# Chapter 6
# Optical Character Recognition

Distributed Algorithms
for Image and Video Processing

## Overview

- Motivation
- OCR in images and videos
  1. Detection of text lines / text regions
  2. Segmentation of characters
  - Detection of single characters
    - Pattern matching
    - Zoning
    - Shape contexts
    - Contour profiles
    - (Topological) Skeleton
    - Scale Space Image
- Summary



## Motivation

In videos, important semantic information is delivered through texts:

- Actors' names in feature movies
- In news broadcasts, the images shown next to the narrator are described by a text
- Place or time in a feature movie
- Questions in quiz shows
- Name / profession of the participants in a discussion forum
- Movie title



## Detection of text regions (I)

**Procedure**

1. Search for blocks with sharp edges
2. Combine adjacent blocks to text regions
3. Use horizontal profiles of projection for the detection of single text lines

**Search for blocks with sharp edges (sum edge strength per bloc > T)**



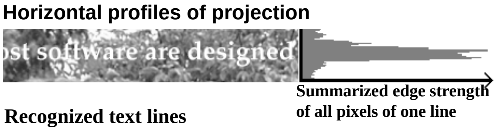
## Detection of text regions (II)

**Combination of blocks**




## Detection of text regions (III)

**Horizontal profiles of projection**

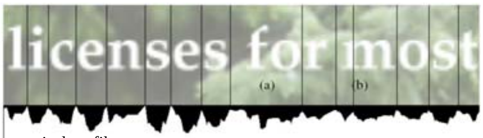



**Recognized text lines**

# Segmentation of letters (I)

**First approach**

Analyze vertical profiles of projection in order to recognize *separators* between letters.



vertical profile of projection — (a) unrecognized separator — (b) separated letter

**Problem:** separated or connected letters



# Segmentation of letters (II)

**New approach**

- Analyze *shortest paths* and use these as separators between the letters.

- For each column from the top line, search a shortest path to the bottom line.
- The costs for the path are to be minimal.
- The costs are defined as sum of the absolute differences between adjacent path pixels.



# Segmentation of letters (III)

**Shortest-path-algorithm**

- The *shortest-path-algorithm* of *Dijkstra* is used in order to find the path with the lowest costs.
- Aim: Calculation of a shortest path between a starting knot (pixel in the top line) and an arbitrary knot in an edge-weighted graph (arbitrary pixel in the bottom line).
- Initialize: Set distance for all knots to infinite.
- Examine knot *u* with least distance to starting knot
  - If knot u is in bottom line: shortest path found.
  - Otherwise:
    - Examine all reachable pixels, i.e. the three adjacent (in the line beneath) pixels.
    - Check if the path to every pixel via the current pixel is favorable to the so far known paths to this pixel.
    - Set new path if this one is favorable.



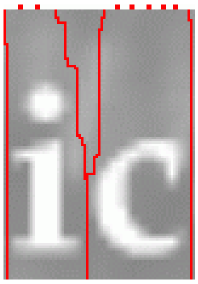
# Optimization of the shortest-path-algorithm

**Problem**

- The effort for the calculation of the shortest path for each pixel is very high.

**Optimization**

1. Initialize possible starting pixels.
2. Mark left and right pixel.
3. Calculate path for marked pixels.
4. Remove starting pixel between two paths if these converge.
5. Choose next starting pixel.
6. Repeat with 3.





# Selection of letter pixels (I)

Identify letter pixels with the aid of a **modified region-merging-algorithm**

1. Calculate the histogram of a text region and identify one to two dominant colors.
   **Assumption:** One of these colors is the text color.
2. Identify regions with a region-growing-algorithm.
3. Every region can assume one of the three conditions: *text, background, undefined.* First of all, all regions are *undefined*.
4. Set all regions with text color to *text regions*.
5. Undefined regions at the top or bottom margin of the text line are defined as background.



# Selection of letter pixels (II)

6. Calculate distance $D_{i,j}$ between an undefined region $i$ and a known region $j$ (text or background) by means of the colors $C_i$ and the center of gravity of a region $G_i$:

$$D_{i,j} = |C_i - C_j| + |G_i - G_j|.$$

7. Choose minimal distance $D_{i,j}$ and define region as *text* or *background*.
8. Repeat with step 6 until all undefined regions are known.

# Recognition of single letters (I)

Recognition of single letters (OCR)

- Pattern matching
- Zoning
- Shape contexts
- Contour profiles
- Skeletons
- Scale space images

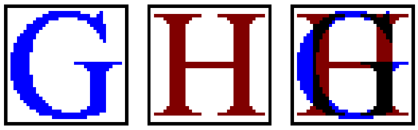


# Recognition of single letters (II)

**Pattern matching**

Calculate the difference between two binary images:

$$D_{Q,J} = \frac{1}{n_x \cdot n_y} \cdot \sum_{x=1}^{n_x} \sum_{y=1}^{n_y} \begin{cases} 0 & \text{falls} \quad Q_{x,y} = J_{x,y}, \\ 1 & \text{sonst.} \end{cases}$$



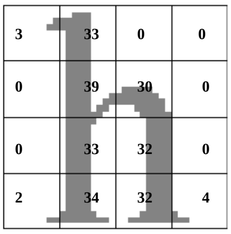
**18 % of the pixels differ from each other**



# Recognition of single letters (III)

**Zoning**

- Define a lattice with N x M cells.
- Count the number of text pixels in each cell.
- Compare two vectors which are defined by the number of text pixels of each cell.



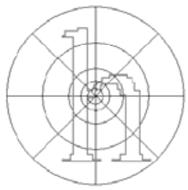

**vector: (3,33,0,0,0,39,30,0,0,33,32,0,2,34,32,4)**



# Recognition of single letters (IV)

**Shape contexts**

- The shape-context-algorithm is a special zoning-algorithm.
- A round grid is used for the definition of the cells.
- A contour pixel defines the center point of the grid.
- The number of ***contour pixels*** (not of the text pixels) in each cell defines the feature vector.
- As letters of reference, characteristic vectors are saved for ***each*** single contour pixel of a letter.

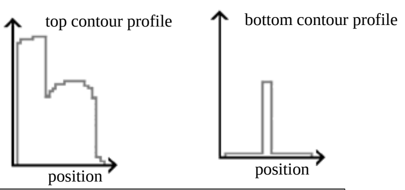




# Recognition of single letters (V)

**Contour profiles**

- Horizontal profile: Analyze top and bottom contour pixels.
- Vertical profile : Analyze left and right contour pixels.
- Aggregate these profiles into one vector.

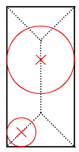


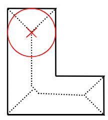



# Recognition of single letters (VI)

**Skeletons**

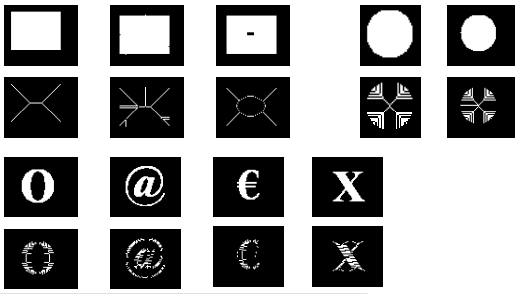
- Idea: Ascertain a region's structure
- Procedure: Skim the regions iteratively
- In 1967, the *medial-axis*-transformation algorithm was introduced:
  - R: region, B: margin
  - For all p ∈ R: Search next neighbor (e.g. via city-bloc-distance) in B.
  - If (number of neighbors > 1): pixel is medial axis (skeleton) of R.





## Recognition of single letters (VII)

**Example for skeletons by use of the city-bloc-distance**





## Recognition of single letters (VIII)

**Problems concerning the generation of skeletons**

- Minimal modifications of the contour result in very diverse skeletons.
- The computing time is very high since for every pixel the distance to every other pixel must be calculated.

**Optimization**:

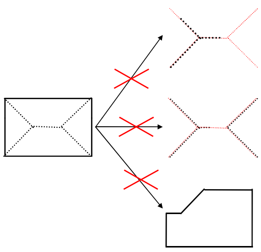
- Skim regions (remove margin pixels) so that:
  - Endpoints of the skeleton are skimmed as less as possible,
  - one region is not divided into two regions,
  - all regions of the object are skimmed equally strong (one region is not to be skimmed extremely strong).



## Recognition of single letters (IX)

**Conditions**

- Endpoints of the skeleton are to be skimmed as less as possible,
- one region is not to be divided into two regions,
- all regions of the object are to be skimmed equally strong (one region is not to be skimmed extremely strong).





## Recognition of single letters (X)

***Thinning*-algorithm for the generation of skeletons**

| | | |
|---|---|---|
| $p_9$ | $p_2$ | $p_3$ |
| $p_8$ | $p_1$ | $p_4$ |
| $p_7$ | $p_6$ | $p_5$ |

- Given: binary image (background=0, object=1)
- 8-pixel-neighborhood
  current pixel: $p_1$

1. Consider each margin pixel and mark pixels in case that all conditions are fulfilled:
   - $3 \leq N(p_1) \leq 6$
   - $S(p_1) = 1$
   - $p_2 * p_4 * p_6 = 0$
   - $p_4 * p_6 * p_8 = 0$
2. Delete marked pixels

*$N(p_1)$*: number of object pixels $p_2 \ldots p_9$ in the area of $p_1$

*$S(p_1)$*: number of transitions from background pixels to object pixels during expiration of $p_2, p_3 \ldots p_9, p_2$



## Recognition of single letters (XI)

***Thinning*-algorithm for the generation of skeletons**

1. Consider each margin pixel and mark pixels in case that all conditions are fulfilled:
   - $3 \leq N(p_1) \leq 6$
   - $S(p_1) = 1$
   - $p_2 * p_4 * p_8 = 0$
   - $p_2 * p_6 * p_8 = 0$
2. Delete marked pixels.
3. Go to 1. if at least one pixel was deleted.

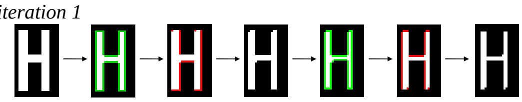
| | | |
|---|---|---|
| $p_9$ | $p_2$ | $p_3$ |
| $p_8$ | $p_1$ | $p_4$ |
| $p_7$ | $p_6$ | $p_5$ |



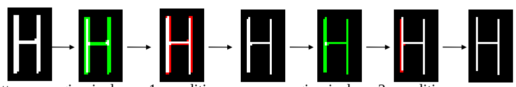
## Recognition of single letters (XII)

***Thinning*-algorithm for the generation of skeletons**

*iteration 1*



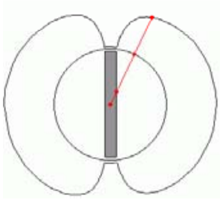

*iteration 2*



# Recognition of single letters (XIII)

**Text recognition with scale-space-images**

- Highly convex curved regions become concave regions.



**Procedure:**

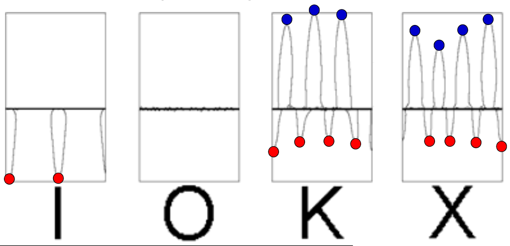
- Identify center of gravity
- Put circle around letter
- Mirror the letter's contour pixel at the circle line



# Recognition of single letters (XV)

**Text recognition with scale-space-images**

- Calculate normal scale-space-image
- Calculate scale-space-image for mirrored contour






# Experimental results (I)

**Database**

- Letters of four fonts were used.
- The scale-space-images were allowed to be turned maximum ~20 degree to recognize italic signs.

**Challenges**

- Text recognition concerning segmentation errors:




# Experimental results (II)

| segmentation errors | projection profiles | shortest paths |
|---|---|---|
| divided letters | 9.9 % | 3.8 % |
| connected letters | 7.5 % | 5.4 % |
| segmentation errors | 17.4 % | 9.2 % |

| text recognition algorithms | recognition results |
|---|---|
| pattern matching | 69 % |
| zoning | 64 % |
| contour profiles | 71 % |
| scale space images | 76 % |
| commercial OCR-software (scanner) | 75 % |



# Experimental results (III)




# Questions ?