ESTIN

Pattern recognition for image analysis

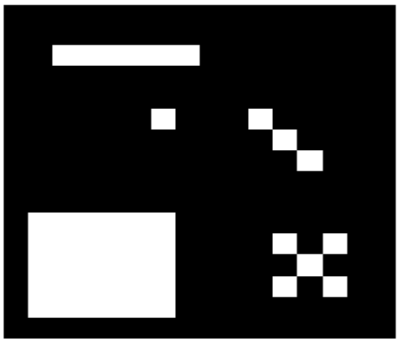
S5 2024-2025

Lab° 01

1.

- Create a function that labels all connected regions with a unique label, and computes the number of these connected components. (Use **scipy.ndimage.measurements**)

- Test this function on the following image



```
I=np.zeros((16,16)).astype(int)

I[2,2:8]=I[10:15,1:7]=255

I[5,6]=I[5,10]=I[6,11]=I[7,12]=255
I[12,12]=I[11,11]=I[13,13]=I[13,11]=I[11,13]=255
```

- Use **cv2.connectedComponentsWithStats** (from OpenCV)

2.

a) Write a python code (using **OpenCV**) to detect shapes in an image.

Approach

- Draw multiple shapes using paint (rectangle, triangle, and circle)
- Import this image
- Convert it to grayscale
- Apply thresholding and then find out contours (using **cv2.findContours**).
- Run a loop in the range of contours and iterate through it.

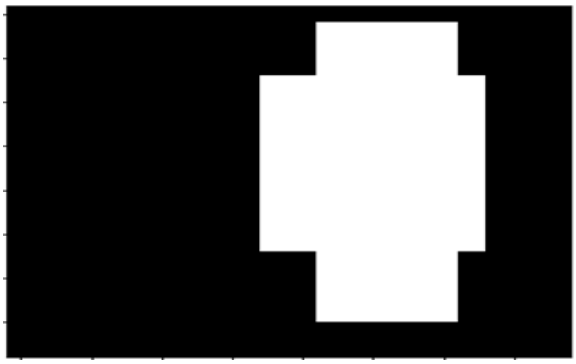
  In this loop:
  - Approximate the shape(using **cv2.approxPolyDP, cv2. arcLength**)
  - Draw contours (using **cv2.drawContours**) and find center point of the shape.
  - Classify the detected shape and put its name at the center point of the shape.

b) Recognize round objects in the previous image.

3.

- Implement Freeman chain code using python.
  - **Shape contours:**

    Before determining the Freeman chain code of the shape, it is necessary to extract its contour.
    - ✓ Generate or load a simple shape as a binary image A:



```
A = np.zeros((20,20)).astype('bool')
A[4:14, 9:17] = True
A [1:18,11:16]= True
```

    - ✓ Extract its contour $C4(A)$ ou $C8(A)$ according to the 4-connectivity or the 8-connectivity, respectively.

(You can erode the object and subtract this erosion to it, with a structuring element that corresponds to *N*4 or *N*8 if you want to have 8 or 4 connectivity, respectively. (use **binary_erosion** from **skimage.morphology**)

- **Freeman chain code:**
  From the shape contours, the Freeman chain code can be calculated.
  - ✓ From the binary array of pixels, extract the first point belonging to the shape (from left to right, top to bottom). (You can use **np.argwhere** to locate the first point.)
  - ✓ From this initial point, determine the Freeman chain code *c* (counterclockwise direction) using the *N*4 or *N*8 connectivity.