Chapter 5

Segmentation

$$\underbrace{\text{Segmentation}}_{\text{dividing into regions}} \rightarrow \underbrace{\text{Feature Extraction}}_{\text{measuring}} \rightarrow \underbrace{\text{Classification}}_{\text{deciding}}$$

Toy example: sorting fruit (cherries, apples, lemons, grapefruit) using a color camera, and image processing:

- determine fruits and background on image $\rightarrow$

$\rightarrow$ determine size, shape color

$\rightarrow$ decide which fruit is of which kind

Image Segmentation
means

- dividing image into regions[1] (= connected sets of pixels) according to
  - grey level
  - color (RGB, HSI)
  - texture (see Section 6.4)
  - ...

  →Fig. 5.1

- often, the regions are "objects" and background
- also: different objects
- or, in satellite images, e.g., different surface cover types
- In the following: Segmentation based on grey level or color (i.e., pixel values) – simplest case

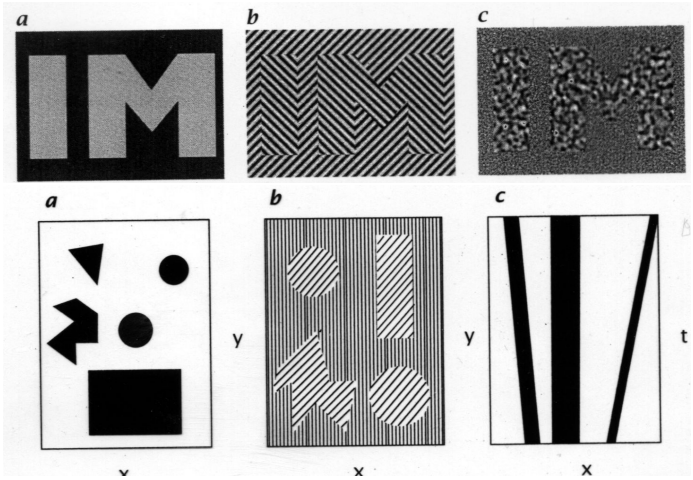

Fig. 5.1: *Objects and background, distinguished by grey level or texture/pattern, (from Jähne, 2002, Fig. 11.1 and 11.2)*

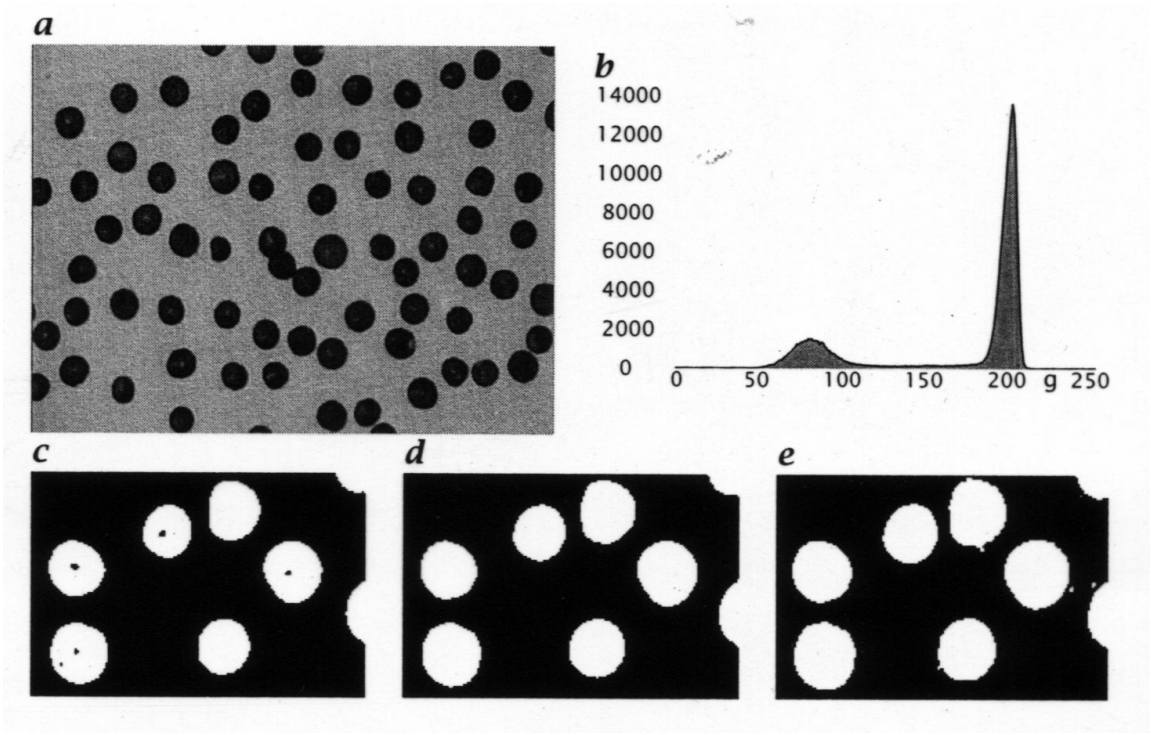

Fig. 5.2: *Segmentation with a global threshold. (a) Original image; (b) histogram; (c)–(e) upper right section of (a), segmented with threshold values 110, 147, 185, respectively. From Jähne (2002)*

5.1 Image segmentation by thresholding

use a threshold value to decide to which region a pixel belongs

→Fig. 5.2

- determining threshold from histogram
- if histogram noisy: first smooth the histogram (not the image!)
- threshold value:
  - relative minimum between corresponding peaks in histogram
  - halfway between the peaks
  - ...

Global thresholding ↔ Adaptive thresholding

- Global thresholding:
  - one threshold value for the whole image
  - good only if background contrast is constant and all objects have similar contrast to background

  else →Fig. 5.3

- Adaptive thresholding:
  - divide image into blocks, choose thresholds according to individual histograms

    or: 2 Steps
    1. Global thresholding to roughly find the objects
    2. local thresholding of objects plus some surroundings
- Other methods, e.g.,
  - correct non-uniform background by location-dependent grey-level manipulation (as in Fig. 5.3

5.2 Region-growing

- select seed points/pixels
- each seed pixel is the starting point for a region
- compare each seed point with its neighbors: if neighbor has similar *properties*,(grey level, local variance, color etc.) as seed pixel, include it into the region of the see pixel.
- continue with the neighbors of the neighbors etc. ⇒regions grow
- merge regions with the same property if the touch
- continue until the regions cannot grow any further

[1] such regions are called segments, hence the name

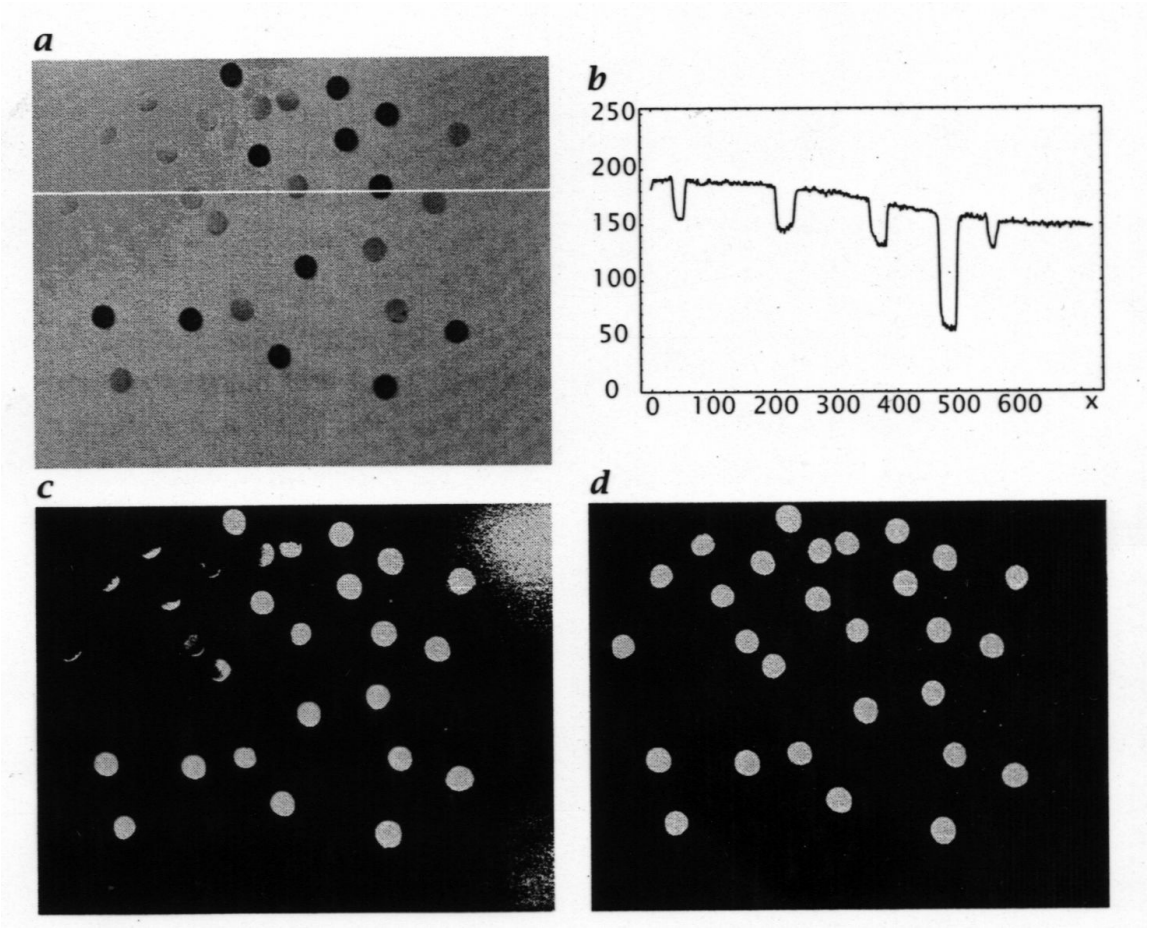

Fig. 5.3: *Segmentation of image with non-uniform illumination. (a) Original image; (b) profile of the indicated line; (c) Segmentation with global threshold (d) Segmentation with global threshold after correction of grey levels for non-uniform illumination From Jähne (2002)*

- Note:
  - seed pixels have to be chosen well, typically many more than regions to be expected
  - property can also be a local property such as variance or texture (see later)
  - computationally expensive

5.3 Gradient-based segmentation, edge detection

- find the boundary/edge of objects

$\Rightarrow$ look for points of rapid spatial change of grey level (or color etc.)

- various methods

Gradient

- linear filter with kernels like

$h = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$ or $\begin{bmatrix} -1 & 1 \end{bmatrix}$
i.e., horizontal gradient, $\frac{\partial}{\partial x}$
or $h = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$ or $\begin{bmatrix} -1 \\ 1 \end{bmatrix}$
i.e., vertical gradient, $\frac{\partial}{\partial y}$

- Then calculate gradient magnitude

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{5.1}$$

- This means:
  1. filter original image with horizontal gradient filter
  2. filter original image with vertical gradient filter
  3. take the pixel-wise square of each gradient image
  4. add the two squared gradient images
  5. take the square root (pixel-wise)

Laplacian edge detection

- Laplace operator is a second derivative

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{5.2}$$

- Corresponding filter kernel (see also Section 2.2.3):

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.3}$$

- Edge produces adjacent positive and negative values

`Figure: Sketch 1D: Edge, Gradient mag., Laplace`

`→Fig. 5.4`

Other edge detection filters

Sobel filter Two kernels,

$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ and } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

apply both and take the maximum

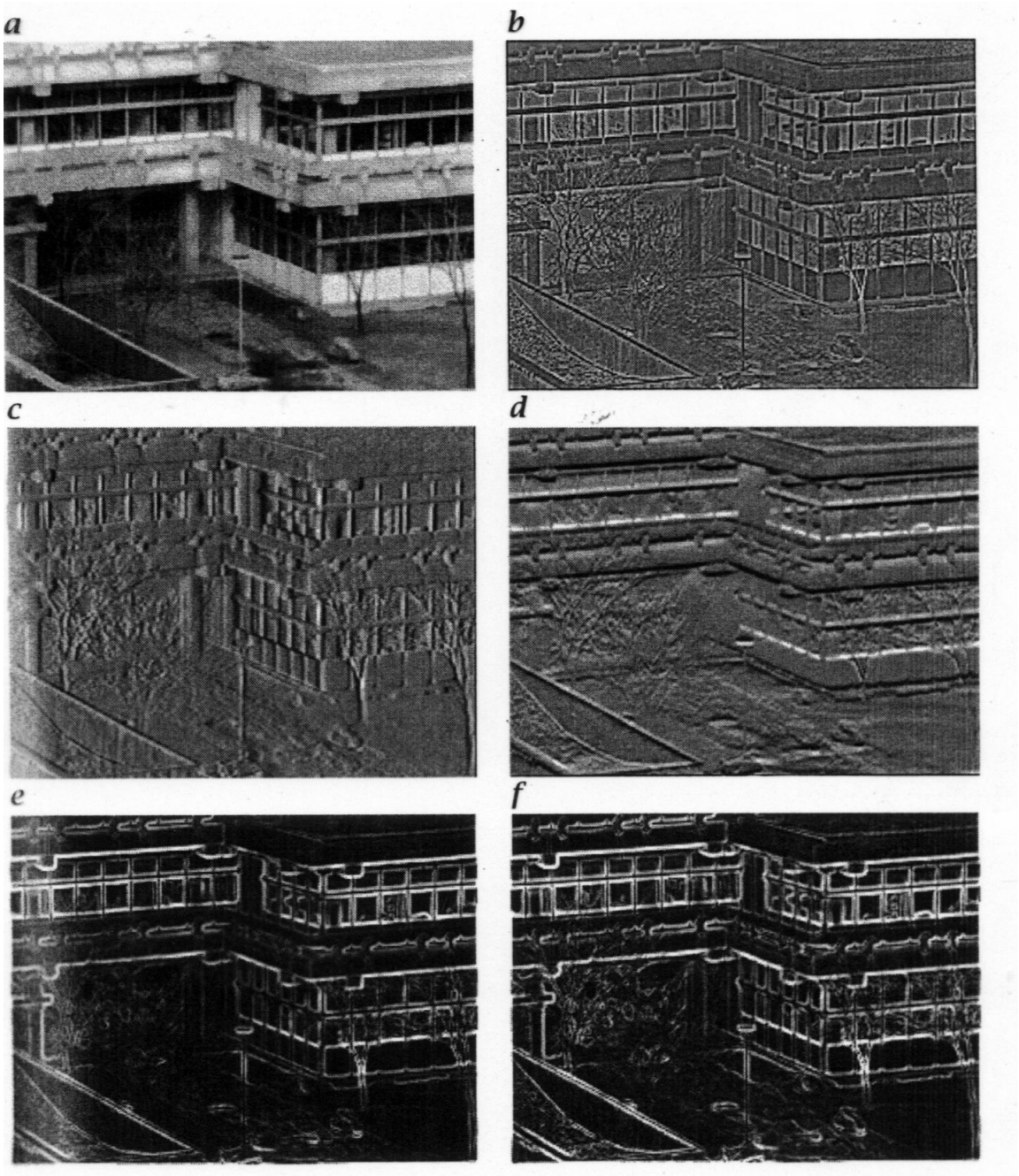

Fig. 5.4: *(a) Original image. Edge detection with (b) Laplace operator, (c) horizontal gradient, (d) vertical gradient, (e) gradient magnitude, sum of absolute values of (c) and (d). After Jähne (2002, Fig 10.20)*

Prewitt

Kirsch

...

5.4 Template Matching

→Fig. 5.5

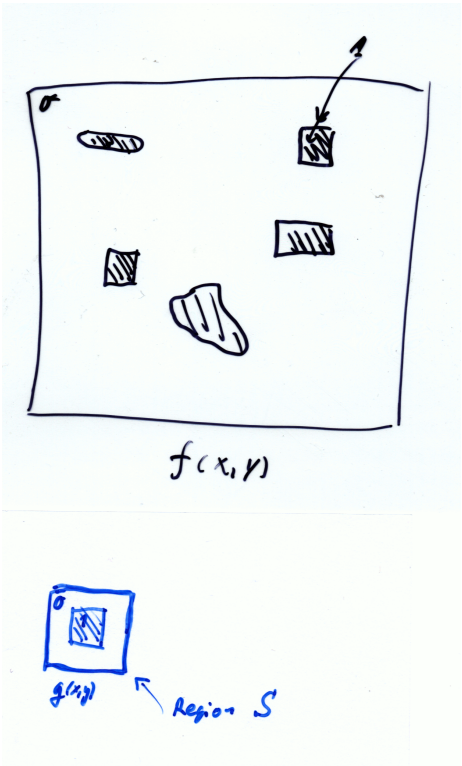

Fig. 5.5: *Image f with some differently shaped objects, and template g (blue) which here is a small square.*

- Define a template g, i.e., a small image containing a prototype of the object to find in image f
- Calculate normalized cross correlation D_{gf} of image f with template g

$$D_{gf}(u,v) = \frac{\int_S g(x,y) f(x+u,y+v)\mathrm{d}x\mathrm{d}y}{[\int_S g^2(x,y)\mathrm{d}x\mathrm{d}y \int_S f^2(x+u,y+v)\mathrm{d}x\mathrm{d}y]^{1/2}} \tag{5.4}$$

  where "integration" is actually just sum over domain S covered by template g.

- Perfect match: $D_{gf} = 1$
- Look for maximum = good match

5.5 Preprocessing

- Above segmentation methods for object and background which are characterized pixel-wise by typical values (grey level, color)

- If objects and background are characterized by different texture/pattern (see the examples in Fig. 5.1):
- Preprocessing: Analyze texture[2] ⇒transform it into some value so that all pixels in one texture area have the same value
- Then do segmentation as described above

5.6 Postprocessing: Morphological Filters

- Result of segmentation: An additional image telling which pixel belongs to which region or object
- often a binary image: "1" for "object", "0" for "background"
- Problem: Segmentation result not ideal because of noise in the original image: Object have small holes, edges ragged, too wide, not continuous

⇒ need to clean or straighten them: this can be done by morphological filters

`Figure: Non-ideal segmentation result`

- Interpret binary image B as a set defined on a Cartesian grid, the "1"s are the elements of the set.
- Define a small mask ("structural element") S_{xy} centered on position (x,y) - the mask contains just "1"s, for example

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad \text{(8-neighborhood)}$$

- 2 main ways to apply the mask to the image:

Erosion

Resulting image E defined by

$$E = B \ominus S = \{(x,y) | S_{xy} \subseteq B\} \tag{5.5}$$

- Sometimes, symbol $\otimes$ is used instead of $\ominus$
- This means: A point (x,y) is only kept if the structural element S_{xy} centered on this point lies entirely within a region of "1"s

⇒ omitting border/edge points of object

- Example with a square 3×3 structural element `→Fig. 5.6`

`→Fig. 5.7`

[2]Texture analysis looks at neighborhoods and characterizes the texture, e.g., the direction of a hatch pattern, the density/size of the pattern, shape of the elements of the pattern ...

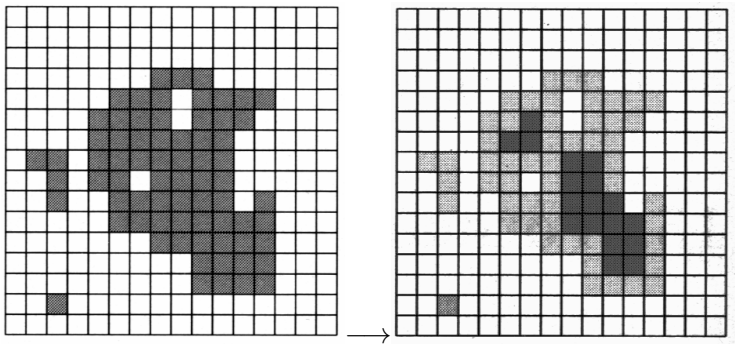

Fig. 5.6: *Erosion by a square 3×3 structural element. Pixels that have been removed are shown in grey (after Jähne, 2002, Fig. 15.1)*

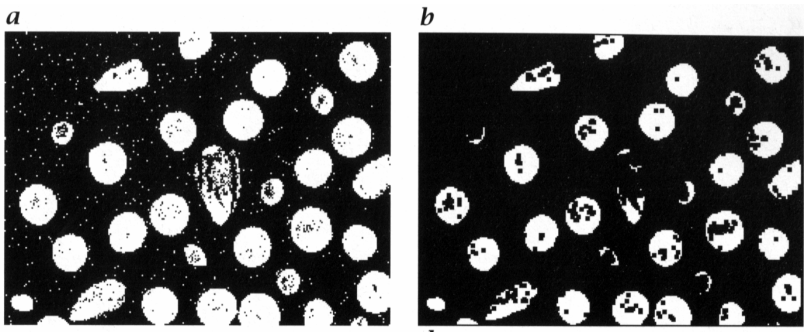

Fig. 5.7: *Erosion of a real image, 3×3 structural element (after Jähne, 2002, Fig.15.2)*

Dilation

Resulting image E defined by

$$E = B \oplus S = \{(x,y) | S_{xy} \cap B \neq \emptyset\} \tag{5.6}$$

$\emptyset$ is the empty set

- This means: A point (x,y) is kept or added if the structural element S_{xy} centered on this point has overlap with region of "1"s

$\Rightarrow$ adding points to border/edge of object

- Example with a square 3×3 structural element →Fig. 5.8

→Fig. 5.9

Other operations

Dilation and erosion can be combined or modified:

Opening Erosion followed by dilation:

$$B \circ S = (B \ominus S) \oplus S$$

$\rightarrow$ eliminates small, thin objects, smooths object edges

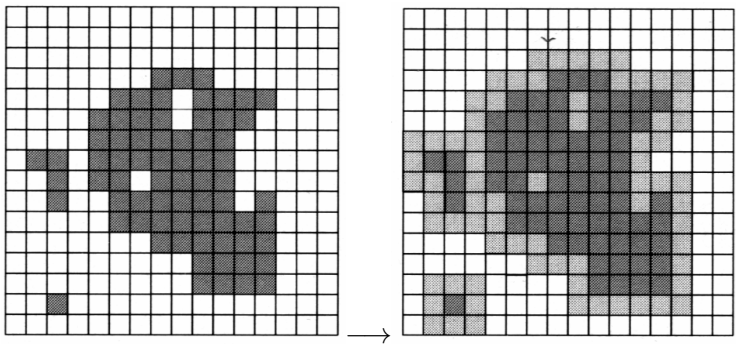

Fig. 5.8: *Dilation by a square 3×3 structural element. Pixels that have been added are shown in grey (after Jähne, 2002, Fig. 15.1)*

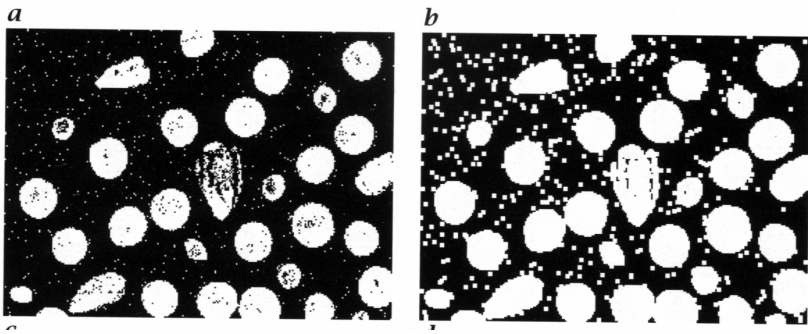

Fig. 5.9: *Dilation of a real image, 3×3 structural element (after Jähne, 2002, Fig.15.3)*

Closing Dilation followed by erosion:

$$B \bullet S = (B \oplus S) \ominus S$$

Shrinking Erosion, but leaving single pixels intact → objects ultimately reduced to single pixels (for counting objects)

Thinning Erosion, but avoiding to break objects → objects become lines

5.7 Postprocessing: Identifying connected components

- usually an "object" is a region of pixels which are connected
- to identify and label them, there is a simple two-pass algorithm:

1st pass: go through image pixel by pixel, line by line, and label all object pixels →Fig. 5.10

- pixels without object neighbor: new label
- pixels with object neighbor: neighbor's label
- note (in a list) labels that have become neighbors during the 1st pass

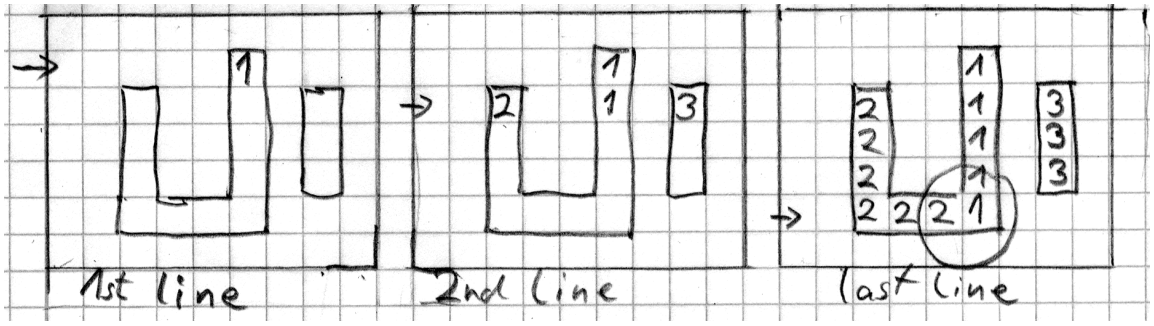

Fig. 5.10: *First pass, labeling object pixels; first line (left), second line (middle), last line (right). New label if no neighbor known; note labels that become neighbors (see right-most sketch).*

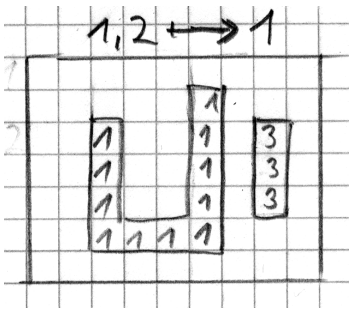

Fig. 5.11: *Second pass, merge labels that have become neighbors in first pass.*

2nd pass: go through the image again, pixel by pixel →Fig. 5.11

- replace all labels that have been noted as neighbors by a new, common label

Bibliography

B. Jähne. *Digital Image Processing*. Springer, 2002.