UNIVERSITY OF CALIFORNIA
SANTA CRUZ

CANCER CELL GROWTH MEASUREMENT USING COMPUTER VISION AND MACHINE LEARNING

A thesis submitted in partial satisfaction of the
requirements for the degree of

Master of Science

in

COMPUTER ENGINEERING

by

Amir R. Pourshafiee

June 2016

The Thesis of Amir R. Pourshafiee
is approved:

Table of Contents

List of Figures

List of Tables

Abstract

Cancer Cell Growth Measurement using Computer Vision and Machine Learning

by

Amir R. Pourshafiee

We have devised a pipeline of computer vision and machine learning algorithms and developed a cell growth rate detection software. The algorithm has the ability to perform single cell detection.

Our cell segmentation mechanism uses the Canny edge detection method to detect the foregrounds of an image; after processing and filtering the foregrounds of the image, the living cell images are stored. In each iteration, we run a very fast object recognition algorithms to detect and match the image of the cells that have had been stored in the previous iteration with the ones that are stored in the recent iteration. If the algorithm fails to find any matches stored from the prior iteration, we run a slower algorithm with that provides a higher likelihood of recognition. After finding the matches we calculate the rate at which the cell's area and perimeters have changed.

We also considered automated cell injection and have made our program forward compatible with possible future automated injection software or devices.

Acknowledgments

I would like to thank my committee for spending their precious time reviewing this document and providing me guidance. I would like to specifically thank professor Mircea Teodorescu, my Principal investigator for providing 3 years of guidance and insight in my research career. Also, Professor Nader Pourmand, Emrah Ozel Akshar Lolith, and Scott Rad for providing support and feedback and helping me with data collection. I would like to extend my appreciations to my graduate reviewing committee and CITRIS for providing us with lab space.

Chapter 1

Introduction

1.1 Description of the thesis

In this section, we describe the flow of the content in this thesis. In the rest of this chapter we explain our motivations and the objectives of this research. From this perspective, in the next section we highlight some of the state-of-the-art cell detection papers. We explain the literature review that we did about foreground detection, cell segmentation, and object detection. Chapter 3 discusses our methods and challenges with foreground detection, filtering, and cell interpretation. Chapter 4 describes our implementation of object detection and a survey of three of the most prominent object detection algorithms and explain how we calculate the rate of growth in cells.

Chapter 5 is dedicated to automated injection and our attempts and methods to perform it as well as the challenges we faced and our solution. In chapter 6 we have concluded the thesis by describing the computer vision and machine learning pipeline.

We also discussed the future work and the expansions that are possible for this research.

1.2 Goals

Our highest level goal was to detect cells and determine their rate of growth using the images obtained by a digital microscope camera. We wanted to be able to distinguish each cell and be able to detect it rapidly. Knowing that cells can only live for several minutes before they need to be placed in an incubator, we knew there might be challenges in terms of positioning the cells exactly where they were before (without rotation). Our goal was to find algorithms that can tolerate the noise and are rotation and scaling invariant. We also wanted to be able to automatically inject the cells.

Chapter 2

Literature Review

After reviewing the literature and understanding the challenges, we performed algorithms for cell detection and individual-cell identification. In this section we explain how we devised the algorithms and we show our results, comparing different algorithms. We tried two methods to distinguish between the cells and the background, edge detection and blob detection. Unfortunately, other methods like color discrimination and shape detection could not be coupled with our algorithms to improve the detection and its robustness. By adding dye to cells, we might alter the results of the future experiments. Since the cells are alive, they could react to the foreign agent and either die or demonstrate a different reaction than would those unmodified by coloring. Regarding the shape, cells, even from the same cell line can have varying shapes that could not be unified.

In terms of the identification algorithms, the challenges were the rapid deformation of cells (24 hours to 48 hours) and the short shelf life of cell lines. We have listed a few algorithms that we found in the literature, explained their methods, and have concluded the reasons each might or might not suit our applications. In the next section we surveyed various edge detection methods and determined that Canny edge detection delivers the best results in our application.

2.1 Foreground Detection and Cell Segmentation

In this section we first did a survey of the most predominant edge detection methods and concluded that the Canny edge detector is best suited for our application. We briefly introduce these methods and then show the results of our survey and compare it with a related study[14].

2.1.1 Canny edge detection

The Canny edge detector is an algorithm that follows a stochastic procedure for the design of arbitrary edge profiles [5]. The detector uses adaptive thresholding, which changes based on the noise level of the image, with hysteresis to eliminate the streaking of edge contours [5].

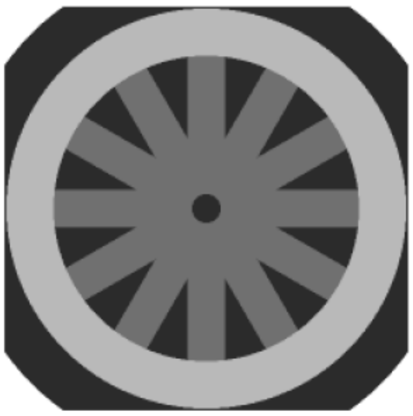
According to John Canny, the inventor of the algorithm, "In two dimensions, it was shown that marking edge points at the maxima of gradient magnitude in the gradient direction is equivalent to finding zero-crossings of a certain nonlinear differential operator" [5]. The algorithm is devised to do the following: [16]

1. Smoothing: Noise reduction by blurring the image

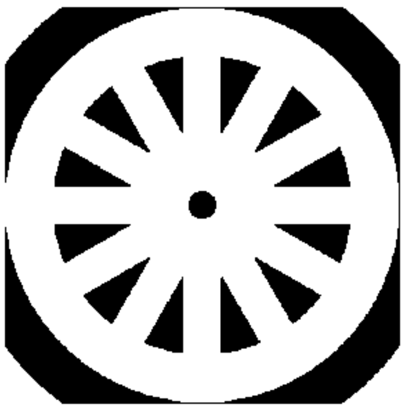
2. Finding local gradient anomalies: By marking the areas that have gradients with high magnitudes
3. Local maxima markup: markings that were not a result of local maxima are suppressed
4. Thresholding: Edges are determined by thresholding
5. Hysteresis: By applying a hysteresis that discriminates against edges that are not connected to strong edges, the algorithm returns the likely edges of the image.

However, a drawback of the algorithm for our purpose was that the performance of the algorithm is hindered when the image operator is locally circular as opposed to straight. This disadvantage has become less important with today's technology; with higher resolution cameras, the image operators locally appear to be more linear.

Figure 2.1 illustrates the image that Maini used to survey the detection along with the binary conversion of the image for the edge detection algorithms to run. We have reproduced the results of an edge detection survey by Maini [14] in Matlab which can be seen in Figure 2.4. The survey suggests that Canny edge detection delivers the best results. By reproducing the survey result and doing experimental edge detection testings on a cell using the Canny edge detection and comparing it to the Matlab's cell segmentation tutorial (2.4), we confirmed that the Canny edge detection was the best method. The other algorithms are briefly described in the rest of this section.



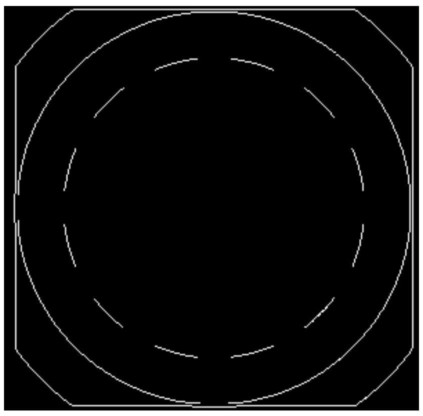
(a) Original image in Maini's survey

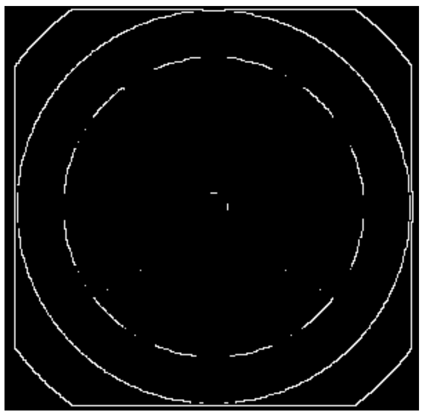


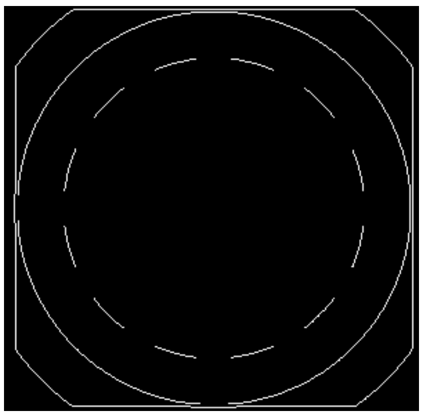
(b) Maini's sample image converted to binary

Figure 2.1: *Sample image for surveying and its corresponding binary image for image processing. 2.1*

In both methods we used the Canny edge detection which filters the images with the derivative of their Gaussian filter mask. The Gaussian filter mask is generated by the local standard deviation (which at edges, results into higher values). The value of the standard deviation with respect to the mean determines the width of the filter matrix. Thus, standard deviation controls the amount of smoothing produced by the Gaussian filter, and using the Gaussian filter make the detection less prone to noise.


(a) Prewitt edge detection


(b) Robert's edge detection


(c) Sobel edge detection

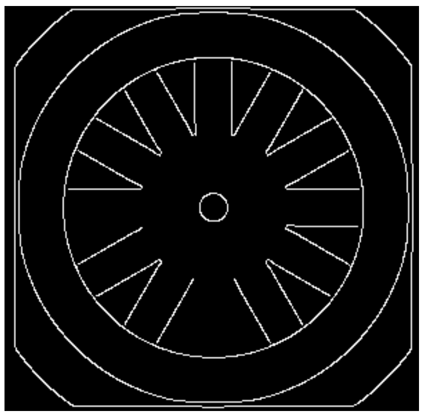

(d) Canny edge detection

Figure 2.2: *Reproduction of Maini's survey with 4 different edge detection algorithms. It can be seen that Canny edge detection has performed significantly better than the rest of the algorithm*

2.1.2 Sobel-Feldman Filter

Irwin Sobel and Gary Feldman presented a talk in Stanford in 1968 "on a relatively isotropic 3x3 gradient operator", Which was inspired by the work of Larry

Roberts [25, 8]. In this method, an image is treated as a density function and "for a 3x3 neighborhood each simple central gradient estimate is a vector sum of a pair of orthogonal vectors. Each orthogonal vector is a directional derivative estimate multiplied by a unit vector specifying the derivative's direction.[26, 7]" A weighted unit vector is chosen that represents the neighbor's directions and the threshold is calculated that outlines the edge of the element[26, 7].

We have used the Matlab implementation of the algorithm in Figures 2.2c and 2.4 (re-implementation of a Matlab cell segmentation example) to compare it with some other edge detection algorithms.

2.1.3 Roberts cross operator

One of the earliest edge detection techniques, Larry Roberts presented a 2x2 gradient estimator as a PhD thesis in 1963 under the title of "Machine Perception Of Three-Dimensional Solids"[7]. In this algorithm, the image is convolved with $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$ and $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$. When they are put together they result in the edge detection. The square root of the sum of the convoluted images yields the gradient.[8].

The limitations of the algorithm were known and Roberts proposed a well defined edges, low noise, and high contrast[8]. These limitations cannot provide a solution for cell detection as the background environment is uncontrollably noisy because of the cell culture, and most cells do not have defined edges. The performance of the algorithm can be seen in figure 2.4.

2.1.4 Prewitt

Developed by Judith M. S. Prewitt, Prewitt algorithm convoluted the two matrices in equation 2.1, which are derived from the following partial derivatives, extract the edges of the image. [22, 6]

$$\frac{\delta I}{\delta x} \approx I(i-1,j+1)+I(i,J+1))+I(i+1,j+1)-I(i-1,j-1)-I(i,j-1)-I(i+1,j-1)$$

and,

$$\frac{\delta I}{\delta y} \approx I(i1,j-1)+I(i+1,J))+I(i+1,j+1)-I(i-1,j-1)-I(i-1,j)-I(i-1,j+1)$$

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, M_y = \begin{bmatrix} -1,-1,-1 \\ 0,0,0 \\ 1,1,1 \end{bmatrix} \tag{2.1}$$

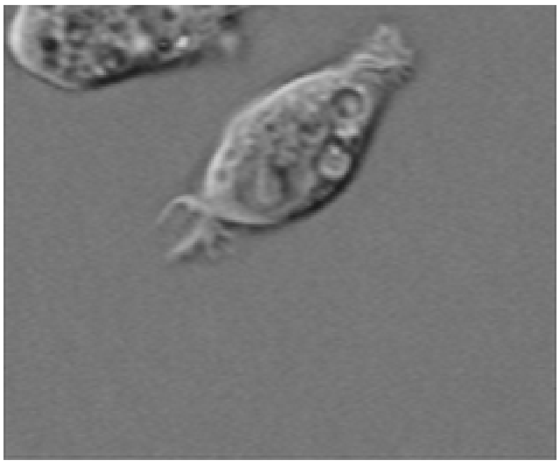
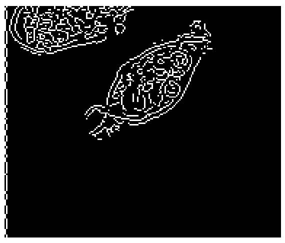
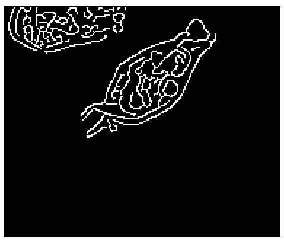


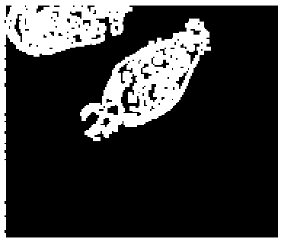
Figure 2.3: Sample cell image. Image courtesy of Alan Partin, John Hopkins University



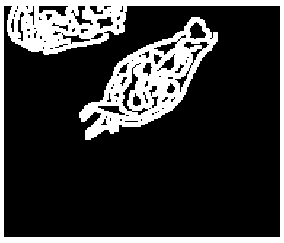
(a) Sobel binary gradient mask



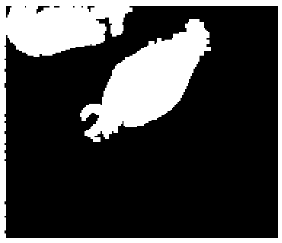
(b) Canny binary gradient mask



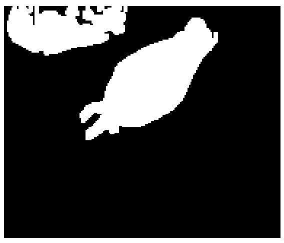
(c) Sobel dialted gradient mask



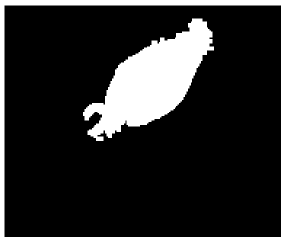
(d) Canny dialted gradient mask



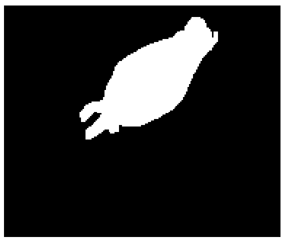
(e) Sobel with filled holes



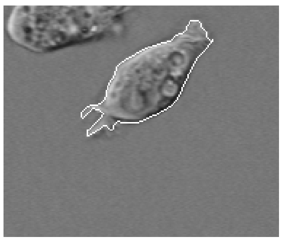
(f) Canny with filled holes



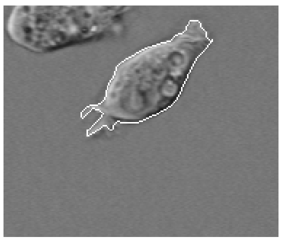
(g) Sobel binary gradient mask



(h) Canny binary gradient mask



(i) Sobel segmented and outlined cell border



(j) Canny segmented and outlined cell border

Figure 2.4: *Cell segmentation by reproducing Matlab's cell detection image processing pipeline and its comparison to replacing the Canny edge detection with Sobel edge detection in the pipeline. Image courtesy of Alan Partin, John Hopkins University.*

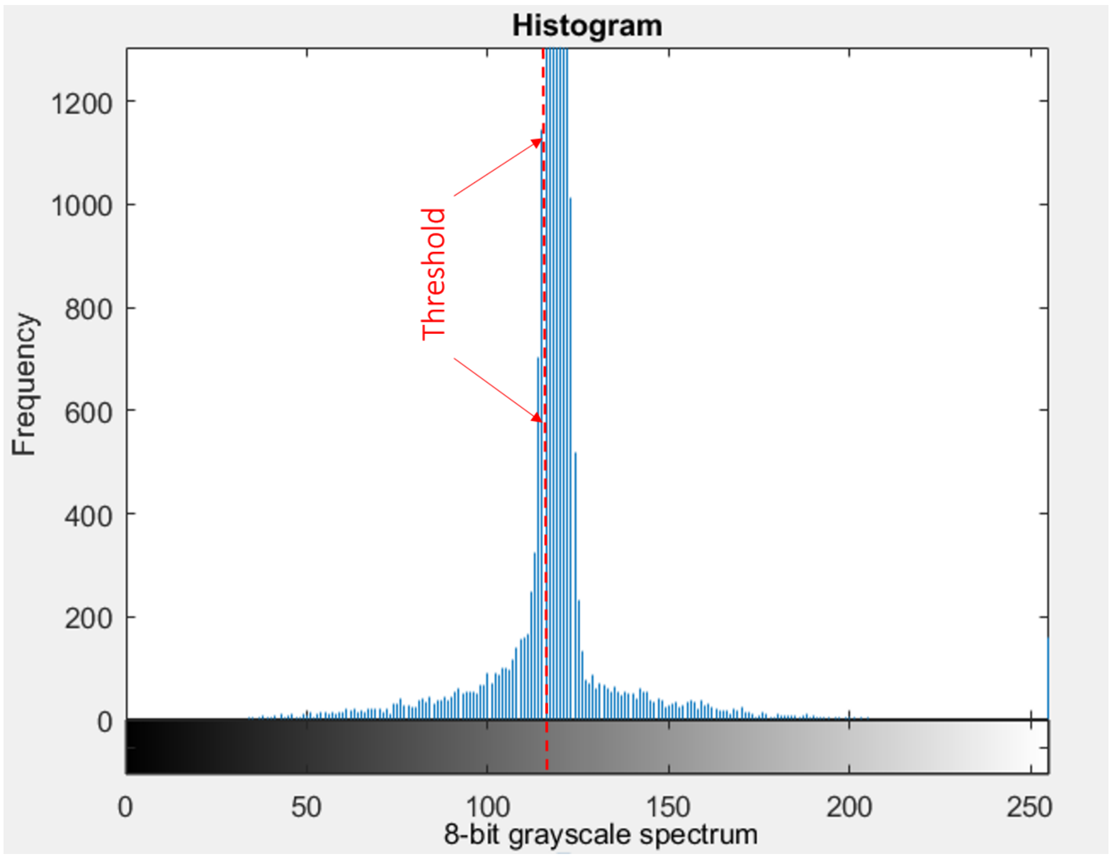


Figure 2.5: The histogram of the figure 2.3. Thresholding the background image

2.2 Object Detection

In this section we have reviewed several object recognition methods and investigated which ones are well suited for our application, cell detection. The challenges that cell identification impose include the physical structure of the cells, their relatively fast growth rate, and the environmental image capturing conditions such as the lighting and whether the cell cultures have been rotated. The literature review and our brief survey rendered many of the methods unsuitable for our research topic. In this section we have used the terms detection and identification interchangeably; these terms refer to 'single objects' and thus we no longer mean clustering. The following are a few object recognition algorithms that we have investigated:

2.2.1 Haar cascade classifier

Haar cascade is a robust machine learning algorithm that has gained significant popularity in detection algorithms. It was one the earlier face-detection algorithms that has been used by the industry. Its popularity is in part due to the fact that after the learning algorithm has produced a model using the training set, it can detect the object in real-time.

In this algorithm, Paul Viola describes that they use an "Integral Image" that allows for a rapid detector computation. The algorithm then performs AdaBoosting on a few distinctive points (like edges and points with high contrast intensity). Then classifiers are cascaded to remove the background and detecting the object.[29]

In essence the algorithm is trained by different image features and classified using AdaBoosting. Then the classifications are cascaded and the algorithm's model converges to a model that can be run in real-time to detect the same object. In order to make a robust model, many images need to be taken in different lighting conditions and from various positions. The algorithm is not rotation-agnostic, and classifications and cascading on a training set usually takes a few days.[4] However, most human cells tend to go through mitosis and even full cell cycle in less than 24 hours, not enough time for the algorithm to converge to a robust cascaded classifier. Also, it is very difficult to capture images of the cells without them being rotated since they cannot survive for more than a few minutes at room temperature. Putting cells in incubators for each trial and recapturing images from them can cause them to rotate in their petri dish.

2.2.2 Speeded-Up Robust Features (SURF)

SURF is an image processing algorithm developed by Dr. Herbert Bay et al. from Eidgenssische Technische Hochschule Zrich in 2006[2]. It analyzes the distinctive features of a source image under different perspectives, creates a descriptor for it using descriptor vectors of neighboring features, and attempts to match the descriptor vectors in the target image.

One of the distinctive characteristics of SURF is that it is rotation, scale, and contrast agnostic[12], making it an optimal algorithm for cell identification in microbiology applications where it is not possible to fixate the cells and it is difficult to maintain the contrast, light intensity, and light color temperature levels constant.

2.2.3 Scale Invariant Feature Transform (SIFT)

SIFT extracts and stores distinctive image features in a database. The features of the target picture (the image that we want to match) are compared to the stored features in the database. If enough vector distances between the features match, the algorithm stochastically determines if there is a match between the two images. The fact that the length of vectors doesn't change with rotation, and that vectors are scalable, the algorithm can perform flawlessly when the image is rotated or scaled. Feature extraction is done using Hough transform which employs a voting procedure of the local maxima in images. [13]

Chapter 3

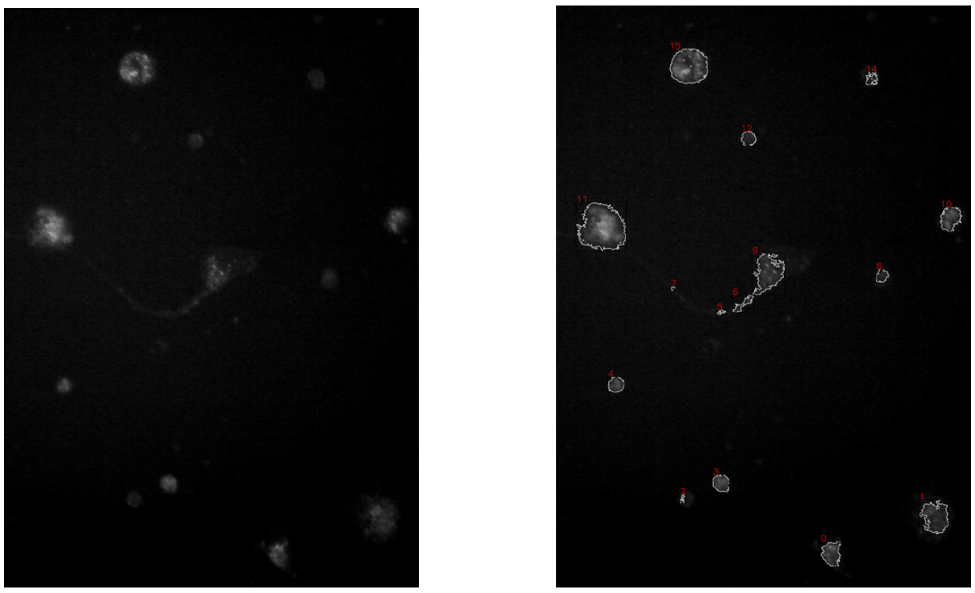
Cell Detection

3.1 Edge detection

We have developed the cell segmentation programs in both Matlab and OpenCV. However, because of the differences in the underlying programming environment and the library functions available the pipelines are slightly different.

The image processing pipeline for the OpenCV code is as follows: Since the edge detection operator can only calculate one channel, the image is converted to a binary black and white (grayscale) image to be passed to the thresholding filter that can dynamically detect the color of the background. After calculating the threshold, a foreground detection (background subtraction) method runs to detect portions of the image with "bags" of similar features (color intensity, gradient). This method is packaged in OpenCV as the Blob detection module. The blob detection algorithm detects any deviations from the thresholded background, and thus not all the foregrounds ("blobs") represent living cells as it can be seen in figure 3.4.

In cases where there is a high contrast between the background and the cells, the edge detection method performs much better as it can be seen in figure 3.1. The image was taken with a fluorescent filter.



(a) fibroblast tissues

(b) detected foregrounds

Figure 3.1: *Edge detection of an image with high contrast using a florescent filter*

The segmentation pipeline using Matlab is inspired by "Detecting a Cell Using Image Segmentation" by Mathworks. Our algorithm replaces the Sobel filter with Canny edge detection. The segmentation image processing pipeline is illustrated step by step in figure 2.4 which compares our algorithm and the one by Mathworks. After converting the RGB image to grayscale, we run the Canny edge detection to find the Canny gradient mask. By dilating the mask, we produce more robust borders. This is done by adding a white pixel to every neighboring white pixel that has resulted from the Canny gradient mask. Then we fill in the holes (i.e. the enclosed portions of the image). Since we cannot calculate the correct parameters of cells that are not entirely in a frame, we ignore them. This can be illustrated in figure 2.4(h). Using this method we can find the foreground in Matlab. The next sequences of the pipeline are identical for Matlab and OpenCV.

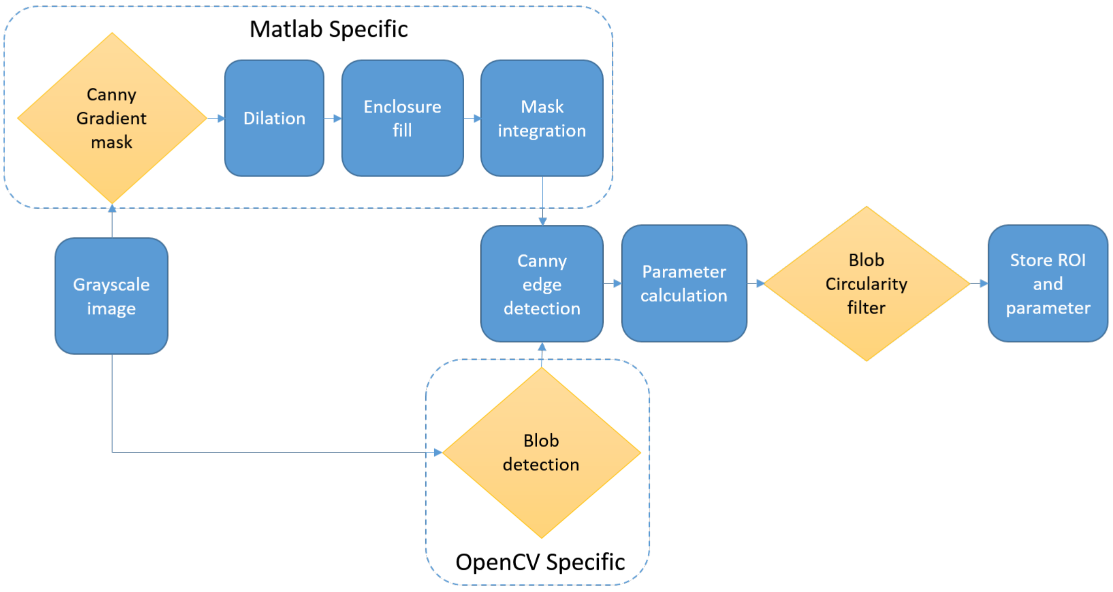


Figure 3.2: cell segmentation image processing pipeline

Both OpenCV and Matlab enable us to calculate the enclosed area of a detected segment and the perimeter of the enclosure after applying the edge detection method. Knowing that dead cells form spherical shapes, and cell flagella form elongated linear shapes, we devised a filter, that by setting the thresholds, can filter the foregrounds that are either too circular or too linear. The formula to calculate the circularity is as follows:

$$Circularity = \frac{4A}{P^2} \tag{3.1}$$

Where A is the area enclosed by the blob, and P is its perimeter. The closer the Circularity is to one, the more circular the image is and the closer it is to 0 the more linear it is. We empirically chose a threshold range between 0.5 and 0.8 to eliminate the cells and cell flagella. Figure 3.3 shows the post-processed and filtered cells. Note that although the living cell recognition has improved drastically, there are still false positives and false negatives in the picture. The algorithm does not perform well when images are out of focus because the blur causes the edge detection to fail finding the edges.

Finally by calculating the cell's physical parameters (areas and perimeters) and eliminating some of the false positives, we enumerate the blobs, store their physical parameters in a Comma Separated Value (CSV) file, and store each cell's image as a Region of Interest (ROI) in the local drive.

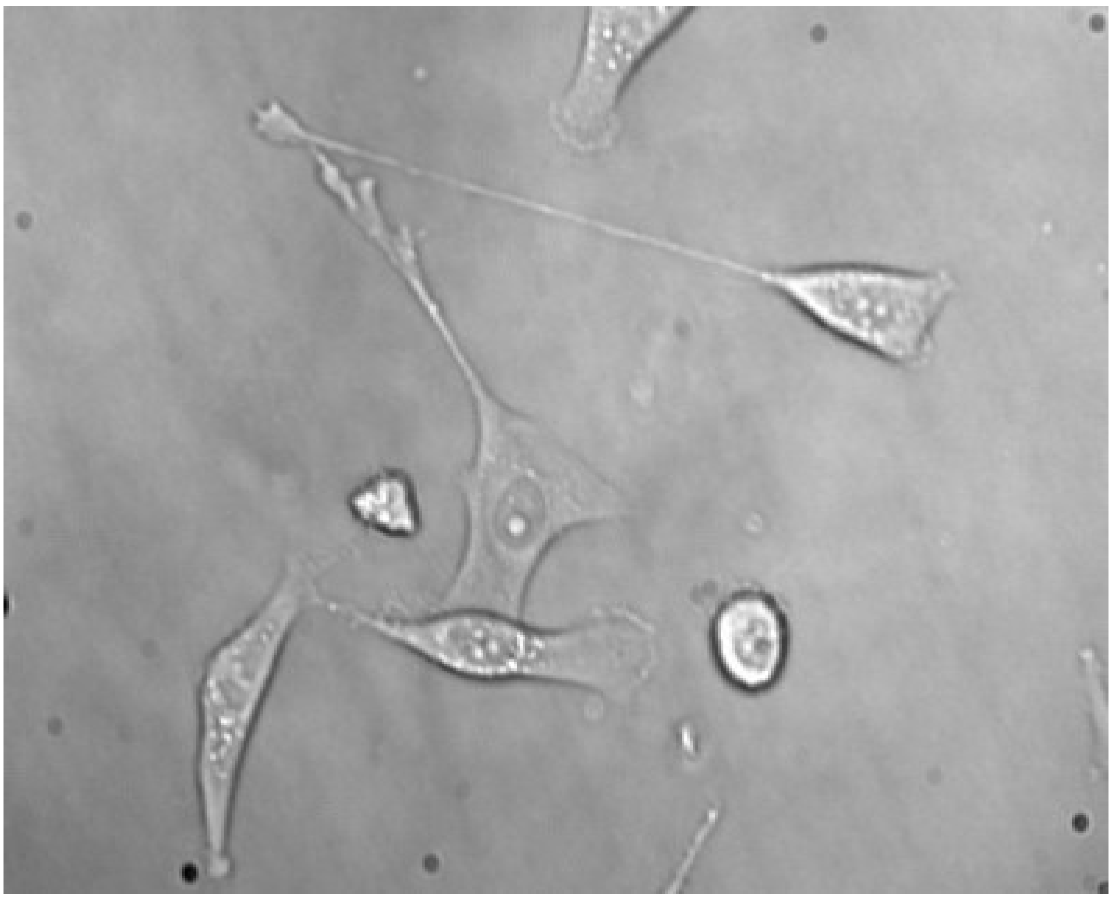


Figure 3.3: Image of HeLa cell culture. Image courtesy of Akshar Lolith from University of California, Santa Cruz

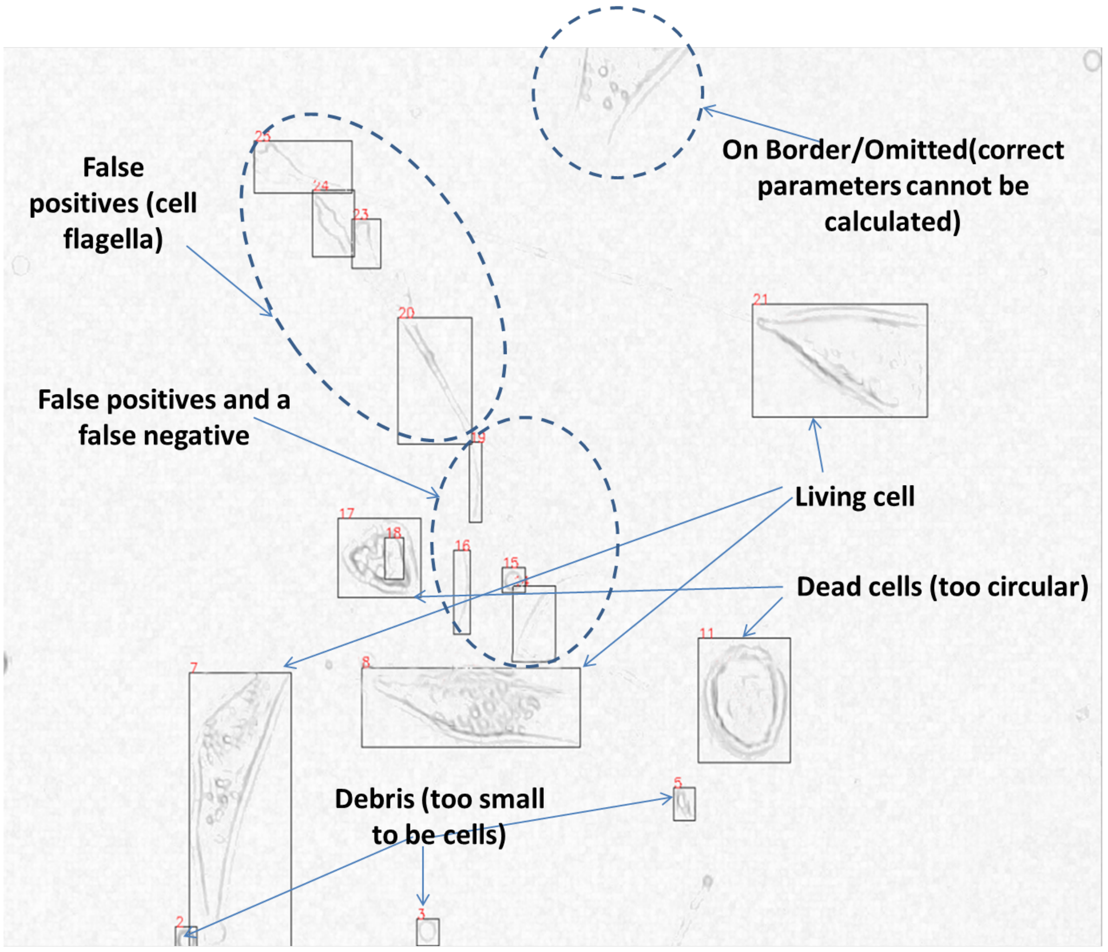

False positives (cell flagella)
On Border/Omitted(correct parameters cannot be calculated)
False positives and a false negative
Living cell
Dead cells (too circular)
Debris (too small to be cells)

Figure 3.4: pre-processed foreground detection. The false positives and false negatives are indicated in this picture

Table 3.1: Detected cell's physical parameters and center point coordinates

Cell ID	Area	Perimeter	x-center	y-center
1	1414	1131	317	766
2	1381	1096	584	682
3	586	406	467	531
4	1264	1010	1039	328

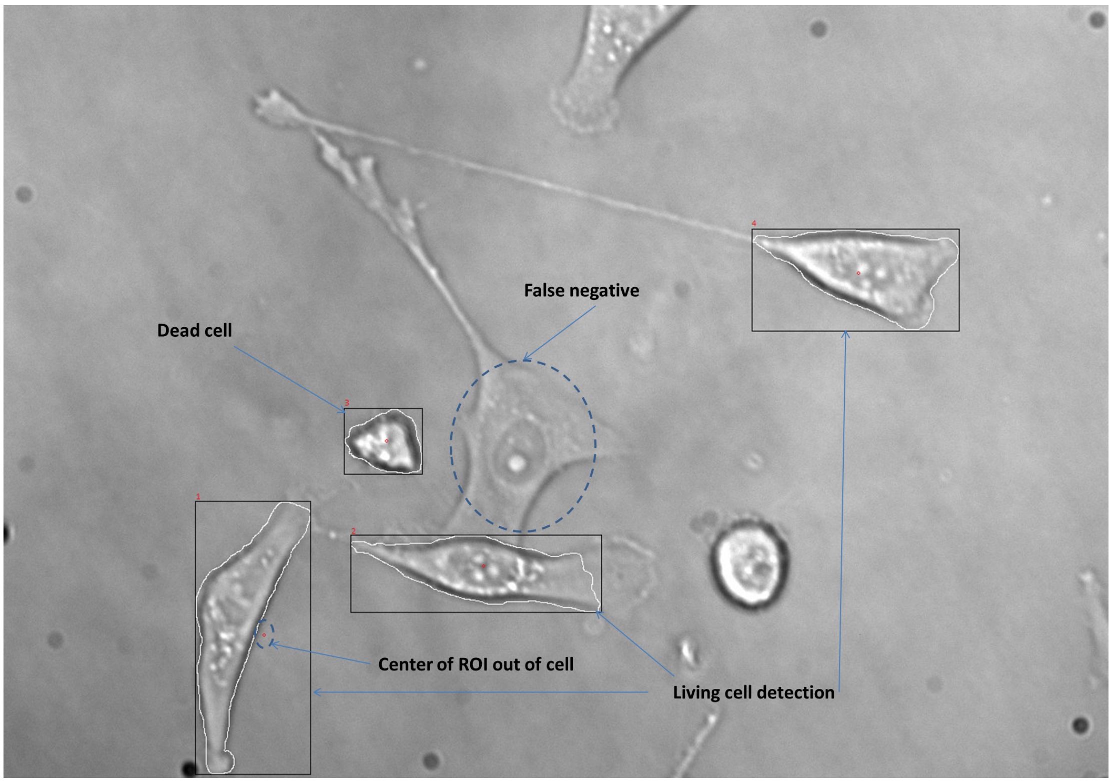


Figure 3.5: Processed and filtered detected cells. There are still a few false positives and false negatives. One of the reasons for the false negative might be because it is not in focus and there is another cell overlapping

Chapter 4

Single Cell Identification

4.1 Speeded-Up Robust Features (SURF)

One of the challenges in utilizing the SURF machine learning algorithm is that there are not many distinctive features in a pool of different cells. The cells are seen as blobs of similar colors, often with similar curvatures, and without many distinctive features, which makes feature extraction more difficult than macro-scale objects. However, in our implementations, we have been able to successfully distinguish between cells and identify the target cell by reducing the number of features required to match the images. Figure 4.2 illustrates a cell detection using the SURF algorithm. The algorithm found only one matching point but was able to find the Region Of Interest (ROI) in the image.

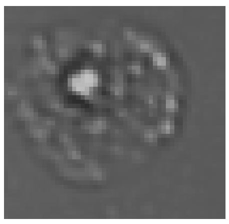


Figure 4.1: Sample cell (ROI)

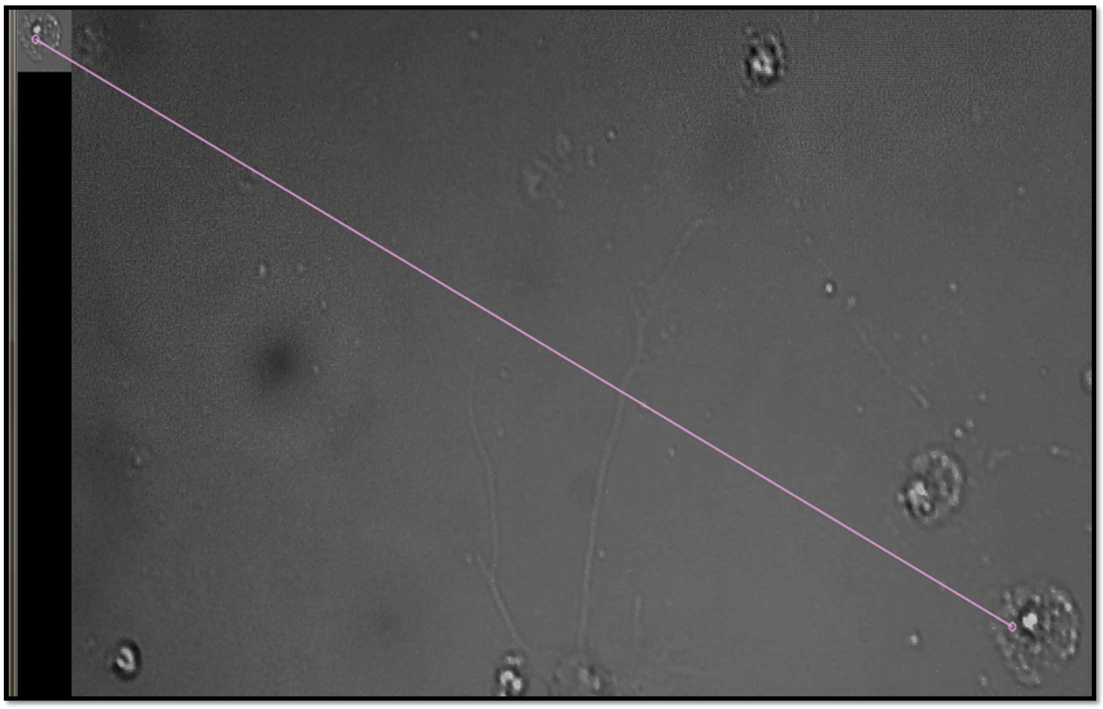


Figure 4.2: Detection and matching of a ROI using SURF

4.2 Scale Invariant Feature Transform (SIFT)

The SIFT algorithm provides many more matching points, increasing the likelihood of a detection, and although that might increase the chance of finding a match, it also increases the chance of a false positive. Moreover, SIFT is much slower than the SURF algorithm. Because of the algorithm's higher likelihood of finding a match, if in an iteration SURF cannot find a match, SIFT is run. Finally, if SIFT can not find a match, we conclude the search without a matched cell. Figure 4.3 Illustrates a cell detection using the SIFT algorithm. It can be observed that the number of the matches has increased significantly compared to the detection performed by the SURF algorithm (in figure 4.2).

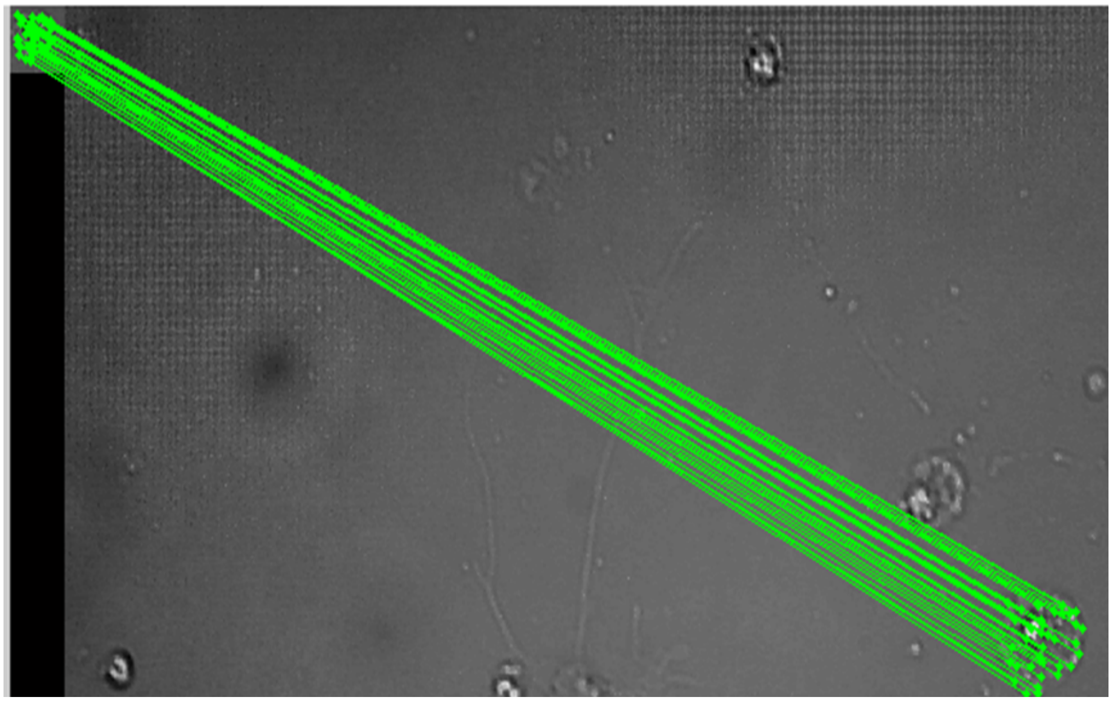


Figure 4.3: Detection and matching of a ROI using SIFT

4.3 Descriptor Outlier and Mismatch Rectification

In order to rectify the mismatches and increase the accuracy of the SURF and SIFT descriptors, we investigated between the Fast Approximate Nearest Neighbor(FLANN)[18] which is a variation of the nearest neighbor algorithm and Random sample consensus (RANSAC)[9], based on Fischler's discriminator method. We found out that RANSAC provides better results while FLANN is a faster method for finding and dismissing the mismatches. We chose to run FLANN with the SURF and SIFT descriptor algorithms.

4.4 Comparisons and survey results

SURF is the fastest descripting algorithm that can be used for matching images and although SIFT performs better in terms of finding the matching features, when it comes to rotation tolerance, SURF is more robust when the illumination changes, and is more than 4.5 orders of magnitude faster in finding matching images [11]. A more comprehensive comparison is shown in table 4.1. Inspired by [11] we used K Nearest Neighbor to perform the matches and to dismiss the outliers from the correct matches.

Table 4.1: Comparison of matching algorithms

Method	**Rotation**	**Scale**	**Time**	**Blur**	**Illumination**
SIFT	best	very good	good	best	fair
Haar Cascade	very poor	very good	very slow training but fast detection	good	good
SURF	very good	good	best	good	best

4.5 Time-Lapsed Calculation of Cell Growth

In order to find cells in an image, we first perform image processing to detect the foreground in the image. After applying a circularity filter and performing a foreground size thresholding we reduce the number of false positives and eliminate the dead cells from the rest of the calculation to work only with the living cells. Each living cell is indexed and its physical parameters are calculated using edge detection. We draw an outline around each detected cell and store the enclosed content (image of the detected cell) as an image in a database.

To find the cell growth, we use the matching algorithms to determine which ROI corresponds to the secondary set of images. We first run the SURF algorithm which is very fast and illumination tolerant and if SURF can not find the matches, we run the SIFT algorithm to detect the corresponding cell. Because cells grow rapidly and divide, the images are only compared to the previous iteration of the algorithm.

We can find the difference in the physical parameters of each cell in each iteration and thus calculate the cell growth. Figure 4.4 illustrates the chain process of calculation of rate of growth of the cell culture.

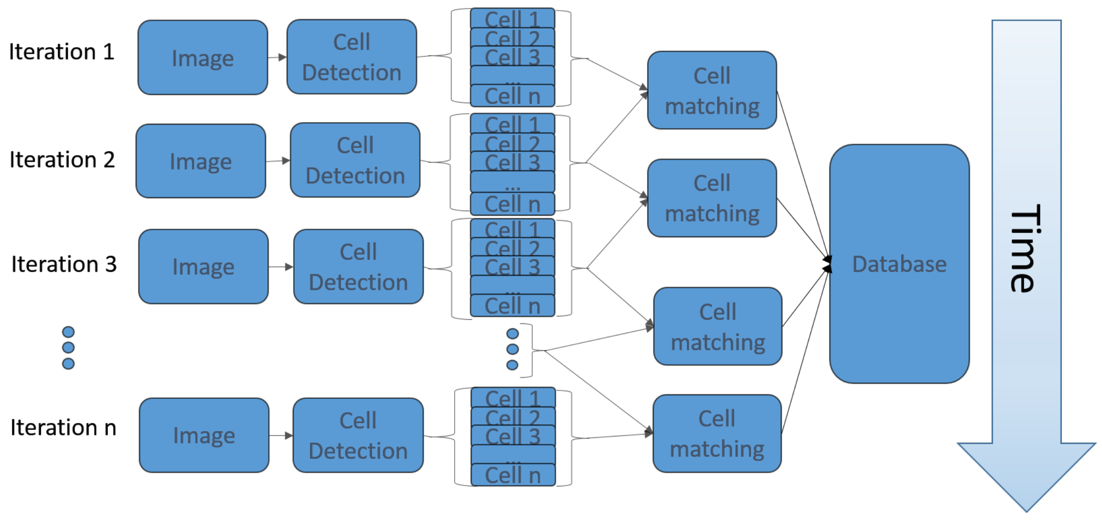


Figure 4.4: Time-lapsed cell parameter calculation.

Chapter 5

Automation

In order to perform cell injections using the cell segmentation, we attempted to perform cell aspiration/injection on two different micro-manipulator instruments, the Sutter Instrument Company's MP-285 manipulator and the BioStinger, a manipulator unit designed by Adam Seger, a UCSC alumnus. Both devices provide supervised and manual cell injection and aspiration. The user calibrates the device under each focus and can click on the cells for the end effector to move to the position under the semi-automatic injection. Under automatic injection, the cell segmentation provides the center points of the bounding rectangle of the cell and the software can inject those points[1] .

[1]Unfortunately although every part of the underlying software has been developed and tested in an isolated manner, due to various hardware shortcomings, automated cell injection has not been tested.

5.1 Mapping and Calibration

To automate the system, we implemented a pixel to x and y coordinate mapping. Using this mapping scheme, the X and Y components of each pixel on the screen gets mapped to the relative x and y of the last position of the microcontroller end effector. This mapping and calibration process has to be executed every time the focus objectives changes. It runs when the program starts, which also provides a compensation for the accumulated stepper motor's missed steps, thus increasing the accuracy of the injection. Equation 5.1 and 5.2 shows the pixel to the location mapping scheme where a and b are the mapping factors for x and y coordinates, respectively, and x and y denote relative position of the end effector while X and Y represent the pixel seen on the screen. Figure 5.1 illustrates a screen shot of the calibration interface. The user is instructed to move the end effector to the top left target and click on it and then move it to the bottom right target and click on it. The mapping factors are populated and when the user clicks on any new point on the screen, the mapping factor is multiplied by the pixel coordinates.

$$\Delta x = a(\Delta X) \tag{5.1}$$

and

$$\Delta y = b(\Delta Y) \tag{5.2}$$

Because different magnification objectives can change the mapping and to ensure that the drift associated with the stepper motors has a lower impact in the automated cell injection process, the mapping and calibration are done every time when the program starts.

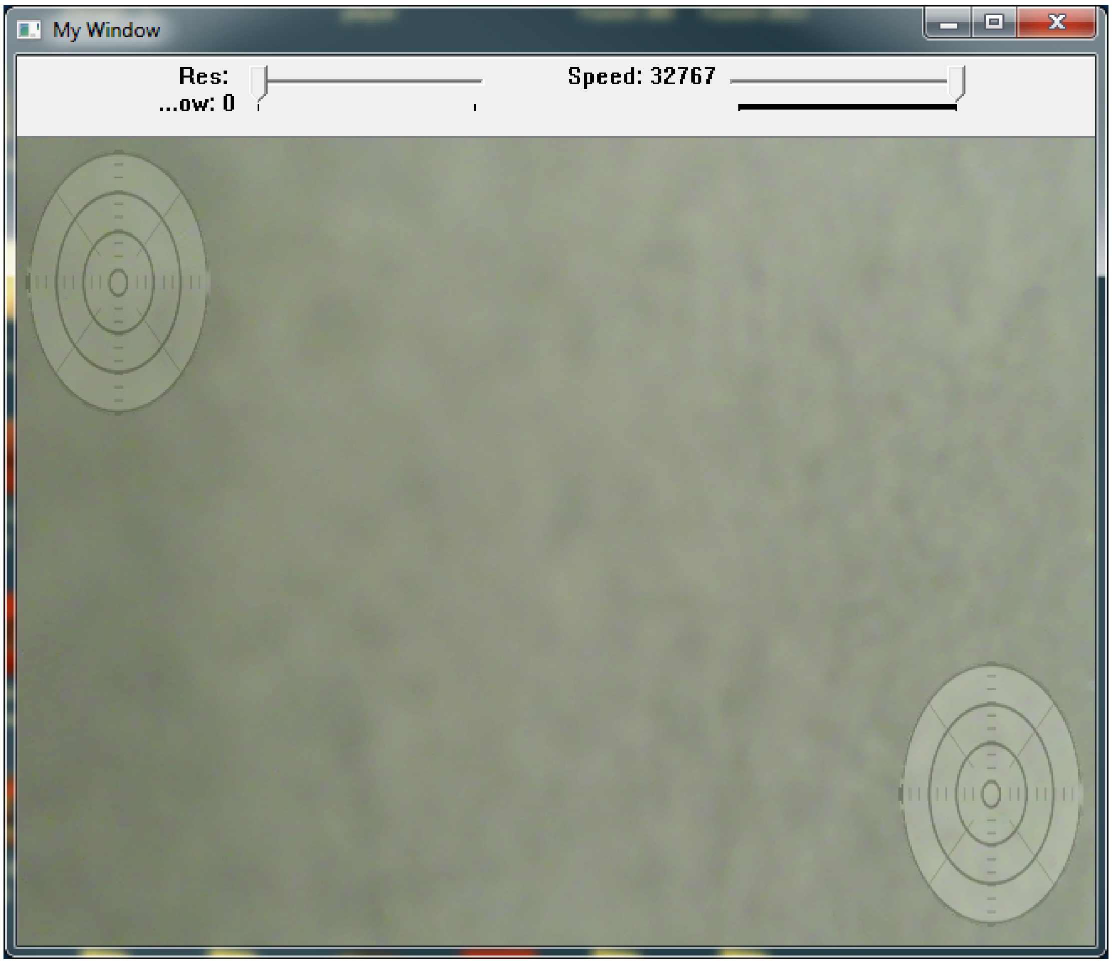


Figure 5.1: Calibration and mapping stage

5.2 Sutter Instrument LabVIEW Integration

We integrated our automated injection code that has been written in C++ (in conjunction with the cell detection module that was written using OpenCV) to the

LabVIEW driver that was provided by Sutter Instrument Company.

In order to run the OpenCV code and do the pixel to position mapping, we communicated using the TCP-IP protocol. In this communication scheme, the C++ detection and calculation program works as the server while the LabVIEW Virtual Instrument (VI) is the client. The Sutter Instrument MP-285 uses the RS-232 Serial interface to communicate with the PC. The MP-285 had various bugs that did not allow for the automatic or manual injection as was advertised. Trying to resolve the problems with the Sutter Instrument Company and them upgrading the VIs several times did not fix these issues.[1, 23]

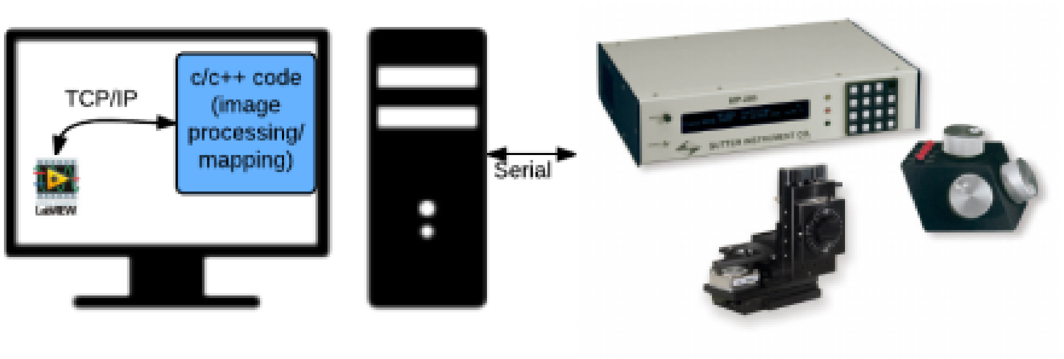


Figure 5.2: Serial interface using LabVIEW driver for MP-285 for automatic cell injection and aspiration

5.3 C/C++ Driver using MP-285 Micro-manipulator Unit

After several failed attempts at making the supplied VI performing the automatic injection, despite VI upgrades, we could not make a significant improvement in moving the manipulator using serial communication. We integrated the serial commands in our C++ code. In this scheme, there was no need for any communication protocols other than the communication between the MP-285 and the computer.

Although we were able to send short commands such as resetting the position and changing the velocity, we have not been able to move the manipulator. After several discussions, we and our point-of-contact with Sutter Instrument Company concluded that the bugs relate to the firmware of the MP-285. [1, 23]

We modified our code to generate a text file with the pixel coordinates of the center points of the bounding rectangles (with the respect to the screen resolution). Scott Rad, a perspective graduate student at UCSC, has been working on the BioStinger unit and he has written a TCP-IP server-client program to read the center points to inject cells.

Chapter 6

Conclusion and Future Work

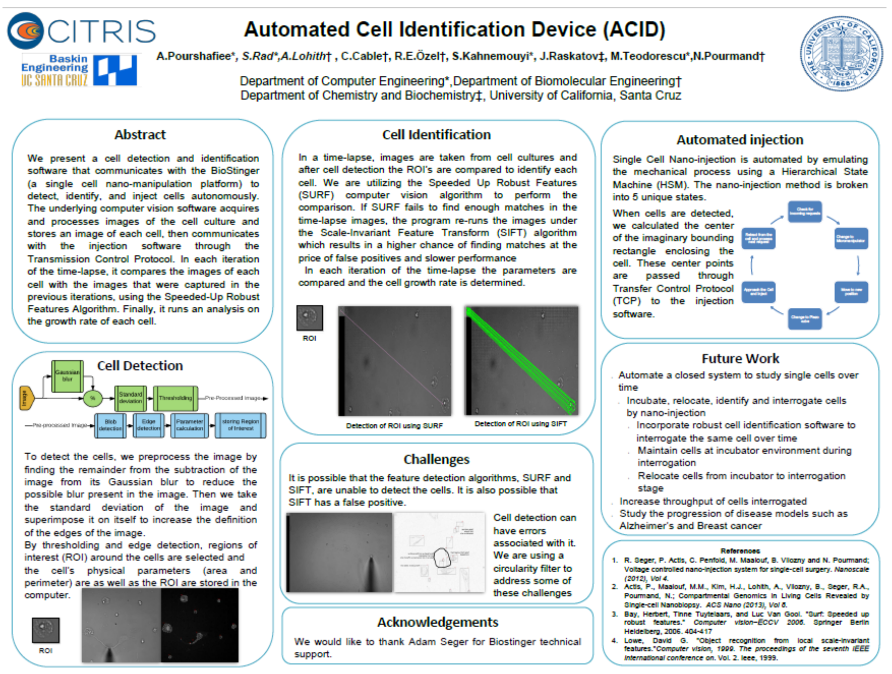
In this thesis, we have been able to use machine learning and computer vision methods to segment cells and detect cells individually. Our research involved reviewing various literature and resources and performing trade studies and surveys of the available methods. We implemented filters to distinguish cells from other objects in the foreground. After segmenting the cells and storing the images of each individual cell, we perform a SURF descriptor detection to detect the cells in the new scenario (in a few hours) and if the Surf detector is unable to find matches, we run the SIFT algorithm to find the matches. When the match is found the physical parameters of a cell is compared over time to update the growth rate of the cell.

We also attempted to perform automated cell injection using micro-manipulators and an injection piezo-cube. However, we have not completed due to limitations posed by the hardware and its associated firmware. We provided a solution by outputting the center points as a text file, to enable future projects to be able to work on the injection phase as an isolated problem and integrate it with the cell detection. This also makes the cell detection hardware agnostic and allows for expansion in the future.

Appendix A

CITRIS Day Poster, Oct. 2015

Berkeley, CA



Automated Cell Identification Device (ACID)

A.Pourshafiee*, *S.Rad*,A.Lohith*† , C.Cable†, R.E.Özel†, S.Kahnemouyi*, J.Raskatov‡, M.Teodorescu*,N.Pourmand†

Department of Computer Engineering*,Department of Biomolecular Engineering†
Department of Chemistry and Biochemistry‡, University of California, Santa Cruz

Abstract

We present a cell detection and identification software that communicates with the BioStinger (a single cell nano-manipulation platform) to detect, identify, and inject cells autonomously. The underlying computer vision software acquires and processes images of the cell culture and stores an image of each cell, then communicates with the injection software through the Transmission Control Protocol. In each iteration of the time-lapse, it compares the images of each cell with the images that were captured in the previous iterations, using the Speeded-Up Robust Features Algorithm. Finally, it runs an analysis on the growth rate of each cell.

Cell Detection

To detect the cells, we preprocess the image by finding the remainder from the subtraction of the image from its Gaussian blur to reduce the possible blur present in the image. Then we take the standard deviation of the image and superimpose it on itself to increase the definition of the edges of the image.
By thresholding and edge detection, regions of interest (ROI) around the cells are selected and the cell's physical parameters (area and perimeter) are as well as the ROI are stored in the computer.

Cell Identification

In a time-lapse, images are taken from cell cultures and after cell detection the ROI's are compared to identify each cell. We are utilizing the Speeded Up Robust Features (SURF) computer vision algorithm to perform the comparison. If SURF fails to find enough matches in the time-lapse images, the program re-runs the images under the Scale-Invariant Feature Transform (SIFT) algorithm which results in a higher chance of finding matches at the price of false positives and slower performance
In each iteration of the time-lapse the parameters are compared and the cell growth rate is determined.

Challenges

It is possible that the feature detection algorithms, SURF and SIFT, are unable to detect the cells. It is also possible that SIFT has a false positive.

Cell detection can have errors associated with it. We are using a circularity filter to address some of these challenges

Acknowledgements

We would like to thank Adam Seger for Biostinger technical support.

Automated injection

Single Cell Nano-injection is automated by emulating the mechanical process using a Hierarchical State Machine (HSM). The nano-injection method is broken into 5 unique states.

When cells are detected, we calculated the center of the imaginary bounding rectangle enclosing the cell. These center points are passed through Transfer Control Protocol (TCP) to the injection software.

Future Work

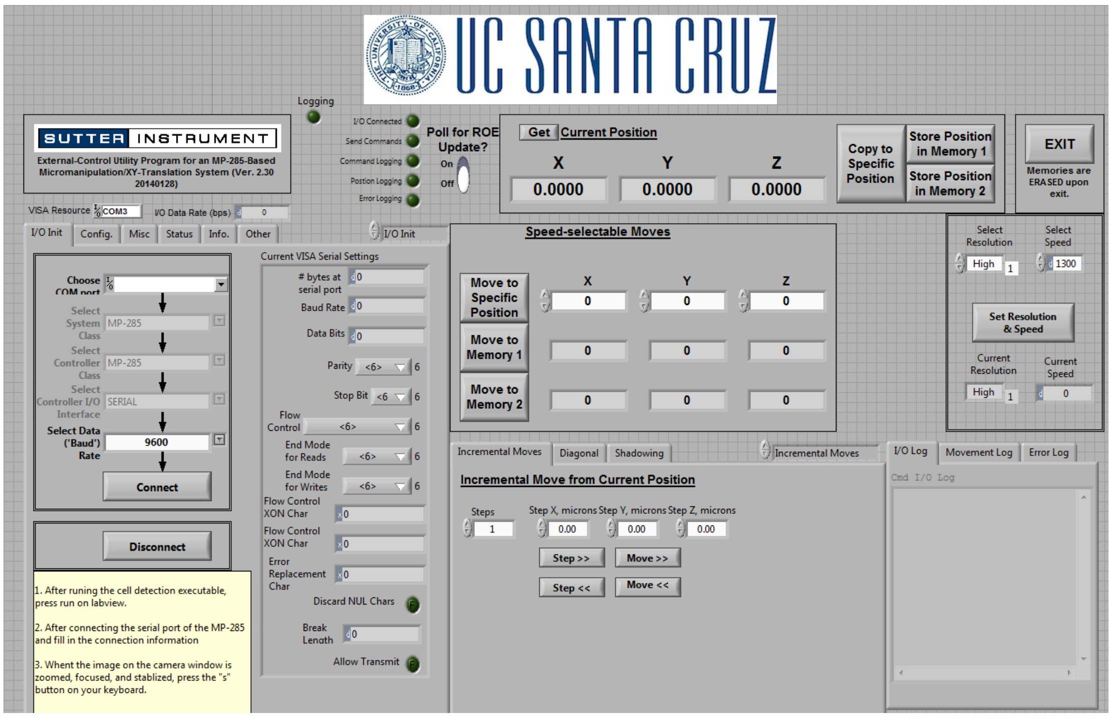
- Automate a closed system to study single cells over time
  - Incubate, relocate, identify and interrogate cells by nano-injection
    - Incorporate robust cell identification software to interrogate the same cell over time
    - Maintain cells at incubator environment during interrogation
    - Relocate cells from incubator to interrogation stage
- Increase throughput of cells interrogated
- Study the progression of disease models such as Alzheimer's and Breast cancer

References

1. R. Seger, P. Actis, C. Penfold, M. Maalouf, B. Vilozny and N. Pourmand; Voltage controlled nano-injection system for single-cell surgery. *Nanoscale (2012), Vol 4.*
2. Actis, P., Maalouf, M.M., Kim, H.J., Lohith, A., Vilozny, B., Seger, R.A., Pourmand, N.; Compartmental Genomics in Living Cells Revealed by Single-cell Nanobiopsy. *ACS Nano (2013), Vol 8.*
3. Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." *Computer vision–ECCV 2006.* Springer Berlin Heidelberg, 2006. 404-417
4. Lowe, David G. "Object recognition from local scale-invariant features."*Computer vision, 1999. The proceedings of the seventh IEEE international conference on.* Vol. 2. Ieee, 1999.

Appendix B

Sutter Instrument MP-285 LabVIEW Interface



SUTTER INSTRUMENT
External-Control Utility Program for an MP-285-Based Micromanipulation/XY-Translation System (Ver. 2.30 20140128)

VISA Resource: COM3
I/O Data Rate (bps): 0

Choose COM port:
Select System Class: MP-285
Select Controller Class: MP-285
Select Controller I/O Interface: SERIAL
Select Data ('Baud') Rate: 9600

Current VISA Serial Settings
# bytes at serial port: 0
Baud Rate: 0
Data Bits: 0
Parity: <6> 6
Stop Bit: <6 6
Flow Control: <6> 6
End Mode for Reads: <6> 6
End Mode for Writes: <6> 6
Flow Control XON Char: 0
Flow Control XON Char: 0
Error Replacement Char: 0
Discard NUL Chars: F
Break Length: 0
Allow Transmit: F

1. After runing the cell detection executable, press run on labview.

2. After connecting the serial port of the MP-285 and fill in the connection information

3. Whent the image on the camera window is zoomed, focused, and stablized, press the "s" button on your keyboard.

Get Current Position
X: 0.0000
Y: 0.0000
Z: 0.0000

Poll for ROE Update? On / Off

Memories are ERASED upon exit.

Speed-selectable Moves

	X	Y	Z
Move to Specific Position	0	0	0
Move to Memory 1	0	0	0
Move to Memory 2	0	0	0

Select Resolution: High 1
Select Speed: 1300
Current Resolution: High 1
Current Speed: 0

Incremental Move from Current Position
Steps: 1
Step X, microns: 0.00
Step Y, microns: 0.00
Step Z, microns: 0.00

Cmd I/O Log

Figure B.1: LabVIEW Interface

Bibliography

[1] Paolo Actis, Michelle M Maalouf, Hyunsung John Kim, Akshar Lohith, Boaz Vilozny, R Adam Seger, and Nader Pourmand. Compartmental genomics in living cells revealed by single-cell nanobiopsy. *ACS nano*, 8(1):546–553, 2013.

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[3] Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

[4] Geoff Budd, Amir Pourshafiee, Sina Kahnemouyi, Samir Mohammed, Tuan Ho, and Leo Bravo. *UCSC Autonomous Rover*. Bachelor thesis, University of California, Santa Cruz, 2013.

[5] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[6] Suxia Cui, Yonghui Wang, Xiaoqing Qian, and Zhengtao Deng. Image processing techniques in shockwave detection and modeling. *Journal of Signal and Information Processing*, 4(03):109, 2013.

[7] Per-Erik Danielsson and Olle Seger. Generalized and separable sobel operators. *Machine vision for three-dimensional scenes*, pages 347–379, 1990.

[8] Larry S Davis. A survey of edge detection techniques. *Computer graphics and image processing*, 4(3):248–270, 1975.

[9] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] Rafika Harrabi and Ezzedine Ben Braiek. Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images. *EURASIP Journal on Image and Video Processing*, 2012(1):1–11, 2012.

[11] Luo Juan and Oubong Gwun. A comparison of sift, pca-sift and surf. *International Journal of Image Processing (IJIP)*, 3(4):143–152, 2009.

[12] Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 501–506. IEEE, 2011.

[13] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[14] Raman Maini and Himanshu Aggarwal. Study and comparison of various image edge detection techniques. *International journal of image processing (IJIP)*, 3(1):1–11, 2009.

[15] Chris Messom and Andre Barczak. Fast and efficient rotated haar-like features using rotated integral images. In *Australian Conference on Robotics and Automation*, pages 1–6, 2006.

[16] Thomas B. Moeslund. Canny edge detection. 2009.

[17] Amaleena Mohamad, Noorain A Jusoh, Lei Win Shoon, et al. Bacteria identification from microscopic morphology: a survey. *International Journal on Soft Computing, Artificial Intelligence and Applications (IJSCAI)*, 3(2):1–12, 2014.

[18] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[19] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2:331–340, 2009.

[20] Marius Muja and David G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410, 2012.

[21] Jim R Parker. *Algorithms for image processing and computer vision.* John Wiley & Sons, 2010.

[22] Judith MS Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[23] R Adam Seger, Paolo Actis, Catherine Penfold, Michelle Maalouf, Boaz Vilozny, and Nader Pourmand. Voltage controlled nano-injection system for single-cell surgery. *Nanoscale*, 4(19):5843–5846, 2012.

[24] R Adam Seger, Paolo Actis, Catherine Penfold, Michelle Maalouf, Boaz Vilozny, and Nader Pourmand. Voltage controlled nano-injection system for single-cell surgery. *Nanoscale*, 4(19):5843–5846, 2012.

[25] Irwin Sobel. History and definition of the so-called "sobel operator",moreappropriately named the sobel-feldman operator. *Research Gate*, 2015.

[26] Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *a talk at the Stanford Artificial Intelligence Project (SAIL)*, 1968.

[27] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[28] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[29] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001.*

Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I–511. IEEE, 2001.