Practical session for "Advanced Image Analysis"

D. Legland

September 16, 2023

Abstract *This document is the guideline for the practical session about "advanced image analysis". It is composed of two parts that follow the talk outline. In the first part, it is proposed to investigate errors resulting from image analysis algorithms. The second part is devoted to image texture analysis. Both parts can be treated independenlty.*

Contents

1 Validation of image analysis

This practical proposes to investigage errors made by image analysis algorithm, and to compare the behaviour of several algorithms.

1.1 Requirements

The practical will use the Fiji software. The following plugins are necessary:

- MorphoLibJ (installed by default)
- Digital_Shapes[1], a library that allows generating binary images of discretized shapes in 2D and 3D.
- ij_Geometry[2], a geometry processing library for ImageJ, required by Digital_Shapes.

A basic knowledge of ImageJ and of macro will be necessary.

1.2 Comparison of perimeter measures

In this session, we will compare the results of several methods for measuring the perimeter of binary shapes. We will focus on basic shapes with known theoretical values of perimeter to be able to evaluate resulting errors, and compare the results obtained with native ImageJ's "Analyze Particles" and the "Analyze Regions" from the MorphoLibJ library.

We will evaluate (relative) errors based on the following equation:

$$\delta_P^{\text{method}} = \frac{P_{\text{method}} - P_{th}}{P_{th}} \tag{1}$$

1.2.1 Disk

Let us start with the most simple shape: the disk. The perimeter of a disk with radius r is $2\pi r$. When using a disk with radius 50 pixels, the expected value of perimeter is around $P_{th} = 314.15$.

- Create a new image, with size 200x200
- Run the Plugins ▹ Digital Shapes ▹ Fill Disk
- Choose a center around $(100, 100)$, a radius equal to 50, and click "OK"
- Note that you can slightly shift the center to make the disk less "symetric". Use for example $(100.23, 100.34)$.

You shoud obtain a disk in the middle of your image (Fig. 1).

[1] `https://github.com/ijtools/ijDigiShapes`
[2] `https://github.com/ijtools/ijGeometry`

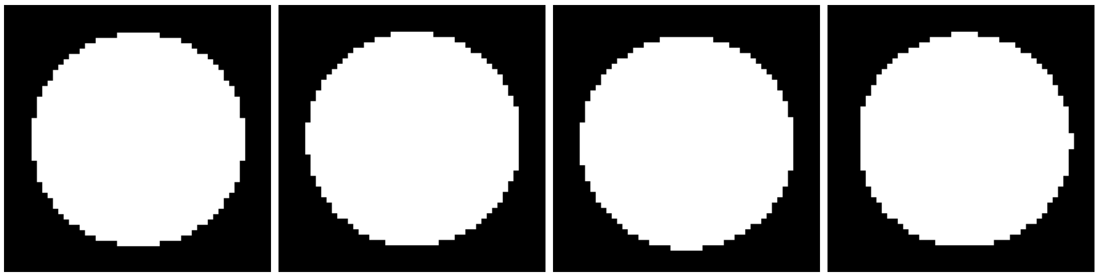


Figure 1: Examples of discretized disks with same radius (R=20). The left-most image corresponds to a disk centered with pixel grid.

Result with ImageJ

You can measure perimeter with standard functions of ImageJ via the “Analyze Particles” plugin. You may need to set the threshold before (Image ▹ Adjust ▹ Threshold..., click “yes”), and to make sure the perimeter is selected in the list of measures (Analyze ▹ Set Measurements, select “Perimeter”). Check the value of perimeter. Evaluate error as

$$\delta_P^{IJ} = \frac{P_{IJ} - P_{th}}{P_{th}}$$

Result with MorphoLibJ

Compare the error with that obtained with MorphoLibJ. Use the Plugin “Plugins ▹ MorphoLibJ ▹ Analyze ▹ Analyze Regions...”, and make sure the option for perimeter is checked. Evaluate error as

$$\delta_P^{MLJ} = \frac{P_{MLJ} - P_{th}}{P_{th}}$$

One should obtain a smaller relative error with MorphoLibJ. Note that relative error depends on shape and on relative position and orientation of shape with respect to discretization grid.

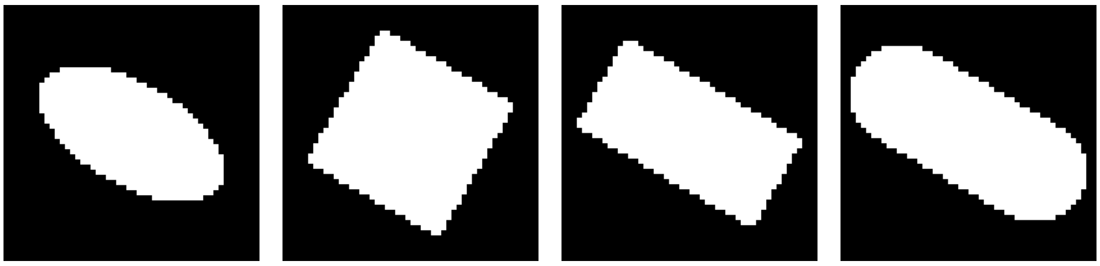
Shape	P_{th}	ImageJ		MorphoLibJ	
		P	δ_P	P	δ_P
disk R=20	125.66	130.1	3.55%	125.0	-0.5%
disk R=50	314.15	329.7	4.95%	314.2	0.03%

Table 1: Sample results that can be obtained for perimeter of a disk.

Comparisons between both methods on a sample image are reported in Table 1. Note that the results depends on the position of the center of the shape.

1.2.2 Other shapes

You can repeat the workflow with other shapes: square, ellipses, rotated rectangles... It can be interesting to let vary the different shape parameters, as well as the orientation.



(a) Ellipse, $a = 20, b = 10$ **(b)** square, $s = 30$ **(c)** box, $s_1 = 40, s_2 = 20$ **(d)** capsule, $l = 30, t = 20$

Figure 2: *Examples of discrete shapes.*

Note that the computation of ellipse perimeter requires numerical integration. Expected perimeter values for disks and ellipses with various sizes are given in table 2.

Radius	Perimeter
10	62.83
20	125.66
30	188.49
40	251.33
50	314.15

(a) Disks

Radius1	Radius 2	Perimeter
20	10	96.88
30	10	133.65
40	20	193.77
50	10	210.10
50	20	230.13
50	30	283.62

(b) Ellipses

Table 2: *Expected perimeter values for disks and ellipses with various radiusses.*

In order to facilitate repetitive runs, a macro can be used. Run “Plugins New Macro” to open the macro editor, and choose the language of your choice. The following script[3] provides a macro that can be used to run the analysis on an ellipse by randomizing position and orientation.

```
// shape parameters
r1 = 40;
r2 = 20;
perimTh = 193.77;

// check with fixed orientation
imgEllipse = "ellipse40x20";
newImage(imgEllipse, "8-bit_black", 200, 200, 1);
centerX = 100 + random;
centerY = 100 + random;
```

[3] https://github.com/dlegland/image-analysis-practical-2023/blob/main/practical/validation/macros/ellipse_perimeter.ijm

```
theta = random * 180;
run("Fill␣Ellipse", "center␣x=&centerX␣center␣y=&centerY␣major=&r1␣minor=&r2␣orientation=&theta␣fill
     =255");

// Run analysis with ImageJ
setAutoThreshold("Default␣dark");
run("Analyze␣Particles...", "display␣clear");

// retrieve perimeter and compute error
perimIJ = Table.get("Perim.", 0);
errorIJ = 100 * (perimIJ - perimTh) / perimTh;
print("Relative␣Error␣using␣ImageJ:␣" + errorIJ + "%");

// Run analysis with MorphoLibJ
run("Analyze␣Regions", "area␣perimeter␣equivalent␣ellipse");
selectWindow(imgEllipse + "-Morphometry");

// retrieve perimeter and compute error
perimMLJ = Table.get("Perimeter", 0);
errorMLJ = 100 * (perimMLJ - perimTh) / perimTh;
print("Relative␣Error␣using␣MorphoLibJ:␣" + errorMLJ + "%");
```

1.3 Average error

It may be interesting to check the variability of the measures around the theoretical values or/and the average values provided by the plugins. For this, we can generate a collection of images representing shapes with same size parameters, but different positions (and orientations if relevant).

The best is to use macro programming. An example is given below[4], for rotated ellipses with semi-axis lengths equal to 40 and 20.

```
// shape parameters
r1 = 40;
r2 = 20;
perimTh = 193.77;

// the image name
imgName = "ellipse40x20";

// number of repetitions
nRepets = 10;

// initialize array
perimIJArray = newArray(nRepets);
perimMLJArray = newArray(nRepets);

run("Clear␣Results");

for (i=0; i < nRepets; i++) {
    // check with fixed orientation
    newImage(imgName, "8-bit␣black", 200, 200, 1);
    centerX = 100 + random;
    centerY = 100 + random;
    ori = random * 180;
```

[4] https://github.com/dlegland/image-analysis-practical-2023/blob/main/practical/validation/macros/ellipse_perimeter_variability.ijm

```
    run("Fill␣Ellipse", "center␣x=&centerX␣center␣y=&centerY␣major=&r1␣minor=&r2␣orientation=&ori␣fill
        =255");

    // Run analysis with ImageJ
    setAutoThreshold("Default␣dark");
    run("Analyze␣Particles...", "display");

    // retrieve perimeter and compute error
    selectWindow("Results");
    perimIJ = Table.get("Perim.", i);
    print(perimIJ);
    perimIJArray[i] = perimIJ;
    errorIJ = 100 * (perimIJ - perimTh) / perimTh;
    print("Relative␣Error␣using␣ImageJ:␣" + errorIJ + "%");

    // Run analysis with MorphoLibJ
    run("Analyze␣Regions", "area␣perimeter");
    selectWindow(imgName + "-Morphometry");

    // retrieve perimeter and compute error
    perimMLJ = Table.get("Perimeter", 0);
    perimMLJArray[i] = perimMLJ;
    errorMLJ = 100 * (perimMLJ - perimTh) / perimTh;
    print("Relative␣Error␣using␣MorphoLibJ:␣" + errorMLJ + "%");
    // clean up
    close(imgName);
}

Array.getStatistics(perimIJArray, min, max, mean, std);
print("ImageJ␣stats");
print("␣␣␣min:␣"+min);
print("␣␣␣max:␣"+max);
print("␣␣␣mean:␣"+mean);
print("␣␣␣std␣dev:␣"+std);

Array.getStatistics(perimMLJArray, min, max, mean, std);
print("MorphoLibJ␣stats");
print("␣␣␣min:␣"+min);
print("␣␣␣max:␣"+max);
print("␣␣␣mean:␣"+mean);
print("␣␣␣std␣dev:␣"+std);
```

1.4 Impact of orientation

Choose an oriented box, with fixed side lengths, and generate discrete versions of the shape with various orientations.

The best is to create a macro. An example of expected result (for a rotated rectangle / oriented box with side length 150 and 50) is given in Figure 3). The corresponding macro can be found online[5].

```
// shape parameters
side1 = 150;
side2 = 50;
perimTh = 400;
centerX = 100 + random;
```

[5] https://github.com/dlegland/image-analysis-practical-2023/blob/main/practical/validation/macros/rectangle_perimeter_byTheta.ijm

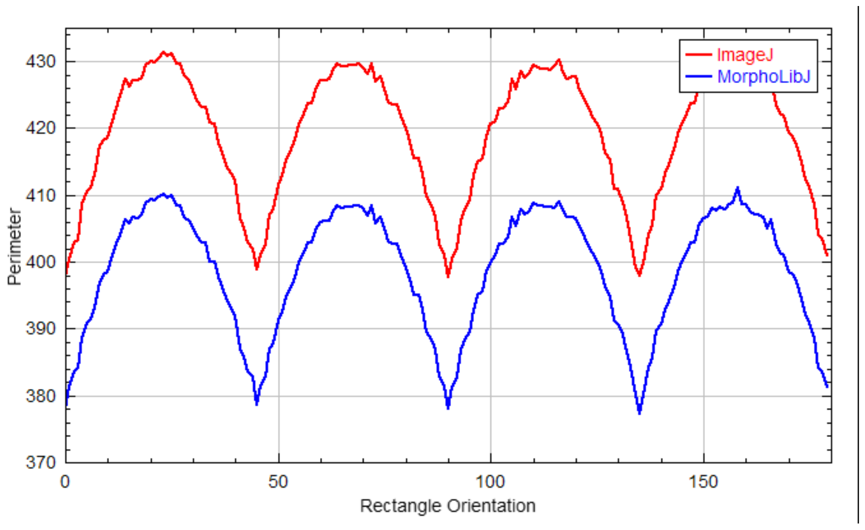


Figure 3: *Impact of orientation for computing perimeter of a rotated rectangle*

```
centerY = 100 + random;
// the image name
imgName = "ellipse40x20";
// number of repetitions
nTheta = 180;
// initialize arrays
thetaArray = newArray(nTheta);
perimIJArray = newArray(nTheta);
errorIJArray = newArray(nTheta);
perimMLJArray = newArray(nTheta);
errorMLJArray = newArray(nTheta);
run("Clear␣Results");

for (i=0; i < nTheta; i++) {
    // check with fixed orientation
    newImage(imgName, "8-bit␣black", 200, 200, 1);
    theta = i;
    run("Fill␣Oriented␣Box", "center␣x=&centerX␣center␣y=&centerY␣box␣length=&side1␣box␣width=&side2␣
         orientation=&theta␣fill=255");
    thetaArray[i] = theta;

    // Run analysis with ImageJ
    setAutoThreshold("Default␣dark");
    run("Analyze␣Particles...", "display");

    // retrieve perimeter and compute error
    selectWindow("Results");
    perimIJ = Table.get("Perim.", i);
    print(perimIJ);
    perimIJArray[i] = perimIJ;
    errorIJ = 100 * (perimIJ - perimTh) / perimTh;
    errorIJArray[i] = errorIJ;

    // Run analysis with MorphoLibJ
    run("Analyze␣Regions", "area␣perimeter");
    selectWindow(imgName + "-Morphometry");

    // retrieve perimeter and compute error
    perimMLJ = Table.get("Perimeter", 0);
    perimMLJArray[i] = perimMLJ;
    errorMLJ = 100 * (perimMLJ - perimTh) / perimTh;
    errorMLJArray[i] = errorMLJ;
    close(imgName);
}

Plot.create("Variations_of_Perimeter", "Rectangle_Orientation", "Perimeter");
Plot.setLimits(0, 180, 370, 435);
Plot.setLineWidth(2);
Plot.setColor("red");
Plot.add("line", thetaArray, perimIJArray);
Plot.setColor("blue");
Plot.setLineWidth(2);
Plot.add("line", thetaArray, perimMLJArray);
Plot.setLegend("ImageJ\tMorphoLibJ", "top-right");
```

1.5 Extension to 3D

The same workflow can be applied to 3D images as well. The “Digital Shapes” library provides discretization of various 3D shape models: ellipsoids, cuboids, cylinders, capsules... The computation may be somewhat slower however.

2 Image texture analysis

This practical session proposes to investigate image texture analysis using gray level granulometry with mathematical morphology filters.

2.1 Requirements

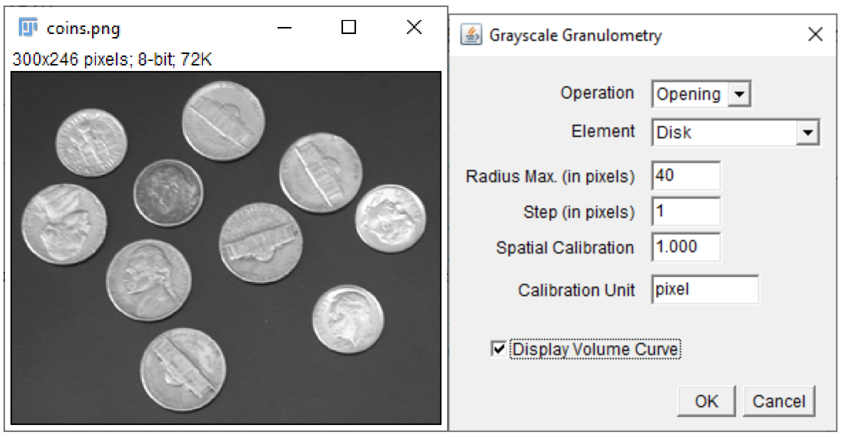
The practical will use the Fiji software. The following plugins are necessary:

- MorphoLibJ (installed by default)
- ijGranulometry[6] a grayscale granulometry plugin for ImageJ based on MorphoLibJ

2.2 Basic image

This sections presents the basic of granulometry.

We will start with the image "coins.png" (Fig. 4). This image presents coins with various diameters.



coins.png
300x246 pixels; 8-bit; 72K

Grayscale Granulometry

Operation: Opening
Element: Disk
Radius Max. (in pixels): 40
Step (in pixels): 1
Spatial Calibration: 1.000
Calibration Unit: pixel
[✓] Display Volume Curve

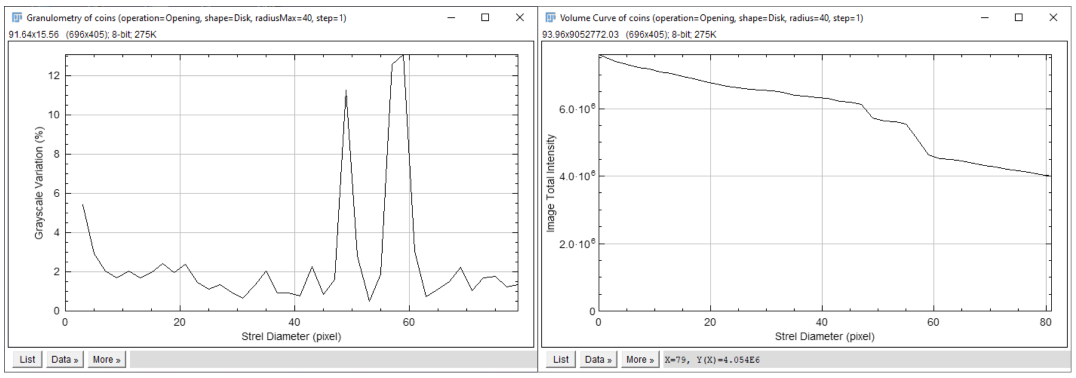
Figure 4: *Test image for granulometry*

Run the plugin "Granulometry ▹ Granulomety by Radius...", fill in the dialog with appropriate settings:

- as the coins are bright over a dark background, choose "opening" operation
- use disk structuring element to match the size of the coins
- choose a max radius larger than that of the coins, say 40 pixels
- for large images, using a step larger than 1 reduces computation time.
- for this image, leave calibration as is.

[6]`https://github.com/ijtools/ijGranulometry`

The resulting granulometry curve is presented on Fig. 5. The abscissa presents the diameter of the structuring element (computer as $2 \cdot \text{radius} + 1$). Two peaks can be noticed around 50 and 60 pixels. You may check that this corresponds to the diameter of the coins by using the “Plot profile” tool.


Granulometry of coins (operation=Opening, shape=Disk, radiusMax=40, step=1)
91.64x15.56 (696x405); 8-bit; 275K

Volume Curve of coins (operation=Opening, shape=Disk, radius=40, step=1)
93.96x9052772.03 (696x405); 8-bit; 275K

X=79, Y(X)=4.054E6

Figure 5: *Result of granulometry Plugin on coins image.*

If the “Display Volume Curve” option was selected, the sum of gray levels for each opening result is also displayed. Two “shoulders” may be noticed, that correspond to the diameters when large regions disappear with opening.

2.3 Cellular morphology of plant tissue

We will now investigate cellular morphology of plant tissue based on cross section of maize stems observed with macroscopy imaging. Open one of the images in the “maize” directory (Fig. 6). We can observe many polygonal cells, as well as bright structures (vascular bundles).

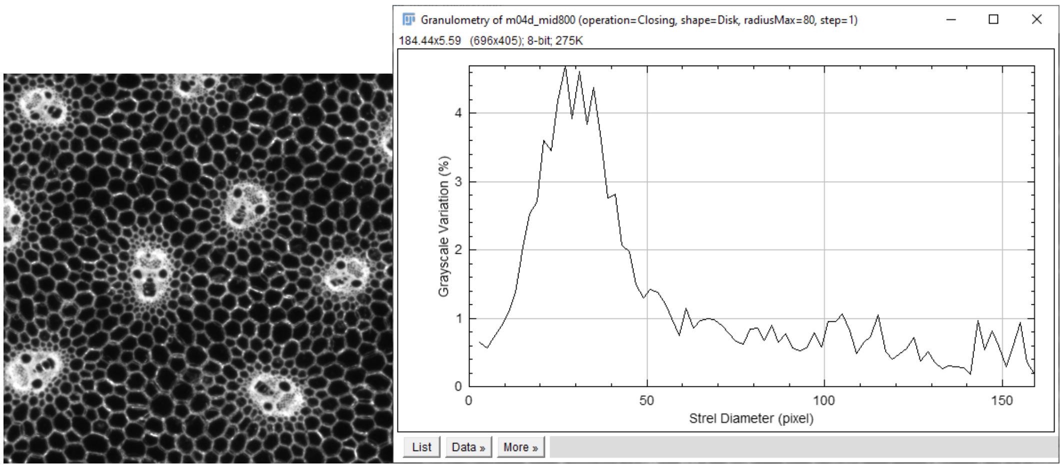

Granulometry of m04d_mid800 (operation=Closing, shape=Disk, radiusMax=80, step=1)
184.44x5.59 (696x405); 8-bit; 275K

Figure 6: *Application of gray-scale granulometry on an image of plant tissues.*

We run the “Granulomety by Radius...” plugin as follow:

- as cells are dark and separated by bright cell walls, we will use the “closing” operation
- use disk structuring element
- choose a max radius larger than that of the cells, say 80 pixels
- use a step equal to 1
- leave calibration as is

The result of granulometry presents a large peak around 30 pixels, corresponding to the typical diameter of cells.

2.4 Anisotropic image

Granulometry may also be used to identify preferred orientations within images. Let us open the image “f7a1.tif” image in the tomato directory. The image represents a slice of tomato pericarp, obtained from macroscopy imaging.

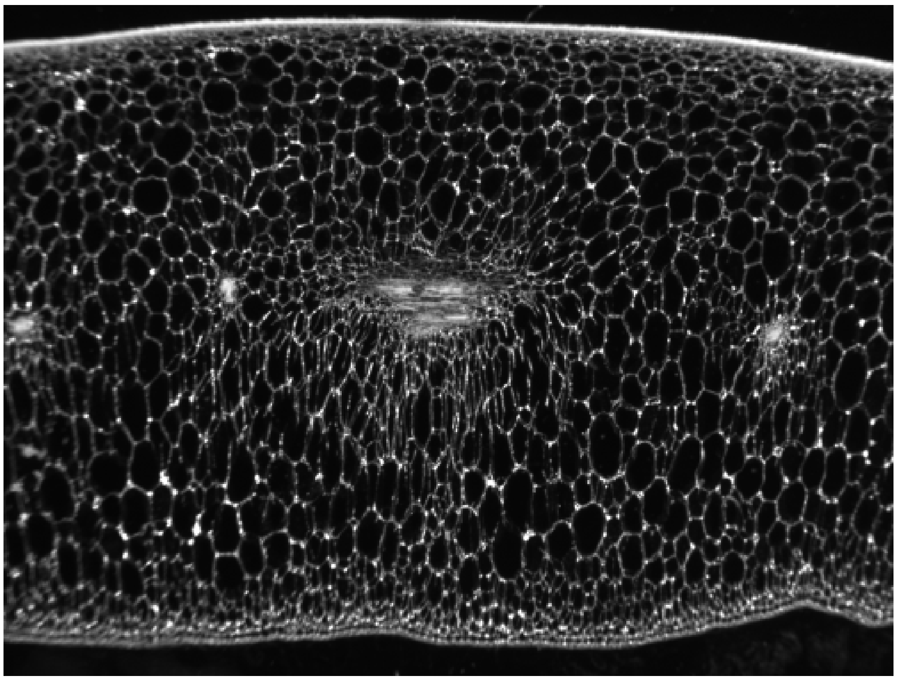


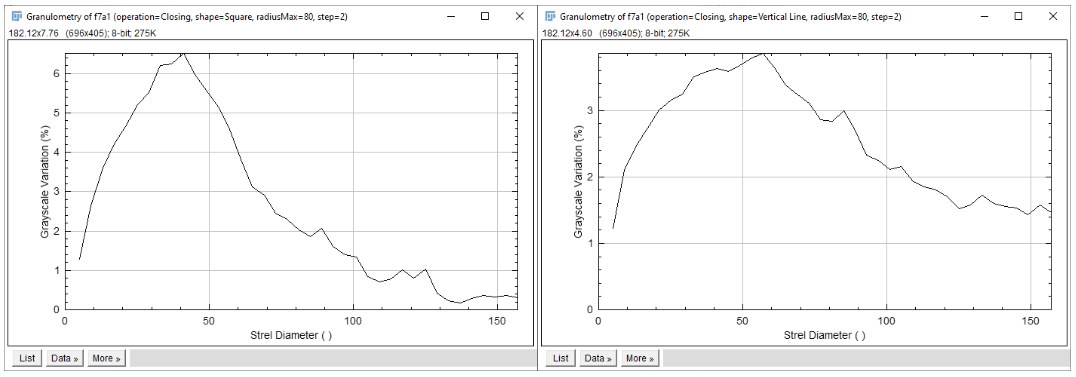
Figure 7: *Sample image of tomato slice.*

Let us apply granulometry, and compare the results obtained when changing the shape of the structuring elements:

- use the “Closing” operation
- repeat the process with each one of the following shapes: “square”, “horizontal line”, and “vertical line”.

- choose a max radius "larger enough", say 80 or 100 pixels
- use a step equal to 1 or larger to reduce computation time.
- leave calibration as is

The three resulting curves have different shapes. The curve obtained with a square or an horizontal line presents a thinner distribution around 40 pixels. This corresponds to variation around the typical thickness of cells. Using a vertical linear structuring elements results in a thicker bulb, and an shift to the right. This may be interpreted as a larger variability in cell sizes in the vertical direction, and as a larger typical size in the vertical direction.


Granulometry of f7a1 (operation=Closing, shape=Square, radiusMax=80, step=2)
182.12x7.76 (696x405); 8-bit; 275K

Granulometry of f7a1 (operation=Closing, shape=Vertical Line, radiusMax=80, step=2)
182.12x4.60 (696x405); 8-bit; 275K

Figure 8: *Granulometry curves on image of tomato tissue using different shapes of structuring elements.*

2.5 Collection of images

When working with a large number of images, it may be tedious to repeat the process of running the granulometry analysis. One possibility is to run a macro. Another possibility is to run the "batch Granulometry" plugin. The plugins allows to select a directory, the granulometry analysis options, and to perform the same analysis on the whole set of images within the directory.