Exercise: Image and Video Processing

Sheet 8 – OCR

Exercise 1 – Class OCR

Develop the class *OCR* to compare two characters:

- `void getPatternMatchingDistance (Image &img1, Image &img2, double &dist);`

  Calculate the distance of two images based on *pattern matching*. Assume that the size of both images is the same. Load images ocr_*1* to ocr_*4* and calculate the distances between each other.
- `void getZoningDistance (Image &img1, Image &img2, double &dist);`

  Calculate the distance of two images based on *zoning*. Assume that the size of both images is the same, and use regions of size 4x4 pixels. Load images ocr_*1* to ocr_*4* and calculate the distances between each other.
- `void getSkeletonMATCityBlock (Image &src, Image &dest);`
  `void getSkeletonMATEucledian (Image &src, Image &dest);`

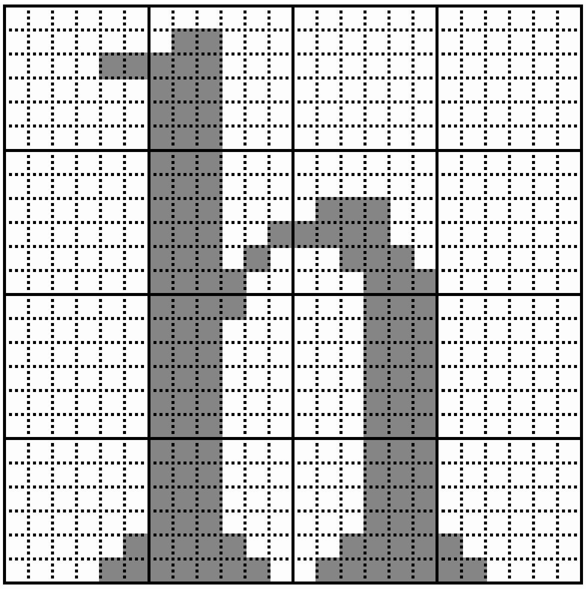
  Calculate the *skeleton* of a character based on the m*edial axis transformation* (*MAT*). Object pixels are white (255), background pixels black (0). Use *the city block distance* in one function and the *Euclidian distance* in a second function. At least two nearest neighbor pixels should have identical distance values to be part of the skeleton. It is only necessary that the distance is similar (*<0.06*) in case of the Euclidian distance. Calculate the skeletons for images *skeleton_01* to *skeleton_14*.
- `void getSkeletonThinning (Image &src, Image &dest);`

  Implement the *thinning* algorithm to create skeletons. Calculate the skeletons for images *skeleton_01* to *skeleton_14* and compare the results with the previous skeletons.

Exercise 2 – General questions (OCR)

1. Which information is made available by overlaid text in videos? Which information cannot be derived from the visual content?
2. Name the basic steps of OCR.
3. What is a projection profile? In which context is a projection profile used?
4. The *Dijkstra* algorithm can detect separators between characters. Describe the idea.
5. The computational effort to calculate the shortest paths for each text line is very high. Why? How could the computational effort be reduced.
6. A small image should depict just one character. How do you identify the pixels that are part of this character?

7. What is the feature vector for the following character if *zoning* (4x4 regions) is used:



8. How do the regular scale space images (without using mirrored shapes) look like for the characters „**I**“ and „**O**“? Why do the mapped scale space images give better results? Is it possible to distinguish the characters „**U**“ and „**C**“ based on scale space images?