Click Intensity V4 python program user manual, by Ziqing Ye

Table of contents

1. Introduction (what the program does) -- 1

2. How to use the code (step by step instruction) ---------------------------------- 2

1. Introduction

The program is used to take pixel intensity data. Here’s a brief summary of what it does:

a. You open the program and the image that you want to analyze opens up.

b. You left click somewhere on the image

c. A circle with a customized radius is generated there. It has a number on it.

   a) While the program is running, you can change the display brightness, zoom in and out, as well as adjust the radius of your circle

d. The **average*** pixel intensity within that circle is calculated.

e. This intensity information is saved into an excel spreadsheet

f. You click somewhere else, and step b-e are repeated.

g. Once you are done, you press esc to close the image.

h. When you close it, the image with all those circles is saved to your folder for future reference， along with an excel spreadsheet that stores all of the pixel intensity data.

Some notable features:

1. Can easily adjust the **display brightness** for seeing dimmer signals without causing fluctuations in the recorded intensity data

2. Can freely **adjust the radius** of circular ROIs (region of interest) depending on the sample conditions

3. Can **zoom in and out** and recenter while making accurate ROI selections.

2. How to use the code

Part 1: things to do before running the code for the first time

1. Make sure you have python and OpenCV installed. Instructions can be found here: https://youtu.be/d3AT9EGp4iw

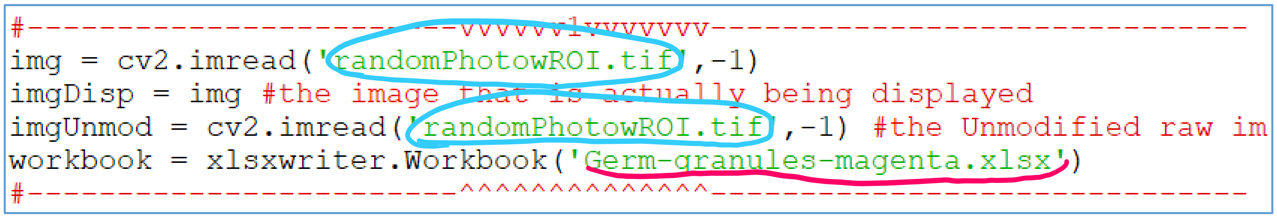
2. Install XlsxWriter by entering **pip install XlsxWriter** in Windows command prompt, just like how the Youtuber installed OpenCV in the tutorial linked above.

Part 2: running the code

Step 1: put the images that you want to analyze in one folder, and put the python code in that same folder with your images.

Step 2: Open the code using python IDLE: Open IDLE (the python ide) first (it should be in the start menu, or search for “IDLE” in the windows 10 “Type here to search” place), then within IDLE, go to File>>Open.

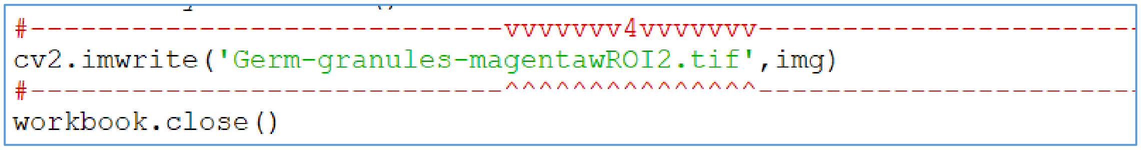
Step 3: Look for this section in the code (or ctrl+F and search for “v1v”), and change the contents inside ‘….’ (circled in blue) to match the exact file name of the image being analyzed:



```
#------------------------vvvvvvv1vvvvvvv--------------------------------
img = cv2.imread('randomPhotowROI.tif',-1)
imgDisp = img #the image that is actually being displayed
imgUnmod = cv2.imread('randomPhotowROI.tif',-1) #the Unmodified raw im
workbook = xlsxwriter.Workbook('Germ-granules-magenta.xlsx')
#------------------------^^^^^^^^^^^^^^--------------------------------
```

Step 4: Change the name of the output excel spreadsheet in the same section (underlined in pink above) so it reflects which image it corresponds to.

Step 5: Scroll to the very bottom of the code and name the output image file:



```
#----------------------------vvvvvvv4vvvvvvv-------------------------
cv2.imwrite('Germ-granules-magentawROI2.tif',img)
#----------------------------^^^^^^^^^^^^^^^-------------------------
workbook.close()
```

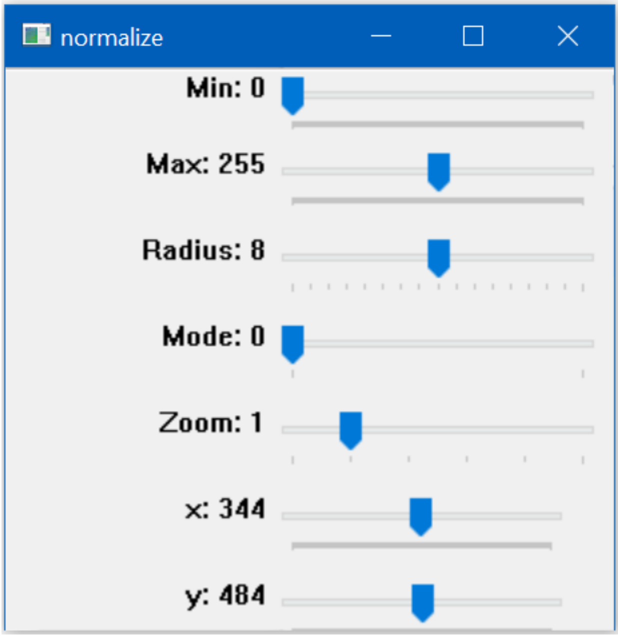
Step 6: Start running the program.

a) In the python IDLE, go to Run>>Run Module (or some equivalent operation in other python IDEs). An image and a window with three scrollable bars will pop up if it runs successfully.

b) Start left clicking on the image. The python shell (in another window) will also print out the coordinates and the calculated pixel intensity as you click so you can check if the intensity looks right or not.

c) The trackbars:

–



normalize

Min: 0

Max: 255

Radius: 8

Mode: 0

Zoom: 1

x: 344

y: 484

levels (brightness) → for display only, won't affect data

radius of circle

mode will be explained later

Zoom: zoom in & out

x, y } move the view when zoomed in

d) When finished, press esc to stop the program close the image.

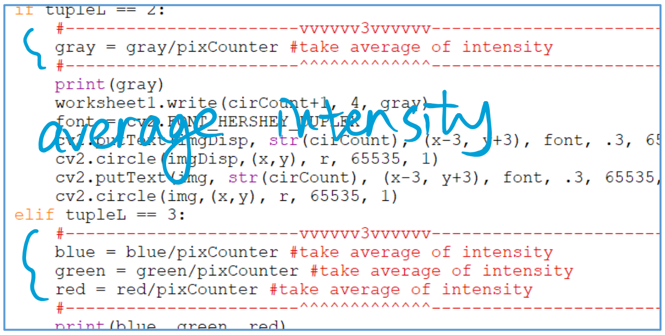
Step 8: You can check the produced excel file and the output image files in the same folder that the python code and the input image files are in.

Step 9: Repeat steps 3 through 7, remember to change the file names after each run. If the output file names are not changed after one round, running the code will overwrite the old file with that file name (so you'd lose the data from the previous round).

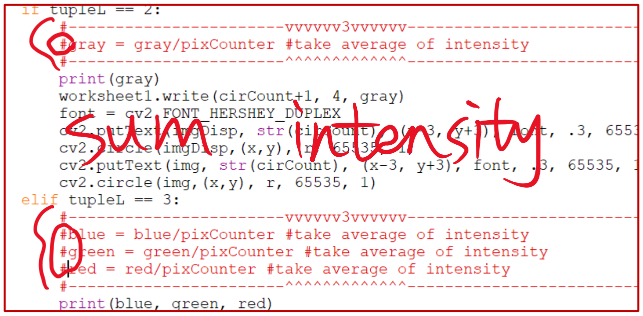
3. Advanced features:

A. What if you want to find out the sum intensity rather than the average intensity?

a) Put a "#" (a.k.a. commenting something out) before all the lines that are located in between something like #---------vvv**3**vvv------- and #--------^^^^^^-----. For example:



```
if tupleL == 2:
    #-----------------------vvvvvv3vvvvvv-----------------------
    gray = gray/pixCounter #take average of intensity
    #-----------------------^^^^^^^^^^^^^-----------------------
    print(gray)
    worksheet1.write(cirCount+1, 4, gray)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(imgDisp, str(cirCount), (x-3, y+3), font, .3, 65
    cv2.circle(imgDisp,(x,y), r, 65535, 1)
    cv2.putText(img, str(cirCount), (x-3, y+3), font, .3, 65535,
    cv2.circle(img,(x,y), r, 65535, 1)
elif tupleL == 3:
    #-----------------------vvvvvv3vvvvvv-----------------------
    blue = blue/pixCounter #take average of intensity
    green = green/pixCounter #take average of intensity
    red = red/pixCounter #take average of intensity
    #-----------------------^^^^^^^^^^^^^-----------------------
    print(blue, green, red)
```

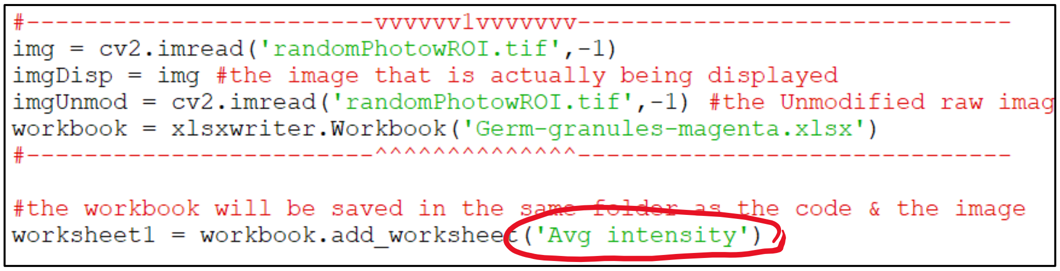


```
if tupleL == 2:
    #-----------------------vvvvvv3vvvvvv-----------------------
    #gray = gray/pixCounter #take average of intensity
    #-----------------------^^^^^^^^^^^^^-----------------------
    print(gray)
    worksheet1.write(cirCount+1, 4, gray)
    font = cv2.FONT_HERSHEY_DUPLEX
    cv2.putText(imgDisp, str(cirCount), (x-3, y+3), font, .3, 6553
    cv2.circle(imgDisp,(x,y), r, 65535, 1)
    cv2.putText(img, str(cirCount), (x-3, y+3), font, .3, 65535, 1
    cv2.circle(img,(x,y), r, 65535, 1)
elif tupleL == 3:
    #-----------------------vvvvvv3vvvvvv-----------------------
    #blue = blue/pixCounter #take average of intensity
    #green = green/pixCounter #take average of intensity
    #red = red/pixCounter #take average of intensity
    #-----------------------^^^^^^^^^^^^^-----------------------
    print(blue, green, red)
```

There are 6 such locations and be sure to comment them all out! Only do this for areas with number v3v as indicators. Do NOT comment out any other sections. (v3v is just an arbitrary code I chose so it is easy to locate these lines)

You only need to do this once. After added those #s, just save the code and from now on all of your runs would give you the sum intensity.

After making this change, it is also a good idea to change the excel workbook's worksheet name (inside the '…'), which is found right below the file name section, so you won't confuse yourself…

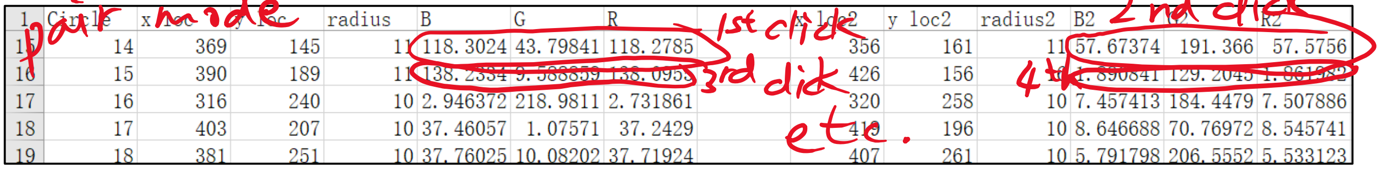


```
#-------------------------vvvvvv1vvvvvvv-----------------------------
img = cv2.imread('randomPhotowROI.tif',-1)
imgDisp = img #the image that is actually being displayed
imgUnmod = cv2.imread('randomPhotowROI.tif',-1) #the Unmodified raw imag
workbook = xlsxwriter.Workbook('Germ-granules-magenta.xlsx')
#-------------------------^^^^^^^^^^^^^^-----------------------------

#the workbook will be saved in the same folder as the code & the image
worksheet1 = workbook.add_worksheet('Avg intensity')
```

B. What is the mode thing on the trackbar panel?

a) If you drag the mode trackbar all the way to the right, it enters a **"pair mode"**: your consecutive two clicks will have the same numbering, and in your output excel spreadsheet, the second click within each pair will be displayed to the right of the first click. This is great for pairwise analysis.



1	Circle	x loc	y loc	radius	B	G	R		x loc2	y loc2	radius2	B2	G2	R2
15	14	369	145	11	118.3024	43.79841	118.2785		356	161	11	57.67374	191.366	57.5756
16	15	390	189	11	138.2334	9.588859	138.0955		426	156	10	1.890841	129.2045	1.861982
17	16	316	240	10	2.946372	218.9811	2.731861		320	258	10	7.457413	184.4479	7.507886
18	17	403	207	10	37.46057	1.07571	37.2429		419	196	10	8.646688	70.76972	8.545741
19	18	381	251	10	37.76025	10.08202	37.71924		407	261	10	5.791798	206.5552	5.533123

4. Other Information

A. Tiff files with z-stacks need to be separated into individual layers stored in separate tiff files before opening with this program.

B. Sometimes when using the trackbar to move the view when zoomed in, the view might not be moving even if you are dragging the trackbar, or might refuse to zoom out, and that is because you are presumably already at the edge of the original image. In this case, the Python shell window should print out "stuck." And you should move the x and y toward the center to address this issue.

C. The code automatically recognizes whether the input image is a grayscale image or an RGB image, and adjusts other things accordingly. If it is a grayscale image, the output excel spreadsheet will have "grayscale" in the first row and have a single column for its intensity; if it is an RGB image, the output spreadsheet will have "RGB" and have 3 columns for the pixel intensities: Blue, Green, and Red. Also if the image is RGB, then the track bar adjusting the display will have a default max value of 255 whereas a 16 bit grayscale image will have a default max value of 65535.

D. Due to some mechanisms of the code (rounding), if you repeatedly zoom in and out drastically and move around, there is a rare chance that the ROI coordinates will be mis-calibrated (e.g. you click somewhere, but the circle shows up to the upper-right of where you clicked). To deal with this issue, reset the zoom to 1. If that still doesn't solve the issue, please contact the author.

E. Email the author (zye20@jhu.edu) for questions or feedbacks.