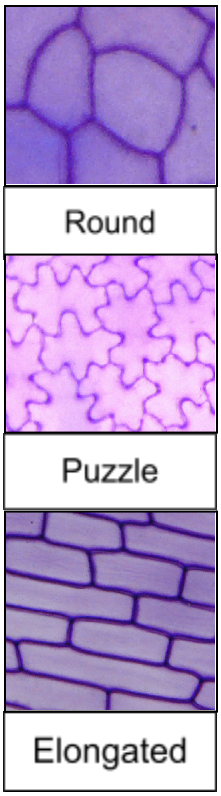
Quitterie
Inès
Yovan

Report : analyze leaf epithelium

Introduction :



The goal of this project is to analyze the leaf epithelium of several species by looking at its cells. When a plant grows, the cells already present stop multiplying. In order to continue to grow, the plant needs its cells to grow themselves since they are no longer dividing. The cells can grow up to 100 times their original size ! Depending on the direction of growth of the plant, the cells will take different shapes (puzzle, elongated, round...). To characterize the growth of the different leaves in this project, we will count the cells, look at the directions of expansion of the cells, and look at their connections between them.

To better understand the project and how we can study the cells, we will introduce to you some vocabulary about plants and their cells. Here we consider cells on the leaf, and particularly the structure of their epithelium. Epithelium is a lining or secretory tissue of the internal (cavities) and external surfaces made up of juxtaposed cells strongly linked, on a layer (simple epithelium) or several layers (stratified epithelium). The external layer is called epidermis, which characterizes a layer of cells on the surface of a plant. Its role is to protect the plant from mechanical damage, and to avoid water loss, thanks to the stomata. The stomata, which we have just talked about, are small orifices present in the epidermis of plants. They allow gas exchanges between the plant and the ambient air (oxygen, carbon dioxide, water vapor...) as well as the regulation of evapotranspiration and osmotic pressure. On a leaf, we distinguish the two faces, abaxial and adaxial, according to the number of stomata they contain. Abaxial is said for the lower face of the leaf, and it contains many stomata. Conversely the adaxial face is the upper face and does not have many stomata.

Now that we described the structure of a leaf, we will talk about cell growth. Each cell has a membrane, also called a cell wall (the outer layer of the cell) that maintains the cell's shape. This membrane undergoes a pressure, the turgor pressure, which will change its shape. In fact, the turgor pressure of a cell defines the tension at its walls when it absorbs a flow of water in its vacuole by osmosis. This pressure stiffens the non-rigid parts of the plant. This happens when the internal concentration of the cell is very high compared to the outside of the cell. We recall that vacuole is a compartment of a plant cell that contains an aqueous solution of mineral salts and organic compounds. As mentioned, it plays an important role in the volume changes of the cell. The cell structure is also modified by microtubules. Indeed, microtubules, together with microfilaments, constitute the cytoskeleton of the plant cell. They have a hollow cylinder structure, formed by dimers, alpha and beta tubulins. In addition, their main functions are to manage the organization of the compounds and the rigidity of the cell, but they also determine the shape of the cell.

The cells that we study here were stained by toluidine blue staining. Toluidine blue, also called trimethylthionine hydrochloride or tolonium chloride, is a blue cationic dye, close to methylene blue. Toluidine blue staining is used to stain different elements of a cell blue, purple or green. It is generally used for histological and intravital staining. Thanks to this dye, all the cells of our pictures are blue or purple !

Quitterie
Inès
Yovan

Now that the essential terms are defined, and that the structure of the cell and their growth is well understood, we can analyze the leaf epithelia. The first step will be to segment the different images. The segmentation used will be different in function of the cell shape, and also if there are stomata or veins for example. After segmenting the different images, we will be able to answer three questions successively :
1) How many cells ?
2) How can we infer the cell expansion directions ?
3) How are cells connected ?

Python libraries :

This project was made on Jupyter Notebook.

We need to use the several libraries as : `napari` , `numpy`, `pandas`, `matplotlib`, `seaborn`,`scipy`,`scikit-image`,`sklearn`

`Pip install “napari[all]”`
`pip install nearest-neighbors`
`pip install pandas`
`Pip install numpy`
`Pip install matplotlib`
`Pip install scipy`
`Pip install Scikit-image`
`Conda install seaborn - c conda-forge`

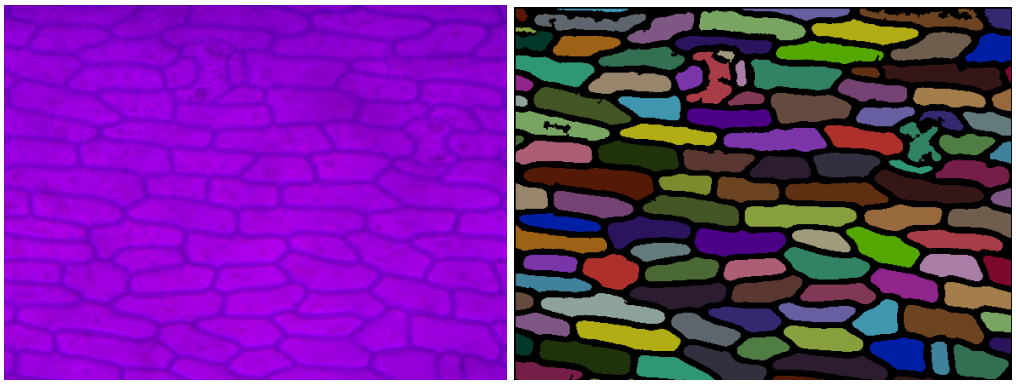
We need to also to import : `glob` and `tifffile`

“How to use” :
Inès :
Segmentation :
Segmentation is essential for this project. Indeed, segmentation is necessary for the three big questions of this project.
Steps :

1- Sort all images into different cell categories (puzzle, crocodile, long, veins and stomata).
2- Import all the necessary libraries to run the jupyter notebook.
3- Run the segmentation functions for each category, where we put the image in black and white, then we apply the median blur in it. Next, we do the threshold and remove all the small holes and objects. If it’s necessary (for some categories) we do the dilatation, erosion and fill holes. To finish, we labeled the mask or we did the watershed.

Results in pictures :

Quitterie
Inès
Yovan



For a **cell image** to **segmented image.**

Limitations : For the limitation of the segmentation, some images had the luminosity rather low or on the contrary too high. So the parameters are totally different, unfortunately by lack of time we could not treat some images where the luminosity is bad (too low). Moreover, some images in addition to the brightness had veins or stomata, so the segmentation was too complicated that's why this type of image was not processed either. Since we made functions for several categories of cells, we could not optimize to the maximum for all the images.

Yovan :
Clean CSV

At first, we cleaned all the images by deleting the one that were not enough clear, and we sorted them by shape and if they have veins or stomata as the following: croco, puzzle, longues, stomates, veines.
I took the csv file 'global_database.csv' and I put NaN where I delete images, put a column with our categories, and I sorted and delete the NaN lines with python and panda.

Inès :

1) **How many cells?**

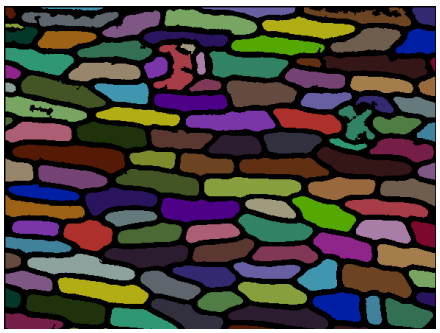
In this question we want to determine the number of cells in each image. So we have to make the segmentation of each picture then label the mask and finally count the number of labels.

Steps :

1- Import all the necessary libraries to run the jupyter notebook.
2- Run the code for each category (longue, puzzle, veine, stomate and croco) where there is the segmentation function of the corresponding category. In the same code, we count the number of labels, so the cells then put the centroids for all the images and put it in a CSV file separately (in the folder result_per_image). Then all the results for each image are collected in a dictionary (the image name, the number of cells and the centroid CSV name), and converted to a dataframe.
3- After having all the dataframes for all the categories, it is necessary to concatenate them all.
4- Convert The final dataframe to a CSV file.

Quitterie
Inès
Yovan

Results in pictures :



	A	B	C	D
		image_fname	nb_cell	centroids_fname
	0	7-15-400x-2-05.tif	131	centroids_7-15-400x-2-05.csv
	1	7-01-700x-t-02.tif	99	centroids_7-01-700x-t-02.csv
	2	7-24-400x-1-05.tif	117	centroids_7-24-400x-1-05.csv
	3	7-01-700x-t-01.tif	114	centroids_7-01-700x-t-01.csv
	4	7-01-700x-t-04.tif	97	centroids_7-01-700x-t-04.csv
	5	7-01-700x-t2-01.tif	159	centroids_7-01-700x-t2-01.csv
	6	7-24-400x-1-02.tif	103	centroids_7-24-400x-1-02.csv
	7	7-15-400x-2-03.tif	136	centroids_7-15-400x-2-03.csv
	8	7-24-400x-1-04.tif	119	centroids_7-24-400x-1-04.csv
	9	7-01-700x-t2-05.tif	166	centroids_7-01-700x-t2-05.csv
	10	7-01-700x-t2-04.tif	157	centroids_7-01-700x-t2-04.csv
	11	7-01-700x-t2-03.tif	161	centroids_7-01-700x-t2-03.csv
	12	7-24-400x-1-01.tif	117	centroids_7-24-400x-1-01.csv
	13	7-01-700x-t-03.tif	113	centroids_7-01-700x-t-03.csv

From **segmentation** to **CSV file**.

Limitations :
Regarding the limitations we can say that we can not make the parameters for each image so the segmentation is not optimized for each image. As a result, we have some values that appear to be outliers in the CSV file.

Quitterie :

2) **How can we infer the cell expansion directions?**

In this question, we want to determine the cell expansion directions for each picture. To do that, we will analyze the area and the lobeyness of all cells, and compare species with these characteristics. The lobeyness can be defined as the perimeter of the cell divided by the perimeter of its convex hull. A cell with a high lobeyness value has a significant lobed shape. Thus, plants containing puzzle cells will have a high lobeyness because they have a lobed shape. The rounder cells will have a smaller lobeyness.

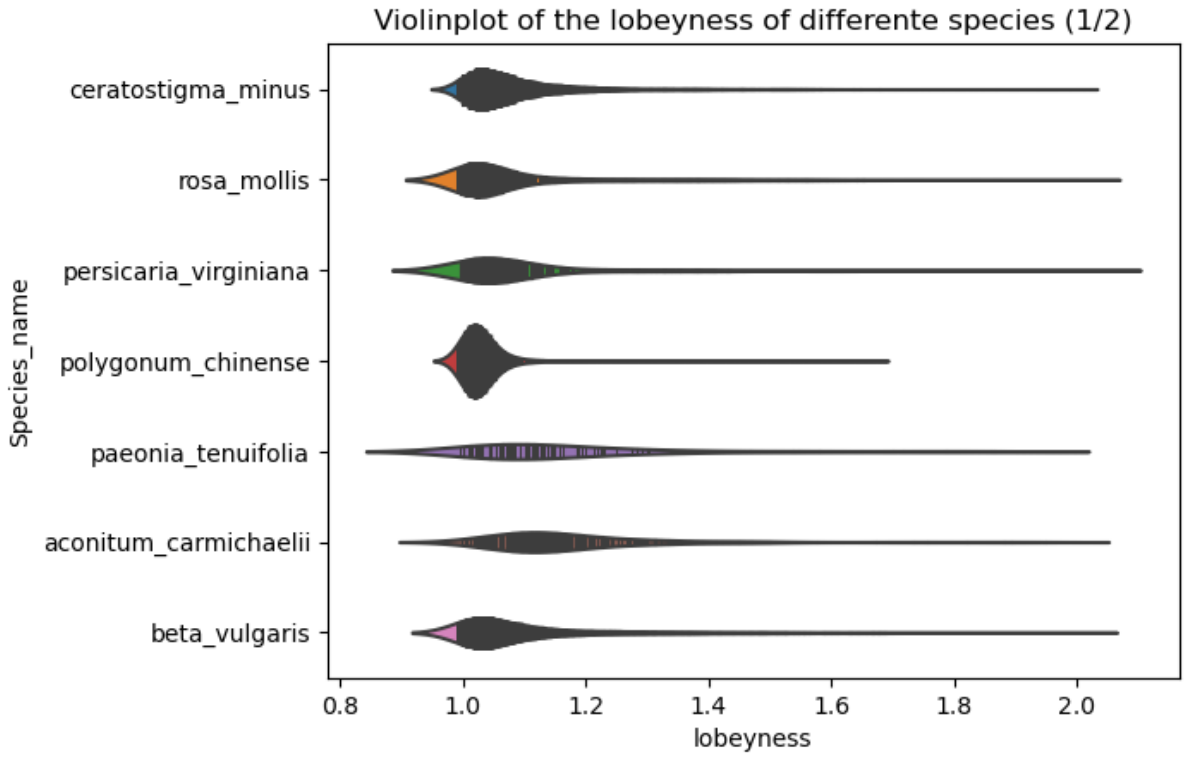
Steps :

- Import all the necessary libraries to run the jupyter notebook.
- Using the segmentation functions, save the colored masks of each image in a new folder (bonus section)
- Sort the saved masks by creating a folder by species.
- For each specie folder, run the function ‘species’. This will return a dataframe by species, containing for each cell its area, perimeter, lobeyness…
- Create a new dataframe to put the dataframes of each species in the same table (here for a better visibility of the graphs, we have created two different big dataframes).
- Remove the outliers of each dataframe to better see the graph (here we estimate them to be above 2 and below 0.99). This operation allows us to obtain readable graphs. Indeed, without removing the outliers, the violin plots looks like lines, which does not allow you to analyze them correctly afterwards.
- Plot the violin for each clean dataframe returning the lobeyness for each species.

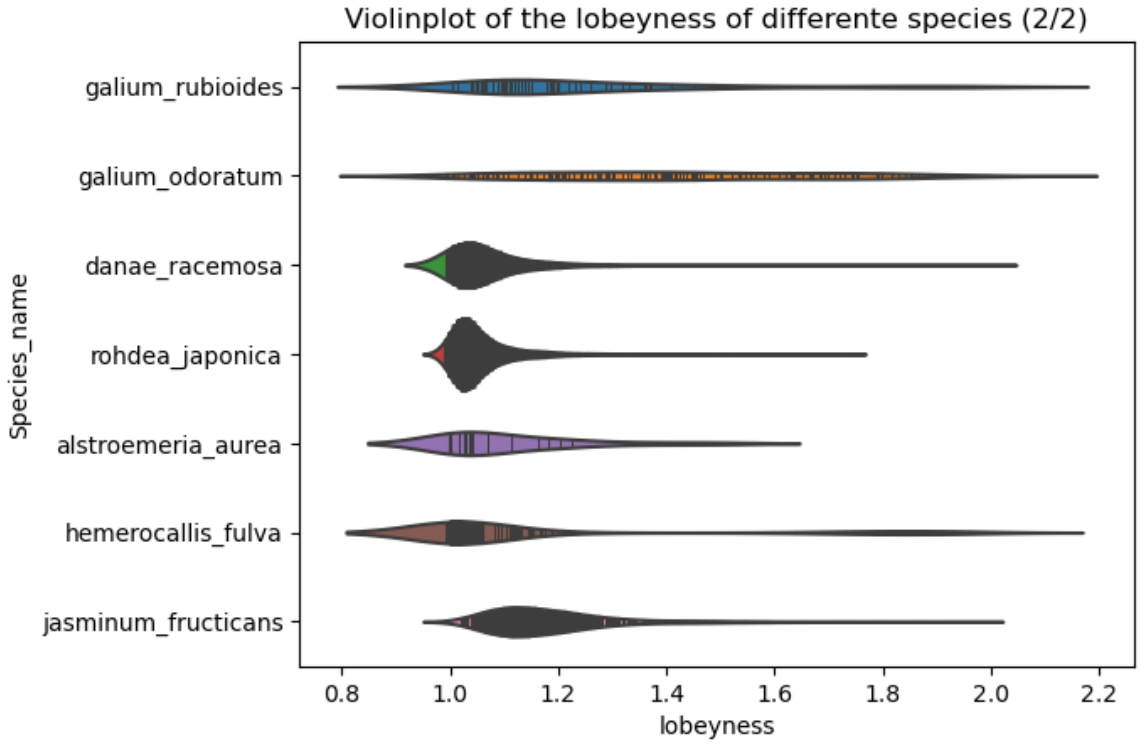
Quitterie
Inès
Yovan

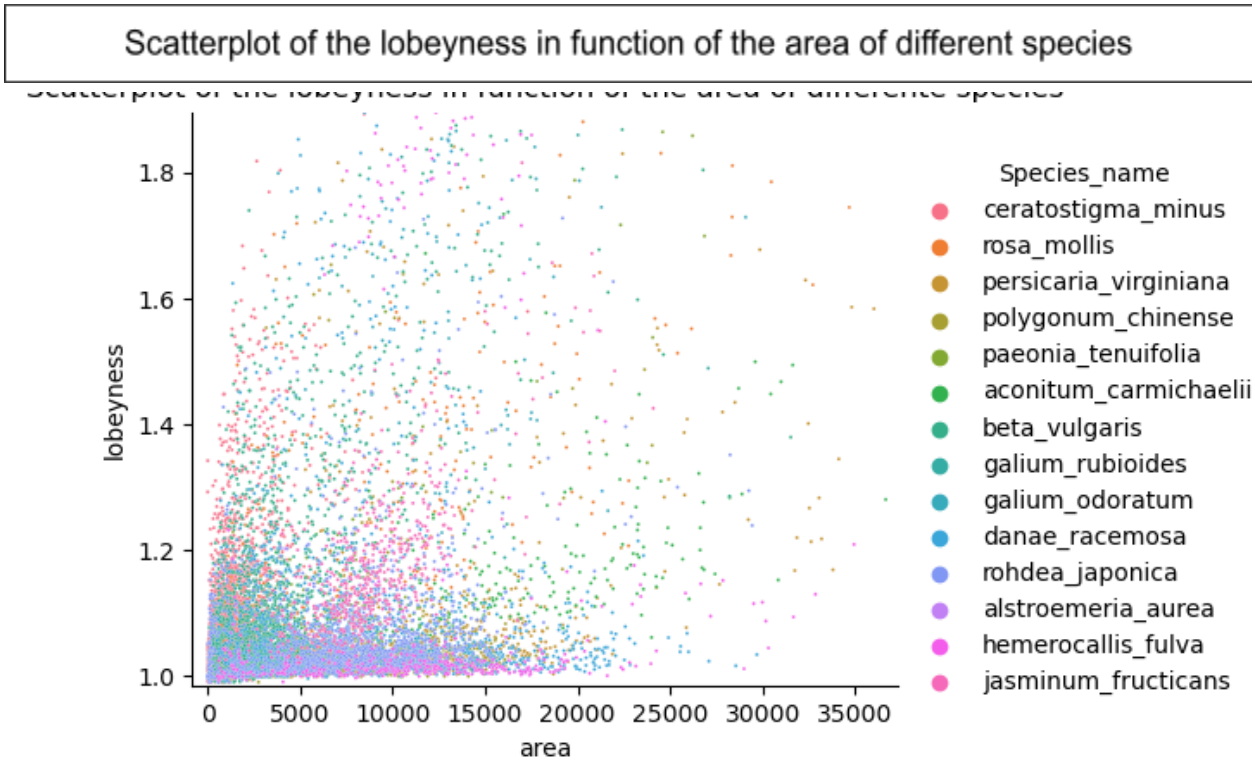
- Plot in a scatter plot the lobeyness in function of the area of each cell. We use one color per species.
- Bonus : to check if the lobeyness is correct for an image, you can use the parametric map function (bonus section at the end of the jupyter notebook). This will open a napari viewer and show the parametric map of the chosen picture in turbo color.

Results in pictures :



Quitterie
Inès
Yovan





Limits :

- For the sake of time, we have segmented the photos by group using the same parameters for similar images. However, the segmentations of some images are not ideal, because some cells are stuck... This gives outliers that we have corrected in the violin graphs. A more adapted segmentation for some images can reduce the outliers.
- For the scatterplot, the image displayed here is a zoom of the original graph, which shows the interesting part (without the outliers).

Yovan :

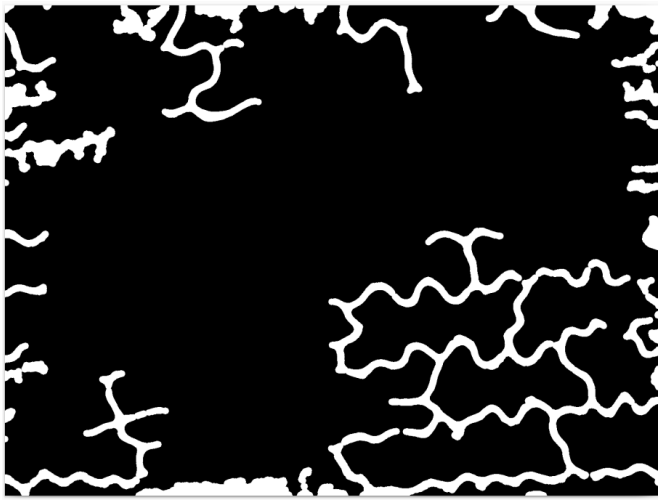
3) **How are cells connected?**

Steps :

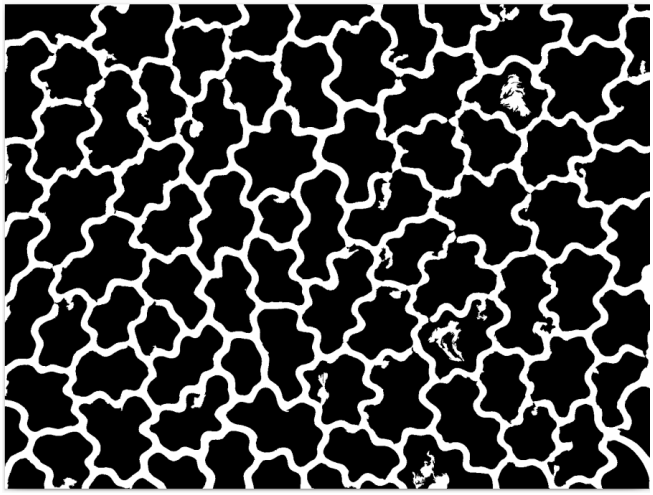
- we open the cleaned csv file as a dataframe containing all the names, categories, etc, of the cell images
- We sort the dataframe by ascending values in the column "categories" (which contains the names of the categories we chose to sort the images and clean our data)
- The deleted cell images have NaN values in the column "categories". We delete them here by deleting the lines containing NaN in the column "categories" using the panda function .dropna
  We import the csv file that we created in excel as a dataframe. It's the csv file of the first point of the To-do list of "How cells are connected ?"
- As the column "image_name" are empty, we take the values of the column "cell_fname" of the global_df_cleaned2 dataframe and use the panda function .assign to put them in the column "image_name" of the cells_connections dataframe.
- To replace "image_name" in 'non_triangular_junctions-image_name.csv' with excel
- I use the code did by Inès to get all the final masks for each categories without the colors (without watershed or labels)
- Then I save the results per images per categories in ‘sauvegarde’
- I Create a function to reverse the mask and get an image with the membranes only
- Refine the membranes in one pixel size line to count the different surrounding pixels of each pixel using skeletonize function
- After that I Turn the array in an array with just 0 and 1
- Inès already wrote a code to create all the centroid, count them, and save these data in a csv file, so I use these csv files to continue
- Get euclidian distance using NearestNeighbors function
- Then I should get one dataframe (that we will export to csv files) per image that I convert into a csv file named non_triangular_junctions-image_name.csv that contains a list of centroid of each occurence per image
- Finally I create a bar plot showing in x the species and in y the measured (non_triangular / total) value for each image

Quitterie
Inès
Yovan
Results in pictures :

Result for **croco** after taking segmentation results without watershed or label, reverse the mask and skeletonize the image :

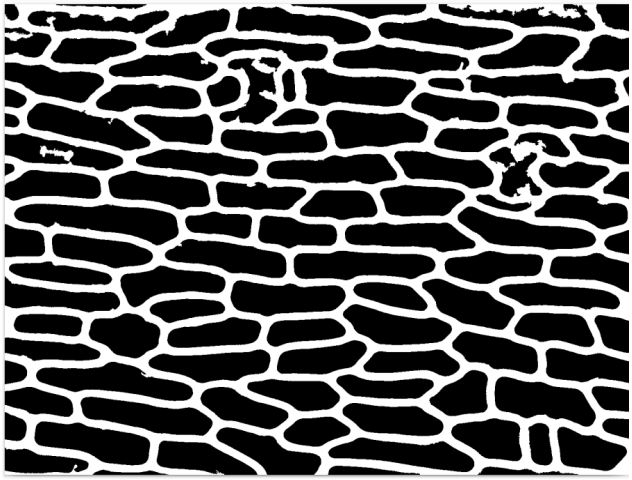


Result for **puzzle** after taking segmentation results without watershed or label, reverse the mask and skeletonize the image :

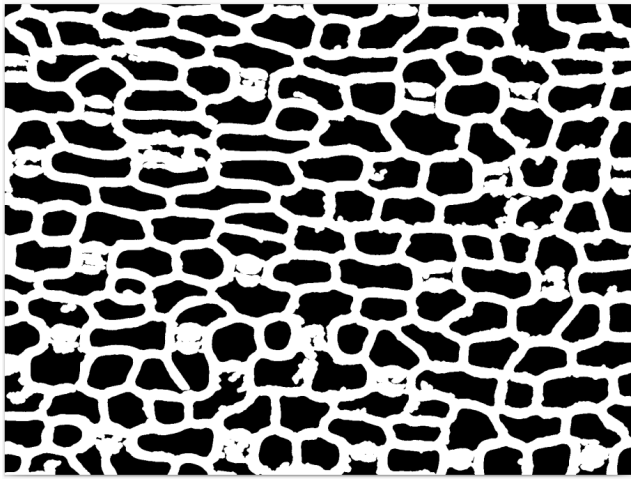


Quitterie
Inès
Yovan

Result for **longues** after taking segmentation results without watershed or label, reverse the mask and skeletonize the image :

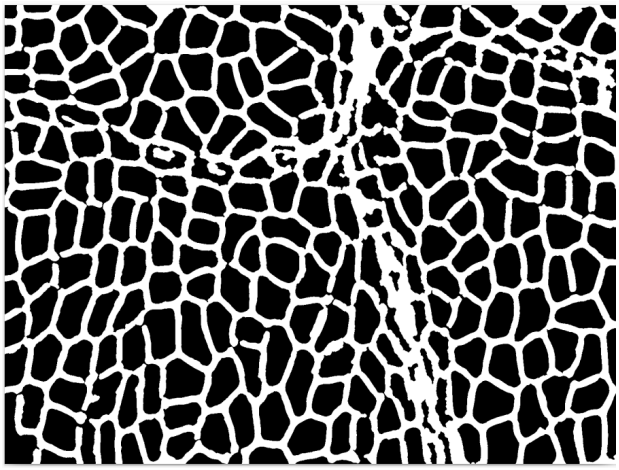


Result for **stomates** after taking segmentation results without watershed or label, reverse the mask and skeletonize the image :



Result for **veines** after taking segmentation results without watershed or label, reverse the mask and skeletonize the image :

Quitterie
Inès
Yovan



Limits :

- Due to the limit from inès question, how many cells ? : “Regarding the limitations we can say that we can not make the parameters for each image so the segmentation is not optimized for each image. As a result, we have some values that appear to be outliers in the CSV file.” I have the same limits on my mask reversed and skeletonize result on the images.
- Also, I should have last bugs in my code because I got empty dataframe and then csv files at the end using NearestNeighbors function. I’ll try to resolve that after the deadline.

Quitterie :
How to improve this project :

- Give images to train before the individual exam (to see if the functions are correctly working for any image)
- Establish different segmentations for other types of images (with veins and stomata, bad lightning…)

Sources :

- A. Sapala et al., “Why plants make puzzle cells, and how their shape emerges,” eLife, vol. 7, p. e32794, Feb. 2018, doi: 10.7554/eLife. (Why plants make puzzle cells, and how their shape emerges | eLife)