PSD from Powders – Guidelines

Scope and summary

The scope of this code is the analysis of particles' size distribution using as input an SEM image of the associated powder. An example of a suited image for this code is reported in Figure 1. The better the particles are well separated, the better the code will work.

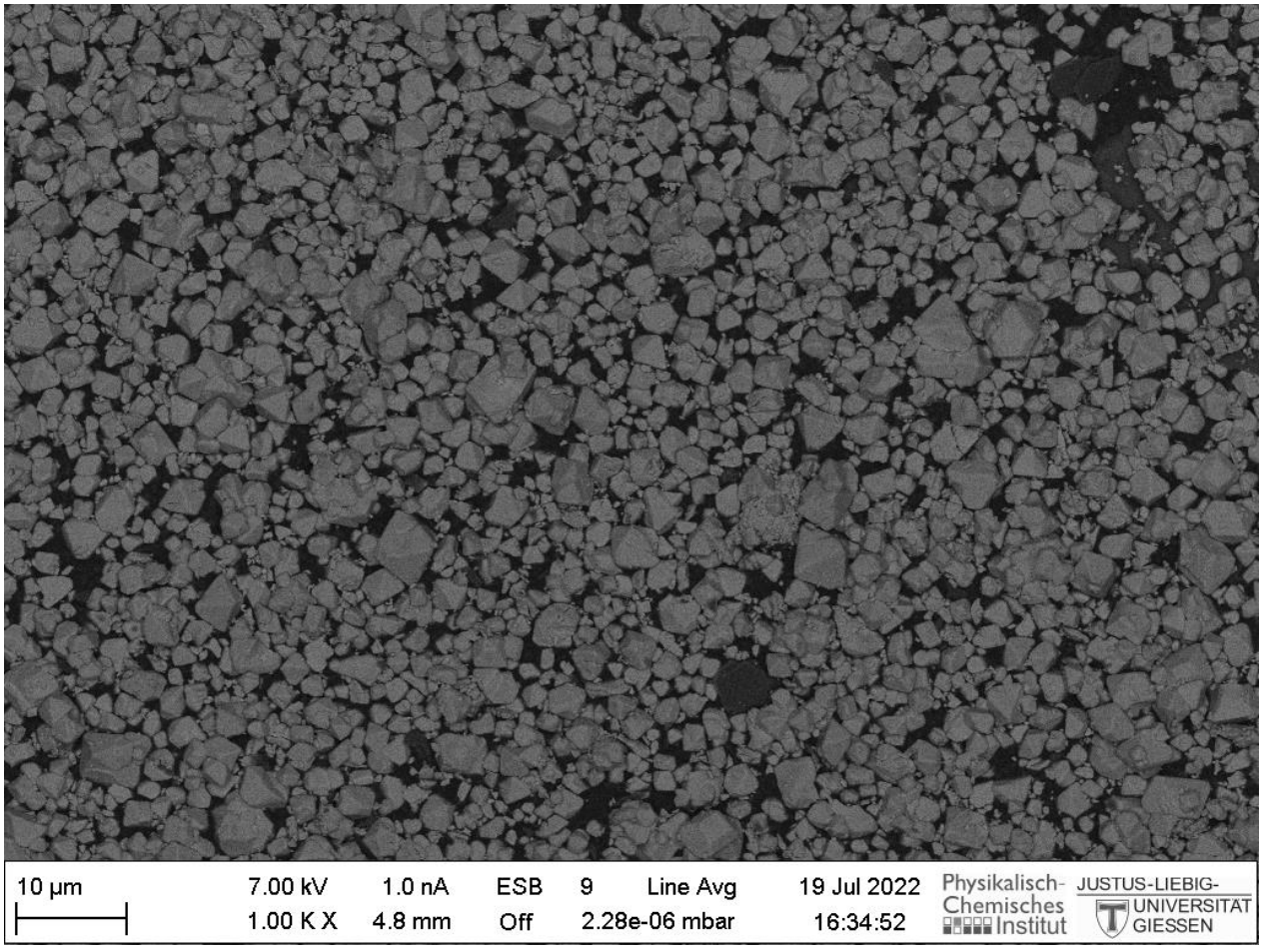

Figure 1. Example of SEM image suited as an input for this code.

To run the code, you have to place in the same folder both the code (python or jupyter) file and the input file, named, respectively, "Particle Size from powders-Final.py" (or .ipynb) and "Inputs_powder_PSD.xlsx".

Afterward, you can fill the input file (from hereafter, the “Excel”) with the parameter requested, including the file location and name, the color and length of the scalebar (for the determination of the image resolution), and the parameters needed for the Voronoi-based instance segmentation, *i.e.*, the identification of every particle in the image.

Particles touching the border of the image can be excluded from the analysis (as they are likely to be cut, it is advised to do so) through the “Remove Particles on Borders” parameter (Excel file).

You can find more information on the Voronoi-based approach, upon which this code is built, in the following YT videos:

https://youtu.be/g5FnaNtcCzU

https://youtu.be/wtrKToZNgAg

https://youtu.be/evgRgDfVuEc

The objective and meaning of each parameter (together with some reference value) are discussed in the Excel file in the “Parameters’ description column.

Once the Excel is filled, save it and run the python (or Jupyter notebook) code (a message will appear a few seconds after you do so, and another one when the analysis is finished). The analysis should be fast, likely less than a minute.

Before running the code (If not done already)

Before running the code, you will have to install either Python or Jupyter (an interactive notebook based on Python) and the libraries (pieces of code already done and made public in the Python community) used by the PSD from powders code. Both steps are very easy and you find many tutorials online. The easiest path is probably installing Anaconda (containing Jupyter Notebook, among other things):

https://www.anaconda.com/products/distribution

After that, you can launch the Anaconda Navigator (just search it after the installation) and from there you can run Jupyter Notebook (Figure 2).

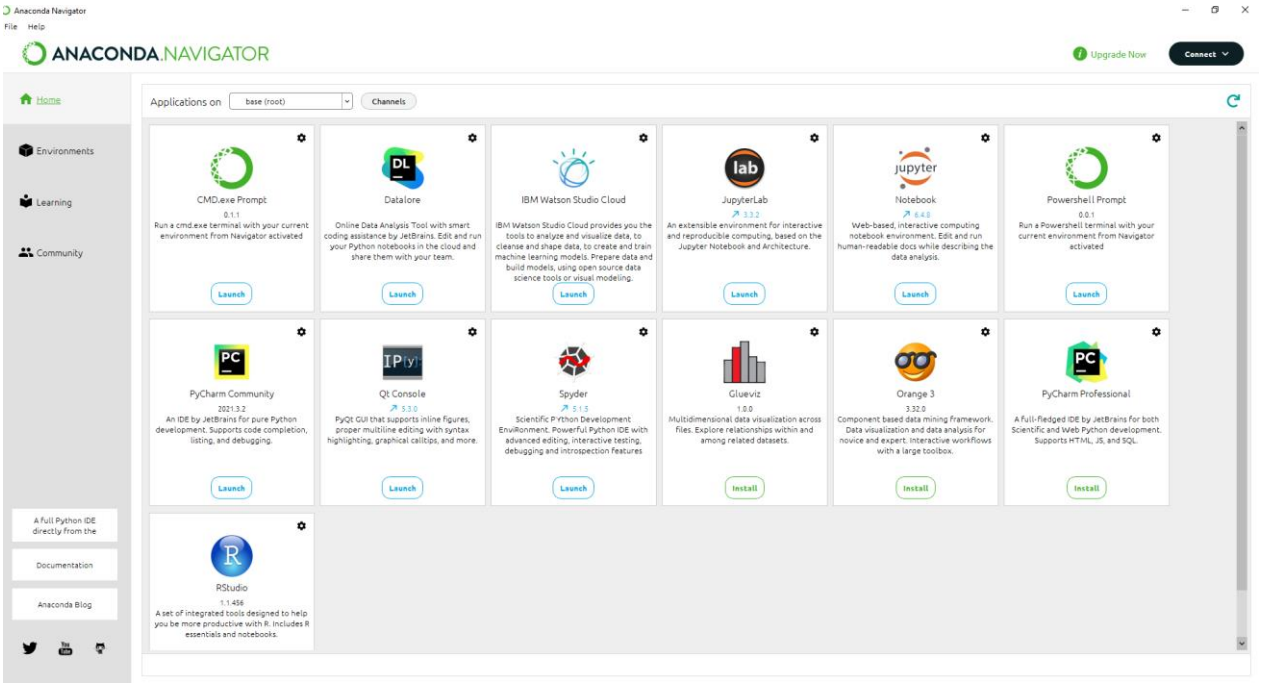

Figure 2. Anaconda Navigator.

The libraries to be installed are the ones reported in the very top of the code (“import Xxx” or “from Xxx”). It is very easy to install them (you can also google, library by library, how to install them, but typically it works for all in the same way):

https://youtu.be/Yr_ihLKq_yY

If you want to understand a bit better and learn more about Python (that can really be handy, so I definitely advise it!), you can look at the following videos series:

https://youtu.be/7uE6hypji0o

Results

The first important result is how well this procedure worked for your case stud. For this, you can look at the “Recognized and overlapped particles – Voronoi.png” file. One example of it is reported in Figure 3. When looking at the image try also to zoom to better identify the borders of the identified particles (the colors are random, then it may seem that certain borders are not there because of low contrast with the particle, but by zooming you can eventually see them).

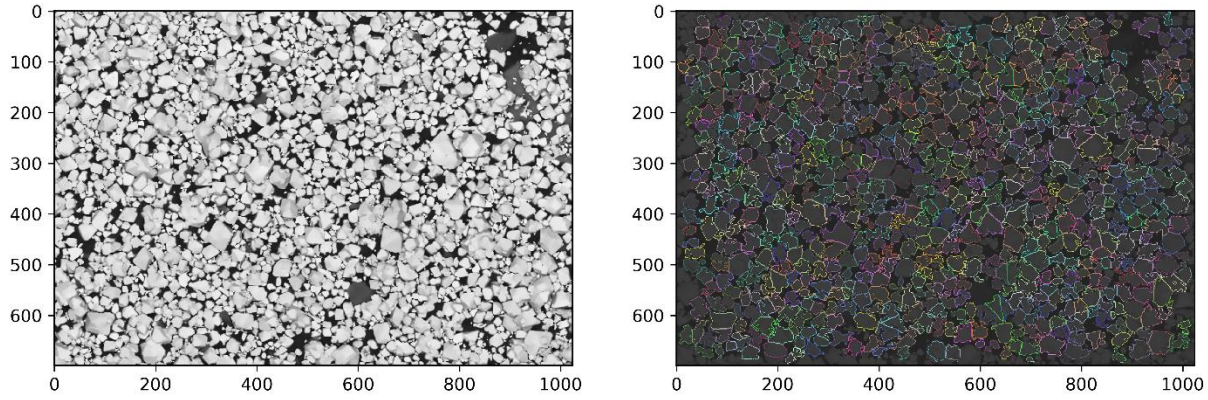

Figure 3. Example of identified particles (right) and a comparison with the original image (left) after some image processing to enhance contrast.

If you are not satisfied with the results, you can try playing with the model parameters. If none of them satisfies you, it may be that your images are not suited for this analysis through the approach developed here, and you might think if prepare differently your SEM sample (*e.g.*, trying to disperse them more) or if trying a different particles identification analysis.

The main results offered by the code are the particle area and (effective) radius distribution (Figure 4 A, B), as well as their aspect ratio (Figure 4 C, D). The effective radius is calculated starting from the area of the particles and assuming a circular shape. The aspect ratio is defined as the ratio between the major and minor axes (aspect ratio equal to 1 in a circle). The smaller the aspect ratio the more spherical-like the particle, while higher values indicate elongated (elliptical).

Other two results are the solidity and Wadell circularity.

Basically, the solidity tries to quantify how "regular" your particles are (how well their size fits within a regular polygon of any number of sides. The value reported refers to the calculation with respect to the regular polygon whose number of sizes fits the best that given particle).

The Wadell circularity is a quantification of the sphericity (and then regularity) of a particle.

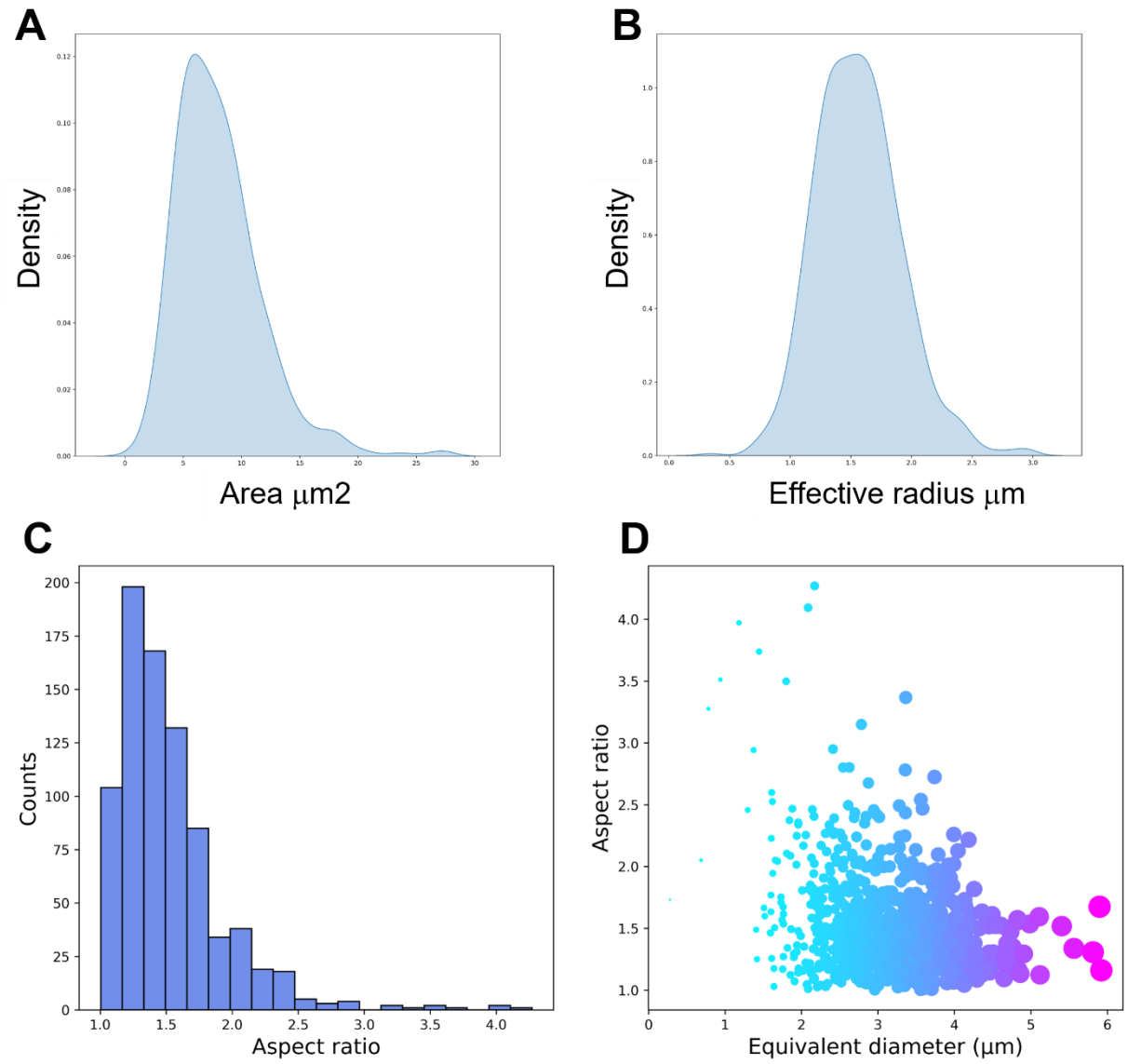

Figure 4. A, B) Area and effective radius continuous probability density distribution (seaborn.kdeplot — seaborn 0.12.2 documentation (pydata.org) , Probability density function - Wikipedia , statistics - How can a probability density function (pdf) be greater than 1? - Mathematics Stack Exchange). The code plots also these results in terms of

histograms. **C, D)** Histogram of the particles' aspect ratio and its distribution as a function of the particle size (reported in the x axis of Figure D, as well as in the data point size – proportional to the particles' effective radius – and color – the more violet the bigger the particle).

Lastly, the code also outputs two csv files including all the information calculated from the identified particles.

Contact for problems/doubts

If you have any problem using the code, feel free to contact me on my personal email: teo.lombardo3@gmail.com