A Tutorial For Measuring Edge Tensions

Thomas Portet
email: thomas.portet@gmail.com

June 11, 2009

Contents

This document provides instructions for applying the edge tension measurement method I developped during my stay in Potsdam between April and June 2009. It describes the different processes that have to be applied, and how to use the programs performing these tasks. Feel free to contact me at the email address above if any help or additional information is required.

The document is organised as follows. I first present briefly the theory on which this measurement method relies in Section 1, along with our specific experimental technique. Next I explain in Section 2 what kind of images can be processed. Section 3 describes the directories organisation, which must be respected when processing images for membrane detection (cf Section 4) and measuring pore sizes (cf Section 5). The last step, edge tension extraction, is finally described in Section 6.

I stress that this is just an instruction manual you are reading, not a scientific paper. The first section is then lacking many references to other peoples' work, and detailed explanations of the presented theory. However, I think that it is sufficient to understand the principle of this method and to successfully apply it.

1 Procedure overview

The measurement method of the edge tension γ presented here is based on the theory introduced in [Brochard-Wyart et al.(2000)]. The authors studied theoretically the full dynamics of growth and closure of a macroscopic pore of radius r in a giant vesicle of radius R. They describe the life of a transient pore as follows: a growth period up to a maximal radius, a slow closure limited by the leakout of the internal medium, and a fast closure when leakout becomes negligible. During this stage of slow closure which represents the majority of the pore lifetime, the plot of $R^2 \ln(r)$ versus time t[1] should be linear with a slope $-2\gamma/(3\pi\eta_0)$, η_0 denoting the viscosity of the bulk solution. The determination of the edge tension is then very simple. One just needs to fit $R^2 \ln(r)$ with a function of the form $y = at + b$, which gives one value for a and one value for b, and γ is obtained with the relation $\gamma = -(3/2)\pi\eta_0 a$.

This theory was later used in [Karatekin et al.(2003)] to measure edge tensions of DOPC vesicles, either pure, containing cholesterol, or some surfactant molecules. The authors used visible light illumination to stretch the vesicles until the sudden opening of a large pore, whose closure was monitored with fluorescence microscopy. This was made possible by using viscous solutions containing 66 % volume glycerol in order to sufficiently slow down the inner fluid leakout, and thus the dynamics of the pore closure.

Our experimental method differs from that one in two major points. First, the pore is opened by the application of a DC electric pulse of 5 ms duration and 200-800 kV/cm amplitude. It has been reported in [Portet et al.(2009)] that such pulses can create a macropore on the pole of the vesicle facing the cathode (the negative electrode), which is crucial for this application. The second difference lies in the use of ultrafast phase contrast imaging. Thanks to this technique, we do not have to use glycerol[2]. Phase contrast imaging may not be the very best method to distinguish between a large pore and many smaller ones, but former fluorescence observations had already established the presence of a single macropore at the cathode-facing side.

The first part of Brochard's theory describing the pore opening does not apply to our system, because the underlying physics of the mechanisms involved is not the same. However, once the pore

[1] It is assumed that only the pore radius r changes with time, not the vesicle radius R.

[2] In our analysis, we thus take $\eta_0 = 10^{-3}$ $Pa.s$ because we work with aqueous solutions.

is open and the surface tension has relaxed, the equations governing the closure dynamics are the same and we can use this theory to infer the edge tension γ from pore size measurements. Compared to their experimental curves, ours sometimes seem to indicate that the linear decrease regime appears later, after some time during which the pore radius seems quite stable.

2 Image acquisition and storage

Images must be transfered from the camera in tiff format. Acquisition must be performed with a linear timescale. Images relative to an experiment (*i.e.* to the application of *one* pulse on *one* vesicle) must be saved as an image stack in tiff fomat, in the directory raw (cf. next section). The name of this file must start with a letter. **The user must ensure that the cathode-facing hemisphere of the vesicle is situated on the *right-hand* side of the images.** Converting images to a stack and rotating them if necessary can be performed with ImageJ.

3 Directories organisation

Programs used for pore size measurements described in Section 5 require a specific organisation of the directories, which is illustrated in Figure 1. The root directory, whose name can be changed, is called demo. It contains three subdirectories[3]: preselection, processing and raw. Directory raw contains stacks of raw images acquired with the microscope, and directory preselection contains stacks cropped around the vesicle of interest (see Section 4 for further explanations). The last directory, processing, is composed of three directories: binaries, code, and results. As their names indicate, directories results and code contain results files and code files respectively. Processed images resulting from the procedure described in the next section will be saved in the directory binaries, which will also contain the files `info.xls` and `infos.txt`.

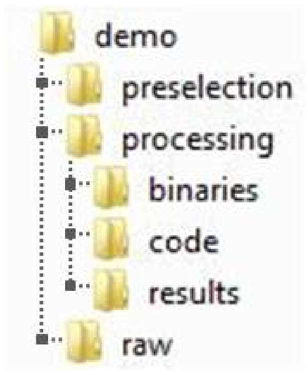


Figure 1: Directories arborescence.

4 Image processing with ImageJ for membrane detection

The procedure starts with a raw file located in the folder raw, let's say `boston_1.tif` for example (see Figure 2A). You first have to open this stack with ImageJ, define a rectangular selection enclosing the vesicle, and crop the image (shortcut **shift + X**). The result of this operation is shown in Figure 2B. You have to make sure that for each slice of the stack, the center of the image lies inside the vesicle, and that the membrane is not porated neither above nor below the center of the image. Save this new stack (with the same name) in the directory preselection.

As a first step in this membrane detection stage, perform an image background correction by using the background subtraction rolling-ball algorithm [Sternberg(1983)], with ball radius set to 50 pixels[4] (**Process, Substract Background, Sliding Paraboloid, 50 pixels**). Then locate the membrane

[3]It also contains a file, `macros_processing.txt`, whose use will be described in Section 4.
[4]A ball radius of 50 pixels gave good results with my images, other values can also be tried if problems.

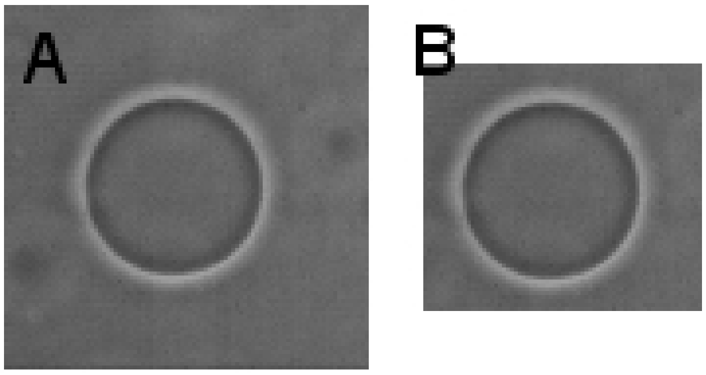


Figure 2: A: Raw image. B: Cropped image.

position by using a common Sobel edge detector to highlight sharp changes in intensity (**Process, Find Edges**). The last step is the image binarization (**Process, Binary, Make Binary, Calculate Threshold for each Image, Black Background**) where the threshold value is calculated using the classical Isodata algorithm [Ridler and Calvard(1978)]. Of course all these three operations should be performed on the entire stack. Figure 3 show some images before and after this procedure (the membrane is represented by the inner circle). I wrote a macro for performing automatically these three procedures in the file `macros_processing.txt`, located in the root directory (demo). For using this macro, first install it in ImageJ (**Plugin, Macros, Install,** choose file `macros_processing.txt`), then simply use the shortcut **p**. (This macro named **process_image** will also appear in the menu **Plugins, Macros**.)

In some cases, we have to remove residual white pixels appearing inside the vesicle, because they could lead to errors in the pore radius measurements. Such pixels are problematic if they are located on the vertical line passing through the center of the image and if they are not connected to the membrane. The small line in the top part of the vesicle in Figure 4A must either be removed, or connected to the membrane. This second solution had been applied to obtain Figure 4B. I just drawed a white rectangle which fills the black part between these residuals pixels and the membrane. This solution is often easier to apply than the one consisting in removing the residual pixels, and can be easily applied to the whole stack. The image is now ready for the pore measurement with Matlab, and must be saved (again with the same name) in the directory binaries. Do not worry if the membrane looks strange on the left-hand side of the picture, only the right-hand side is used for pore size measurements.

It is now time to enter some information about the image in the file `infos.xls` located in the folder binaries. These different informations can be seen in Figure 5. Each line represents an experiment and must contain the image name without the `.tif` extension, the number of slices of the stack, the time between two consecutive acquisitions in μs (50 for an acquisition at 20000 fps, 100 for an acquisition at 10000 fps, 200 for an acquisition at 5000 fps...), the pixel size in μm, the index of the vesicle (an integer identifying the analyzed vesicle), the initial radius of the vesicle in μm, and the two times in ms delimiting the linear part of $R^2 \ln(r)$. At this moment, we do not yet know these times, so we can enter any value (0 and 0 for example), but we *must* enter some values for these two numbers. Matlab will not directly load this file, so we have to copy this array (except the first column with file names) as shown by the red rectangle in Figure 5, into a simple text file named `infos.txt` and located in the same directory, binaries. This file should look line the one shown in Figure 6.

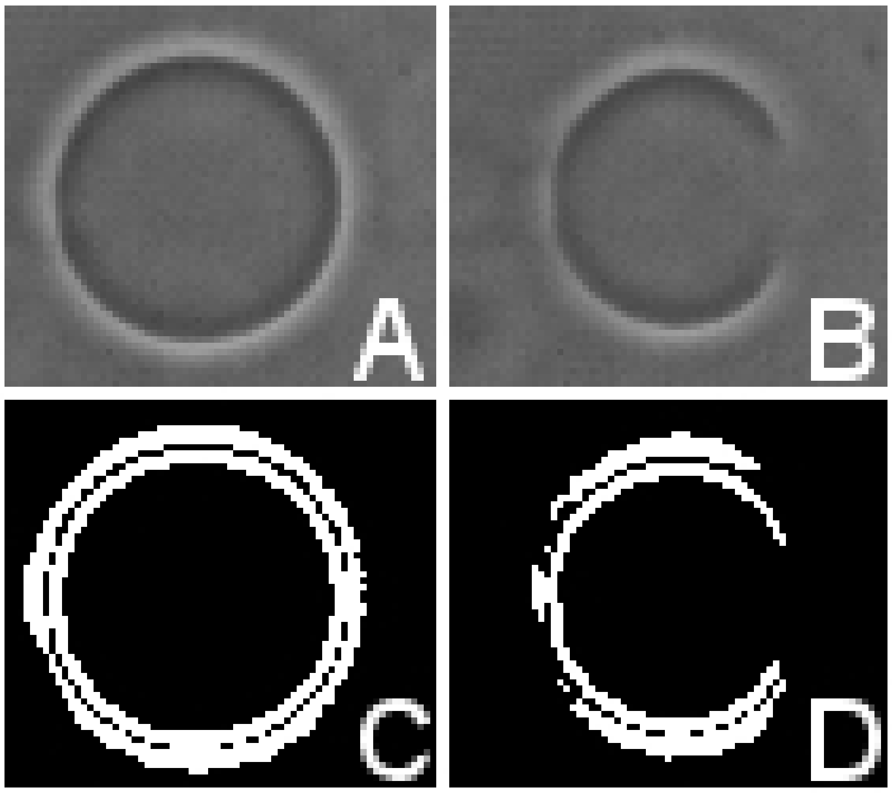


Figure 3: A,B: Images before processing. C,D: Images after processing. (The inner contour represents the membrane.)

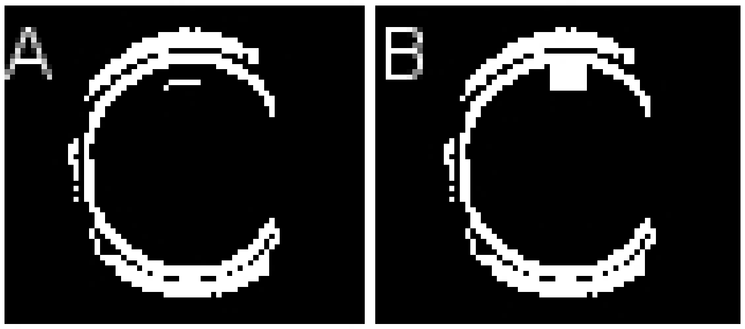
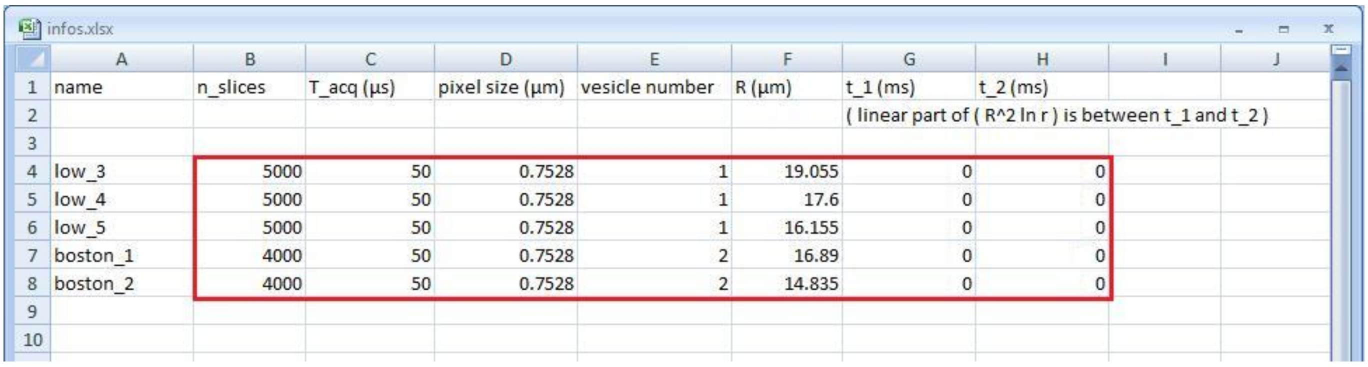


Figure 4: A: Bad image. B: Good image.


infos.xlsx

	A	B	C	D	E	F	G	H	I	J
1	name	n_slices	T_acq (μs)	pixel size (μm)	vesicle number	R (μm)	t_1 (ms)	t_2 (ms)		
2							(linear part of (R^2 ln r) is between t_1 and t_2)			
3										
4	low_3	5000	50	0.7528	1	19.055	0	0		
5	low_4	5000	50	0.7528	1	17.6	0	0		
6	low_5	5000	50	0.7528	1	16.155	0	0		
7	boston_1	4000	50	0.7528	2	16.89	0	0		
8	boston_2	4000	50	0.7528	2	14.835	0	0		
9										
10										

Figure 5: Example of the contents of the file `infos.xls`.

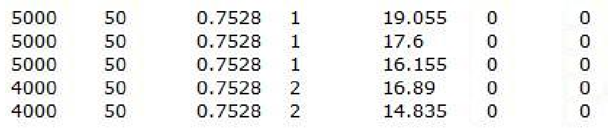

```
5000    50    0.7528   1    19.055   0    0
5000    50    0.7528   1    17.6     0    0
5000    50    0.7528   1    16.155   0    0
4000    50    0.7528   2    16.89    0    0
4000    50    0.7528   2    14.835   0    0
```

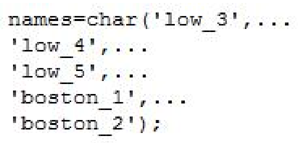
Figure 6: Example of the contents of the file `infos.txt`.

5 Pore measurements with Matlab

Once the tasks described in Section 4 have been carried out for all vesicles, we can launch Matlab for measuring pore sizes. First, define the directory processing as the current directory. Then, add the code directory to the current path by typing the line `path('.\code',path);`. Before launching the fonction `trous` which computes the pore radiuses, open the file `chargement.m`, and update the variable *names* containing the names of the image files, using a syntax similar to the one shown in Figure 7. Save this file.

Pore size measurements can then be performed with the function `trous`, which takes a single argument as input, named *njump*. This argument must be a non zero integer, defining which slices have to be taken into account for the pore radius measurements. Only slices with numbers 1, $njump + 1$, $2 \times njump + 1$, $3 \times njump + 1$, $4 \times njump + 1$ (...) will be considered. For using all slices, *njump* must be taken equal to 1. This feature is useful to evaluate how long it will take to make all measurements. The computation time is roughly proportional to the number of slices to analyze, so if the algorithm takes 50 seconds to run with $njump = 100$, the computation will last 5000 seconds if you consider all slices with $njump = 1$, because the total slice number is a hundred times greater. I recommand to first perform one calculation with a high *njump* (500 or 1000, by typing `trous(500);` or `trous(1000);`) to make sure that the program produces no error (it would be the case if the file `infos.txt` is not filled properly, or if there is a discrepancy between the actual names of the images and the ones entered in the *names* variable in `chargement.m` for example). After a succesful execution, Matlab displays the elapsed time in seconds, which can be used to evaluate the time required for a computation involving more slices (lower *njump*). The algorithm I wrote is far to be the most efficient and could of course be improved to run faster, but I did not have enough time to do this. However, when I analyzed more than 50 stacks in a row, each containing on average several hundreds of slices, the results were always obtained whithin less than 12 hours, which is still reasonable and much faster than if one had to do these measurements by hand !

The results are saved in the results directory as text files named `njump_xxx_yyyyyy.txt`, where `xxx` is the value of *njump*, and `yyyyyy` is the name of the corresponding image. Each text file is an array composed of 6 columns. The first column contains the radiuses of the pore in μm, and the last column the corresponding slice numbers. Columns 2 to 5 contain the pore edges coordinates (two couples of two integers) which can be of use if one wants to locate the pore on the cropped image located in the folder preselection.

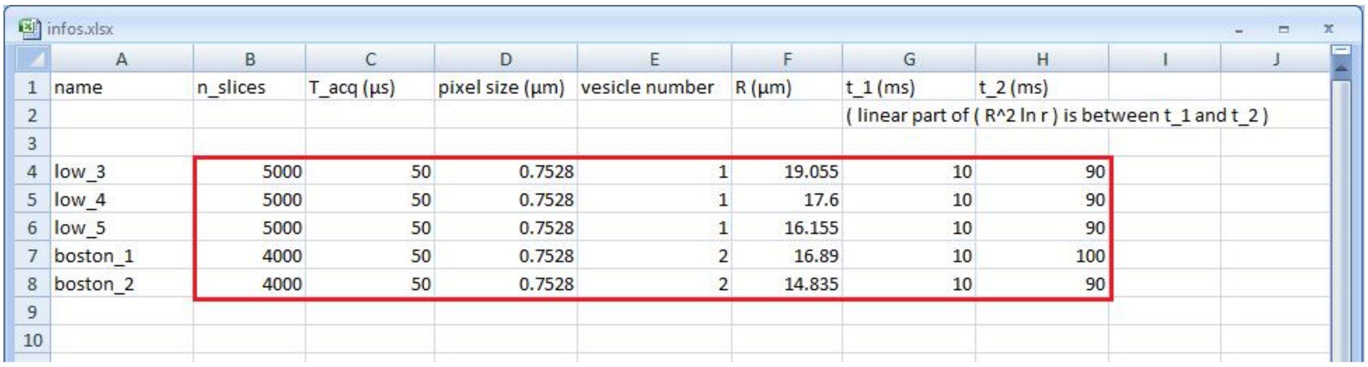


```
names=char('low_3',...
'low_4',...
'low_5',...
'boston_1',...
'boston_2');
```

Figure 7: Example of the syntax for updating the variable *names* in the file `chargement.m`.

6 Extraction of the edge tension

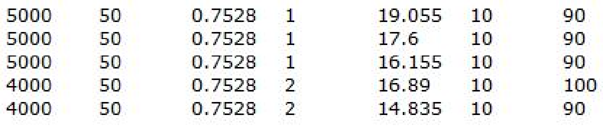
Now that we have measured the pore radiuses, we just need to find where is the linear part of the function $R^2 \ln(r)$, and to compute its slope a to finally obtain the edge tension γ, as described in Section 1. We can use the Matlab function `affiche(njump);` to plot all the functions $R^2 \ln(r)$. We have to locate the decreasing linear part, and enter its time boundaries t_1 and t_2 in the files `infos.xls` and `infos.txt` as shown in Figures 8 and 9. After that, we just have to launch the function `pentes`, whose syntax is `[gamma]=pentes(njump)`. It returns the measured edge tensions in the variable gamma, and also displays its mean value and its standard deviation.



infos.xlsx

	A	B	C	D	E	F	G	H	I	J
1	name	n_slices	T_acq (µs)	pixel size (µm)	vesicle number	R (µm)	t_1 (ms)	t_2 (ms)		
2							(linear part of (R^2 ln r) is between t_1 and t_2)			
3										
4	low_3	5000	50	0.7528	1	19.055	10	90		
5	low_4	5000	50	0.7528	1	17.6	10	90		
6	low_5	5000	50	0.7528	1	16.155	10	90		
7	boston_1	4000	50	0.7528	2	16.89	10	100		
8	boston_2	4000	50	0.7528	2	14.835	10	90		
9										
10										

Figure 8: Example of the contents of the file `infos.xls` after having added time limits of the linear part of $R^2 \ln(r)$.



```
5000    50    0.7528  1      19.055  10     90
5000    50    0.7528  1      17.6    10     90
5000    50    0.7528  1      16.155  10     90
4000    50    0.7528  2      16.89   10     100
4000    50    0.7528  2      14.835  10     90
```

Figure 9: Example of the contents of the file `infos.txt` after having added time limits of the linear part of $R^2 \ln(r)$.

References

[Karatekin et al.(2003)] Karatekin, E., O. Sandre, H. Guitouni, N. Borghi, P.-H. Puech, and F. Brochard-Wyart, 2003. Cascade of transient pores in giant vesicles: line tension and transport. *Biophys. J.* 84:1734–1749.

[Brochard-Wyart et al.(2000)] Brochard-Wyart, F., P. de Gennes, and O. Sandre, 2000. Transient pores in stretched vesicles: role of leak-out. *Physica A* 278:32–51.

[Portet et al.(2009)] Portet, T., F. Camps, J.-M. Escoffre, C. Favard, M.-P. Rols, and D. S. Dean, 2009. Visualization of membrane loss during the shrinkage of giant vesicles under electropulsation. *Biophys. J.* 96:4109–4121.

[Sternberg(1983)] Sternberg, S., 1983. Biomedical Image Processing. *IEEE Comput.* 16:22–34.

[Ridler and Calvard(1978)] Ridler, T. W., and S. Calvard, 1978. Picture Thresholding Using an Iterative Selection Method. *IEEE Trans. Syst. Man. Cybern.* 8:630–632.