Lab 5 – Morphological Operations

A common image processing task consist in finding given objects in a digital image and extracting simple features from these objects. During this lab we will work with morphological operations and component labeling algorithms.

Learning Objectives

- You know how to apply basic morphological image processing operations.

- You know how to locate simple geometric objects inside a digital image.

- You are familiar with the concept of component labeling.

1 Locate Squares of Given Size

Starting point is the image `squares.tif` that contains structures of various sizes– from 2×2 to 15×15. It is our goal to **locate and count** all squares with size 5×5.

1. Use the morphological operations *dilation* and *erosion* to remove all squares whose size is not 5×5. The corresponding Python functions are `binary_erosion(...)` and `binary_dilation(...)`—they are contained in the `scipy.ndimage` package. As usual, read the documentation.

2. Find and count all squares of size 5×5 using the concept of connected components.

   (a) Start with the image containing only (white) squares of size 5×5.

   (b) Select an arbitrary white pixel (seed points).

   (c) Find all white pixels that are connected to this seed point. As shown in the lecture, this can be done iteratively, with

$$X_{k+1} = (X_k \oplus B) \cap I, \; B = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$
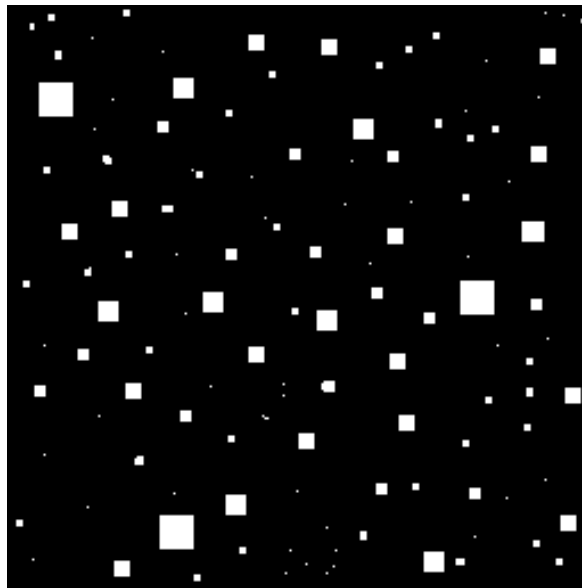
   The algorithm ends when $X_{k+1} = X_k$.

Figure 1: Your task: find all squares with size 5×5.

    (d) When you found all white pixels connected to your seed point this means that you found a 5×5. The next step consists in increasing the counter by one and setting all pixels you just found to black (background).

    (e) Proceed with the first step until there are no more white pixels left.

3. If we only cared about the number of 5×5 objects in the given image. Can you think of a simpler approach?

2   Counting Blood Cells

Starting point for this exercise is the gray scale image `bloodCells.tif`, shown in Figure 2. It is our goal to count and locate all blood cells that are inside the picture and remove all cells that are located at the boundary.

1. Binary image segmentation: For simplicity, we want to operate on a black and white (BW) image. The easiest way to convert a gray level image into a BW image is by global thresholding – all gray values above a given threshold are set to one (foreground), all other pixels are set to zero (background). Convert the gray scale image into a BW image by setting the threshold to the arithmetic mean between minimum and maximum gray level. Alternatively you can try out the Otsu thresholding method.

2. Remove all cells at the boundary. You can use for example `clear_border` from `skimage.segmentation`.

3. Due to the hard thresholding some of the blood cells feature holes. Try to fill these holes using morphological operations. Alternatively, you can use `binary_fill_holes(...)` from the `scipy.ndimage` package.
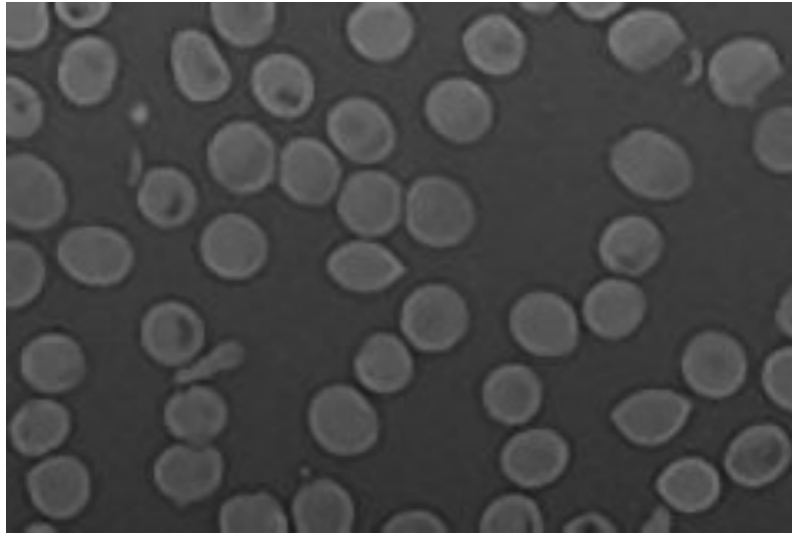
Figure 2: Blood cells.

4. Label and count all blood cells. As before, this can be done by finding connected components – in a similar way as before when we counted the squares. Yet, this procedure is slow, and it is faster to generate a label map. You can use the `ndimage.label(...)` function. Display the number of blood cells you found.

5. Object features. For each blood cell you found, determine the number of pixels.

6. Display a histogram showing how often each size occurs.