Lab 7 & 8 Retina recognition

1. Retina vessels segmentation

We will use method inspired by:

Z-W Xu, X-X Guo, X-Y Hu, X Cheng,*The blood vessel recognition of ocular fundus*, Proceedings of the 4th International Conference on Machine Learning and Cybernetics, pp. 44934498, 2005: https://ieeexplore.ieee.org/abstract/document/1527730

Download RIDB dataset:

https://drive.google.com/drive/folders/1FtIX04PLdeDe7I_n8zG2xhzVngwnnlLU?usp=sharing

For each image in RIDB do the following steps:

- convert image to the grayscale
- prepare mask to remove the black border from the image:
    - use floodfil algorithm (cv2.floodFill) from the point (0, 0)
    - remember to work on the gray image copy:

      ```
      img_floodfill = img_gray.copy()
      ```

    - to obtain proper mask one can use the following code:

      ```
      h, w = img_floodfill.shape[:2]
      floodfill_mask = np.zeros((h+2, w+2), np.uint8)
      ```

    - dilate the floodfill result to cover pixels near the ROI (retina image) - use kernel of size 11 or 13:

      ```
      dil_kernel = np.ones((13, 13),np.uint8)
      ```

    - finally, do the bitwiste_not and binarization with small threshold (e.g. 1) to obtain the proper mask
- enhance contrast of the grayscale image using CLAHE algorithm (clipLimit = 2.0, tileGridSize = (8,8))
- invert enhanced grayscale image (255 - img) and once again use the CLAHE algorithm
- blur image with the gaussian kernel (size 7x7, sigma calculated from size)
- do the adaptive thresholding with gaussian-weighted sum of the neighbourhood values (cv2.ADAPTIVE_THRESH_GAUSSIAN_C, C = 0)
- combine threshold output with previously prepared mask (cv2.bitwise_and)
- blur image with the median filter (size 5x5) and invert values (cv2.bitwise_not)

- fill the holes with morphological close operation - you can use kernel from previous dilation
- again invert the image
- get objects stats and remove background object by:

```
nb_components, output, stats, centroids =
cv2.connectedComponentsWithStats(img_closed_inv, connectivity=8)
sizes = stats[1:, -1]
nb_components = nb_components - 1
```

- remove smaller objects (area < 500)
- finally, get the skeleton of the preserved objects - you can use the following code:

```
skel_element = cv2.getStructuringElement(cv2.MORPH_CROSS, (3,3))
skel = np.zeros(img_obj_filtered.shape, np.uint8)
while True:
    opened = cv2.morphologyEx(img_obj_filtered, cv2.MORPH_OPEN,
skel_element)
    temp = cv2.subtract(img_obj_filtered, opened)
    eroded = cv2.erode(img_obj_filtered, skel_element)
    skel = cv2.bitwise_or(skel, temp)
    img_obj_filtered = eroded.copy()
    if cv2.countNonZero(img_obj_filtered)==0:
        break
```