CMED Image Analysis – Report

Presented to: Prof. Christopher Moraes

Prepared by

Rahul Behal [260773885]

CHEE 494 – Research Project and Seminar

Department of Chemical Engineering

McGill University

2021.04.21

ABSTRACT

The development of a means to measure miniscule tissue stresses is incredibly useful for the general understanding of dynamics at play during tissue formation. As demonstrated in the 2019 paper, polyacrylamide microspherical stress gauges (MSGs) can be dispersed into 3D multicellular spheroid (MCS) cultures to map radial and circumferential stresses [1]. The methodology for mapping these stresses included tracing out circles and ellipses on the microscopy images of the MCS and MSGs using the ImageJ analysis software to extract the dimensions of the fitted shapes in order eventually yield the stresses. This process of individually tracing out the shapes was labour intensive and inefficient. The purpose of this research was to develop a programmatically assisted method of calculating internal stresses from microscopy images of MCS and MSGs. This was done by employing utilizing the OpenCV Python package to fit shapes to detected contours in binary thresholded images, provided constraint parameters through a GUI developed with PyQt5. This process is shown to be much faster and more efficient than manual mapping through ImageJ, with a tradeoff of perfection and error tracking; further work is able to be done to explore rectifications of those drawbacks.

TABLE OF CONTENTS

1 INTRODUCTION

Hydrogel force sensors have been a particular study of interest to better understand internal stresses in tissue formation. Compressive 3D stress profiles have been developed, mapping the axial and radial strain by analyzing the deformation of polyacrylamide microspherical stress gauges (MSGs) in multicellular spheroids (MCS) [1]. These stress profiles were developed through analyzing high resolution (2048 x 1536), 16-bit, TIFF images with multiple colour channels. These images were taken over a period of time starting with the first day, Day 0. The shapes of MSGs and MCS changing over subsequent days were compared to the shapes on Day 0 in order to calculate stress over time. Days as a metric here is used to represent an arbitrary period of regimented time; some, for example, were done over 88-hour time lapses. The bright field colour channels revealed visible MCS in approximate circle shapes, whereas the red channel revealed the MSGs. The process being used included viewing these images in scientific image analysis software ImageJ in order to fit shapes to the image. Circles were fit to MCS and Day 0 MSGs, whereas ellipses were fit to subsequent day MSGs to better capture the stress. Fitting the shapes on ImageJ allowed for the extraction of relevant parameters like area, centroids, angle, radius, and major/minor axes. These could subsequently be used to easily calculate radial, axial, and circumferential strains.

The goal of this report is to assist with this process of mapping shapes to the images, extracting the parameters, and calculating the appropriate strains using numerical methods. Python was used to interface with many open source scientific and image analysis libraries like OpenCV, NumPy, and Matplotlib to effectively aid in the current labour intensive and inefficient process of data extraction.

2 TOOLS AND METHODS

Python was chosen as the programming language to use to accomplish the task at hand. The primary reason it was chosen was due to the ease-of-access to popular and robust image analysis libraries, namely OpenCV. The accessibility in terms of interfacing with any number of applications was also appealing and useful for tasks like writing to human-readable Excel files. A full list of Python modules using with purposes is found as follows:

Table 1 Python libraries used in CMED Image Analysis application

Library Name	General Description	Specific Usecase	Reference
OpenCV	Cross-platform library used for real-time computer vision tasks. Used in image processing, video capture, and analysis.	Thresholding images, finding contours, and fitting circles/ellipses	[2]
PyQt5	Python binding for Qt; C++ libraries that allow access to high-level APIs for modern operating systems allowing for GUI creation.	Creating Graphical User Interface (GUI)	[3]
XlsxWriter	Python module used to read, write, format, and generally manipulate data in modern Excel spreadsheets	Writing both raw and calculated data to Excel files for human-readability	[4]
NumPy	Scientific computing package for python containing a variety of mathematical functions and operations.	Handling manipulation of image arrays	[5]

Extensive use was made out of the OpenCV library. It is used for several operations in the program such as reading, writing, and processing for visualization for the general image viewer. It also handles the entirety of detecting specific shapes in the image, extracting appropriate parameters, and drawing the shapes on the image. PyQt5 is the framework used

to create the GUI so that the user can interact with the images, set appropriate parameters to fit shapes, and extract the data in a specified format. XlsxWriter is used to output the data in a meaningful way to the user. It displays both the raw and calculated data from all of the shapes in the images in an organized and readable format. NumPy was used for storing image data in arrays as well as general mathematical operations.

Several methods were explored in order to map circles and ellipses to an image. One of the most popular methods used for circle detection is the Hough Circle Transform by which imperfections and boundaries are found through analyzing the image with a voting procedure in the Hough parameter space to find values that optimize for best fitting circles. This method was successfully employed from the OpenCV package but was not used due to slow and inefficient calculation time for the high-resolution images being used. Ellipse detection could also prove to be challenging with the Hough Transform, which is typically used on circles.

Other methods tried include various Maximally Stable Extremal Regions (MSER) blob detection techniques like the native SimpleBlobDetector in OpenCV as well as functions implemented in the SkLearn module like the Laplacian of Gaussian and Determinant of Hessian algorithms. The former simply killed the kernel upon running with no discernible errors. The latter successfully mapped circles; however, the fits were poor and extraneous circles were plentiful. This method would also be unable to detect ellipses.

The most successful method explored, and the one implemented in the final program, involves thresholding the image, finding the contours, and then fitting a shape to it. This was done entirely with OpenCV functions: threshold(), findContours(), fitEllipse(), and minEnclosingCircle(). The first step in the process was binary thresholding. This takes in a

pixel value, and any value in the image brighter than the given value is turned white, and values lower are turned black. This is extremely helpful as the MCS are significantly darker than the rest of the image in the bright field channel, and the MSGs are significantly brighter than the rest of the image in the red channel. Next, the contours were found in the binary image using an algorithm developed on the basis of border following [6]. The contours are stored as a series of OpenCV vectors containing points that can be further used and manipulated by other OpenCV functions. For mapping circles, these contours are then passed to functions that attempt to fit a smooth circle or an ellipse to the contours and returns parameters and dimensions sufficient to draw the image. For a circle, this includes the x, y pixel position of the center and the radius of the circle. For an ellipse, this includes the x,y pixel position of the center, the major and minor axes pixel lengths, as well as the angle of rotation. These sets of parameters are sufficient to derive radial, axial, and circumferential strains overtime from the microscopy images.

Challenges of this methodology primarily include the difference in necessary input parameters to map appropriate shapes between different images. Each image effectively requires different thresholding values to exclusively expose shapes of interest. Different values for minimum and maximum radii to consider are also required for each image. Failure to seek appropriate parameters results in copious amounts of noise in the data and nearly unusable fits. Other challenges include the inability to quantify the accuracy of the fit between the contours and the shape. This leads to difficulties when it comes to error propagation of the strain results.

In order to deal with the main challenge, a GUI program was developed allowing users to choose specific parameters for each image and see the results of shape fitting in real-

time. This allows the user to go through each image in the series and set appropriate parameters by visually verifying the fits are accurate. The interface also allows the user choose to export the data in one of several ways. First, the user can first export the raw data of a single pair of bright field and red channel images. This simply extracts the raw dimensions of the shapes and outputs it to an Excel file; notably this does not calculate strain data as it is simply one set of images. The user can also export a set of single marked down images with identification labels for each shape. The second way a user can export the data is in-terms of all images in the series. This exports the raw data for all shapes across all images, as well as calculates the strain data, and writes them to an Excel spreadsheet. The marked down images can also be exported for all of the images as well.

The methodology of mapping shapes to images does not address the other critical question of the ability track and identify the same shapes across different images in order to calculate the strain data. This was done through establishing the concept of a “Base Image’ and “Base Shapes”. The base image is an image for which the shapes are identified and labelled arbitrarily. The value in having this image, is that the shapes for all other images can now be identified and labelled according to the shapes in the base image. This is done by finding the closest shapes in an image to the shapes of the base image; these shapes are then known as the base shapes for that image. All shapes other than the base shapes found in the image are then ignored for the purposes of data collection and export. As such, it is favourable to choose a base image that successfully maps all of the MCS and MSGs that the user is interested. If a shape is not mapped on the base image, it will not be mapped on any other image. Challenges with this method primarily include choosing an appropriate “distance cutoff” when deciding which shapes on an image are closest to the base image shapes.

3 RESULTS AND DISCUSSION

The GUI of the final application developed looks as follows:

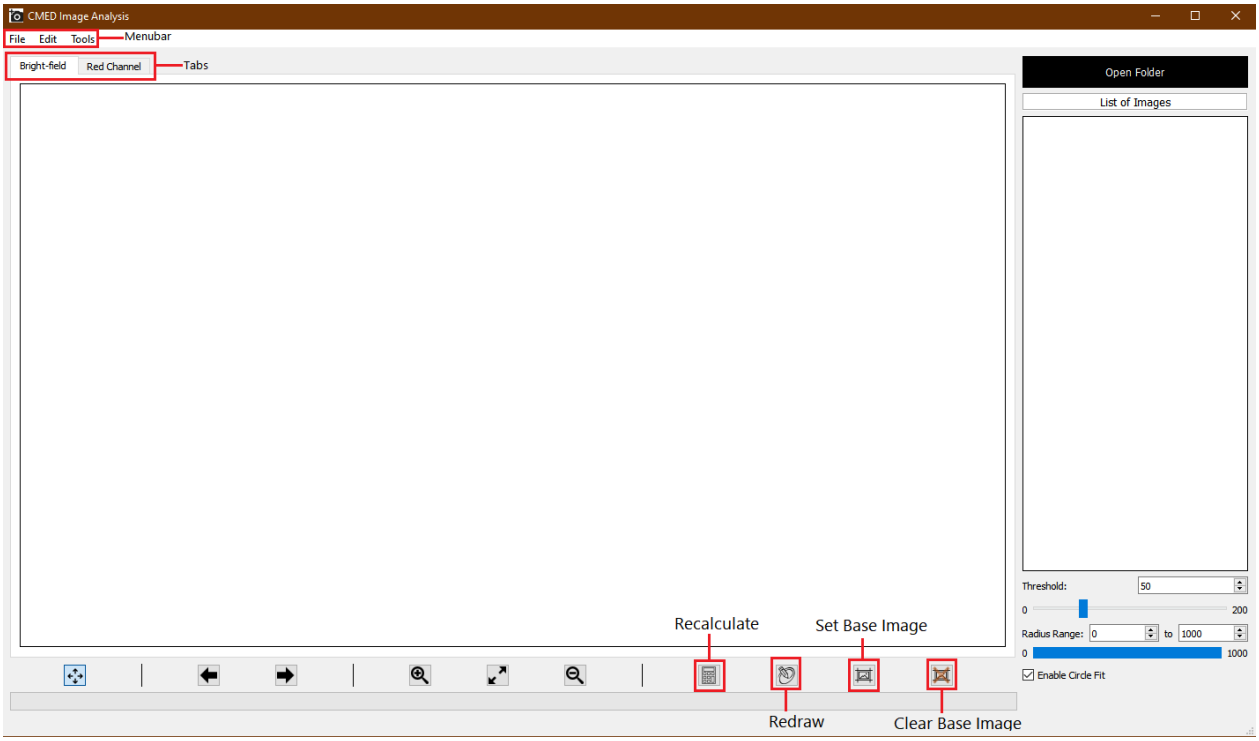


Figure 1 GUI without any image set loaded

A directory is first opened through the “Open Folder” button. Two different folder structures are supported for mapping shapes to images and extracting data from it. An additional folder structure is supported for the viewing and analysis of images termed “Z-Stack” images that measure the MCS and MSGs on the z-axis. The program supports this structure for the calculation of image sharpness and general viewing of the images. The full folder structure and regular expressions being used to parse text can be found in the Appendix.

After a folder has been selected, the program loads in the image sets and immediately displays the image and begins calculating fits and drawing shapes based on default

parameters. The default thresholding parameter for red channel images is set to 120, with minimum and maximum radii of 10 and 100 pixels respectively. For the bright field images, the thresholding defaults at 120 as well, but with a minimum and maximum radii of 40 to 500 pixels respectively.

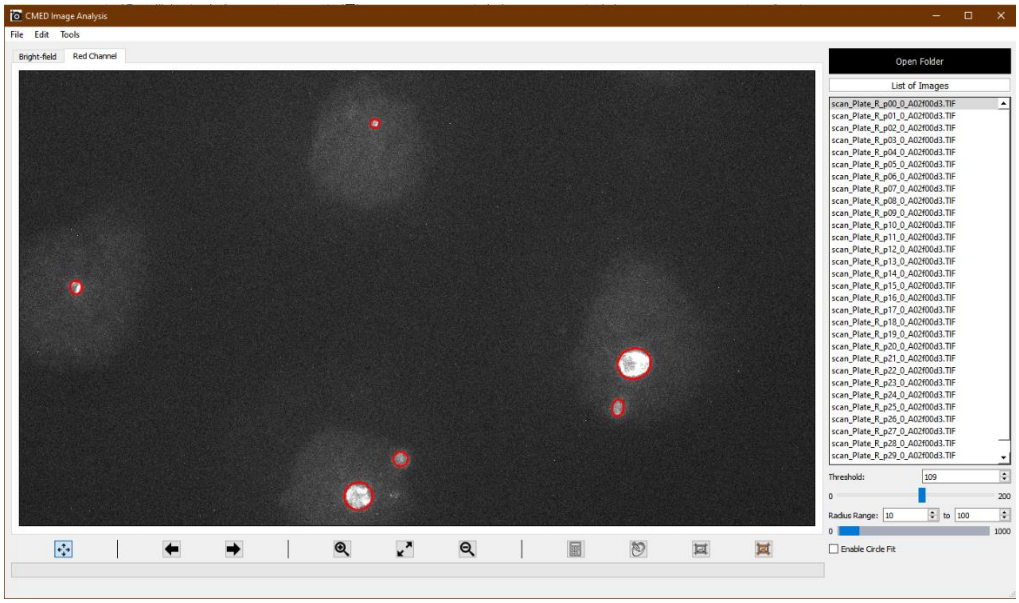


Figure 2 Red channel image tracing MSGs as ellipses in GUI

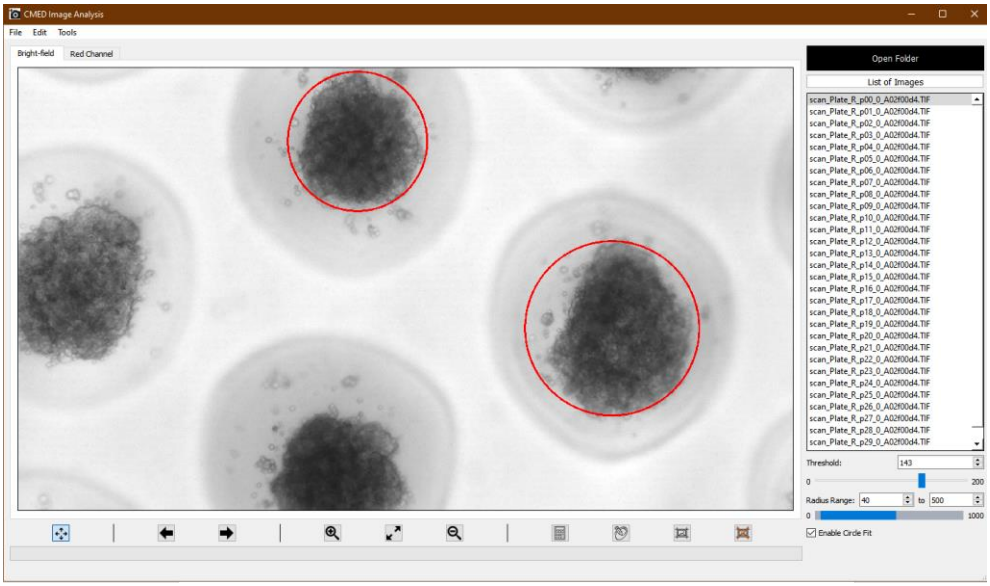


Figure 3 Bright field channel image tracing MCS as circles in GUI

The list of images on the right allows a user to quickly identify which image in the set they're on and navigate between different points in the set with ease. The user can swap between the bright field and red channel image sets using the tabs on the top left. All input parameters will be saved specific to the image being analyzed, and default parameters will adjust accordingly. By default, all MCS and Day 0 MSGs will be calculated and fit to circles, whereas any other MSGs will be calculated and fit to ellipses. However, the checkbox on the bottom right ensures that the user can fit whichever shape they please manually.

The threshold slider and number box and radius double slider and number box allows the user to select specific input parameters for the calculation and fit of shapes on the images. The aforementioned default parameters are selected initially, but it is expected that the user adjusts until the shapes fit appropriately. It is important to note that red channel image shapes should be fit before the bright field image shapes, as MCS shapes are determined by if they have detectable MSGs in them. This was decided because strain calculations are only useful if the context can be traced through the MCS the MSGs are located within.

After the user has gone through the list of images, fitting shapes appropriately to the red channel images, and subsequently the bright field ones, they can set the base image. The base image can be set at any time, but as soon as it is set, the only shapes that will be detected from that point forward are shapes that can be related back to the shapes on the base image. As such, it is advisable to select the base image at the end when the user has context for which images would be best suited. As soon as the base image is set, identification numbers appear for the images as can be seen in Figures 4 and 5.

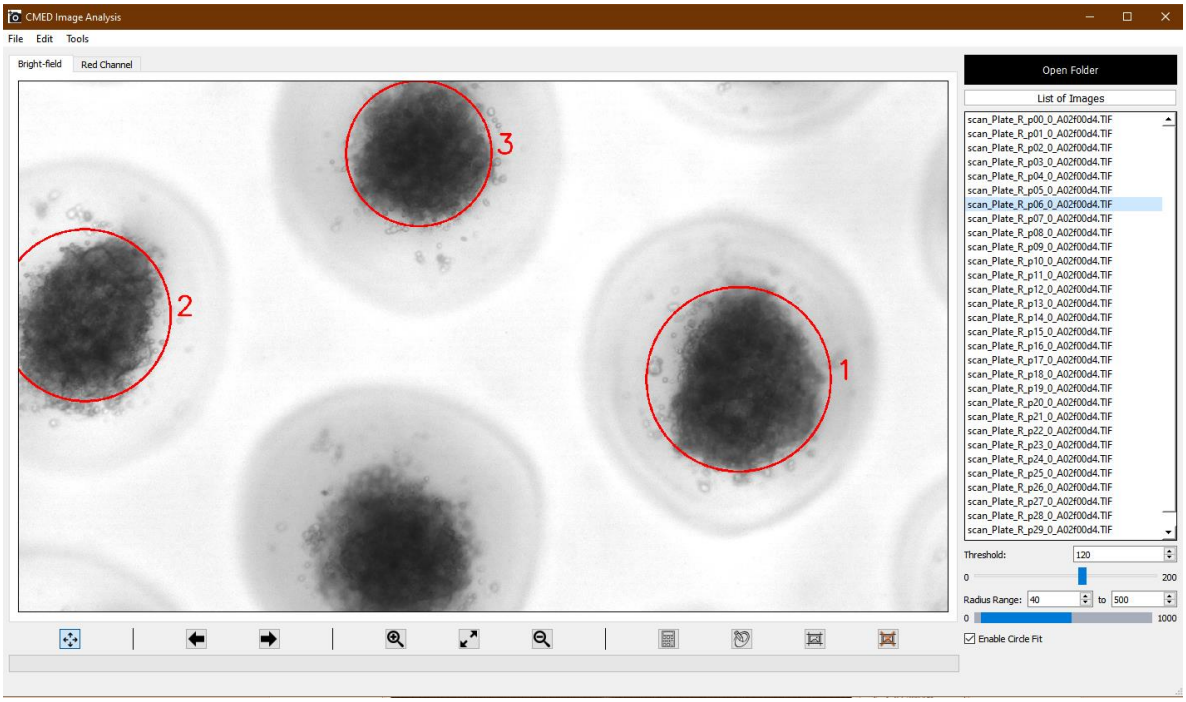


Figure 4 Bright field channel image identifying MCS as circles after setting base shape

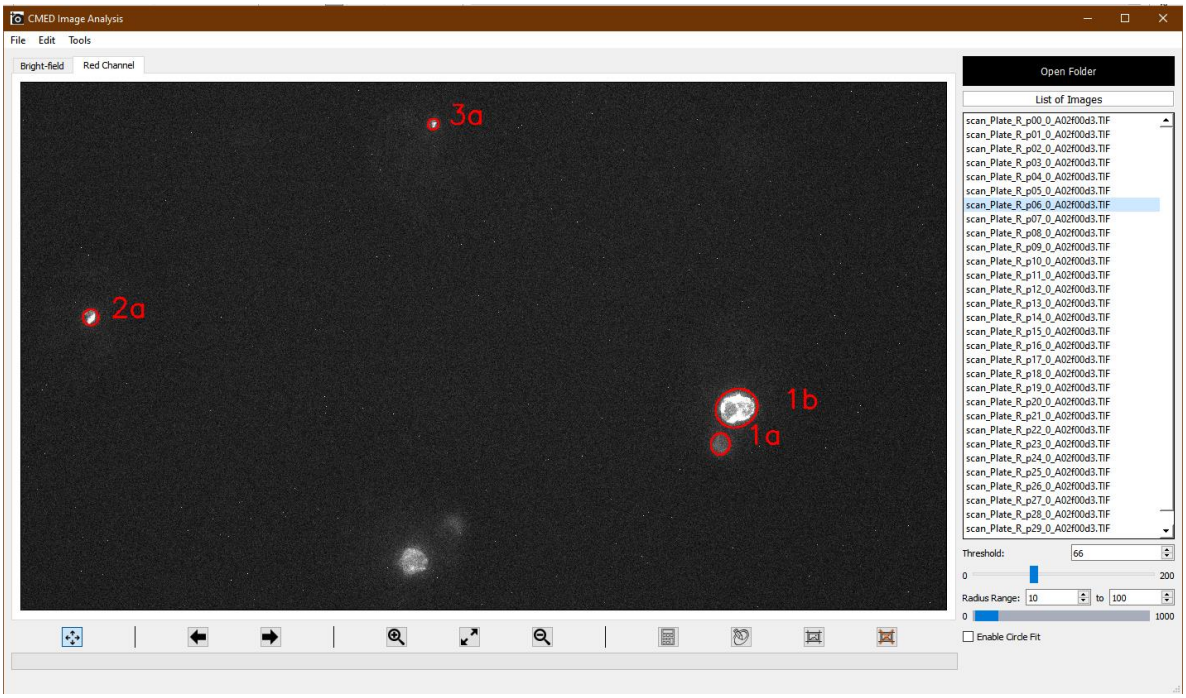


Figure 5 Red channel image identifying MSGs as ellipses after setting base shape

The base image is set and cleared through the buttons on the bottom right below the image window. It should be noted that when setting or clearing a base image, it is applicable to the pair of images, one from the red channel, and one from the bright field. The recalculate button is used to run the thresholding, finding contours, and fitting of shapes functions again. The redraw button is used to recorrelate the image shapes with those of the base image, i.e. redraw the base shapes of the image.

After the base image is set, all of the calculated shapes will be identified and related based on the base image. Any shapes that could not be correlated to base image shapes will be discarded for drawing and data collection purposes. Data can then be exported using the options in the menu bar.

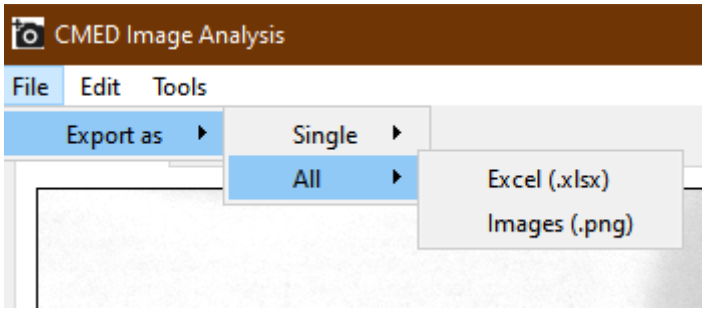
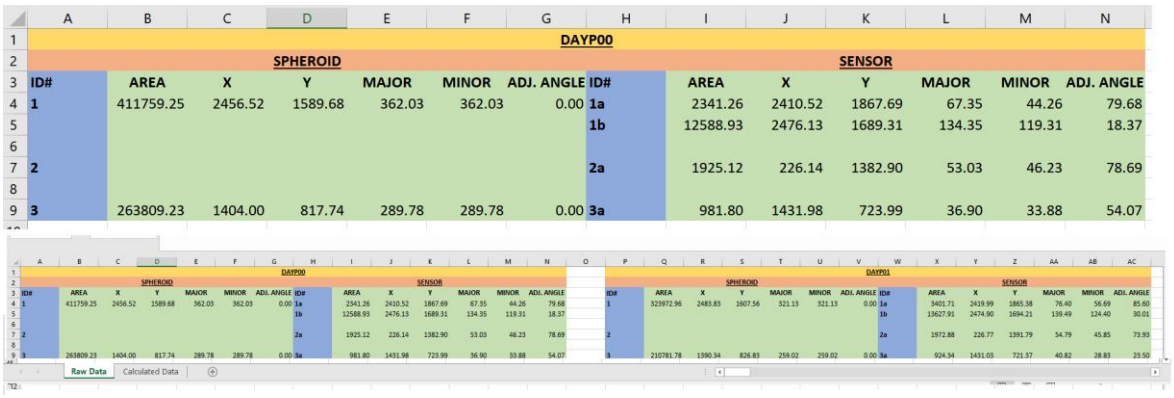


Figure 6 Menu bar export options

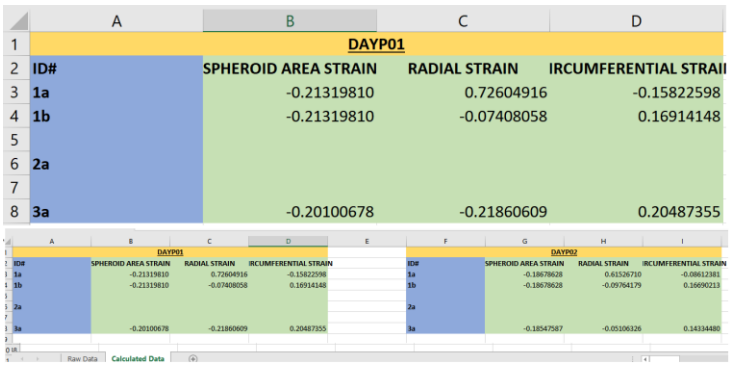
The single export option will set the current image being viewed as the base image and just export the raw shape parameters from that image. Exporting all images will produce the data for all images in the series, with shapes identified consistently through the base shape concept. A sample of the raw data sheet from an Excel file of a series of images can be seen in Figure 7.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1						DAYP00								
2				SPHEROID							SENSOR			
3	ID#	AREA	X	Y	MAJOR	MINOR	ADJ. ANGLE	ID#	AREA	X	Y	MAJOR	MINOR	ADJ. ANGLE
4	1	411759.25	2456.52	1589.68	362.03	362.03	0.00	1a	2341.26	2410.52	1867.69	67.35	44.26	79.68
5								1b	12588.93	2476.13	1689.31	134.35	119.31	18.37
6														
7	2							2a	1925.12	226.14	1382.90	53.03	46.23	78.69
8														
9	3	263809.23	1404.00	817.74	289.78	289.78	0.00	3a	981.80	1431.98	723.99	36.90	33.88	54.07

Figure 7 Raw data format enlarged (top) and zoomed out (bottom)

The raw data is provided on one excel sheet, with each day of data being divided between the “Spheroid” (MCS) and “Sensor” (MSGs) headings respectively. They are formatted and labelled deliberately so that it is easy to see exactly which sensors belong in which spheroids and the dimensions belonging. This data has been converted from pixel values to μm using a scale value of $0.638\,\frac{pixels}{\mu m}$. For each day, the data for all shapes in the base image are outlined, even if one of the shapes could not be found on that specific day. If that was the case, blanks will be filled. This is to ensure consistent formatting across days, and is also intended so that researchers can easily see which days the program failed to detect shapes and supplement or discard that data manually as they see fit. Data is scrollable horizontally, as each days data is produced in the same format, with one column of blank space before then data for the next day in the series gets written.



	A	B	C	D
1		DAYP01		
2	ID#	SPHEROID AREA STRAIN	RADIAL STRAIN	IRCUMFERENTIAL STRAI
3	1a	-0.21319810	0.72604916	-0.15822598
4	1b	-0.21319810	-0.07408058	0.16914148
5				
6	2a			
7				
8	3a	-0.20100678	-0.21860609	0.20487355

Figure 8 Calculated data format enlarged (top) and zoomed out (bottom)

The calculated data is provided in the same Excel workbook, but on a separate worksheet labelled “Calculated Data”. The format is similar to that of the raw data in that the data for each day is calculated and put in the same format, with one column of blank space in between subsequent days. Data that could not be found is again put in blank for manual review. It is important to note that in the calculated data, the days start at Day 1 instead of Day 0. The reason being is that strains can only be computed relative to another day, and so each strain is computed relative to Day 0. This can be seen through the following formula:

$$\epsilon = \frac{\Delta L}{L_o} = \frac{L - L_o}{L_o} \tag{1}$$

where L can be taken as length parameter of a given image and L_o is the parameters of the Day 0 image. These length parameters can be taken as the major/minor axes in the case of radial or circumferential strain, and as the area in the case of area strain.

This process for the extraction of raw data and calculated strain parameters through the GUI application and automated algorithms for detecting and tracing images immensely improves the efficiency of the process. Researcher Christina-Marie Boghdady estimated that the amount of time taken to manually trace shapes and extract data to the Excel appropriately

for the 60 images in Figures 4 and 5 would take approximately 5-6 hours. Using the program developed throughout this report, this process takes approximately 5 minutes.

4 CONCLUSION

The objective of this report was to aid in the current method of mapping circles and ellipses to microscopy images of MCS and MSGs being manually done in ImageJ. Dimensions of these shapes over time can then be used to calculate strains and derive stress profiles. Throughout this report, a GUI application was developed in order to automatically trace shapes to images based on user-defined parameters. The shapes across all images are identified and labelled through the comparison to a base image that the user sets. The program also calculates the radial, axial, and circumferential strains from the centroid, angle, and dimensions attained from OpenCV. The user can then export the data to an Excel workbook with two separate sheets dividing between raw and calculated data. Drawbacks of the methodology primarily include the inability to propagate error due to the correctness of the shape fits being unknown. Ultimately, it is expected that the CMED Image Analysis tool will reduce the time spent on the shape tracing process by more than 90%.

5 REFERENCES

[1] Lee, W., Kalashnikov, N., Mok, S., Halaoui, R., Kuzmin, E., Putnam, A. J., . . . Moraes, C. (2019). Dispersible hydrogel force sensors reveal patterns of solid mechanical stress in multicellular spheroid cultures. Nature Communications, 10(1). doi:10.1038/s41467-018-07967-4

[2] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. "Reilly Media, Inc."

[3] Riverbank Computing. (2019). PyQt5 Reference Guide. Retrieved April 20, 2021, from https://www.riverbankcomputing.com/static/Docs/PyQt5/

[4] McNamara, J. (2013). Creating Excel files with Python and XlsxWriter. Retrieved April 20, 2021, from https://xlsxwriter.readthedocs.io/

[5] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2. (Publisher link).

[6] Satoshi Suzuki and others. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing, 30(1):32–46, 1985.

6 APPENDIX

Folder/File Structure for mapping shapes:

- One of two folder structures:
  - Root Folder:
    - BF
      - ***p##***.[Valid Image Extension]
    - Texas Red
  - Root Folder
    - ***Day##***
      - ***p00***_[Any 6 Characters]d4.[Valid Image Extension]
      - ***p00***_[Any 6 Characters]d3.[Valid Image Extension]
    - ***Day##***
    - ***Day##***
    - ***Day##***
- Code being used for detection:
  - Checks if BF and Texas Red folders exist in selected directory
    - or checks if folder exists with Day Regex: DAY(\d{1,2})
  - Checks for id using ID Regex: (p\d{1,4})
  - For daily folders, BF and TR image sets are separated through the following Regex: _.{6}d(\d), checking for the digit at the end (“4” = BF, “3” = TR)

Folder/File Structure for z-stack:

- Root Folder
  - ***z##***d2.[Valid Image Extension]
  - ***z##***d4.[Valid Image Extension]
- Code being used for detection:
  - Regex in selected folder and to extract id and image type: z(\d{1,4}).*d(\d)