Documentation for

Micropipette Aspiration

Author: Tanmay Pandey

25 July 2023

Contents

1 Getting Started

The code is written in Python programming language. It aims to calculate the desired parameters for the micropipette aspiration project using the profile plots and manual selection of required points.

1.1 Required Libraries

These are the required libraries we need to install.

1. CV2
2. Numpy
3. Pandas
4. Sklearn
5. Matplotlib
6. EasyGUI
7. Scipy
8. Sklearn
9. Skimage
10. Alive Progress

Apart from this, an ascii art library (art) is used to display ascii art in terminal.

1.2 The program

The very first step is to enter the require parameters for the calculation.

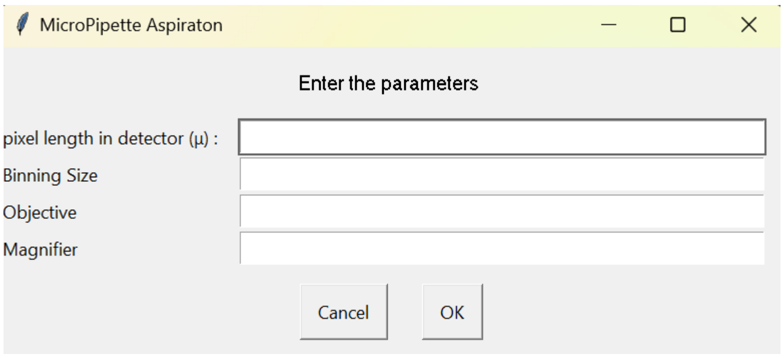

Figure 1.1. Screenshot of the popup enter box to enter the required parameters. We use *easygui.multenterbox()*.

1.2.1 Inputting images

In the very next step, we select the directory where our file is located. Once we select, the program reads the files one by one and store it into an array using *cv2.imread()* function. Since, cv2 reads images in BGR, we convert it into RGB, so that we can use matplotlib to do the further operations. We convert BGR to RGB by using *cv2.cvtColor()* function.

1.3 Background Subtraction

Once we have done the directory selection and colour space conversion, we proceed to background subtraction. We use mean subtraction method to reduce background. But since, it may result into artifacts in some of the images, we ask for if we have to do it or not. A popup using *easygui.ynbox()* is made to select an option.

2 Image Processing and Calculations

2.1 Image Pre-Processing

2.1.1 Rotating Images

We first rotate the image in a way such that we get the pipette start from the left (refer to figure 2.1.1).

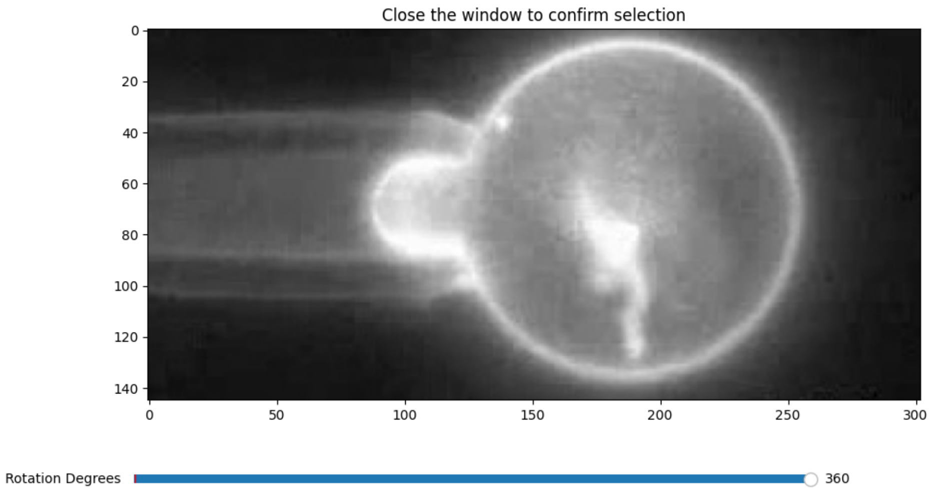

Figure 2.1. Rotating Image to get pipette start from left

Once we have done rotating the first image, we get the degrees of rotation and we apply to all images using *scipy.ndimage.rotate()* function.

2.1.2 Region of interest selection

Once we are done with rotation, we may encounter regions where there is no data, which is created due to rotation of images. So, we use *cv2.selectROI()* to get the coordinates of selected region of interest, and we apply the region of interest selection to all of the images.

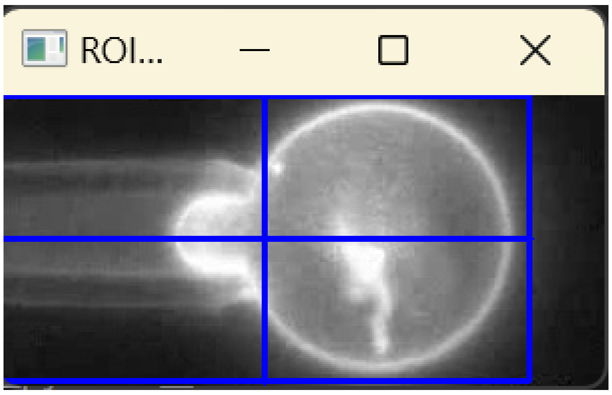

Figure 2.2. Selecting region of interest

Once it is done, we go to process the image and do the calculations.

2.2 Finding the radius of pipette

2.2.1 Asking User for points

We first ask the user to select 4 points, which will give us the coordinates of the area, by which we tend to find the pipette inner edges inside it. Say we get points $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$.

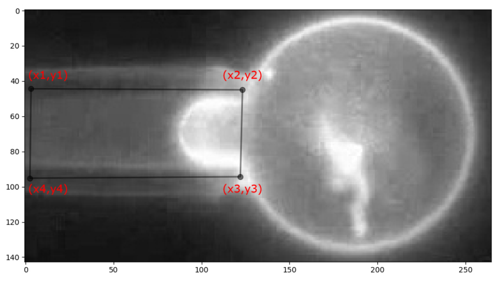

Figure 2.3. Selecting points for processing image

Since we are interested in column profile, so, we need to get a top position and a bottom position, to get a profile between those two points.

$$y_{top} = \frac{(y_1 + y_2)}{2}, y_{bottom} = \frac{(y_3 + y_4)}{2} \tag{2.1}$$

So, we restrict our profile between y_{top} and y_{bottom}.

Once, we get the average top and bottom y positions. We need to do the scan in the area of box only, so we need to restrict it with x-axis.

$$x_{first} = \frac{(x_1 + x_4)}{2}, x_{bottom} = \frac{(x_2 + x_3)}{2}$$

(2.2)

After getting the values for the boundary, we can now proceed for pipette radius calculations.

2.2.2 Calculating radius of pipette (r)

We first convert the images to gray-scale and then invert the image so that we get peaks at darker points and valley at brighter (we use $cv2.bitwisenot$). This way, we can identify the area between the outer and inner wall of pipette. We then select the first and last peak between the points, and do it for all lines in the selected area, and for all images. By this method, we get an approximate diameter of the radius and error associated, which is mean absolute error.

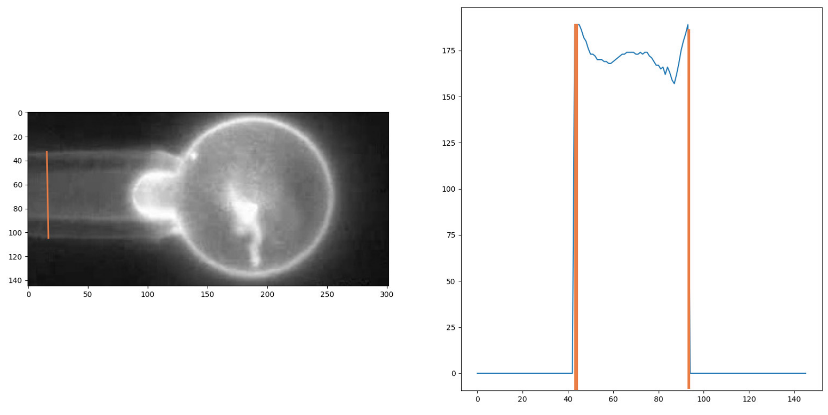

Figure 2.4. The program selects a column (marked in orange on image), and get the profile between the point. On the plot, the orange marked peaks are actually pixels with lowest intensity near inner wall in images. So, we calculate the distance between them and do this for all images.

We get a value of r and its error δr from the images.

2.3 Parameters extraction

Once we have the coordinates and value or r and δr, we can proceed with further calculations.

We ask the user to draw a line where the pipette tip could be or draw a rectangular box where it could be at center of it, so that we can get an approximate value for pipette tip.

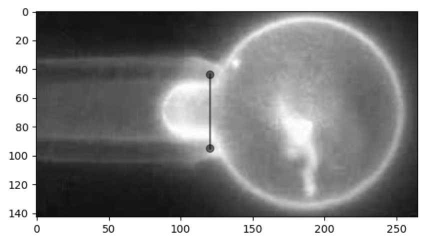

Figure 2.5. We select the line by selecting two points where pipette tip is highly probable to be found and let us denote the the top point as (x_5, y_5) and bottom point as (x_6, y_6) .

We then draw a middle axis passing derived from the points given in 2.2.1. The line starts at (x_{start}, y_{start}) and ends at (x_{end}, y_{end}), where,

$x_{start} = 0$, x_{end} =total number of columns.

And, since we need to fix the y-axis at the center of our selection so,

$y_{start}, y_{end} = \frac{y_{top}+y_{bottom}}{2}$ (from equation 2.1)

(***Please Note***: *The x and y in graph denotes the column number and row number respectively.*)

We then plot the profile along our line, and then inflection points, the rough positions are found by first applying *scipy.ndimage.gaussian_gradient_magnitude()* filter (calculates an absolute value of gradient of gaussian-presmoothed profile) and than using user input for overestimated feature positions. The left and left-most maxima, gives us the aspirated tip position (x_t) and the right-most gives us the guv edge position (x_v), while the pipette tip position is calculated by searching peaks in neighbour position from user inputted point (x_p).

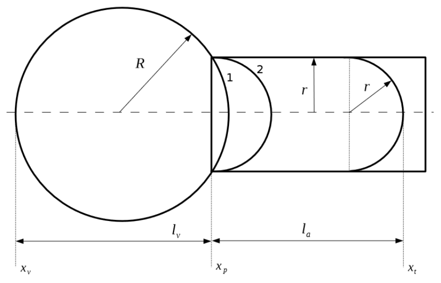

Figure 2.6. Sketch of a guv aspirated

We then calculate l_a and l_v as :
1. $l_a = |x_p - x_t|$, $\delta l_a = \delta x_p + \delta x_t$
2. $l_v = |x_p - x_v|$, $\delta l_v = \delta x_p + \delta x_v$

2.4 Calculations

The radius of outer vesicle is calculated as :

$R = \frac{(l_v^2 + r^2)}{2l_v}$

Corresponding error is then:

$\delta R = \sqrt{\frac{r^2}{l_v^2}(\delta r^2) + \frac{1}{4}(1 - \frac{r^2}{l_v^2})(\delta l_v^2)}$

The surface of inner part is:

$$S_{in} = \begin{cases} 2\pi r l_a & l_a \geq r \\ \pi(l_a^2 + r^2) & 2R - l_v \leq l_a \leq r \end{cases}$$

and its volume is :

$$V_{in} = \begin{cases} \pi r^2(l_a - \frac{r}{3}) & l_a \geq r \\ \frac{1}{6}\pi l_a(3r^2 + l_a^2) & 2R - l_v \leq l_a \leq r \end{cases}$$

with errors:

$$\delta S_{in} = \begin{cases} 2\pi\sqrt{(l_a^2)(\delta r)^2 + (r^2)(\delta l_a)^2} & l_a \geq r \\ 2\pi\sqrt{(l_a^2)(\delta l_a)^2 + (r^2)(\delta r)^2} & 2R - l_v \leq l_a \leq r \end{cases}$$

$$\delta V_{in} = \begin{cases} \pi r\sqrt{r^2(\delta l_a)^2 + (2l_a - r)^2(\delta r)^2} & l_a \geq r \\ \frac{\pi}{2}\sqrt{4r^2 l_a^2(\delta r)^2 + (r^2 + l_a^2)^2(\delta l_a)^2} & 2R - l_v \leq l_a \leq r \end{cases}$$

The surface of outer part of vesicle is:

$S_{out} = 2\pi R l_v = \pi(l_v^2 + r^2)$

and it's volume is:

$V_{out} = \frac{1}{6}\pi l_v(3r^2 + l_v^2)$

with it's errors as:

$\delta S_{out} = 2\pi\sqrt{l_v^2(\delta l_v)^2 + r^2(\delta r)^2}$
and
$\delta V_{out} = \frac{\pi}{2}\sqrt{4r^2 l_v^2(\delta r)^2 + (r^2 + l_v^2)^2(\delta l_v)^2}$

Now we can define,

$S = S_{in} + S_{out}$ and $\delta S = \delta S_{in} + \delta S_{out}$
$V = V_{in} + V_{out}$ and $\delta V = \delta V_{in} + \delta V_{out}$

3 Limitations and Scope

3.1 Limitation

The program has some limitations, some of them which can be reduced by some tweaks in the code.

1. It is tested only for Phase-Contrast Images.
2. No extra data is given to users.
3. Program is highly based upon user inputted points.

3.2 Scope

The program is based upon algorithms which can be tweaked to further add more datas or for different images. Since we are basing our code on python interpreter and language, so the computational time can be reduced by using *just-in-time* interpreter.