task 1.3 [15 points]

estimating the fractal dimension of objects in images

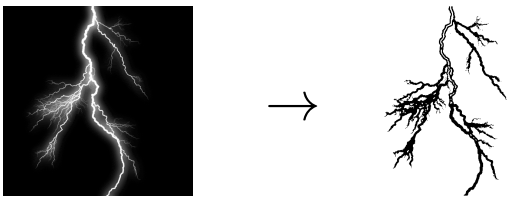
In our lectures, we discussed the use of least squares for linear regression. In this task, we consider a neat practical application of this technique.

background info

Box counting is a method for estimating the fractal dimension of an object in an image. For simplicity, we focus on square images whose width w and height h (in pixels) are integer powers of 2. For instance, if $w = h = 512$, then $w = h = 2^L$ where $L = 9$.

Given an image like this, the box counting procedure involves three steps:

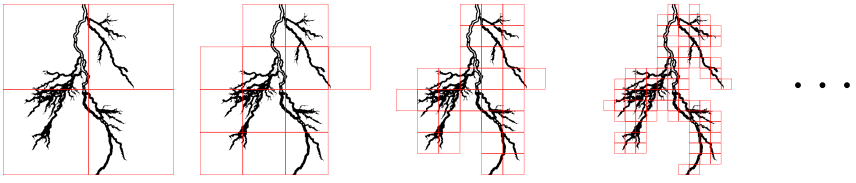
1. apply an appropriate binarization procedure to create a binary image in which foreground pixels are set to 1 and background pixels to 0



2. specify a set S of scaling factors $0 < s_l < 1$, for instance

$$S = \left\{ s_l = \tfrac{1}{2^l} \;\middle|\; l \in \{1, 2, \ldots, L-2\} \right\}$$

and, for each $s_l \in S$, cover the binarized image with boxes of size $s_l\, w \times s_l\, h$ and count the number n_l of boxes which contain at least one foreground pixel

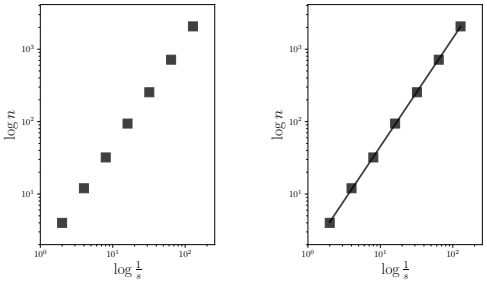


$s_l = \frac{1}{2}, n_l = 4$ $s_l = \frac{1}{4}, n_l = 12$ $s_l = \frac{1}{8}, n_l = 32$ $s_l = \frac{1}{16}, n_l = 94$ . . .

3. plot $\ln n_l$ against $\ln \frac{1}{s_l}$ and fit a line

$$D \cdot \ln \frac{1}{s_l} + b = \ln n_l$$

the resulting estimate for the slope D of this line represents the fractal dimension we are after. Here is an example for how such a line may look like:



In other words, the problem of estimating D is a simple linear regression problem that can of course be tackled using least squares.

here is your task

Implement `Python` / `numpy` / `matplotlib` code for the box counting method and run it on the images `tree.png` and `lightning.png`.

Which fractal dimensions do you obtain? Which object has the higher one, the tree or the lightning bolt?

Tip: We strongly suggest to work with the following imports

```
import numpy as np
import imageio.v3 as iio
import numpy.linalg as la
import scipy.ndimage as img
```

Tip: To read, say, image `tree.png` into a `numpy` array, you can then use

```
imgG = iio.imread('tree.png', mode='L').astype(float)
```

Note: While we do not really care about how you realize image I/O, we do care about how you binarize the images you read.

This is because after binarization, the outcome of the box counting procedure should be deterministic, i.e. the same for every team of exercising students. In other words, if every team uses the same binarization procedure, every team should obtain the same results and these results should be those your instructors got (up to numerical precision). Teams getting different results can rest assured they made a mistake.

Therefore, please use the following snippet

```
def binarize(imgF):
    imgB = np.abs(img.gaussian_filter(imgF, sigma=0.50) - \
                  img.gaussian_filter(imgF, sigma=1.00))

    return img.binary_closing(np.where(imgB < 0.1*imgB.max(), 0, 1))
```