4-1-2024

Development of a Machine Learning-Based Program to Measure Cell Proliferation

Adam J. Jones
University of Dayton

Development of a Machine Learning-Based Program to Measure Cell Proliferation

Honors Thesis

Adam Jones

Department: Chemical and Materials Engineering

Advisor: Russell K. Pirlo, PhD

April 2024

Development of a Machine Learning-Based Program to Measure Cell Confluency

Honors Thesis

Adam Jones

Department: Chemical and Materials Engineering

Advisor: Russell K. Pirlo, PhD

April 2024

Abstract

Many tools have emerged to investigate the functioning of biological systems, especially when in contact with foreign substances. In vitro procedures are often used due to their cost effectiveness and suitability for high-throughput experiments. These procedures collect basic measurements, such as toxicity and biocompatibility, that provide insight into the compatibility and safety of a substance. In vitro toxicity tests are favored for their expediency, affordability, and consistent outcomes. Quantitative methodologies, like colorimetric and fluorometric assays, offer objectivity and high-throughput analysis. However, they require lengthy incubation times and only provide a single metric. Microscopy-based methods provide more information in terms of cell morphology and localization and can be captured quickly without the need for reagents and incubation. Yet, this method requires specialized expertise and is prone to subjective biases and variations based on the region of interest. Given the limitations of microscopy-based approaches, there is a growing interest in leveraging machine learning (ML) to streamline and enhance cell analysis. This study aims to develop an ML-based approach to evaluate cell count and confluency from microscope images and compare its performance to the colorimetric assay, CCK8. The CCK8 assay, which releases a dye when metabolized by live cells, served as the benchmark for comparison. The ML-based method developed using Ilastik, CellProfiler, and Python, segments microscopy images into cell and background regions, followed by erosion for cell boundary enhancement. CellProfiler subsequently quantifies the cell count and confluency from the processed images. This novel ML-based approach offers expedited analysis, while mitigating the inherent subjectivity and error associated with conventional techniques. This approach also eliminates the need for excess reagents and waste associated with quantitative assays. In conclusion, this technique presents an alternative in scenarios where traditional assays are impractical, such as with low cell counts or when cells must be reused.

Acknowledgements

I would like to thank Dr. Russell Pirlo for his mentorship and guidance throughout this project. I would also like to thank the University Honors Program for funding and support and the Department of Chemical and Materials Engineering for allowing me to use their facilities.

University *of* Dayton

Table of Contents

Introduction

Many tools have emerged to investigate the functioning of biological systems, especially when in contact with foreign substances. These tools include several methods, including dye exclusion, colorimetric, and fluorometric assays [1], as well as microscopy techniques, such as cell counting with a hemocytometer [2,3,4] and examination of cell morphology [5,6]. One specific area in which these tools are utilized is the study of biomaterials, which are materials interfaced with biological systems for a range of medical purposes [7]. Since these materials must not cause harm to the patient, rigorous testing must be performed to ensure potential safety before they are even approved for human clinical trials [7]. Although animal studies can accurately predict the efficacy and safety of a new biomaterial, these tests require a significant cost and a high level of care for the animals [8]. Therefore, in vitro tests are often used as a preliminary investigation of a novel biomaterial [9].

Due to various factors, in vitro tests are suitable for initial cell-based experiments. These tests are superior due to their capacity for high-throughput experiments [10,11], which allows for various conditions to be tested at once. In addition, in vitro experiments are rapid and cost-effective [5,8], allowing a successful biomaterial or treatment to reach animal trials quickly without a significant financial contribution. These tests can also measure various outcomes, such as cell function, enzyme activity, and receptor binding [12]. After one of these outcomes is determined, in vitro assays can determine the mechanism of action for a specific cellular phenotype [10,13]. Therefore, this method is a powerful tool to study the initial response of an organism to a particular substance or event.

One specific area where in vitro tests are often used is in the study of cell toxicity. Toxicity is defined as a disruption in the biochemical function of a cell, where the effect can range in severity from barely detectable to fatal [5]. Toxicity tests are used in the first stage of biomedical device testing to assess the biocompatibility and safety of the materials used [7]. There are two main types of cytotoxicity tests. An assay provides a single absorption, fluorescence, or luminescence metric that can be measured in bulk using a plate reader, whereas a stain discriminates between live and dead cells individually and requires counting. One example of an assay is the MTT assay, which

detects the metabolic activity of cells by the conversion of substrates, affecting a change in the light absorption of the cell media [9]. The PicoGreen and AlamarBlue assays can also determine cytotoxicity through a plate reader by measuring green and blue fluorescence when these reagents bind to specific parts of the cell [2]. As a result, these assays are used to differentiate dead from live cells, allowing for the evaluation of a biomaterial's biocompatibility. Other reagents can be used to stain cells and determine live and dead counts via count. For example, trypan blue and propidium iodide are only absorbed in dead cells but require cell counting due to not emitting a readable signal [2]. The large number of effective toxicity measurement methods shows that toxicity can be effectively deduced using in vitro methods.

Although in vitro toxicity tests are effective, they have numerous limitations. These tests require a specific incubation time, temperature, and medium to be effective and eliminate confounding variables [14]. For cell stains that require cell counting, a standard method is through the use of a hemocytometer. This device has a number of glass chambers with specific area and depth [3]. The cell count in one of these chambers, which is completed manually, is extrapolated to determine the overall cell count. Due to the amount of manual labor involved, this method is time-consuming, prone to human error, and requires a narrow cell concentration to be effective [3,4]. Automated methods, such as the Vi-CELL® XR from Beckman Coulter, eliminate the human aspect of cell counting, but still require specific reagents and have a higher cost [4]. Consequently, microscopy-based approaches have been explored to provide more detailed information about the cells without the need for incubation.

Microscope technology has rapidly developed in its ability to discriminate between small objects, which means it is applicable in the study of cells. The axial resolution for a confocal microscope is a few hundred nanometers, which allows for individual cells to be easily distinguished [15]. As a result, cell viability can also be determined using microscopy as altered morphology is another indication of toxicity [5]. Viability can also be measured by examining the colony formation ability or lysosomal integrity of a group of cells [15]. To obtain more detailed images to examine these factors or for morphology analysis, specific microscopy techniques, such as stimulated emission depletion microscopy (STED), are often used [15]. Scanning electron microscope (SEM)

images also give more information about the surface morphology of a cell [6]. Due to the high amount of light needed to perform in-depth microscopy techniques [15] and the gold-plating of samples needed to create SEM images [6], neither method is a viable option for intermediate testing as the cells may not survive the assessment. In addition, identifying specific morphological features of live and dead cells requires a great deal of expertise. Given these limitations, there is a growing interest in leveraging machine-learning (ML) to streamline and enhance the analysis of microscope images of cells.

ML-based methods seek to improve image analysis with automation, reducing user time and effort, increasing throughput, and reducing user bias and error, improving statistical power [16]. Common applications of ML include car navigation, movie recommendation, and face identification [17]. This wide array of complex tasks shows that ML will easily handle the task of cell counting. Several groups have already utilized ML to count and classify cells based on type. For example, one group used ML to develop a "you only look once" method that could accurately identify and count different types of blood cells [18]. Another group utilized ML to develop a program called HALO, which counted and sorted specific cells within a tissue [19]. These tools involve supervised learning methods, which can label objects after training [17]. Many open-source software packages, such as Ilastik, use this type of ML to segment cell microscope images.

Ilastik is a program that generates probability maps of cell images. The Broad Institute has combined this program with CellProfiler to count Chinese Hamster Ovary cells and determine confluence [20]. This study aims to build upon this approach to evaluate cell count and confluency from TIME GFP cell images. The results from this method will be compared with CCK8, which is a reputable colorimetric toxicity assay [21]. This comparison will determine the ML-based method's efficacy and facilitate the development of a novel cell counter.

Methods & Results

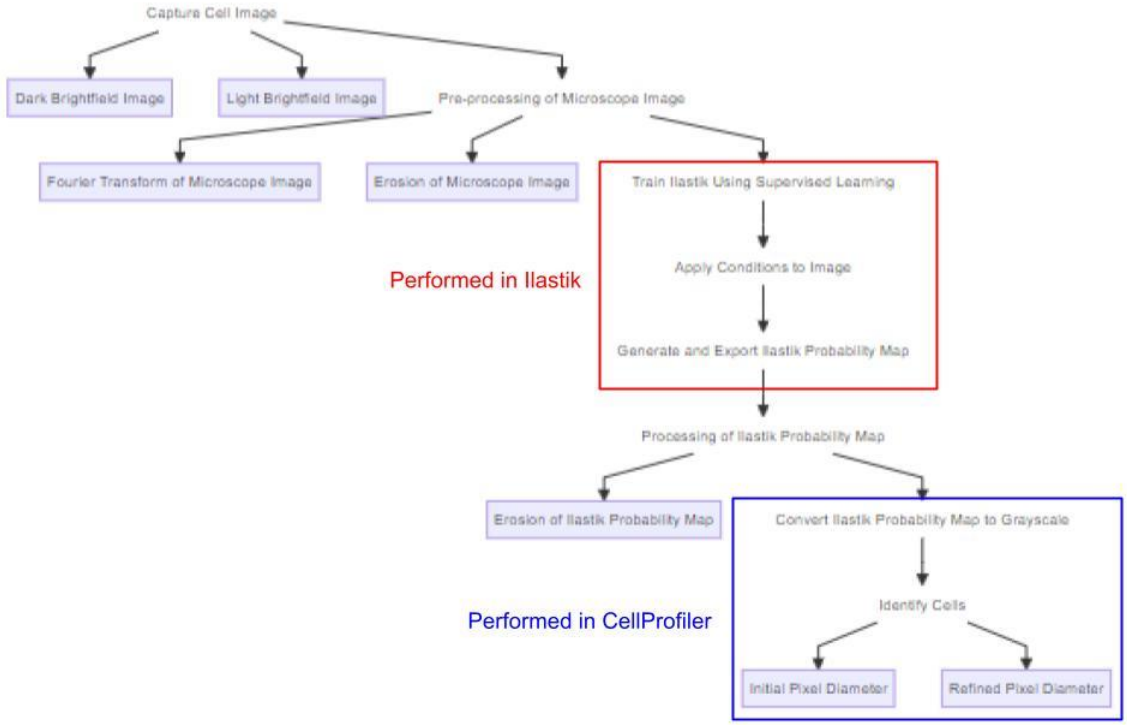

Figure 1. Image Processing Methods

Fig. 1 describes the steps in the processing of a microscope image using the ML-based program. This follows the methods of Karhohs [20] while also adding pre-processing of the microscope image and processing of the Ilastik image steps. The steps in the gray-shaded boxes correspond to parts of the development process as shown in **Fig. 2**.

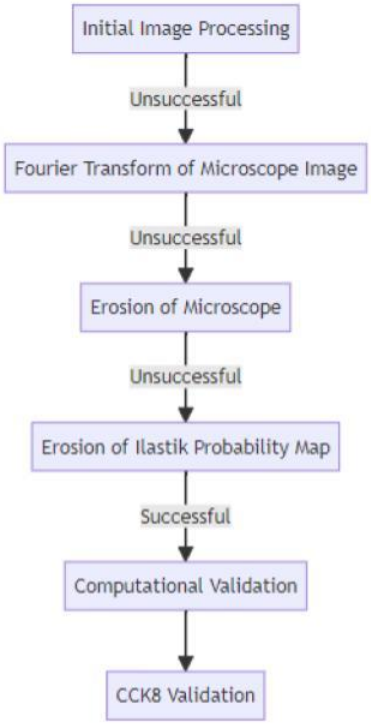

Figure 2. Development of the ML-based program.

Fig. 2 depicts the overall methods for the development of an ML-based program to measure cell confluency. Since the goal of this project was to develop a working program, the Methods & Results section delineates each step of the process with its outcome.

Cell Culture

Time GFP cells were thawed and seeded in a T25 flask. The cells were cultured at 37 degrees Celsius in Time GFP media. The media was changed every other day and cells were passaged every week, reseeding at a density of 50000 cells in a T25 flask with 5 mL of media. The cells were monitored for three weeks.

Cell Images

Cells images were taken every time the media was changed, which led to images of various cell densities. Four methods were used to capture cells each day they were fed, yielding seven viable images for each type of image. All images were taken at 10x

magnification. The types of images were Brightfield (BF) Dark, BF Light, Phase Dark, Phase Light, and GFP. The dark images were focused so the cells showed up darker than the background, whereas the light images were focused so the cells showed up lighter than the background. The GFP images were not used because not all parts of the cell fluoresced, which would yield inaccurate confluency results. The phase images were not used because many images had halos, as shown in **Fig. 3**, which Ilastik would improperly identify as cells.

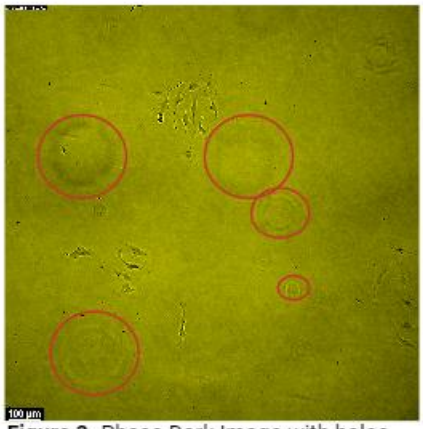

Figure 3. Phase Dark Image with halos (circled in red).

Initial Image Processing

The methods described by Karhohs [20] were recreated. This involved first creating a probability map in Ilastik as shown in **Fig. 4**. This map was generated by training the software through specifying which areas contained cells and which were background. Next, a 'pipeline' was developed in CellProfiler to determine the cell count and confluency. This pipeline involved 'RescaleIntensity', 'ColorToGray', and 'IdentifyPrimaryObjects' modules. The RescaleIntensity module enhances certain areas to allow for easier cell segmentation. The ColorToGray module was necessary to convert the image to grayscale so it could be processed through the IdentifyPrimaryObjects module. That module calculated cell count and confluency by implementing a cell diameter range. Since dead cells will appear smaller in the microscope, this range ensures that only live cells will be counted. The resulting output of the IdentifyPrimaryObjects module is shown in **Fig. 5**.

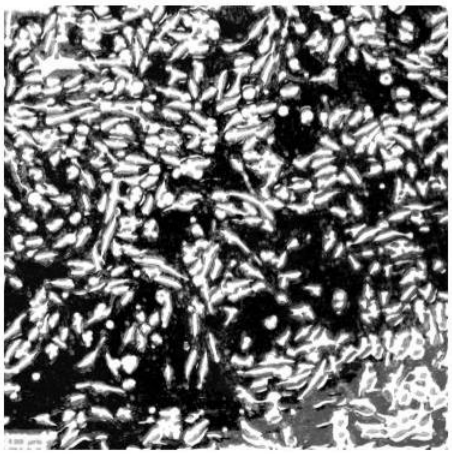

Figure 4. Probability map created in Ilastik

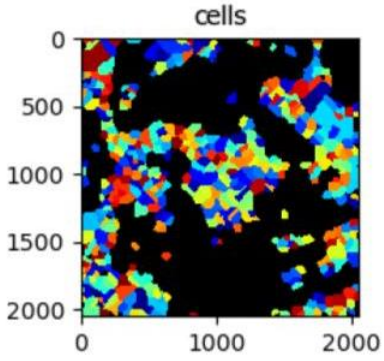

Figure 5. CellProfiler output using unprocessed microscope image

Recreating Karhohs's previous methods yielded the CellProfiler output shown in **Fig. 5**. The calculated cell count of 473 had a 414.13% error from the manual calculations, and the calculated cell confluency of 45.0% had an 847.37% error from the manual calculations. This showed that other imaging processing methods were needed to make this method effective.

Image Processing

Numerous image-processing techniques were employed to improve the efficacy of this ML-based method. The code for all of these operations is shown in **Appendix I**. The first image in the BF Dark folder was used for all preliminary image processing.

Fourier Transform of Microscope Image

The microscope image was passed through a Fourier transform before entering into Ilastik (**Code A1)**. The resulting Fourier-transformed image is shown in **Fig. 6**. Although the human eye could more easily distinguish the cells, this transformation processed an inaccurate Ilastik probability map. This resulted in an improper identification of the cells, as shown in **Fig. 7**. The calculated cell count of 14 had a 84.78% error from the manual calculations, and the calculated cell confluency of 98.6% had a 1975.79% error from the manual calculations.

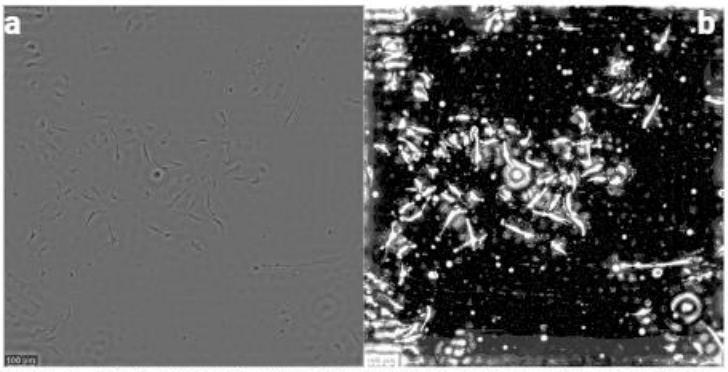

Figure 6. (a) shows the Fourier-transformed microscope image and (b) shows the ensuing Ilastik probability map.

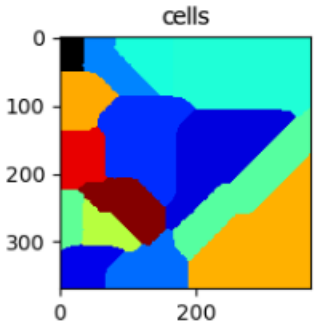

Figure 7. This depicts the improper identification of cells in CellProfiler after the microscope image underwent a Fourier transform.

Erosion of Microscope Image

A variety of erosion and dilations conditions were performed on the microscope image. This included eroding and dilating the image at kernels of 3, 5, 7, 9, and 11 (**Code**

A2). The operations that were performed on the microscope image are shown in **Fig. 8**. Kernel 7 erosion produced the image with the most distinctive cells with the least distortion, so this condition was used for future image processing. The kernel 7 eroded image was processed using Ilastik to predict cell count and confluency as shown in **Fig. 9**. The calculated cell count of 87 had a 5.43% error, which was much closer than any other pre-processing methods. On the other hand, the confluency of 21.4 had a percent error of 350.53% due to the blocky nature of the predicted cells. Therefore, this method was not effective either for identifying cell count.

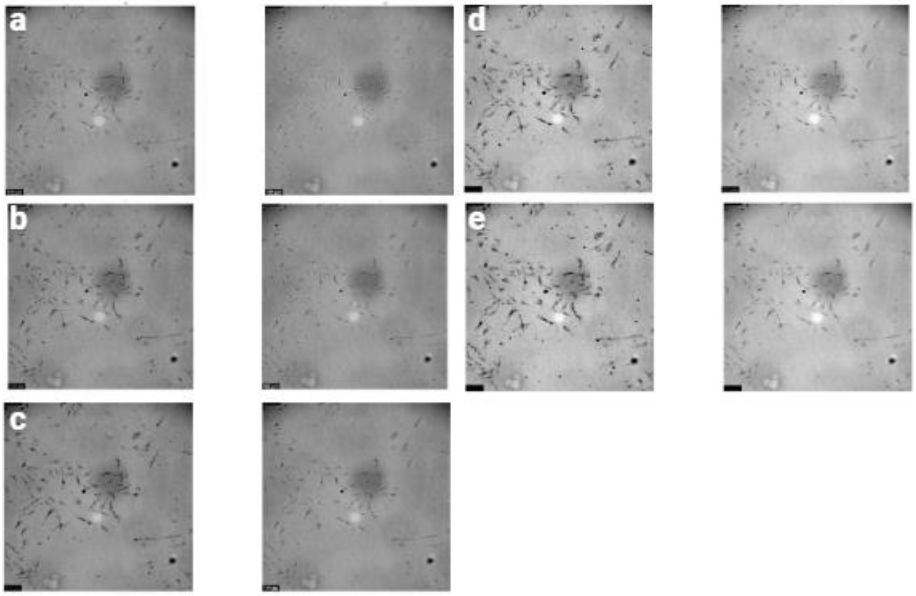

Figure 8. This figure depicts a variety of single erosions and dilations at different kernels. Images in the left column of each figure are erosions and images in the right column of each figure are dilations. (a) is performed at kernel 3, (b) is at kernel 5, (c) is at kernel 7, (d) is at kernel 9, and (e) is at kernel 11.

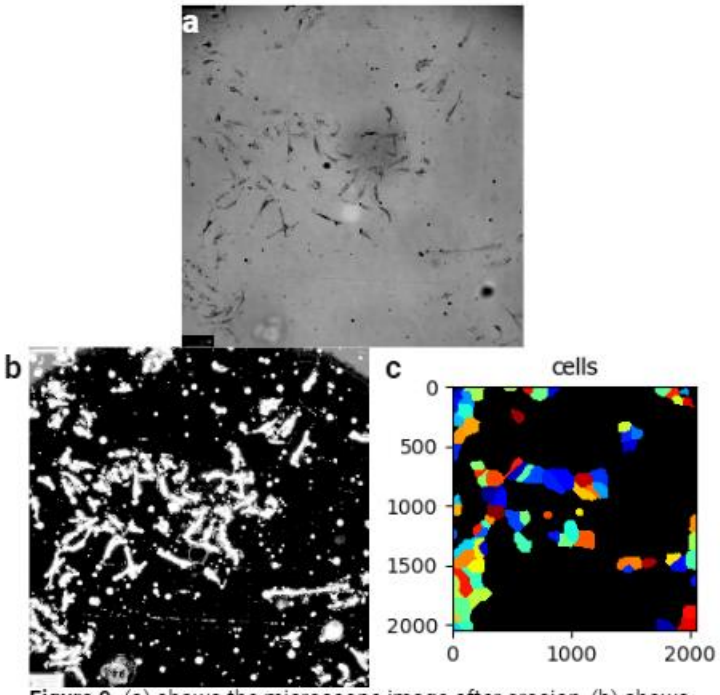

Figure 9. (a) shows the microscope image after erosion, (b) shows the resulting Ilastik image, and (c) shows the CellProfiler output.

Erosion of Ilastik Probability Map

Since processing of the raw microscope image was yielding inaccurate CellProfiler outputs, processing of the Ilastik probability map was explored as an alternative method. Erosions and dilations were performed with a kernel value of 7 after the image was processed through Ilastik (**Code A3**). The CellProfiler outputs from this method were analyzed for clear borders between cells. Example outputs using erosions of 1, and 2 are shown in **Fig. 10**. These methods produced much more accurate CellProfiler outputs, so more cell images were processed using this method.

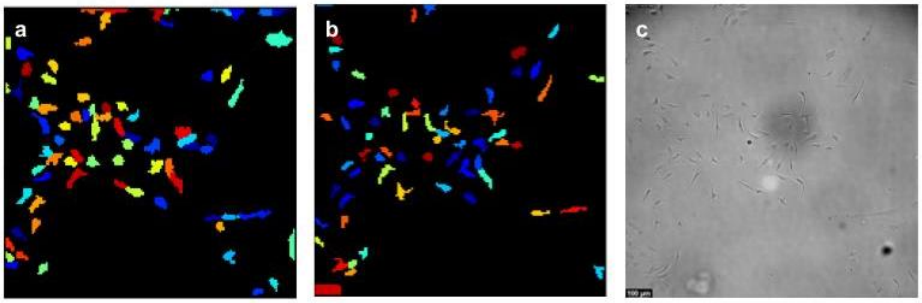

Figure 10. CellProfiler outputs of single (a) and double (b) erosions of the Ilastik probability map. The initial image is shown in (c).

Computational Validation

Seven viable BF Dark and BF Light images were all analyzed using Ilastik, followed by a variety of erosions (**Code A4**), and then analyzed in CellProfiler to determine cell count and confluency. Each image was processed through 1, 2, and 3 erosions (E1, E2, E3) at kernel 7 and every condition was processed separately in CellProfiler. The conditions were processed in CellProfiler first using a cell pixel diameter range shown in **Table 1** and second using a cell pixel diameter range shown in **Table 2**. The diameters in **Table 1** were proposed to capture all of the viable cells in image 1, which had a relatively small number of cells. The diameters in **Table 2** were proposed to capture all of the viable cells in image 3, which had a relatively large number of cells.

Table 1. Initial Pixel Diameter Range

	Min	Max
Dark E1	10	30
Dark E2	9	30
Dark E3	7	30
Light E1	10	30
Light E2	8	30
Light E3	6	30

Table 2. Refined Pixel Diameter Range

	Min	Max
Dark E1	9	22
Dark E2	7	20
Dark E3	6	20
Light E1	8	22
Light E2	7	20
Light E3	6	20

The cell count and confluency were validated using codes shown in **Appendix I**. The first code utilized the cv2 module (**Code A5**), which was used to color over the cells using the computer mouse. **Fig. 11** shows an example output from this method. This code also counted the number of objects created, which corresponded to the cell count. This image was saved, and another code (**Code A6**) was used to determine the cell confluence in the image.

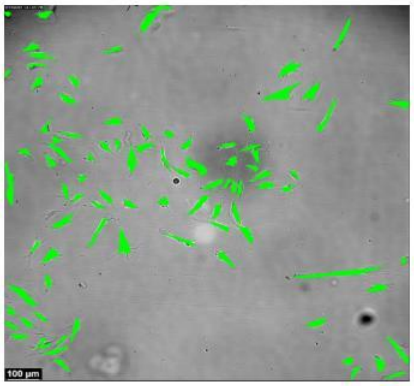

Figure 11. Output from test code.

Tables 3 and 4 illustrate the percentage error elicited from the initial and refined image processing techniques for cell count. The Dark/Light label refers to the type of Brightfield image that was taken. The images processed using the initial pixel diameter in **Table 1** do not have a label before Dark/Light, whereas the images processed using the refined pixel diameter in **Table 2** are labeled as 'ref' before the Dark/Light label. The number after 'E' specifies the number of erosions performed of the Ilastik output. Although there were many conditions with a large percent error, the refDark E1, refDark

E2, and refDark E3 conditions produced the lowest error with values of 12.13 ± 5.81%, 12.63 ± 14.75, and 11.49 ± 10.47, respectively. The refDark E1 and E2 also produced the best fit slope values closest to 1 with values of 1.006 and 1.1021.

Tables 3 and 4. Initial and Refined Error of Cell Counting

3

Image	Dark E1	Dark E2	Dark E3	Light E1	Light E2	Light E3
1	2.17%	10.87%	3.26%	3.26%	7.61%	6.52%
2	34.86%	37.30%	20.81%	75.41%	75.41%	74.05%
3	15.89%	40.19%	42.99%	2.80%	8.41%	17.76%
4	0.00%	*	*	13.64%	0.00%	*
5	8.82%	8.82%	*	20.59%	47.06%	58.82%
6	30.65%	35.48%	23.12%	38.71%	31.18%	15.59%
7	27.78%	38.89%	27.08%	18.75%	2.08%	16.67%
Mean ± STD	17.17 ± 14.13%	28.59 ± 14.62%	23.45 ± 14.23%	24.74 ± 25.43%	24.54 ± 28.26%	31.57 ± 27.72%
R^2	0.987	0.9706	0.9818	0.2919	0.2091	0.1946
Best Fit Slope	0.605	0.6166	0.8452	0.1725	0.1657	0.2206

4

Image	refDark E1	refDark E2	refDark E3	refLight E1	refLight E2	refLight E3
1	9.78%	30.43%	7.61%	9.78%	4.35%	6.52%
2	18.92%	0.27%	5.41%	43.78%	59.19%	74.05%
3	2.80%	4.67%	29.91%	22.43%	20.56%	17.76%
4	18.18%	*	*	40.91%	31.82%	*
5	14.71%	32.35%	*	73.53%	76.47%	58.82%
6	12.90%	1.08%	9.68%	2.15%	8.06%	15.59%
7	7.64%	6.94%	4.86%	16.67%	17.36%	16.67%
Mean ± STD	12.13 ± 5.81%	12.63 ± 14.75%	11.49 ± 10.47%	29.89 ± 24.57%	31.12 ± 27.06%	31.57 ± 27.72%
R^2	0.9942	0.9831	0.9861	0.7786	0.4942	0.1948
Best Fit Slope	0.7736	1.006	1.0121	0.4904	0.3294	0.2206

Tables 3 and 4. Depict % error values of cell count. * represents an image that could not be processed, ref symbolizes the refined pixel range.

Tables 5 and 6 illustrate the percent error elicited from the initial and refined image processing techniques for confluency. Every image processing technique was much less effective for calculating confluency than cell count. This was shown with the lowest percentage errors of 22.50 ± 20.68% for the Dark E3 condition and 22.74 ± 20.24% for the refDark E3 condition. This was also shown through the best fit slopes as the slope closest to 1 was 0.818 from the refLight E1 condition.

Tables 5 and 6. Initial and Refined Error of Confluency Calculations

5

Image	Dark E1	Dark E2	Dark E3	Light E1	Light E2	Light E3
1	135.79%	51.58%	3.16%	74.74%	15.79%	22.11%
2	90.58%	60.21%	43.51%	33.94%	54.44%	65.83%
3	72.66%	8.27%	44.24%	97.84%	47.48%	15.11%
4	885.29%	*	*	150.00%	91.18%	*
5	80.95%	38.10%	*	152.38%	109.52%	71.43%
6	33.60%	2.75%	19.45%	11.98%	14.54%	44.01%
7	75.79%	19.40%	2.16%	100.66%	70.81%	60.86%
Mean ± STD	196.38 ± 305.27	30.05 ± 23.54%	22.50 ± 20.68%	88.79 ± 53.41%	57.68 ± 35.85%	46.56 ± 23.61%
R^2	0.8433	0.8893	0.8845	0.3949	0.2847	0.1757
Best Fit Slope	1.4286	1.4612	1.3996	0.5392	0.3694	0.2861

6

Image	refDark E1	refDark E2	refDark E3	refLight E1	refLight E2	refLight E3
1	133.68%	64.21%	1.05%	72.63%	22.11%	22.11%
2	105.01%	81.47%	48.82%	3.57%	40.77%	65.83%
3	76.26%	7.91%	38.85%	105.04%	52.88%	15.11%
4	238.24%	*	*	223.53%	105.88%	*
5	76.19%	52.38%	*	171.43%	123.81%	71.43%
6	44.40%	10.02%	15.52%	30.65%	0.79%	26.33%
7	95.69%	50.91%	9.45%	127.20%	84.08%	60.86%
Mean ± STD	109.92 ± 63.01%	44.48 ± 29.62%	22.74 ± 20.24%	104.86 ± 77.26%	61.47 ± 44.92%	43.61 ± 25.06%
R^2	0.9239	0.895	0.8905	0.6624	0.4434	0.2375
Best Fit Slope	1.7976	1.6603	1.4572	0.818	0.5117	0.3509

Tables 5 and 6. Depict % error values of cell confluency. * represents an image that could not be processed, ref symbolizes the refined pixel range

CCK8 Validation

The cell count measuring capabilities of the program were also validated using a CCK-8 assay. Two calibration curves at high and low concentrations were generated to validate the replicability of the CCK8 assay in a 24-well plate. The low concentration curve included points of 1000, 2000, 3000, 4000, 5000, and 6000 cells per well. The high concentration curve included points of 10000, 20000, 25000, 50000, 75000, and 100000 cells per well. The low calibration curve yielded a slope of -5E-6. The negative slope showed that this assay was inaccurate. The high calibration curve yielded an equation of

$$number\ of\ cells = \frac{absorbance + 0.0319}{6 \times 10^{-6}}$$

The slope from this equation was very similar to previous calibration curves, so it was used to calculate cell count in the experimental samples.

For the experimental samples, cells were seeded in each well of a 24-well plate with 3000 cells in each well of the first column, 5000 cells in the second column, 10000 cells in the third column, 20000 cells in the fourth column, 40000 cells in the fifth

column, and 60000 cells in the sixth column. The 20000-cell count was selected for being the seeding number in an experiment performed by Cai et al. [22] and the 40000 and 60000 were selected for being double and triple this amount. The 3000, 5000, and 10000 cell counts were selected for their adherence to a previous CCK8 absorbance calibration curve. The cells were incubated for three hours, and the absorbance values were collected. The predicted cell counts from the CCK8 assays were calculated from these absorbance values and the calibration curve. The predicted cell counts using the ML-based program were calculated using the procedure from the **Computational Validation** section for only BF Dark images at 1, 2, and 3 erosions. The refined pixel diameter ranges in **Table 2** were used for processing in CellProfiler.

Table 7 depicts the error of the CCK8 assay on determining cell count. The error was high at lower cell counts, such as 3000 and 5000 cells per well, whereas it decreased as the cell concentration increased. **Table 8** describes the error of the ML-based program in determining the cell count of the same wells tested using the CCK8 assay. The results for this method were the opposite, as the ML-based program had relatively low errors of up to 10000 or 20000 cells per well but had greater errors in the 40000 and 60000 cell wells.

Table 7. Error of CCK8 Assay

Cells	% Error
3000	205.00%
5000	114.67%
10000	34.83%
20000	34.92%
40000	17.75%
60000	13.22%
Mean ± STD	70.06 ± 0.76%
R^2	0.9704
Best Fit Slope	0.724

Table 8. Error of ML-Based Program on a 24-well Plate

Cells	E1	E2	E3
3000	14.93%	14.66%	*
5000	14.56%	0.14%	17.62%
10000	0.97%	14.84%	49.80%
20000	27.88%	14.01%	20.39%
40000	35.50%	24.82%	29.54%
60000	57.37%	48.59%	50.62%
R Square	0.8903	0.9089	0.8765
Best Fit Slope	0.4228	0.5051	0.408

Data Analysis

The percentage error for the cell count and confluency for computational validation were both calculated in Google Sheets. R^2 values and linear fit slopes were calculated using Microsoft Excel. The same analysis was performed for the CCK8 validation experiment.

Discussion

Computational validation of the ML-based program proved that three imaging methods were the most effective: refDark E1, refDark E2, and refDark E3. The refDark E1 condition had a comparable average error to the other two conditions but had the lowest standard deviation by far. Therefore, this condition had the most consistent error, which could be potentially predicted and eliminated by determining the relationship between error and cell amount. Although this seems promising, this condition had a best fit slope of 0.7736, which is much lower and farther from 1 than the refDark E2 and refDark E3 slopes of 1.006 and 1.0121, respectively. This was due to the processing of image 2, which had the highest actual cell count. When processing this image, the refDark E1 cell count was 18.92% too low, the refDark E2 cell count was 0.27% too high, and the refDark E3 cell count was 5.41% too high. In addition, the refDark E2 method only produced an ineffective CellProfiler output in one image, whereas the refDark E3 method produced two ineffective CellProfiler outputs. Due to an accurate best

fit slope and low error at higher cell counts, the refDark E2 method was the most effective method.

The results of this experiment are consistent with previous literature. Erosion has been previously used in cell segmentation to separate overlapping cells [23]. Therefore, the cell count should become more accurate until whole cells are eroded. This was seen in the E3 condition, where more ineffective CellProfiler maps were produced due to over-erosion of the Ilastik probability map.

The CCK8 assay was ineffective at lower cell concentrations, but its accuracy increased as cell count increased. Using the ML-based program on the same cells produced the opposite results as the lower cell counts had lower error, whereas the higher cell counts had higher error. Although the CCK8 assay is more sensitive than other metabolic assays such as MTT [24], the ML-based program must have a higher sensitivity at lower cell concentrations. The lower accuracy of the ML-based method at higher cell counts contradicted the results of the computational validation experiment. This result could be explained by the use of a 24-well plate, which has a recommended seeding density of 50000 cells per well [25]. When the cell concentration per plate approaches this, the cell plate will be at confluency. Any excess cells that divide will be suspended in solution. The ML-based method works by taking an image of only the cells attached to the bottom. As a result, any excess cells will not be captured and factored into the cell count calculations. This affirms that the ML-based method will have a saturation point, especially when using low surface area containers.

Conclusions

The results of this experiment proved that the CCK8 assay and ML-based program occupy different niches in the calculation of cell count. The ML-based program has a higher sensitivity and can more accurately predict lower cell concentrations, whereas the CCK8 assay does not have a saturation point and can more accurately predict higher cell concentrations. Therefore, these two methods can be used in tandem for experiments that require a large range of cell concentrations in multi-well plates whose absorbances can be read in a plate reader. For larger surface areas and unorthodox containers, the ML-based program, specifically the refDark E2 method, offers an

accurate, rapid, and low-cost approach to determine cell count. This experiment validated the efficacy of an ML-based program that determines cell confluency and demonstrated that the program rivals and even in some instances exceeds the CCK8 assay in its accuracy.

References

(1)

Ramasamy, S.; Pakshirajan, K. 15 - Product Evaluation: Cytotoxicity Assays. In *Biomedical Product and Materials Evaluation*; Mohanan, P. V., Ed.; Woodhead Publishing, 2022; pp 373–408. https://doi.org/10.1016/B978-0-12-823966-7.00024-4.

(2)

Wiegand, C.; Hipler, U.-C. Methods for the Measurement of Cell and Tissue Compatibility Including Tissue Regeneration Processes. *GMS Krankenhhyg Interdiszip* **2008**, *3* (1), Doc12.

(3)

Absher, M. CHAPTER 1 - Hemocytometer Counting. In *Tissue Culture*; KRUSE, P. F., PATTERSON, M. K., Eds.; Academic Press, 1973; pp 395–397. https://doi.org/10.1016/B978-0-12-427150-0.50098-X.

(4)

Cadena-Herrera, D.; Esparza-De Lara, J. E.; Ramírez-Ibañez, N. D.; López-Morales, C. A.; Pérez, N. O.; Flores-Ortiz, L. F.; Medina-Rivero, E. Validation of Three Viable-Cell Counting Methods: Manual, Semi-Automated, and Automated. *Biotechnology Reports* **2015**, *7*, 9–16. https://doi.org/10.1016/j.btre.2015.04.004.

(5)

Kirkpatrick, C. J.; Mittermayer, C. Theoretical and Practical Aspects of Testing Potential Biomaterialsin Vitro. *Journal of Materials Science: Materials in Medicine* **1990**, *1* (1), 9–13. https://doi.org/10.1007/BF00705347.

(6)

Zhang, J.-Z.; Saggar, J. K.; Zhou, Z.-L.; Bing-Hu. Different Effects of Sonoporation on Cell Morphology and Viability. *Bosn J Basic Med Sci* **2012**, *12* (2), 64–68.

(7)

Rogero, S. O.; Malmonge, S. M.; Lugão, A. B.; Ikeda, T. I.; Miyamaru, L.; Cruz, Á. S. Biocompatibility Study of Polymeric Biomaterials. *Artificial Organs* **2003**, *27* (5), 424–427. https://doi.org/10.1046/j.1525-1594.2003.07249.x.

(8)

Jain, A. K.; Singh, D.; Dubey, K.; Maurya, R.; Mittal, S.; Pandey, A. K. Chapter 3 - Models and Methods for In Vitro Toxicity. In *In Vitro Toxicology*; Dhawan, A., Kwon, S., Eds.; Academic Press, 2018; pp 45–65. https://doi.org/10.1016/B978-0-12-804667-8.00003-1.

(9)

Kamal, A. F.; Iskandriati, D.; Dilogo, I. H.; Siregar, N. C.; Hutagalung, E. U.; Susworo, R.; Yusuf, A. A.; Bachtiar, A. Biocompatibility of Various Hydoxyapatite Scaffolds Evaluated by Proliferation of Rat's Bone Marrow Mesenchymal Stem Cells: An in Vitro</Em> Study. *Medical Journal of Indonesia* **2013**, *22* (4), 202–208. https://doi.org/10.13181/mji.v22i4.600.

(10)

Yoon, M.; Campbell, J. L.; Andersen, M. E.; Clewell, H. J. Quantitative in Vitro to in Vivo Extrapolation of Cell-Based Toxicity Assay Results. *Critical Reviews in Toxicology* **2012**, *42* (8), 633–652. https://doi.org/10.3109/10408444.2012.692115.

(11)

Radio, N. M.; Mundy, W. R. Developmental Neurotoxicity Testing in Vitro: Models for Assessing Chemical Effects on Neurite Outgrowth. *NeuroToxicology* **2008**, *29* (3), 361–376. https://doi.org/10.1016/j.neuro.2008.02.011.

(12)

Supplements, I. of M. (US) and N. R. C. (US) C. on the F. for E. the S. of D. Categories of Scientific Evidence—In Vitro Data. In *Dietary Supplements: A Framework for Evaluating Safety*; National Academies Press (US), 2005.

(13)

Tennant, R. W.; Margolin, B. H.; Shelby, M. D.; Zeiger, E.; Haseman, J. K.; Spalding, J.; Caspary, W.; Resnick, M.; Stasiewicz, S.; Anderson, B.; Minor, R. Prediction of Chemical Carcinogenicity in Rodents from in Vitro Genetic Toxicity Assays. *Science* **1987**, *236* (4804), 933–941.

(14)

Youdim, K. A.; Dobbie, M. S.; Kuhnle, G.; Proteggente, A. R.; Abbott, N. J.; Rice-Evans, C. Interaction between Flavonoids and the Blood–Brain Barrier: In Vitro Studies. *Journal of Neurochemistry* **2003**, *85* (1), 180–192. https://doi.org/10.1046/j.1471-4159.2003.01652.x.

(15)

Schneckenburger, H.; Weber, P.; Wagner, M.; Schickinger, S.; Richter, V.; Bruns, T.; Strauss, W. s. l.; Wittig, R. Light Exposure and Cell Viability in Fluorescence Microscopy. *Journal of Microscopy* **2012**, *245* (3), 311–318. https://doi.org/10.1111/j.1365-2818.2011.03576.x.

(16)

Jordan, M. I.; Mitchell, T. M. Machine Learning: Trends, Perspectives, and Prospects. *Science* **2015**, *349* (6245), 255–260. https://doi.org/10.1126/science.aaa8415.

(17)

Kan, A. Machine Learning Applications in Cell Image Analysis. *Immunology & Cell Biology* **2017**, *95* (6), 525–530. https://doi.org/10.1038/icb.2017.16.

(18)

Alam, M. M.; Islam, M. T. Machine Learning Approach of Automatic Identification and Counting of Blood Cells. *Healthcare Technology Letters* **2019**, *6* (4), 103–108. https://doi.org/10.1049/htl.2018.5098.

(19)

Hvid, H.; Skydsgaard, M.; Jensen, N. K.; Viuff, B. M.; Jensen, H. E.; Oleksiewicz, M. B.; Kvist, P. H. Artificial Intelligence-Based Quantification of Epithelial Proliferation in Mammary Glands of Rats and Oviducts of Göttingen Minipigs. *Toxicol Pathol* **2021**, *49* (4), 912–927. https://doi.org/10.1177/0192623320950633.

(20)

Karhohs, K. *CellProfiler & Ilastik: Superpowered Segmentation*. Broad Institute: Carpenter-Singh Lab. https://carpenter-singh-lab.broadinstitute.org/blog/cellprofiler-ilastik-superpowered-segmentation (accessed 2024-04-09).

(21)

Jiao, G.; He, X.; Li, X.; Qiu, J.; Xu, H.; Zhang, N.; Liu, S. Limitations of MTT and CCK-8 Assay for Evaluation of Graphene Cytotoxicity. *RSC Adv.* **2015**, *5* (66), 53240–53244. https://doi.org/10.1039/C5RA08958A.

(22)

Cai, L.; Qin, X.; Xu, Z.; Song, Y.; Jiang, H.; Wu, Y.; Ruan, H.; Chen, J. Comparison of Cytotoxicity Evaluation of Anticancer Drugs between Real-Time Cell Analysis and CCK-8 Method. *ACS Omega* **2019**, *4* (7), 12036–12042. https://doi.org/10.1021/acsomega.9b01142.

(23)

Wang, Z. A New Approach for Segmentation and Quantification of Cells or Nanoparticles. IEEE Transactions on Industrial Informatics 2016, 12 (3), 962–971. https://doi.org/10.1109/TII.2016.2542043.

(24)

Xiong, J.; Xiao, H.; Zhang, Z. An Experimental Research on Different Detection Conditions between MTT and CCK-8. Acta Laser Biology Sinica 2007, 16 (5), 559.

(25)

Useful Numbers for Cell Culture - US. https://www.thermofisher.com/us/en/home/references/gibco-cell-culture-basics/cell-culture-protocols/cell-culture-useful-numbers.html (accessed 2024-04-11).

Appendix I

Code A1. Fourier Transform

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft2, ifft2, fftshift, ifftshift
import tifffile
import os

def low_pass_filter(image, cutoff_frequency):
    # Compute 2D Fourier Transform
    image_fft = fft2(image)

    # Shift zero frequency component to the center
    image_fft_shifted = fftshift(image_fft)

    # Get image dimensions
```

```
    rows, cols = image.shape

    # Create a mask for low-pass filtering
    mask = np.ones((rows, cols))
    center = (rows//2, cols//2)
    x, y = np.ogrid[:rows, :cols]
    mask[(x - center[0])**2 + (y - center[1])**2 >
cutoff_frequency**2] = 0

    # Apply the mask to the shifted Fourier Transform
    image_fft_shifted_filtered = image_fft_shifted * mask

    # Shift back to the original position
    image_fft_filtered = ifftshift(image_fft_shifted_filtered)

    # Inverse Fourier Transform to get the filtered image
    filtered_image = np.abs(ifft2(image_fft_filtered))

    return filtered_image

# Get file path from the user
file_path = input("Enter the file path of the TIFF image: ")

# Load the image using tifffile
image = tifffile.imread(file_path)

# Set cutoff frequency (adjust as needed)
cutoff_frequency = 30

# Apply low-pass filter
filtered_image = low_pass_filter(image, cutoff_frequency)

# Specify the output directory
output_directory = r"C:\Users\jones\Documents\Pirlo
Lab\Thesis\2023.8.30 BF Dark Cells"

# Specify the output file name
output_file_name = "filtered_image.tif"

# Create the full output file path
output_file_path = os.path.abspath(os.path.join(output_directory,
output_file_name))

# Display original and filtered images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(filtered_image, cmap='gray')
plt.title('Filtered Image (Low-pass)')
```

```
# Save the filtered image
tifffile.imwrite(output_file_path, filtered_image)

# Show the plot
plt.show()

print(f"Filtered image saved to {output_file_path}")
```

Code A2. Erosion and Dilation of Microscope Image

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

def apply_erosion_dilation(image_path):
    # Read the image
    image_color = cv2.imread(image_path, cv2.IMREAD_UNCHANGED)

    # Convert the color image to grayscale
    image_gray = cv2.cvtColor(image_color, cv2.COLOR_BGR2GRAY)

    # Apply Erosion and Dilation
    # kernel size is altered to test how each kernel size looks.
    kernel_size = 7
    kernel = np.ones((kernel_size, kernel_size), np.uint8)
    image_eroded = cv2.erode(image_gray, kernel, iterations=1)
    image_dilated = cv2.dilate(image_eroded, kernel, iterations=1)

    # Display only the eroded and dilated images
    plt.figure(figsize=(10, 4))

    plt.subplot(1, 2, 1)
    plt.imshow(image_eroded, cmap='gray')
    plt.title('Eroded Image')
    plt.axis('off')

    plt.subplot(1, 2, 2)
    plt.imshow(image_dilated, cmap='gray')
    plt.title('Dilated Image')
    plt.axis('off')

    plt.show()

# Get the image file path as input
image_path = input("Enter the path of the image: ").strip('"')

# Apply erosion and dilation and display the images
apply_erosion_dilation(image_path)
```

Code A3. Erosion and Dilation of Ilastik Image

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tifffile

def main():
    # Input the image path
    image_path = input("Enter the path to the image: ").strip('"')

    # Load the image
    try:
        image = tifffile.imread(image_path)
    except ImportError:
        print("Error: The 'imagecodecs' package is required to read
TIFF files with LZW compression.")
        print("Please install the package using: pip install
imagecodecs")
        return
    except Exception as e:
        print("Error: Could not read the image. Please check the
file path.")
        print("Exception:", e)
        return

    if image is None:
        print("Error: Could not read the image. Please check the
file path.")
        return

    print("Image shape:", image.shape)  # Print the shape of the
loaded image

    # Separate channels
    channel1 = image[:,:,0]
    channel2 = image[:,:,1]

    # Define the kernel size for erosion and dilation
    kernel_size = 7
    kernel = np.ones((kernel_size, kernel_size), np.uint8)

    # Apply erosion and dilation to each channel separately
    erosion1 = cv2.erode(channel1, kernel, iterations=2)
    dilation1 = cv2.dilate(channel1, kernel, iterations=2)
    erosion2 = cv2.erode(channel2, kernel, iterations=2)
    dilation2 = cv2.dilate(channel2, kernel, iterations=2)

    # Display the images
    plt.figure(figsize=(12, 6))
```

```
    # Channel 1
    plt.subplot(2, 3, 1)
    plt.imshow(channel1, cmap='gray')
    plt.title('Channel 1 (Original)')
    plt.axis('off')

    plt.subplot(2, 3, 2)
    plt.imshow(erosion1, cmap='gray')
    plt.title('Erosion 1')
    plt.axis('off')

    plt.subplot(2, 3, 3)
    plt.imshow(dilation1, cmap='gray')
    plt.title('Dilation 1')
    plt.axis('off')

    # Channel 2
    plt.subplot(2, 3, 4)
    plt.imshow(channel2, cmap='gray')
    plt.title('Channel 2 (Original)')
    plt.axis('off')

    plt.subplot(2, 3, 5)
    plt.imshow(erosion2, cmap='gray')
    plt.title('Erosion 2')
    plt.axis('off')

    plt.subplot(2, 3, 6)
    plt.imshow(dilation2, cmap='gray')
    plt.title('Dilation 2')
    plt.axis('off')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    main()
```

Code A4. Variable Erosions Used in Final Analysis

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import tifffile
import os

def main():
    # Input the image path
    image_path = input("Enter the path to the image: ").strip('"')
```

```
    # Load the image
    try:
        image = tifffile.imread(image_path)
    except ImportError:
        print("Error: The 'imagecodecs' package is required to read
TIFF files with LZW compression.")
        print("Please install the package using: pip install
imagecodecs")
        return
    except Exception as e:
        print("Error: Could not read the image. Please check the
file path.")
        print("Exception:", e)
        return

    if image is None:
        print("Error: Could not read the image. Please check the
file path.")
        return

    print("Image shape:", image.shape)  # Print the shape of the
loaded image

    # Separate channels
    channel1 = image[:, :, 0]

    # Define the kernel size for erosion and dilation
    kernel_size = 7
    kernel = np.ones((kernel_size, kernel_size), np.uint8)

    # Prompt the user to enter the number of iterations for erosion
    iterations = int(input("Enter the number of iterations for
erosion: "))

    # Apply erosion to Channel 1
    erosion1 = cv2.erode(channel1, kernel, iterations=iterations)

    # Display the erosion1 image
    plt.imshow(erosion1, cmap='gray')
    plt.axis('off')

    # Prompt the user to enter the path and filename to save the
erosion1 image
    save_path = input("Enter the path to save the erosion1 image:
").strip('"')
    filename = input("Enter the filename for the erosion1 image
(include extension, e.g., .jpg): ").strip('"')
    output_file = os.path.join(save_path, filename)
    plt.savefig(output_file, bbox_inches='tight', pad_inches=0)

    plt.show()
```

```
if __name__ == "__main__":
    main()
```

Code A5. Manual Determination of Cell Areas

```
import cv2
import numpy as np

# Function to calculate confluency
def calculate_confluency(region_area, total_image_area):
    confluency = (region_area / total_image_area) * 100
    return confluency

# Function to handle mouse events
def draw_region(event, x, y, flags, param):
    global points, drawing, cell_count, resized_image
    if event == cv2.EVENT_LBUTTONDOWN:
        drawing = True
        points = [(x, y)]
    elif event == cv2.EVENT_MOUSEMOVE:
        if drawing:
            points.append((x, y))
            # Fill the region on the resized image
            cv2.fillPoly(resized_image, [np.array(points)],
color=(0, 255, 0))
            cv2.imshow('Image', resized_image)
    elif event == cv2.EVENT_LBUTTONUP:
        drawing = False
        points.append((x, y))
        # Check if contour needs to be closed
        if len(points) > 2 and np.linalg.norm(np.array(points[0]) -
np.array(points[-1])) < 100:
            # Close the contour by connecting last point to the
first point
            points.append(points[0])
        # Increment the cell count
        cell_count += 1
        # Fill the region on the resized image
        cv2.fillPoly(resized_image, [np.array(points)], color=(0,
255, 0))

# Get the path to the image from the user
image_path = input("Enter the path to the image: ")

# Strip quotes from the file path if they are present
image_path = image_path.strip('"')

# Load the image
image = cv2.imread(image_path)
```

```
if image is None:
    print("Error: Could not load image. Please check the file
path.")
    exit()

resized_image = cv2.resize(image, (800, 600))  # Resize for better
visualization

# Set default values
points = []
drawing = False
cell_count = 0

# Create a window and set the mouse callback function
cv2.imshow('Image', resized_image)
cv2.setMouseCallback('Image', draw_region)

# Wait for the user to press 'c' to calculate confluency and quit
while True:
    cv2.imshow('Image', resized_image)
    key = cv2.waitKey(1) & 0xFF
    if key == ord('c'):
        break

# Calculate the total image area based on the original size of the
loaded image
total_image_area = image.shape[0] * image.shape[1]

# Calculate the area of the drawn regions
region_area = 0
for contour in points:
    if len(contour) > 2:  # Check if contour has at least 3 points
        contour_array = np.array(contour)
        if contour_array.ndim == 3 and contour_array.shape[1] >= 3
and contour_array.shape[2] == 2:
            region_area += cv2.contourArea(contour_array)

# Calculate and print the confluency
confluency = calculate_confluency(region_area, total_image_area)
print("Confluency: {:.2f}%".format(confluency))

# Print the number of regions drawn (cell count)
print("Number of regions drawn (cell count):", cell_count)

# Close all windows
cv2.destroyAllWindows()
```

Code A6. Determination of Confluency

```
from PIL import Image
```

```
def calculate_green_percentage(image_path):
    # Open the image
    image = Image.open(image_path)

    # Convert the image to RGB mode
    image_rgb = image.convert('RGB')

    # Initialize counters
    total_pixels = 0
    green_pixels = 0

    # Iterate over each pixel in the image
    for pixel in image_rgb.getdata():
        total_pixels += 1
        # Check if the pixel is green (0, 255, 0)
        if pixel == (0, 255, 0):
            green_pixels += 1

    # Calculate the percentage of green pixels
    green_percentage = (green_pixels / total_pixels) * 100

    return green_percentage

if __name__ == "__main__":
    # Get the path to the PNG image from the user
    image_path = input("Enter the path to the PNG image: ")

    # Strip quotes from the file path if they are present
    image_path = image_path.strip('"')

    # Calculate the percentage of green color
    percentage_green = calculate_green_percentage(image_path)

    # Print the percentage of green color
    print("Percentage of green color:
{:.2f}%".format(percentage_green))
```