Merge
  - Hough transformation
    - Straight lines
    - Circles
- Regions
  - Split & merge for regions

Result of the segmentation: *Segmentation objects* with

- Attributes
- Relationships with each other

- Examples of segmentation objects:
  - Lines
  - Regions
  - Vertices
  - in 3D: Surfaces and volumes
  - In video sequences: moving objects

- Examples of attributes:
  - Position in the image (qualitative – in front of, behind, ... quantitative – coordinates)
  - Position in world coordinates
  - Size
  - Quality
  - Gray value, color or texture
  - Motion (2D or 3D; qualitative – from top to bottom; quantitative – with x pixel/s, y m/s,...)
  - Shape (qualitative – round, rectangular; quantitative – gradient, surface normal)

Introduction

Technische Hochschule Rosenheim

- Segmentation is based on **changes in the image**
  (there is nothing to segment in an image with a constant gray value)

- But: Changes are only *indications* of segment boundaries, i.e. they are generally
  - not necessary (1) and
  - not sufficient (2) for segment boundaries

- Examples:
  - (1) 2 objects of the same color located directly adjacent to each other
  - (2) Gray value in the image can change due to shadows/lighting without a segment boundary

Approaches

In general:

- Decide for one or more parameters (e.g. gray value)
- Track changes of these parameters in the image

Two basic approaches to segmentation

- Find **changes**
  - segment border if change in parameters is sufficiently large
- Track **homogeneous areas**
  - Assumption: homogeneous area, if parameters change only slightly

Threshold Method

Technische Hochschule Rosenheim

Objects against a homogeneous background:
Thresholding method Chapter “Preprocessing”

- Calculate threshold value from histogram
- Binarization of the image
- Operations on binarized image (e.g. opening, closing)
- Result: Separation into foreground (object) and background pixels

Contour Lines

Overview

Technische Hochschule Rosenheim

Contour line: not necessarily straight, can be curved

Typical procedure:

- **Filter** (gray) image with edge detector
  - e.g. Sobel, Laplace, ... see previous chapters
- **Threshold** value on edge image:
  only consider edges above a certain strength → binary edge image
- Optional: Determine the points on the contour line
- **Approximation** of the contour
  - through straight lines
  - piecewise linear
  - circles/ellipses
  - other parametric curves

Canny edge detector

Technische Hochschule Rosenheim

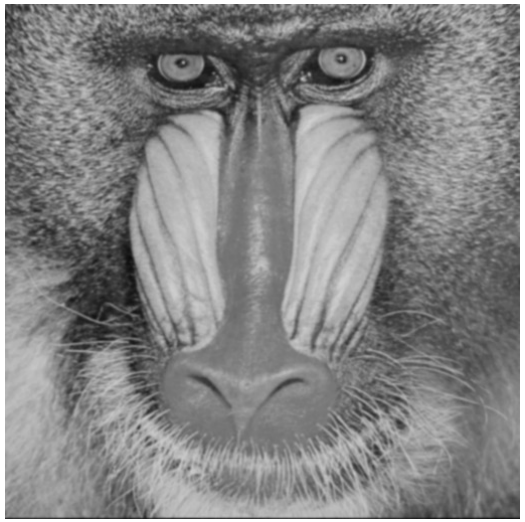
- Criteria for optimal edge detector
  - Good detection
    - Filter responds to edges, not noise
    - No existing edges are overlooked
    - No non-existent edges are detected
  - Good localization
    - Edges should be detected exactly where they appear in the image
  - Single answer
    - the detector should only provide one response per edge (no double edges)

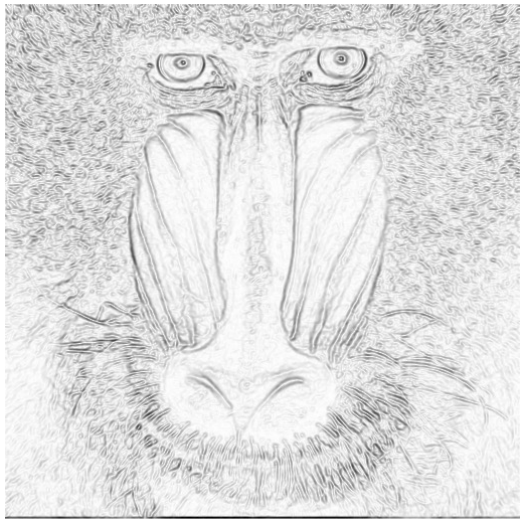
- Procedure
  - Smooth the image with a 5x5 Gaussian filter (removes edges that are too small)
  - Calculate the gradient (strength and direction)
  - Non-maxima suppression (localize the best edge position)
  - Hysteresis threshold method for edge tracing

Canny – Smoothing / Gradient

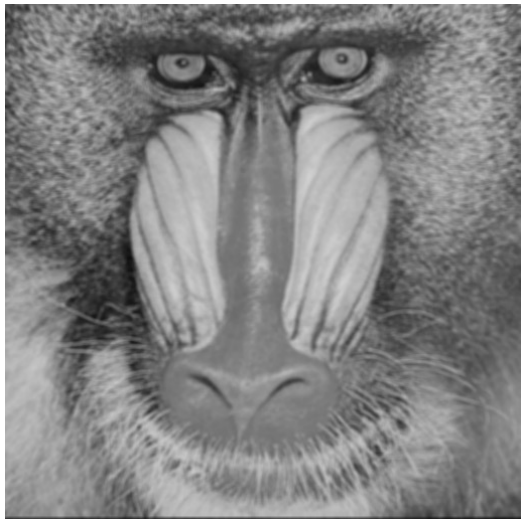
Technische Hochschule Rosenheim

$\sigma = 1.0$

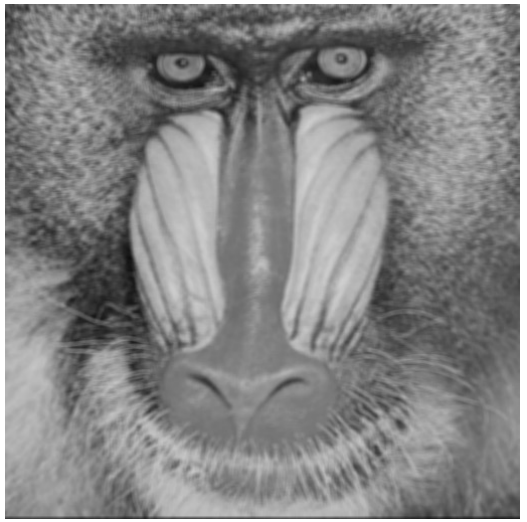




$\sigma = 2.0$



$\sigma = 4.0$



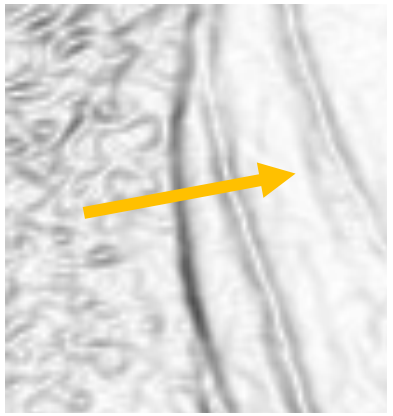
Canny – Thinning

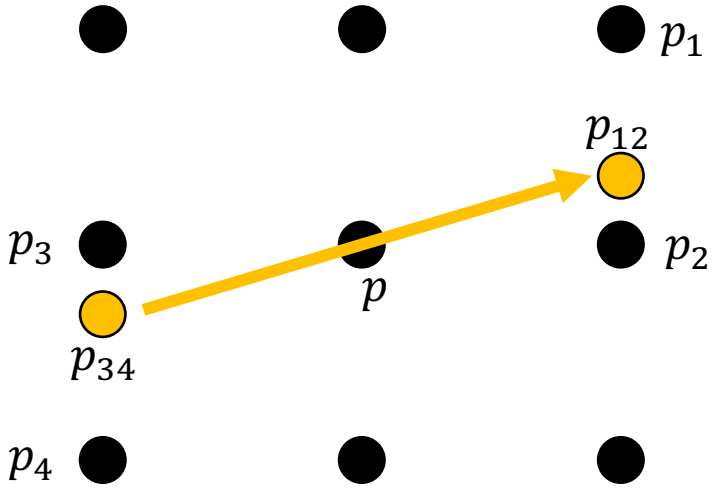
Non-maxima suppression

Where exactly is the edge?

- check edge strength along the gradient
- only use the position with the highest value

Technically: Interpolation of the gradient strength required





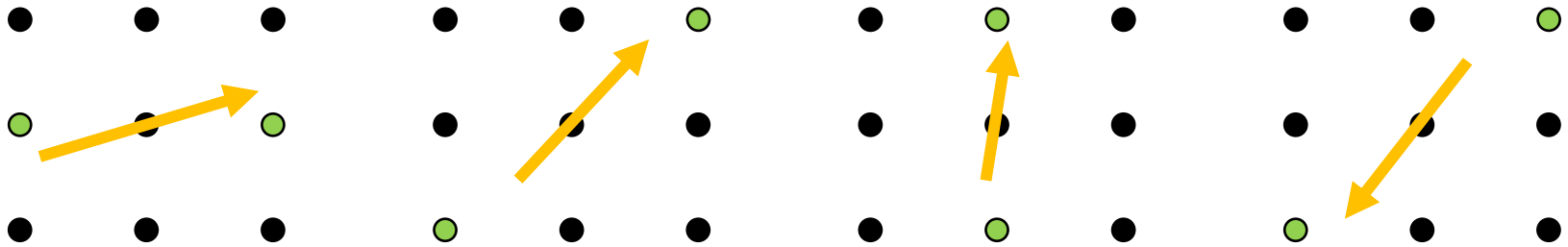
Strength of real pixel

Strength of interpolated pixel

Suppress p if $p < p_{12}$ or $p < p_{34}$
(set p to zerol)

Canny – Thinning

In practice: Instead of interpolation → use discrete directions in 8-neighborhood



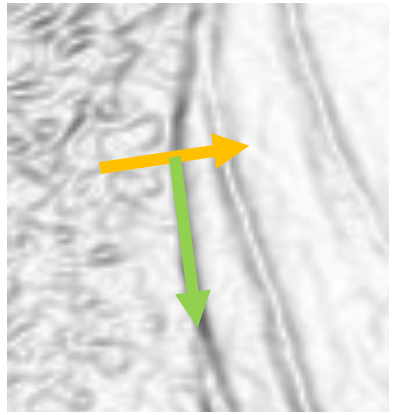
Suppress if the strength is less than one of the two marked neighbors

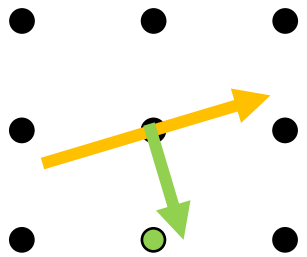
Canny – Edge Tracing

Technische Hochschule Rosenheim

Hysteresis threshold method

- go through the complete thinned edge strength image
- start a new contour line, if the strength is greater than a threshold value h
- Determine the next edge pixel
  - tangential to the edge, i.e. go perpendicular to the gradient
  - Continue to follow the contour, if the strength is greater than a threshold value l.
  - It applies: $l < h$
- save the contour line created in this way (e.g. as chain code or binary image)





Chain Code

- Either
  - 4-Neighborhood or
  - 8-Neighborhood

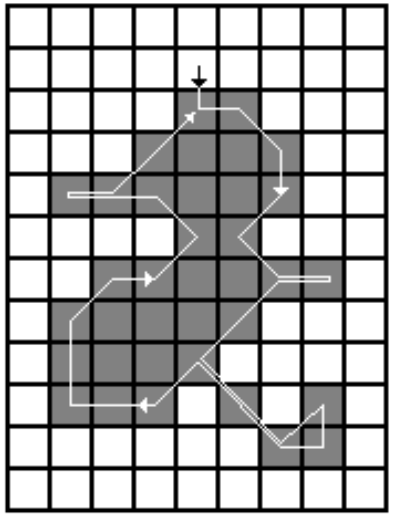
	3	
2	+	0
	1	

5	6	7
4	+	0
3	2	1

- Representation of a contour line as a chain code

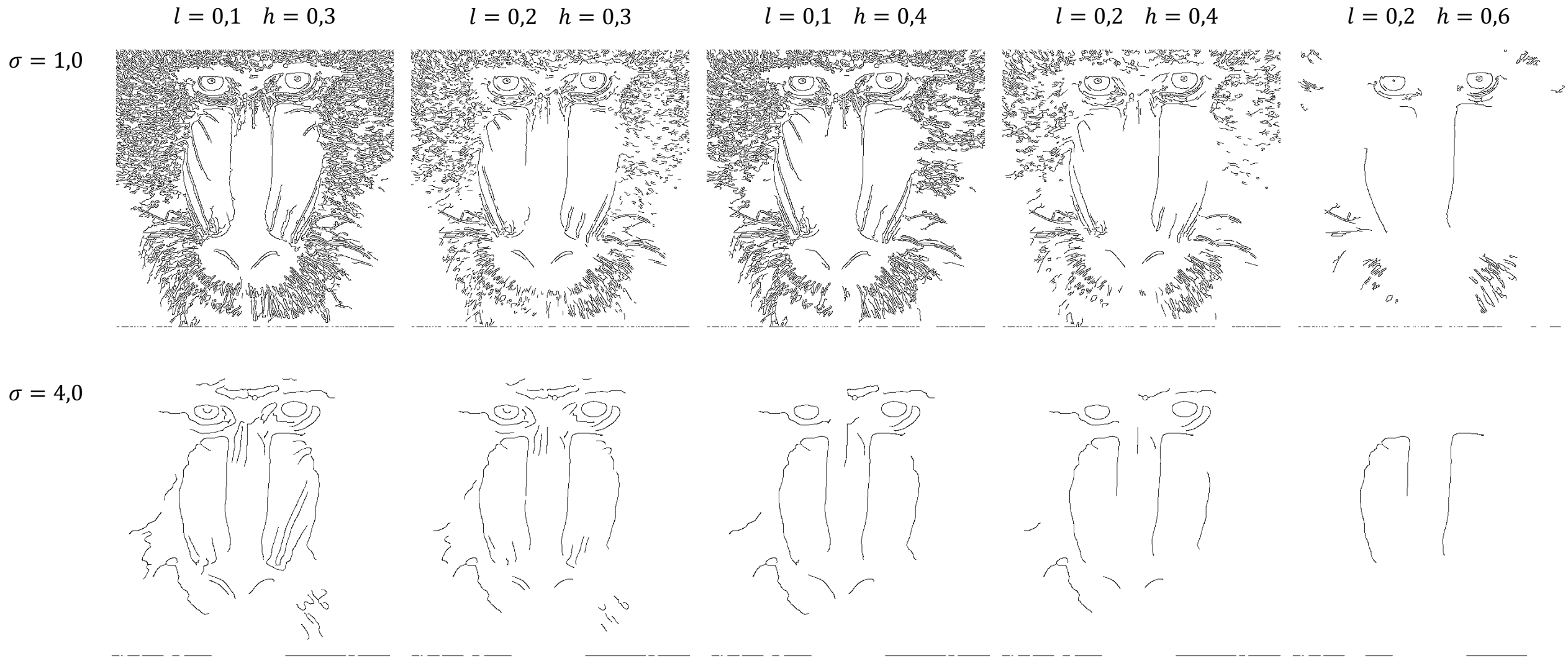
(x, y)
Coordinates of start

$r_1, r_2, \ldots, r_n$
Directions of the following edge pixels



(4, 2), 0, 1, 2, 3, 1, 0, 4, 3, 3, 1, 1, ...

Canny – Examples



Split & Merge

Technische Hochschule Rosenheim

Objective: piecewise linear approximation of the contour line

Given:

- Set of (**ordered**) contour points P
- Residual error measure for determining the approximation quality
- Threshold value θ for maximum permissible error

Steps:

1. Initialization
2. Split
3. Merge
4. Shift

Split & Merge – Initialization

Technische Hochschule Rosenheim

- Start with an **initial approximation** – piecewise linear
- Straight line segments are defined by their start and end points, which must lie on the contour
- Initialization in principle arbitrary, e.g.
  - two adjacent pixels form a line segment
  - only the start and end points of the contour form a line segment
  - Every x^{th} contour point is the start/end of a straight line segment

Split & Merge – Split

- For each line segment, check whether the error between the approximation and the contour > θ
  - no: segment remains as it is
  - yes: **segment is split** into two parts
- Iterate until error for all pieces < θ

- Split e.g.
  - in the center
  - at the point with the largest error
- This process *increases* the number of straight line segments

Split & Merge – Merge

- For each pair of adjacent line segments, check whether the merged segment still has an approximation error < θ
  - no: segments remain as they are
  - yes: **segments are merged**
- Iteration, as long as merging is possible

- This process *reduces* the number of straight line segments

Split & Merge – Shift

Technische Hochschule Rosenheim

- Consider **neighboring pairs** of straight line segments s_1 and s_2.
  Let the corresponding approximation error be ε_1 and ε_2.

- **Move the end point** of s_1 (= start of s_2) one pixel to the left or right and calculate the errors ε_1' and ε_2' or ε_1'' and ε_2''.

- Select the contour pixel that leads to the smallest overall error as the new end/start point.

- Iterate until no more end/start points are changed
- This process reduces the error with a *fixed* number of straight line segments

Residual Error Calculation

- Since vertical line segments can occur: Use of the Hesse normal form

- Given: Start/end point of the line $(x_S, y_S), (x_E, y_E)$
- Line equation: $ax + by = c$
  with

$$a = (y_S - y_E), \qquad b = (x_E - x_S), \qquad c = y_S(x_E - x_S) - x_S(y_E - y_S)$$

- Vertical distance d_i of a point (x_i, y_i) from the line:

$$d_i = \frac{ax_i + by_i - c}{\sqrt{a^2 + b^2}}$$

Residual Error Calculation

Technische Hochschule Rosenheim

Residual error measures for a straight line segment (N: number of points)

- Average distance (L_1-Norm)

$$\varepsilon_1 = \frac{1}{N}\sum_i |d_i|$$

- Mean square distance (L_1-Norm)

$$\varepsilon_2 = \frac{1}{N}\sum_i d_i^2$$

- Maximum distance (L_∞-Norm)

$$\varepsilon_\infty = \max_i |d_i|$$

Hough Transform

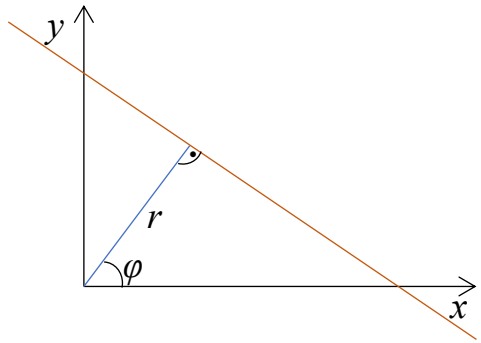
Technische Hochschule Rosenheim

- approximates contours using parametric functions (e.g. straight lines, circles)
- Suitable for functions with 2 - 3 unknown parameters
- No ordered point set required, no contour tracing necessary

Hough Transform – Straight Lines

- Line equation in polar form

$$r = x \sin\varphi + y \cos\varphi$$



Two points of view:

- "Normal" viewpoint:
  - y is the dependent variable, x the independent variable
  - returns a straight line in (x, y)-space for fixed parameters φ_i, r_i
- Dual viewpoint:
  - r is the dependent variable, φ the independent variable
  - returns a harmonic function in (φ, r)-space for fixed parameters x_i, y_i

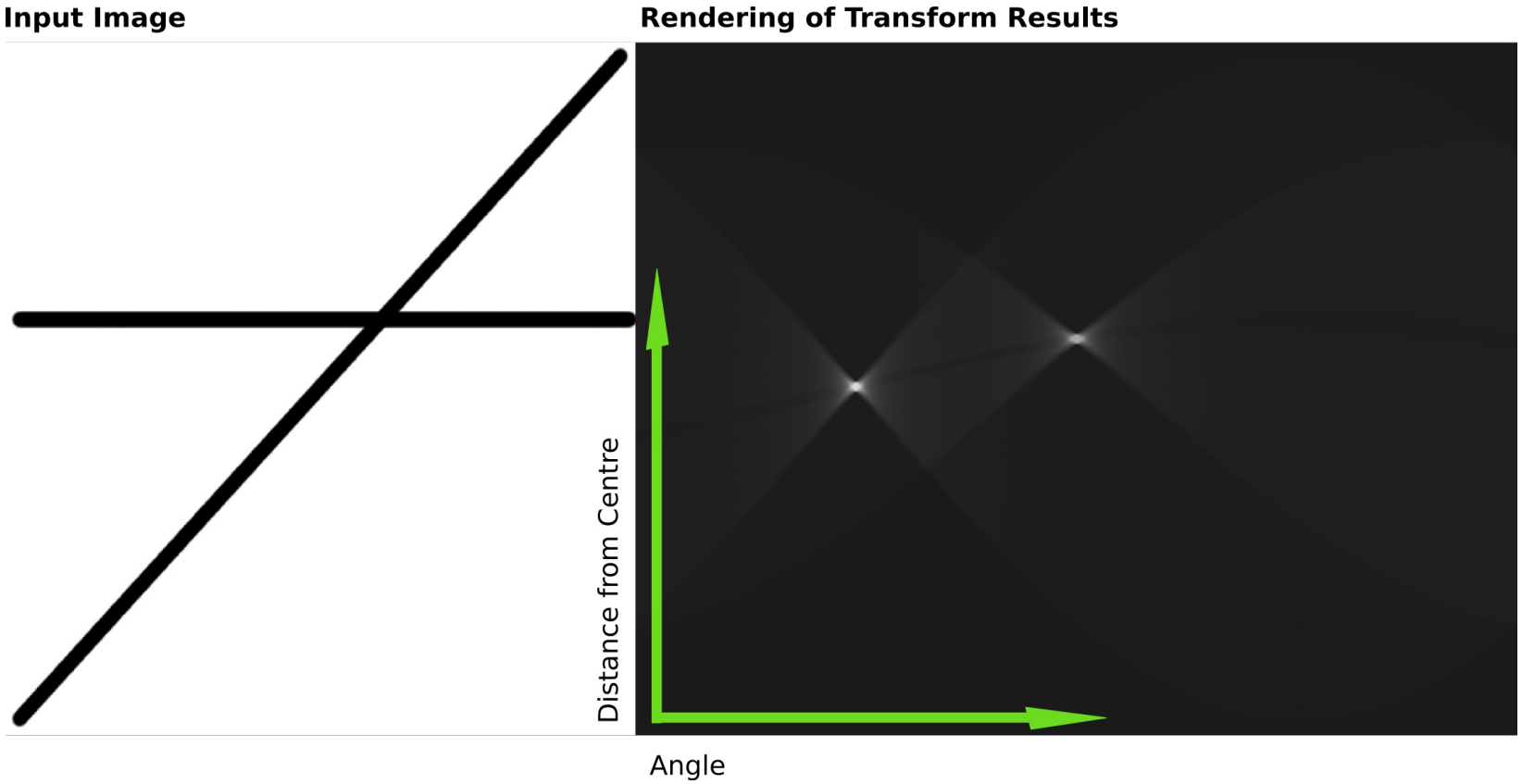
Hough Transform – Straight Lines

Technische Hochschule Rosenheim

Idea:

- Each contour point (x_i, y_i) is assigned a harmonic function in (φ, r)-space
- If 3 or more contour points lie on a straight line with the parameters φ_i, r_i, then all their harmonic functions intersect at this one point
- Notes:
  - Of course, this also applies to 2 points – so why look only at 3 or more?
  - In practice, the edge points will never lie exactly on a straight line → clusters in Hough space
    - Low-pass filter
    - Search maximum in a neighborhood

Hough Transform – Straight Lines



Daf-de, Hough-example-result-en, CC BY-SA 3.0

Hough Transform – Straight Lines – Algorithm

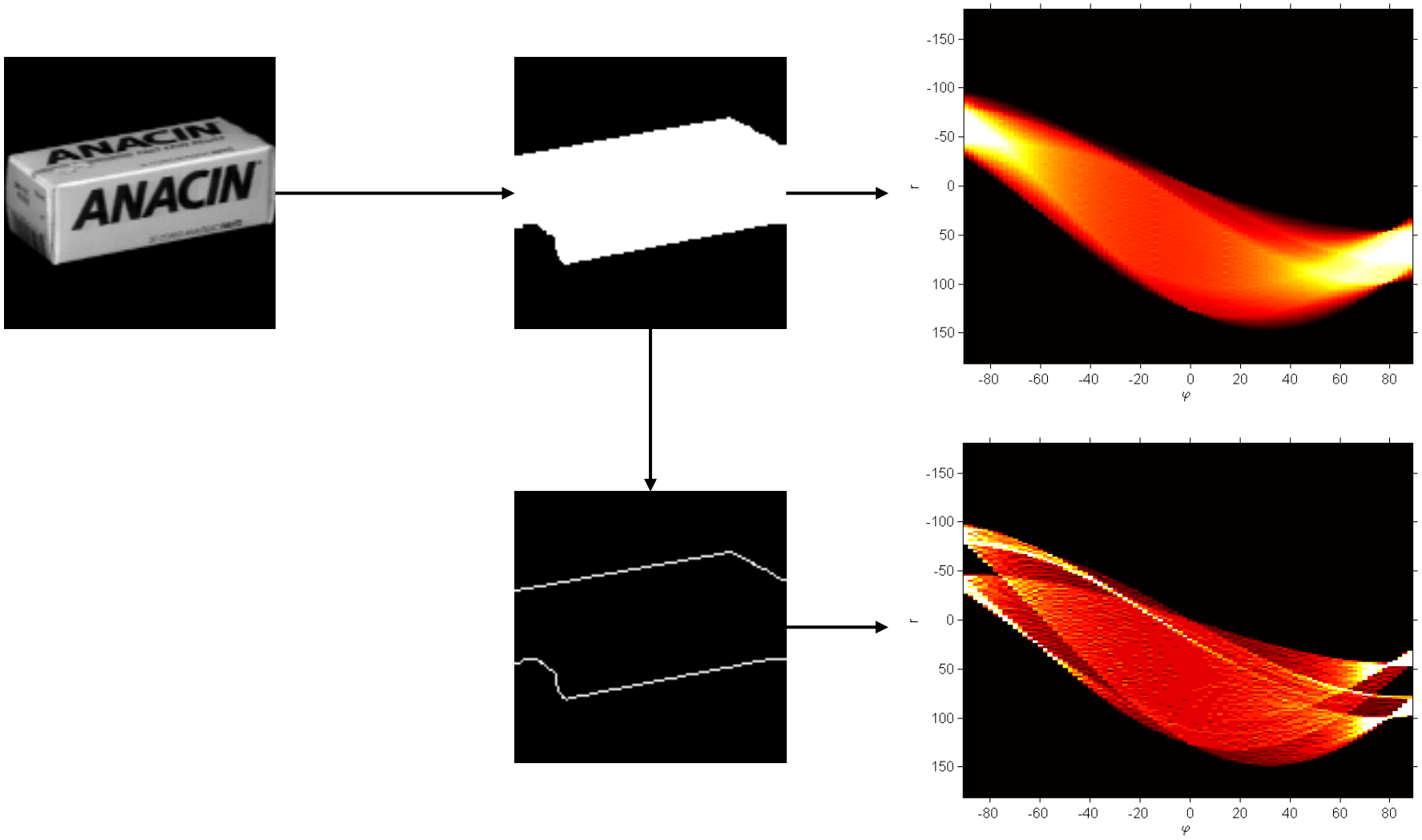
Technische Hochschule Rosenheim

- Quantization of the parameter space
  $\varphi_i,\ i = 1,\ \ldots,\ p$ $\qquad$ $r_j,\ j = 1,\ \ldots,\ q$
- Initialization Hough accumulator $H(i, j)$ with zeros

- For all edge pixels $(x_k,\ y_k),\ k = 1,\ \ldots,\ N$
  - For all discrete values of $\varphi_i,\ i = 1,\ \ldots,\ p$
    - $r_j = x_k \cos \varphi_i + y_k \sin \varphi_i$
    - $H(i, j) = H(i, j) + 1$

- The element $H(i, j)$ of the accumulator contains the number of edge points on the straight line with the parameters $(\varphi_i,\ r_j)$

Hough Transform – Straight Lines



Hough Transform – Straight Lines

Technische Hochschule Rosenheim

- the Hough transform calculates the parameters of a line equation: the start and end points of the line segment are unknown
- Searching in parameter space is very time-consuming
  - therefore, this should have a small dimension
  - better efficiency e.g.
    - through resolution pyramid
    - use knowledge about the edge direction (restricts the search space)
- Parameter space must be finite
  - An array is created for the accumulator

Hough Transform – Circles

- Circle detection in principle works in the same way as for straight lines
- Parametric function: Circle equation

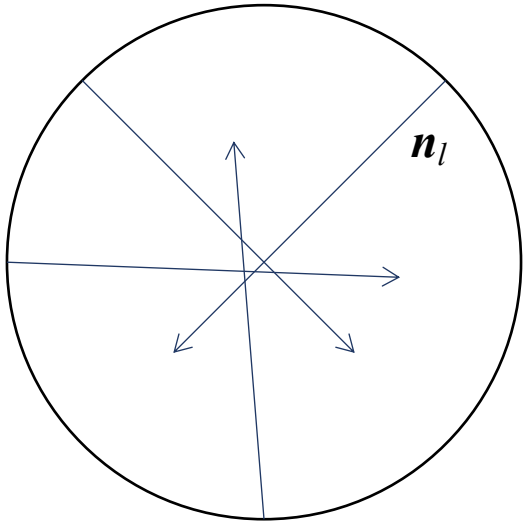
$$(x - x_c)^2 + (y - y_c)^2 = r^2$$

- Problem:
  - Hough accumulator becomes three-dimensional
  - Computational effort very high

- Idea: Use the edge gradient

Hough Transform – Circles

- Consider straight lines on circle contour perpendicular to gradient direction
- all straight lines intersect at the center of the circle
  (for discrete noisy images: at least approximately)



Hough Transform – Circles – Algorithm

Technische Hochschule Rosenheim

- Quantization of the parameter space (x_{ci}, y_{cj}, r_k)
  $i = 1, \ldots, M_x$ $\quad$ $j = 1, \ldots, M_y$ $\quad$ $k = 1, \ldots, p$
- Initialization Hough accumulator $H(i, j, k)$ with zeros
- For all edge pixels (x_l, y_l), $l = 1, \ldots, N$
  - Calculate normal vector $\boldsymbol{n}_l$ from gradient direction
  - For all discrete values of r_k, $k = 1, \ldots, p$

$$\begin{pmatrix} x_{ci} \\ y_{cj} \end{pmatrix} = \begin{pmatrix} x_l \\ y_l \end{pmatrix} \pm r_k \boldsymbol{n}_l$$

    - $H(i, j, k) = H(i, j, k) + 1$
- Search for accumulations in the accumulator $H(i, j, k)$

Hough Transform – Circles

- the quantization for the radius is typically chosen very coarsely, e.g., 4 to 10 different radii

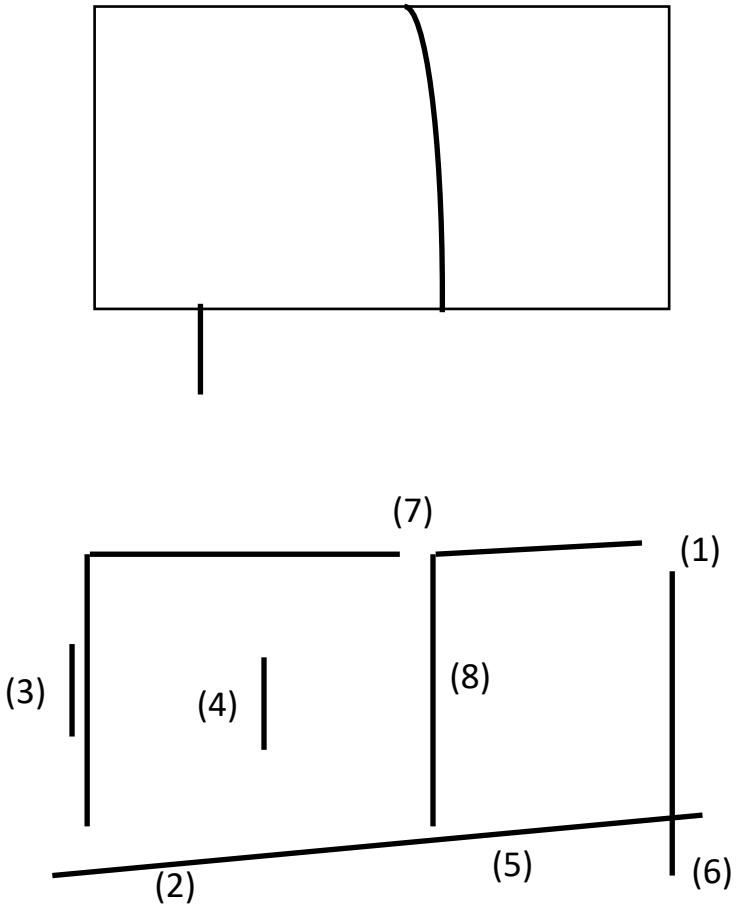
- Instead of two points (x_{ci}, y_{cj}) for each radius, it can also be useful to enter two whole line segments (as not all lines intersect exactly at the center point)

- As with straight lines, a resolution pyramid is also helpful here

Hough Transform – Circles

Hough Transform – Circles

Typical problems with contour lines



(1) closed contour is broken up

(2) a line is not found

(3) another line is found in the vicinity of a line

(4) a non-existent line is found

(5) line direction and/or length are incorrect

(6) corners are not detected correctly

(7) one line is split into several

(8) an arc with a large radius is detected as a straight line (or vice versa)

Regions

Regions vs Contours

Technische Hochschule Rosenheim

- a region is a homogeneous image area
- a region is always surrounded by a closed contour
  segmentation of contours does not always provide a closed contour
- for segmentation based on gray values (simplified):
  - Detection of contour based on sufficiently strong gray value change
  - Region has almost constant gray value
- Segmentation can be done, e.g., based on
  - Gray value
  - Color
  - Texture

Requirements for regions

- image is broken down into contiguous subsets

$$\boldsymbol{f} \rightarrow \{\boldsymbol{f}_i \,|\, i = 1, \ldots, n\}$$

- each pixel belongs to exactly one region

$$\bigcup_{i=1}^{n} \boldsymbol{f}_i = \boldsymbol{f}, \qquad \boldsymbol{f}_i \cap \boldsymbol{f}_j = \varnothing \quad \forall i \neq j$$

- there is an evaluation criterion for homogeneity

$$H(\boldsymbol{f}_i) = \begin{cases} 1 & \text{if region satisfies the criterion} \\ 0 & \text{otherwise} \end{cases}$$

- all regions are homogeneous in the sense defined above

$$H(\boldsymbol{f}_i) = 1 \text{ for } i = 1, \ldots, n$$

- all regions are as large as possible

$$H(\boldsymbol{f}_i \cup \boldsymbol{f}_j) = 0 \text{ for } i \neq j \text{ and adjacent } \boldsymbol{f}_i, \boldsymbol{f}_j$$

Homogeneity Criterion – Example

- Based on the gray values in the region

$$H(\boldsymbol{f}_i) = \begin{cases} 1 & \text{if } |f_{\max} - f_{\min}| < \theta \\ 0 & \text{otherwise} \end{cases}$$

$f_{\max}$ and $f_{\min}$ are the largest/smallest gray value in the region under consideration (pre-filtering required)

- The threshold value $\theta(a)$ increases linearly with the area a of the region

$$\theta(a) = \theta_S - \frac{a}{a_{\max}}(\theta_S - \theta_E)$$

- similar criterion for color (multiple channels) possible

Algorithms - Overview

Technische Hochschule Rosenheim

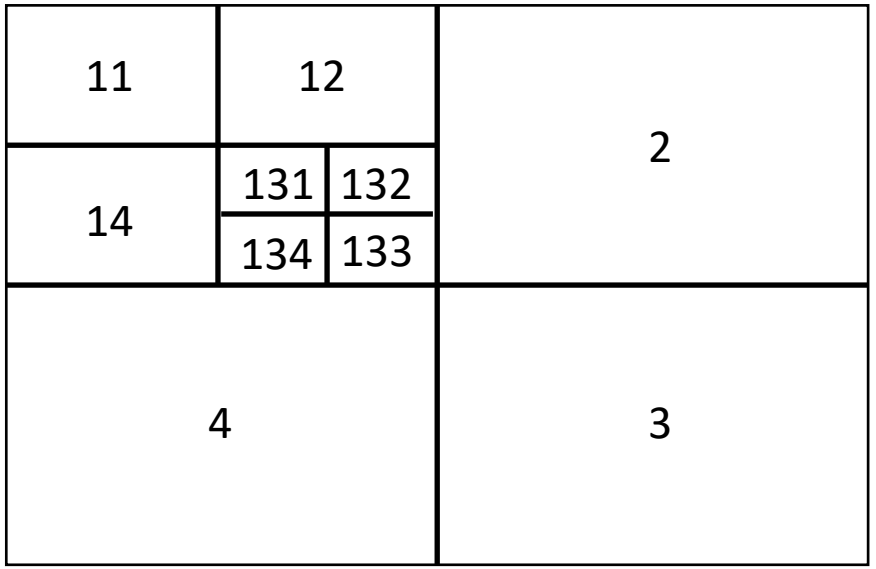
- Split
- Merge
- Split & Merge

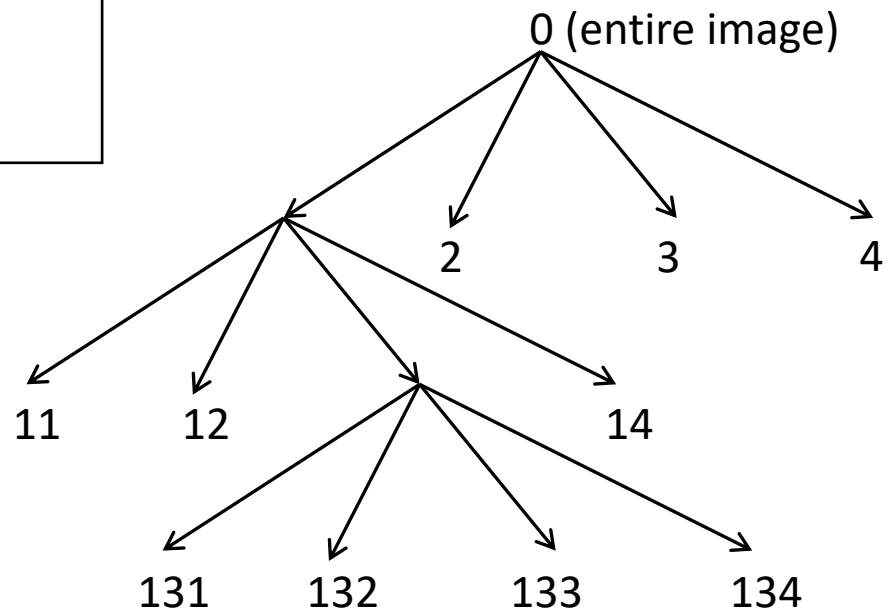
Split

- start with a single region (i.e. the entire image)
- normally the image is not homogeneous → further decomposition
  - randomly into two regions
  - regularly, into four equally sized regions (quadtree)
  - …
- repeat until all regions are homogeneous

- obviously: at some point you always get homogeneous regions, no matter how you decompose them
- the objective is to obtain the largest possible regions as quickly as possible

Split – Quadtree

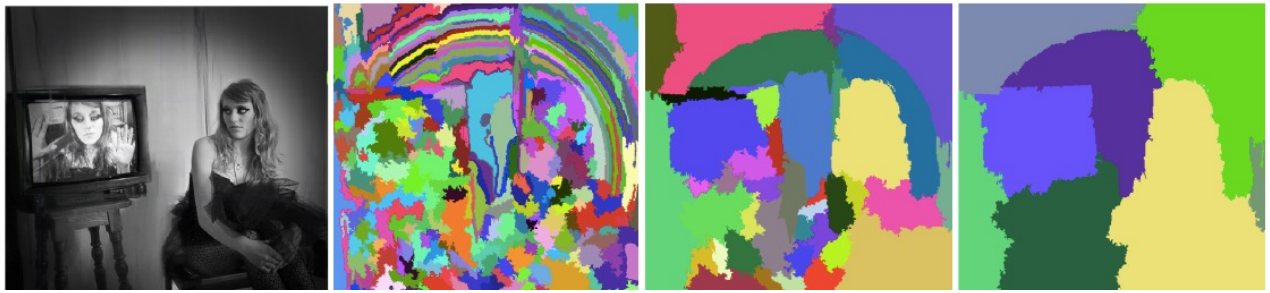




Merge

- Start with a large number of small regions (e.g.: each pixel is a region)
- merge neighboring regions as long as the resulting region is still homogeneous

- This is also the principle of “Selective Search” (segmentation algorithm as the basis of R-CNN)
  - uses a fast graph-based segmentation method as starting regions
  - works (if desired) in several color spaces in parallel
  - 4 homogeneity criteria, which are combined
    - Color (based on histograms of the individual channels)
    - Texture
    - Size of a region (penalizes regions that are too small)
    - Avoiding holes: how well does one region fit into another?



Uijlings, J.R.R., van de Sande, K.E.A., Gevers, T., Smeulders, A.W.M. Selective Search for Object Recognition. *Int. J. of Computer Vision* **104,** 154-171 (2013). http://huppelen.nl/publications/selectiveSearchDraft.pdf

Split & Merge

Technische Hochschule Rosenheim

- Principle like split & merge for contours
- Start with an initial decomposition of the image that corresponds to a complete quadtree up to a defined level

- Check all quadrants of the image
  - **Merge** all four regions if H remains fulfilled
  - until merging is no longer possible
- Check all regions
  - **Split** a region into four parts if H is not fulfilled
  - until splitting is no longer possible
- **Group** neighboring regions
  - as long as H remains fulfilled (no longer in the quadtree)
- **Eliminate** regions that are too small
  - if the area of a region is too small: merge it with the neighboring region that differs least from the region that is too small

Summary

Technische Hochschule Rosenheim

Contours

- Edge images, e.g., Canny
- Split & merge algorithm
  - Piecewise linear approximation
  - Ordered point set required
- Hough transform
  - Straight lines
  - Circles
  - No ordered point set required

Regions

- Region always has a closed contour line
- Split and/or merge for regions
  - e.g., based on Quadtree
  - Requires
    - Initial decomposition
    - Homogeneity criterion
    - Threshold value(s) for splitting/merging