Automatic image analysis and extraction of cell cycle parameters in stem cells with FUCCI expression

J. Yu (jy380@cam.ac.uk), University of Cambridge, Thesis for Computational Biology MPhil

Project Supervisor: Pietro Cicuta, Course Director: Stephen Eglen

August 8, 2017

Contents

Word count: 10469

List of Figures

Abstract:

Fluorescent Ubiquitination-based Cell Cycle Indicator (FUCCI) can help characterise the cell cycle phase according to red and green fluorescence signals. Here, we design a customized automatic image analysis system to help us efficiently characterise the cell phase with FUCCI expression. We successfully extract the relevant parameters from a large amount of imaging data by using a mask-based segmentation technique. We use a digitalized method to compare the quality of the cell mask generated by using different combination algorithms. The combination algorithms are composed of a personalized imaging segmentation procedure combining different filters and edge detection tools. The analysed results from the automated method are evaluated by comparing against a manual selection method. The results show that the fluorescence signal curve fitting generated by automated method matches well with the manual approach, especially at the peak regions of the red fluorescence signals.

I Introduction:

I .1 FUCCI reporter system

Fluorescent Ubiquitination-based Cell Cycle Indicator (FUCCI) is a set of fluorescent probes which enables the visualisation of cell cycle progression in living cells. [1] The cell cycle refers to the different stages in the life of the cell. There are two stages of the cell cycle: interphase and mitosis (depicted in Figure 1 [1]). Interphase is the longest stage of the cell cycle. It can be broken down into three phases: G1, S and G2. Each phase carries out a specific function. G1 is the first growth phase where the cell produces organelles, such as ribosomes and proteins [4]. The S phase stands for synthesis in which DNA synthesis, DNA replication and chromosome duplication occur. G2 is the second growth phase where the cell is preparing for mitosis. Mitosis (M phase) is when active cell division occurs. After M phase, each of the daughter cells will start the cell cycle again.

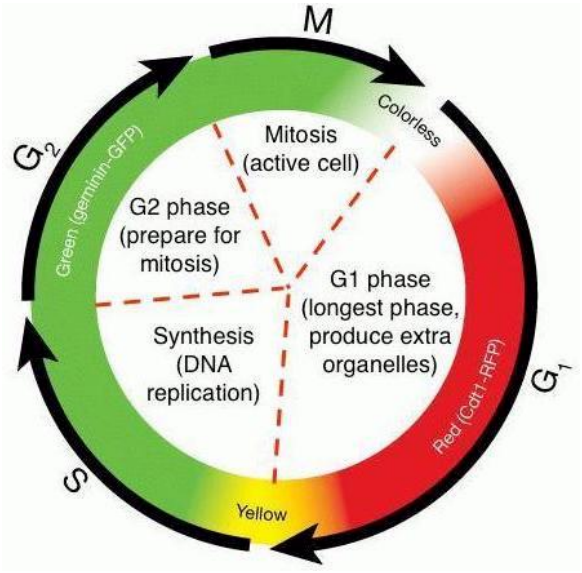

Figure 1: Dynamic color change of the FUCCI Cell Cycle with general procession[1]

FUCCI cells are especially well-suited for real-time spatio-temporal imaging of their cell cycle dynamics [6, 7, 22]. In particular, we can measure the time each cell spends in each phase. The FUCCI reporter system uses two distinct fluorescence-proteins to visualise the cell cycle. mKO2-Cdt1 emits a red signal during the G1 phase. Once the cell enters the S phase, it produces mAG-Geminin which emits a green signal and deactivates mKO2-Cdt1. The cell deactivates the mAG-Geminin once the G1 phase starts anew [4]. These properties of the FUCCI cells help us figure out which phase the cell is in.

[1] Figure 1 is adapted from the manual of the Premo FUCCI Cell Cycle Sensor, ThermoFisher Scientific, re-edited by J.Yu.

I .2 Research overview

The stem cell division rate, population and phase duration are important quantities in biology research. They help predict cellular properties, such as differentiation path, throughout its life. [14, 17]. The motivation of the project is to study the cell cycle phase (G1, S, G2 and M) durations in mouse embryonic stem (mES) cells at single cell resolution using the FUCCI reporter system. Cell cycle dynamics during cell fate changes have not yet been directly measured in the mES cell literature. Thus no clear understanding of the cell cycle progression and the execution of cell fate choices has been established. [14] Our experiment introduces an imaging program developed in-house for multiple hours tracking and imaging of single stem cells with high throughput.

I .3 Research objectives

In this report, we describe a new automated program which combines filtering and edge-detection algorithms to measure the fluorescence of each cell individually. The program automatically analyses the data to compute the duration of each cell cycle phase. The results are evaluated by comparing against the manual approach. In summary, the purpose of designing this automated image analysis system includes:

- developing a new automated image analysis program to better segment individual cells
- extracting relevant parameters from the analysis to characterise cells in each of their phases.

II Automatic image analysis:

II .1 Purpose of design:

Building an automatic image analysis system is important because a large amount of imaging data is generated from the experiment. Each FUCCI experiment will require at least 6-7 days to finish (2 days for cell preparation, 2-3 days for imaging and 2 days for data collection). Finally, all imaging data will be stored in a database for analysis. However, like other experiments, the FUCCI experiment does not always get perfect data from the test. This encouraged us to build an automatic analysis method to help us perform the analysis of the imaging data so that we can obtain the useful data more efficiently. Currently, more than five Terabytes (TB) of imaging data is stored on our server. Dealing with all this data by hand is impossible, hence in this section, we explain the design of our automatic image analysis program which helps us efficiently deal with the large amount of data. In addition, building this automatic program can also reduce user bias to make things more standard across different laboratories.

II .2 Data preparation:

II .2.1 Experiment

The FUCCI experiment[2] required maintaining cells at a healthy state to acquire useful information for the cell dynamics study. We also prepared a high-resolution imaging microscope which is essential for improving the accuracy of the experiment. In our experiment, a microscope platform compatible incubator, a robust gas regulating and feeding device were used to image mES cells with desired optical quality. In order to overcome the challenge of phototoxicity in the long-term single cell imaging experiment, we adjusted the shuttle speed of the fluorescent light exposure and tried to minimise the total duration and amount of light exposure to the cell body. We also extended the time intervals between each fluorescent exposure to further reduce the exposure of excitation light.

[2]The experiment is mainly operated by W. Zhai (the PhD student in our group) and collaborate with the Wellcome Trust Sanger Institute.

Finally, our imaging software system allows adjustment in the fluorescent imaging interval timings, ensuring flexibility in adjusting to different fluorescent signal output requirement to address the biological questions on quantifying the cell dynamics of mES cells. All the imaging data are uploaded to the centre server of Biological and Soft System (BSS Group) in Rutherford building at Cavendish Laboratory on West Cambridge site. The photos of experiment set up are shown in additional figures in **Appendix II**.

II .2.2 Channels in '.movie' file

The original imaging data is stored in a customised format file called '.movie' file. A series of '.movie' files were stored in a folder to record the data for each set of experiments. Each '.move' file is the recording from a particular field of view called a 'frame'. Each movie folder contains around 100-200 frames and each frame contains 9 channels (shown in Figure 2). One frame is collected in 15 minutes, hence an entire movie folder takes 25-50 hours to record.

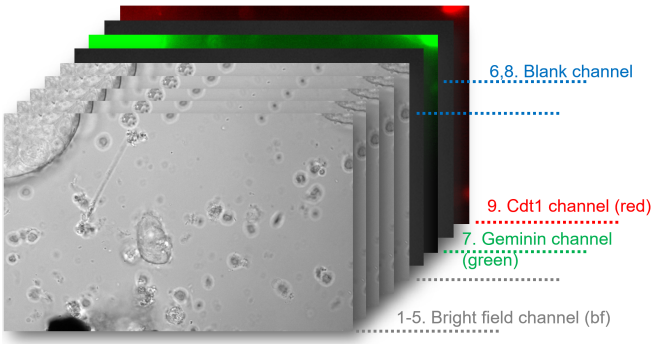

Figure 2: Channels in a frame ('.movie' file)

In each frame, channels 1 through 5 are bright field (bf) channels, which contain an image of the stem cell taken though an optical microscope. Channels 1 through 5 are captured sequentially in a very short timespan with the intent of selecting the best image for analysis. Channel 6 is captured under no lighting to serve as a background calibration for the green signal (channel 7), which records the signal for 'mAG-Geminin'. Channel 8 is captured under no lighting to serve as a background calibration for the red signal (channel 9),

which records the signal for 'mKO2-Cdt1'. We subtract the background channels 6 and 8 from channels 7 and 9, respectively, to reduce the effects of the environment on our data, improving the accuracy of our calculations.

II .2.3 Automatic video generator

The '.movie' files are stored in the folder named by the date of the experiment in ordered. Because there are multiple stacks in each single frame, the separate videos for 'bf', 'green' and 'red' channel are unavailable to watch directly. We wrote a Macros scripts in ImageJ [9] to automatic transfer the stacked '.movie' file into three separate channel videos- bf, green and red (the scripts are attached in **Appendix I**). Generating separate channel video can help us preview the general situation of the experiment. Also, it is helpful for comparing the analysis results of green and red fluorescences with the corresponding channel videos. After separate channel videos are ready, we can be easier to find the best examples in the experiment and check the data in a more intuitive way.

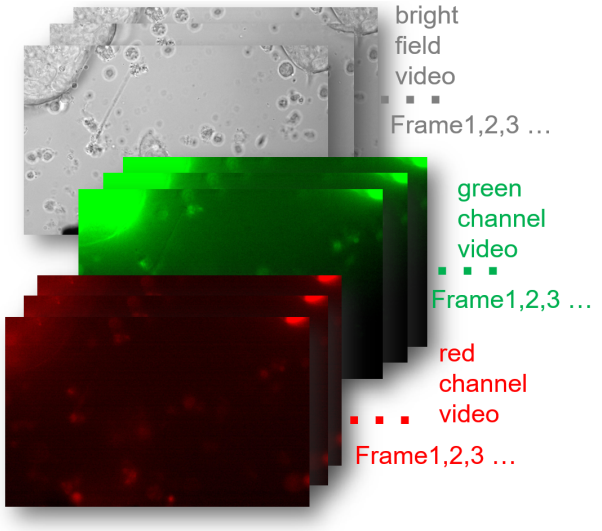

Figure 3: Separate video of bf, green, red channel

II .3 Program design:

The main body of the program is written based on Matlab software [3], the program code is attached in **Appendix I**. In this section, we will talk about the general process of the program and the idea of designing our customized automatic image analysis system. The flow chart of the program is shown in **Section II .3.3**.

II .3.1 Initialization

At the beginning of the program, we do parameters initialization, including the initial value of different regions and the necessary variables to record the coming data. Each read-in imaging data contains 9 different channels and stored into a 3-dimensional matrix (x,y,9). The matrix records the quantitative value for imaging data in each channel. We consider that the imaging data from one set of experiment are stored in a series of '.movie' files. The program can help us automatic search the ordered frames in a movie folder and record the total number of frames (N). In the following analysis, the program will automatically look through these N frames in the folder and record each file name. The searching program can be paused by hand or an occurring error (for example, when the imaging data is missing).

II .3.2 Region selection

One single frame can contain multiple cells on an image, while we expect to analyse the data using a single cell approach. Hence we need to select a region of interest (ROI) on the frame which ideally tracks an individual cell. In order to increase the accuracy of the tracking process. We combine the trackmate plugin in the ImageJ with our program. This plugin can give us a coordinate position of the tracking cells so that we can roughly record the suitable position of the ROI. The program also supports adjusting the region by hand, the basic tracking is required just in case when the cell may run out of the region with the time going by. The reference position of the cell are pre-stored in an array, also we can determine the size of the ROI by adjusting the length of the diagonal of the rectangle region. During the analysis, we ensure that the ROI should always contain an entire cell. If the tracked cell is missing, the program will be paused and start anew.

[3] MATLAB 8.6, the MathWorks Inc., Natick, MA, 2015

II .3.3 Flow chart:

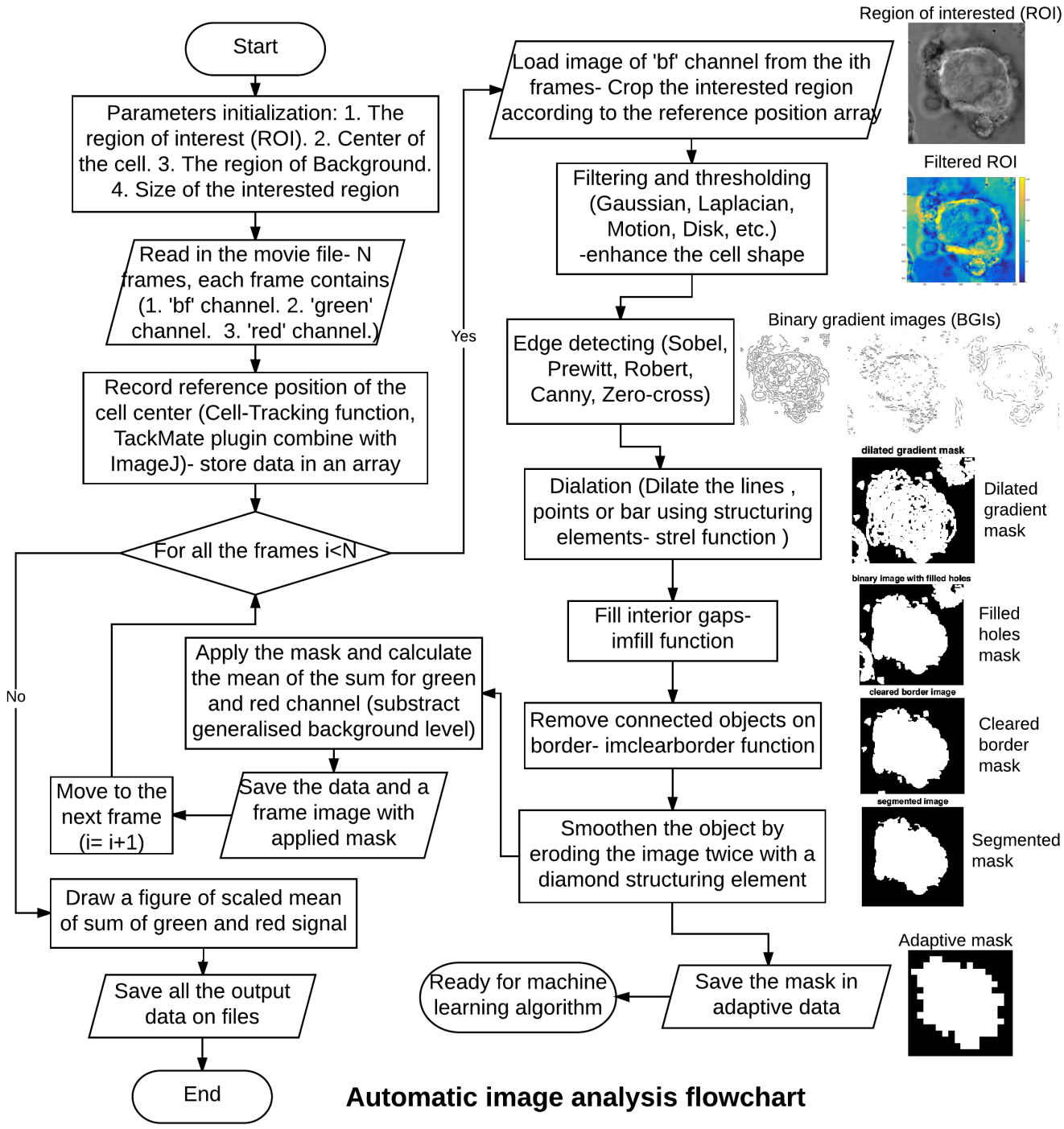

Figure 4: Flow chart of the code to automatic mask building and intracellular fluorescence measuring

II .3.4 Analysis method

In our program, we proposed an active mask segmentation to achieve the single cell analysis. This segmentation method combines the selection of filtering and edge detection methods. We start building our cell masks according to the region of interest. First, we load the image from the bf channel and crop the interested region according to the reference centre. The cropped image is converted into an eight-bits version so that we can set up an auto-level threshold generated by the Matlab function. In the filtering and edge detection part, we tried more than fifty different options (combining use different filtering methods and edge detecting methods) to figure out a better-fitted mask, the results are detailed in **Section II .5**. After applying the combination algorithm, we can get the binary gradient image ready to do an active mask segmentation. The key principle of building the final mask includes edge dilation, erosion and smooth. After the analysis of the current frame being finished, the program will auto-move to the next frame until all '.movie' files in the defined folder have been analysed. The active mask will be updated in each frame according to the selected cell in bf channel, then applied to the green and red channel to calculate the intracellular fluorescence under the mask.

II .3.5 Mask building:

The active mask building is based on cell segmentation techniques. The aim of applying different combination methods is to find a better-fitted mask for the segmented cell. In order to get a neat cell mask, there are four further steps requiring to do after getting the binary gradient image from the combination algorithms There are corresponding outputs being attached in Figure 4. (the right side of the flow chart in **Section II .3.3**)

Step 1: Dilate the lines. The binary gradient mask from the edge detecting shows lines of high contrast in the image. These lines do not quite delineate the outline of the object of interest. Compared to the original image, we can notice gaps in the lines surrounding the object in the gradient mask. These linear gaps will disappear if the edge detected image is dilated using linear structuring elements (the vertical structuring element followed by the horizontal structuring element), which we can create with the *strel* function.

Step 2: Fill interior gaps. The dilated gradient mask shows the outline of the cell quite nicely, but there are still holes in the interior of the cell. To fill these holes we use the *imfill* function.

Step 3: Remove connected objects on border. The interested cell are successfully segmented, but it is not always the only object that has been found. Any objects that are connected to the border of the image can be removed using the *imclearborder* function.

Step 4: Smoothen the object. In order to make the segmented object look natural, we smoothen the object by eroding the image twice with a diamond structuring element. We create the diamond structuring element using the *strel* function. Finally, we display the segmented individual cell by combining a mask drawn by the *alphamask* function.

II .3.6 Fluorescences measure

After we get the segmented individual cell mask from bf channel, we can apply the mask to its green and red channel. The densities of the fluorescences signal are calculated by the mean of the sum within the area covered by the applied cell mask. And then the calculated green and red signal data and a cropped image of the applied mask will be stored. All the information is saved on files. At the end of the analysis, a figure of the scaled data (a fitting curve of the green and red signal) will be generated. The generated results are detailed in **Chapter III** .

II .4 Segmentation techniques:

Optical microscopy is widely used to quantify single cell features, such as cell size or intracellular densities of fluorescences markers. [19] Automated image segmentation for cell analysis is generally a difficult problem due to large variability and complexity of the data. Accurate quantification of such features critically depends on the spatial detection of the cell in the image by cell

segmentation. In addition, cell images may vary widely, depending on the type of microscopy and staining used, as well as the cell type and cell density. This makes the development of a generally applicable cell segmentation method a huge challenge. [11] In our program, we introduced a mask-based method to achieve the cell segmentation applied on cells that are growing isolated from other cells. In the pre-processing of the mask building, select different filtering method and edge detection algorithms can give us many options for combination algorithms to build our cell mask. In order to get better-fitted cell masks, we further investigated the effects of using these combination algorithms to the final mask of the cell segmentation. Finally, we use the segmented mask to calculate the densities of fluorescences signal in Geminin (green) and Cdt1 (red) channel.

II .4.1 The 'combination algorithm':

The 'combination algorithm' mentioned in this report is composed by a personalized imaging segmentation procedure combining different filters and edge detection tools. In our personalized segmentation produce, there are two prerequisite steps to generate the cell mask: Filtering and edge detecting. For each step, there is a possibility to choose from many different algorithms. For example, the possible filtering algorithms include Median, Mean, Gaussian, Laplacian, Motion etc., the possible edge detection algorithms include Sobel, Prewitt, Robert, Canny, Zero-cross etc. [2, 8, 10, 12, 16, 21] We combine these methods to investigate the 'Results of the binary gradient image (BGI) for detecting cell mask with 20 combination algorithms' shown in Figure 8, in order that we can optimize the individual cell segmentation. Figure 5 shows the pipeline of the combination algorithms. The region of interest (ROI) is cropped from the original image which contains the interested cell. In order to compare the results, we use an auto-level threshold by applying the *graythresh* Matlab function. The *graythresh* function computes a global threshold (level) that can be used to convert an intensity image to a binary image. After setting the threshold, we do the filtering algorithm to enhance the general shape of the cell. The filters will produce an interim image, and then we apply the edge detection algorithm to the filtered image. The edge detection algorithm will give us a BGI image which has clear edge features of the interested cell. By using different BGI images, we will obtain different results of the segmented mask. 20 individual masks will be produced if we apply 4 filtering algorithms and 5 edge detecting algorithms. Finally, we can compare the outputs and select the best qualified mask to do further analysis.

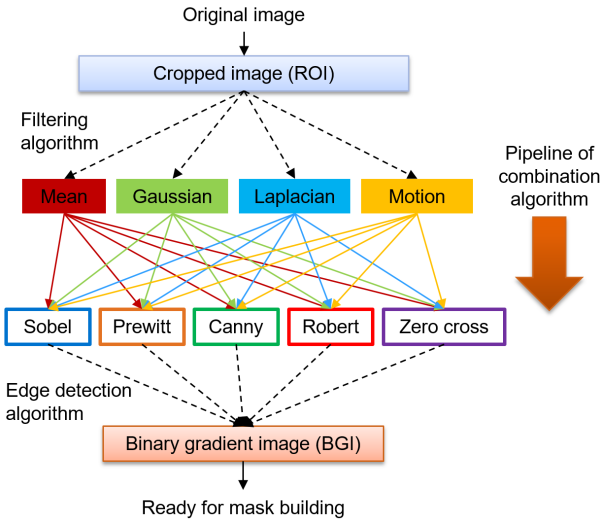

Figure 5: Pipeline of 20 combination algorithms

II .4.2 Filtering method:

A single layer image data can be described as a 2-D matrix (x,y), where x and y represent the coordinates of the pixels. Set a filtering method can help to improve the detectability of important image details, for example enhancing the edge of the single cell in the image. There are many filtering methods and each of them can provide different effects to the image. They are usually implemented by convolutions, which means a process of adding each element of the image to its local neighbours and weighted by the kernel. Kernel represents the shape and size of the neighbourhood to be sampled when calculating. Depending on the element values, a kernel can cause a wide range of effects. [13]

Mean (average):
Mean filtering is a simple, intuitive and easy to implement method of reducing noise images. It is often used to reduce the amount of intensity variation between one pixel and the next. [15] The idea of mean filtering is replacing each pixel value in an image with the mean value of its neighbours (including itself). This has the effect of eliminating pixel values which are unrepresentative of their surroundings. Mean filtering is usually thought of as a convolution filter based on a 3×3 square kernel as below when calculating the mean.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Square kernel of mean filter

Gaussian:
The Gaussian smoothing operator is another 2-D convolution operator that is used to 'blur' images and remove detail and noise. [8] In this sense, it is similar to the mean filter, but it uses a different kernel that represents the shape of a Gaussian function. Each image pixel(x,y) is weighted with the Gaussian functions as *Equation 1*:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{1}$$

where the σ is the width parameter of the function. The idea of Gaussian smoothing is to use this 2-D distribution as a 'point-spread' function, and this is also achieved by convolution. In two dimensions, Gaussian functions are rotationally symmetric, hence the amount of smoothing performed by the filter will be the same in all detections.

Laplacian:
The Laplacian is a 2-D isotropic measure of the second spatial derivative of an image and good for highlighting regions of rapid intensity change. [21] The Laplacian L(x,y) of an image with pixel intensity values I(x,y) is given by *Equation 2*:

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \tag{2}$$

Since the input image is represented as a set of discrete pixels, we have to find a discrete convolution kernel that can approximate the second derivatives in the definition of the Laplacian. Two commonly used small kernels are shown as below:

0	-1	0
-1	4	-1
0	-1	0

or

-1	-1	-1
-1	8	-1
-1	-1	-1

Square kernel of Laplacian operator

Motion:
Motion blur is the apparent streaking of rapidly moving objects in a still image. The motion blur is modelled by convolution with a straight line segment oriented along the direction of motion and with a length equal to the image speed times the exposure duration. [2] The kernel of the motion filter approximates the linear motion of the image on a discrete grid, with a particular choice to weight individual pixels close to the line.

II .4.3 Edge detection:

Edge detection is an image processing technique for finding the boundaries of objects within images. It is used for image segmentation and data extraction to reduce the amount of data and filters out useless information. [10, 16, 18] There are many methods of edge detections, being grouped into first- and second-order derivative expression. The first-order difference operators for estimating image gradient are proposed in Sobel, Prewitt and Robert. And the second-order are proposed in zero-cross. They are computing a measurement of edge strength such as gradient magnitude, and then searching for local directional maxima of the gradient magnitude using estimated local orientation of the edge as the gradient direction.

Sobel:
The Sobel operator consists of a pair of 3×3

convolution kernels as shown below (represents in horizontal and vertical direction, respectively). The operator is convolved with the original image to calculate approximations of the derivatives. Pixel(x,y) at each point in the output represents the estimated absolute magnitude of the spatial gradient of the input image at that point. [12]

G_x:

-1	0	+1
-2	0	+2
-1	0	+1

G_y:

+1	+2	+1
0	0	0
-1	-2	-1

Sobel operator

Prewitt:
Prewitt operator is similar to the Sobel operator, particularly used to detect vertical and horizontal edges in images. At each point in the image, the result of the Prewitt operator is not only the corresponding gradient vector but also the norm of this vector. [16]

G_x:

+1	0	-1
+1	0	-1
+1	0	-1

G_y:

+1	+1	+1
0	0	0
-1	-1	-1

Prewitt operator

Robert:
The Roberts operator consists of a pair of 2×2 convolution kernels, hence can perform a simpler and quicker computation for 2-D spatial gradient measurement on an image. Pixel values at each point in the output represent the estimated absolute magnitude of the spatial gradient of the input image at that point. [5]

G_x:

+1	0
0	-1

G_y:

0	+1
-1	0

Robert operator

The first three operators are designed to respond maximally to edges running vertically and horizontally relative to the pixel grid, one kernel for each of the two perpendicular orientations. The kernels can be applied separately to the input image, to produce separate measurements of the gradient component in each orientation (G_x and G_y). G_x and G_y can then be combined together to find the absolute magnitude of the gradient at each point and the orientation of that gradient. The gradient magnitude is given by $G = \sqrt{G_x^2 + G_y^2}$, and the direction is given by $\theta = atan(G_y, G_x)$. [18] Generally speaking, the gradient magnitude of operator detection represents a number giving the greatest rate of change in light intensity in the direction where intensity is changing fastest.

Canny:
Different from the first three basic detections (Sobel, Prewitt and Robert), the Canny edge detector is a more complex method including a multi-stage algorithm to detect a wide range of edges in images. [3] The algorithm first smooths the image to eliminate and noise. It then finds the image gradient to highlight regions with high spatial derivatives, then doing non-maximum suppression to pick out the best pixel for edges from multiple possibilities in a local neighbourhood. This is smarter than just applying a threshold to an operator detection.

Zero-crossing:
A zero-crossing is a point where the sign of mathematical function changes. At edge points, there will be a peak in the first derivative and, equivalently, there will be a zero crossing in the second derivative. The zero crossing detector looks for places by finding the zero crossings of the second derivative of the image intensity. [10] The crossings often occur at 'edges' in images where the intensity of the image changes rapidly. Zero crossings always lie on closed contours, and so the output from the zero crossing detector is usually a binary image with single pixel thickness lines showing the positions of the zero crossing points.

II .5 Results evaluation:

Combined use of the different filtering algorithm and edge detection algorithms can give us almost infinite choices to describe the cell mask. A simulation of 20 distinguished binary gradient images (BGIs) are plotted in Figure 8. These BGIs can be dilated, filled and smoothed to get an auto-segmented mask, respectively. Although the computer is good at calculation, the human brain is still the best objects-detecting machine, especially for the real life images. In order to evaluate the quality of our auto-segmented masks, we quantitatively compared the auto-segmented masks with a manual-drawing mask. The manual-drawing mask is directly depicted from the original cell image to be a standard mask. And then, we calculate the similarity of the comparison to show how similar our automated mask to the standard mask (Figure 6).

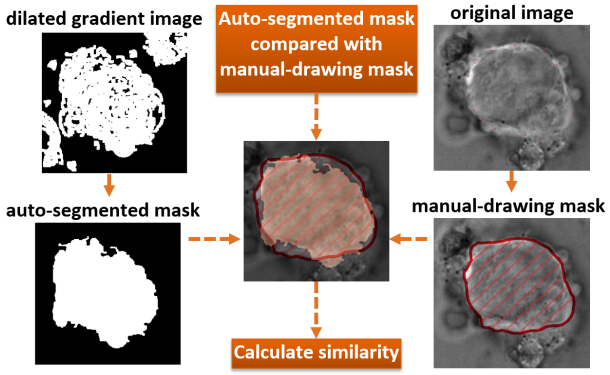

Figure 6: Process of comparing auto-segmented mask to the standard mask

II .5.1 Calculate similarity

The detailed process of calculating the similarity is plotted in Figure 7. We use *imfreehand* Matlab function to get x,y coordinates of the drawing points and record the cell mask as a matrix$[0,1]$. The area covered by '1's is the mask region. We store the information of auto-segmented mask as a matrix called M_{auto} and manul-drawing mask as a matrix called $M_{standard}$. We apply a matrix subtraction to produce the similarity matrix ($M_{auto} - M_{standard}$). The similarity matrix is composed of three daughter regions (exceed, absent and overlap). These factors can affect the final similarity degree. In the similarity matrix, the exceed region produces '1' and absent region produces '-1'. These two factors can both reduce the similarity degree, hence we take the sum of the absolute value. The sum value further divides the sum of $M_{standard}$ to calculate our similarity degree (s%). The formulae can be written as *Equation 3*. Generally speaking, a higher similarity degree indicates a more standard cell mask.

$$s\% = \frac{\sum |M_{auto} - M_{standard}|}{\sum M_{standard}} \times 100\% \qquad (3)$$

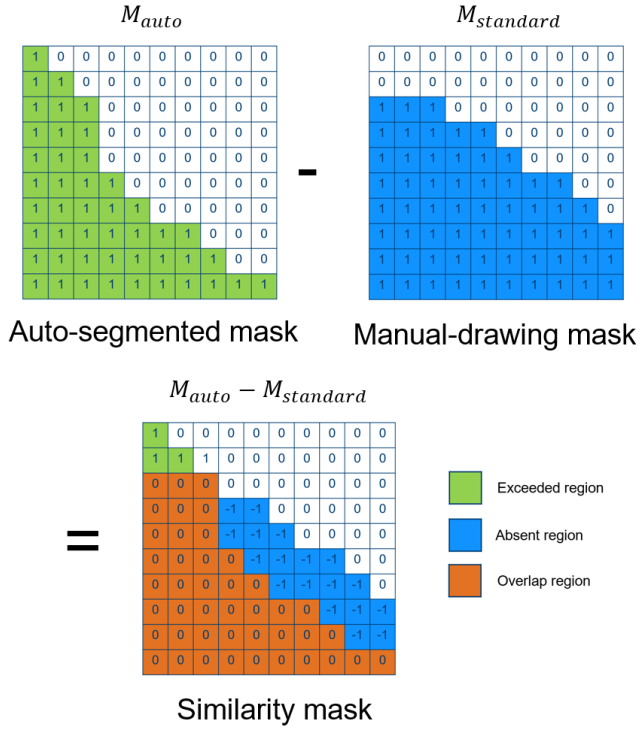

Figure 7: Schematic diagram of calculating similarity degree

This is a digitalized method to evaluate the quality of the auto-generated cells mask. The computer is not powerful to evaluate the imaging data directly. We digitalize the information from the image so that the computer can help us make the decision. By comparing the magnitude of the similarity degrees, we can automatically find the most standard mask generated by corresponding BGIs to do further analysis.

Results of the binary gradient images (BGIs) with 20 combination algorithms:

Figure 8 shows the results of the binary gradient images (BGIs) with 20 combination algorithms generated by combining different filters and edge detection tools. The binary gradient images (BGIs) are the prerequisites to the cell mask. It helps us show the most prominent differences between all applied combination algorithms. In each column, we included different filtering methods (Gaussian, Laplacian, Motion and Disk). In each row, we included different edge detection algorithms (Sobel, Prewitt, Robert, Canny and Zero-cross). Generally speaking, the canny-BGIs give us the most detailed gradient edge, but extra noise occurs in the background when it is applied with Laplacian algorithm. The Robert-BGIs use the simplest operator of the kernel, relatively the detected edges are not detailed enough compare to others. The Sobel and Prewitt algorithms are very similar, even combined with different filtering method. The zero-cross BGIs are especially good at detecting the intracellular gradient changes.

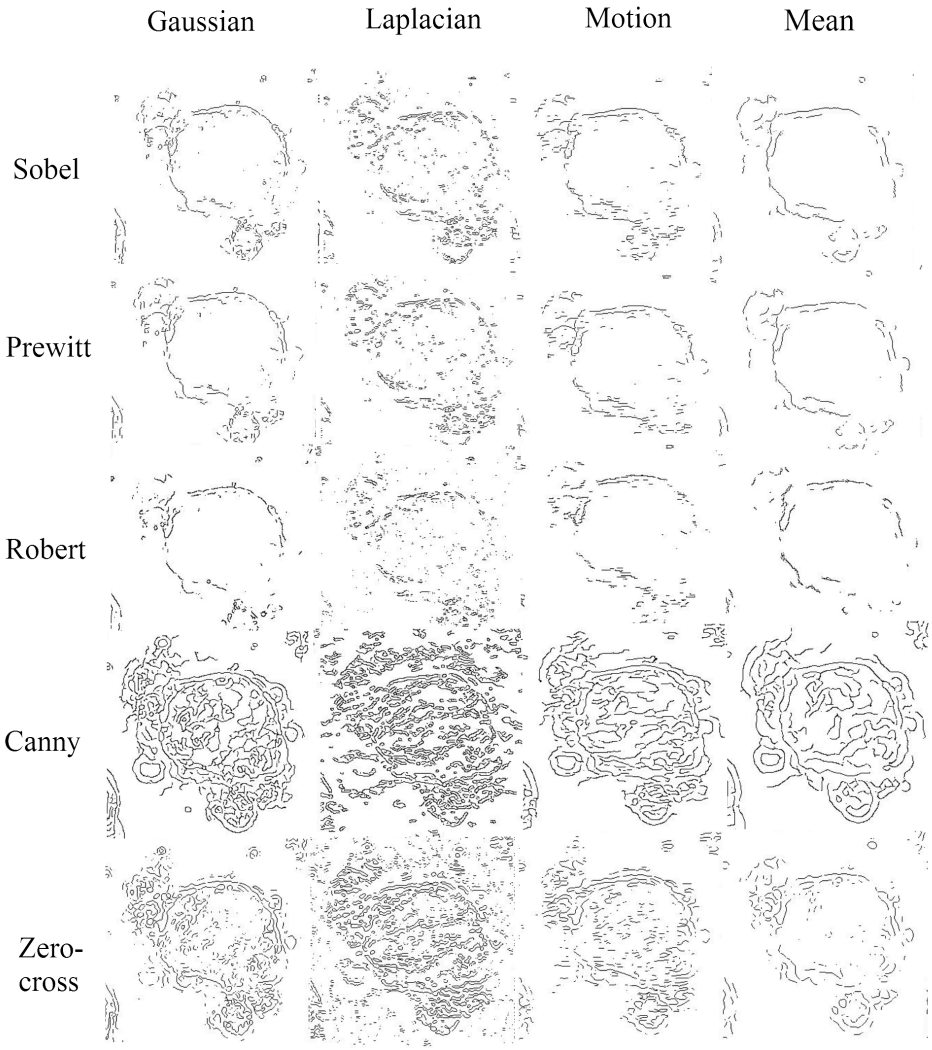

Figure 8: Results of the binary gradient image (BGI) with 20 combination algorithms. Each BGI is generated by one combination algorithm using the algorithm on rows and columns.

Results of the similarity degree for 20 combination algorithms [4]:

The similarity degree table in Figure 9 records the similarity of comparing auto-segmented masks with the standard mask. The auto-segmented masks are started with corresponding BGIs. Because the differences of the cell masks are not very perceptible to the human eye, we use a digitalized method to evaluate their differences. In the table, higher similarity degree value indicates a more standard mask. By comparing the degree values we can evaluate and find the best fitted BGIs to generate the cell mask in our example. Three best 'similarity' masks in this result are further compared in **Chapter III** for extracting cell cycle parameters (Auto mask 1: generated by Gaussian-Canny BGI, Auto mask 2: generated by Laplacian-Sobel BGI, Auto mask 3: generated by Motion-Zerocross BGI).

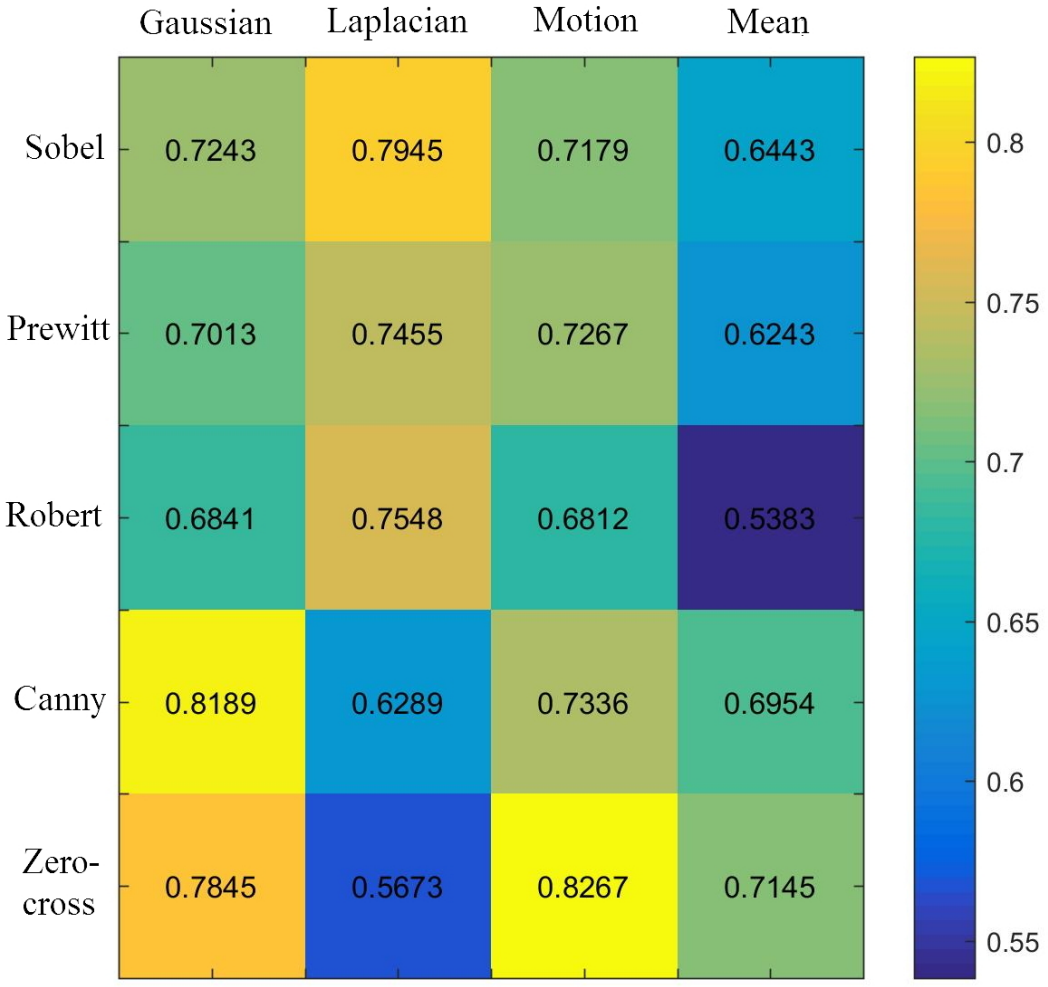

Figure 9: The similarity degree table of cell masks generated by 20 BGIs compared to the standard mask, the highest similarity degree represents the most standard cell mask generated by the corresponding BGI.

[4]Note this similarity degree table is based on analysing a specific set of imaging data. We show the relative comparison results just for an example. In a more general analysis, the similarity degree can be affected by many factors (for example, the format of the imaging data, the cellular state, density of the background noise, resolution of the microscopy etc.). In our final analysis, the program will help us choose the most suitable BGI to build the final cell mask.

III Extraction of cell cycle parameters with FUCCI expression

In this chapter, we compared two methods to extract the cell cycle parameters with FUCCI expression: the manual selection method and the automated method. In both methods, we refer to the same gating information which can help us distinguish different cell cycle phases from the imaging data. Although the manual selection method is much more straightforward to select the signal detected by human eye, considering that we have hundreds and thousands of images in the database, it is easy to imagine that a manual analysis of the whole data would be an extremely hard task. Therefore, the automatic image analysis program is crucial to be designed to release the handwork. In order to evaluate the results of the automatic analysis, we run the results of three representative methods mentioned in the previous chapter to compare with the manual selection method (Auto mask 1- Gaussian canny, Auto mask 2- Laplacian sobel and Auto mask 3- Motion zero cross). The results for comparing the manual analysis and automatic analysis has been discussed at the end of the chapter.

III .1 Gating information:

Since the gene-transfer techniques are highly efficient, contransduction allowed us to obtain stable transformants expressing equivalent levels of Cdt1 (the red fluorescence) and Geminin (the green fluorescence). In each transformant, red fluorescence alternated with green fluorescence in the nucleus. The cell cycle period is variable, presumably due to differences in cell density and serum concentration. [4, 20, 22] Since the green fluorescence disappears rapidly in late M phase and the red fluorescence becomes detectable in early G1 phase, a small gap in fluorescence is observed in newborn daughter cells. By contrast, during red-to-green conversion, red and green fluorescence always overlaps to yield a yellow nucleus. [1, 4] In order to extract the parameters of the cell cycle, we define a suitable gating information to represent the different cell cycle phases from the imaging data (Figure 10). The approximate gating method was advised by Dr. Kedar Natarajan (European Bioinformatics Institute) and also mentioned in [1]. The gating information helps us add the typical cell cycle phases on our analysed data.

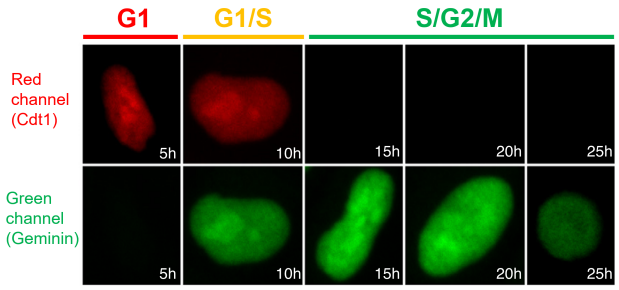

Figure 10: Defination of the gating information to represent the different cell cycle phases from the imaging data, the typical phases are recorded as 'G1', 'G1/S' and 'S/G2/M'. (this figure is adapted from [1], re-edited by J.Yu)

III .2 Manual selection method:

III .2.1 General process:

Step 1: select the interested region containing a single cell. The size of the region should be always larger than the cell size ensure that the region is covering the entire cell (Figure 11). The position of the cell is manually determined by human eye, accurate but very slow.

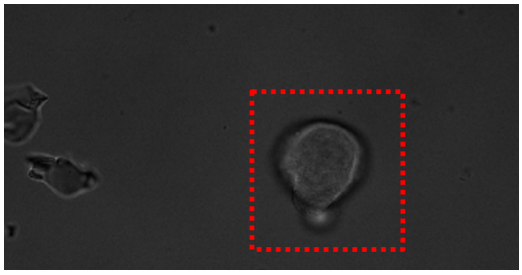

Figure 11: The red dash-line square represents the region of interest (ROI)

Step 2: select a small circle within the cell. The small circle is chosen manually in each frame and can be adjusted according to the position of the cell in each frame- noting the circle can cover as much as the intensive signal either in the green

or red channel. (Figure 12). Tremendous work, if the movie folder contains thousands of frames(In this example, we select 100 frames in the whole movie folder). The interval between each frame is equal to a real-life time for 15 mins. Hence, 100 frames are recording the real-life time for around 25 hours.

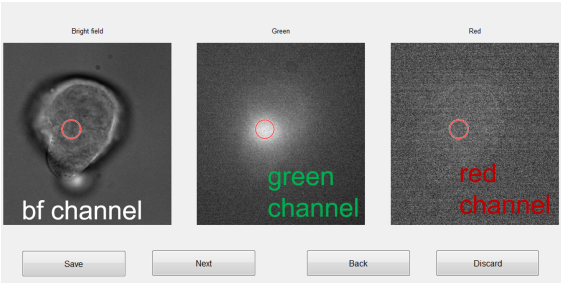

Figure 12: Select a small circle within the cell to cover the green and red signal in each frame

Step 3: calculate the mean of the sum of the signal within the small circle in the green and red channel. The diameter of the small circle can be adjusted by hand. Normally we set the size of the small circle tenth of the entire cell. The position of the circle in each frame is all equal so that the calculation in the green and red channel can be extracted at the same time. Each calculated results are recorded as a scaled value and stored in an array. The results of the recorded value are plotted as Figure 13.

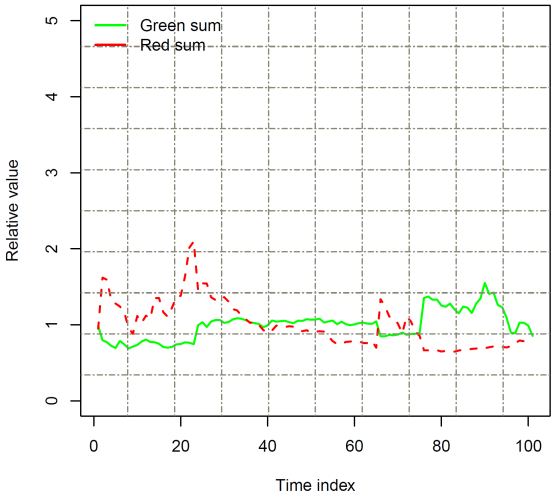

Figure 13: Results of recorded data in **Step 3**, difficult to add gating informaiton in these results

III .2.2 Curve fitting:

Step 4: applied the stored data to a two-term Gaussian fitting. Consider that the data plotted in Step 3 is uncontentious and fluctuated. It is difficult to apply a gating information on such kind of data. The idea is we can apply the process of constructing a curve, or mathematical function, that has the best fit to a series of data points. The curve fitting can involve either interpolation, where an exact fit to the data is required, or smoothing, in which a 'smooth' function is constructed that approximately fits the data. After many trials, we decided to use a multi-term Gaussian smoothing method. The fitted curves can be used as an aid for data visualization to help summarize the relationships among these two variables (green and red sum). The first Gaussian term can display the peak of the data 'mount' the and the second Gaussian term can help describe the general tendency of the data. After applying the curve fitting, we can get smoothed results as Figure 14 (the blue dotted data in Figure 14 is extracted from the results in **Step 3**).

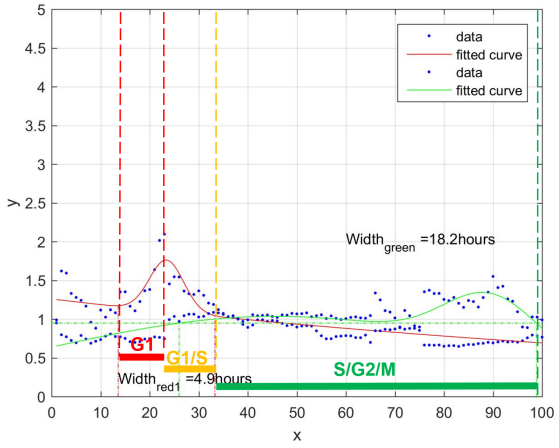

Figure 14: Results of two-term Gaussian fitting for the data from **Setp 3** with its gatting information, the green and red peak regions are more obvious

III .2.3 Gating threshold:

Step 5: set threshold method for gating information. Now it is much easier to apply the gating information to the smoothed data. Before applying, we required a threshold method to define

start points and end points of the different cell cycle phases. According to the gating information, the red fluorescence accumulates in G1 and reduced in transitional period G1/S. The start and end points of the red curve are determined by 95% confidence of the first term in the Gaussian fitting model. The green fluorescence accumulates in S/G2/M (approximately setting the threshold to be 70%-80% of the global maximum value), or otherwise, it can be gated from the end of the G1/S phase to the start of the next G1 phase (shown in Figure 14), considering the peak value of the green curve much more close to the general level.

III .3 Automatic method:

III .3.1 General process:

The detailed process of automatic image analysis has been explained in **Chapter II**. Generally speaking, in the automated method, the computer helps us find the cell and apply the active-mask to calculate the mean of the sum of green and red fluorescence under the entire cell (a photo guide is designed as Figure 24 in **Appendix II**). Unlike the manual selection, the auto-segmented mask do not focus on the intensive region of the fluorescence signal. However, it is much clever to calculate the fluorescence signal within the entire cell. The signal calculation principle is similar to the manual selection method, while the program automated all data storing, multi-term Gaussian fitting and gating threshold in the automatic method. Figure 15 shows the auto-tracking mask apply with a different period of time.

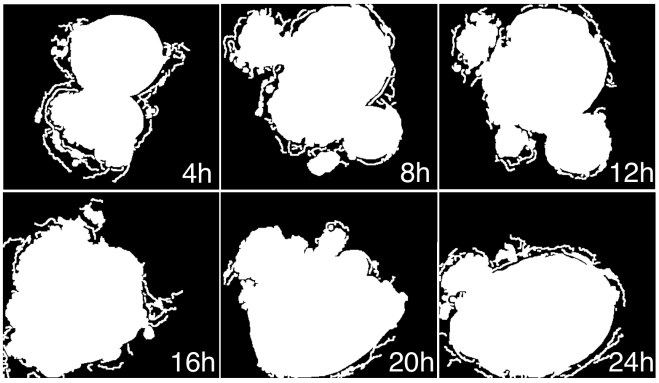

Figure 15: Automatic mask tracking with time

Results for automatic analysis methods:

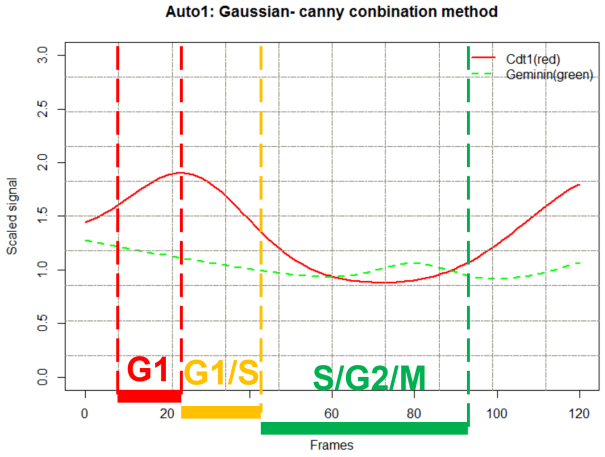

Figure 16: Auto-mask1: Gussian-canny

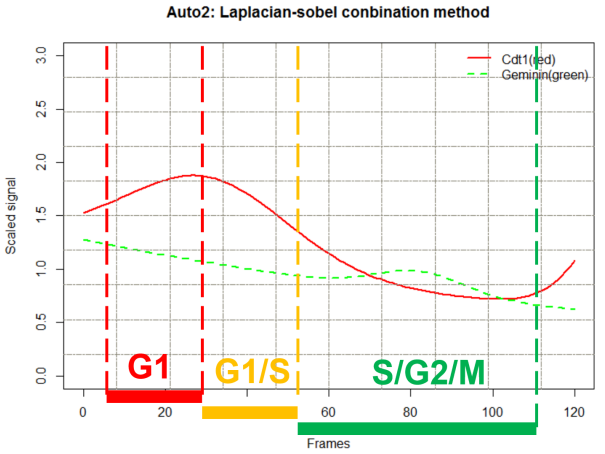

Figure 17: Auto-mask2: Laplacian-sobel

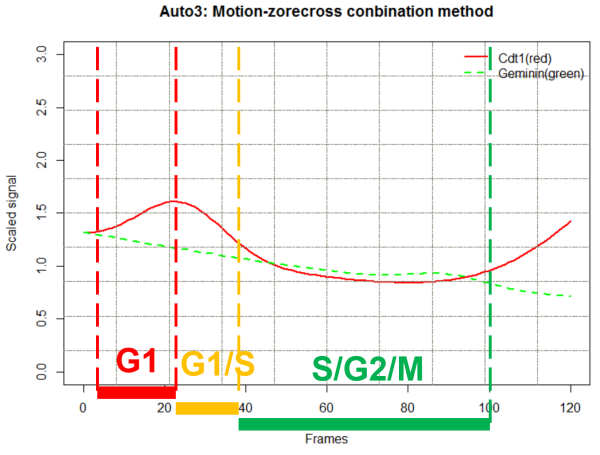

Figure 18: Auto-mask3: Motion-zerocross

III .3.2 Results of automatic analysis methods:

The combination uses of different filtering methods and edge detecting methods give us many options for the mask building. Three recommended methods (based on the top three similarity results described in **Chapter II**) are further compared here by applying the data fitting and gating information (Auto mask 1- Gaussian canny, Auto mask 2- Laplacian sobel and Auto mask 3- Motion zerocross). The results are shown as Figure 16 to Figure 18. The fitting functions of the red signal have slight differences, while the peak values are quite fixable at around 20-25 frames. Considering that the peak value of the green curve is much more close to the general level, the phase of S/G2/M in automatic results are gated from the end of the G1/S phase to the next start point of the G1 phase (when red curve rises again).

III .4 Compare with the manual selection and automatic analysis method:

III .4.1 Method comparison:

The manual approach can get more obvious peaks for red and green curve fitting. Because it can focus on the fluorescences signal within an intensive circle. Flexibly, the focusing position can be adjusted by hand in each frame. It will cost a long time to select but it increases the accuracy of the analysis. The automated method is calculating the green and red signal within the entire cell, which the results are more gentle, compared to the manual selection. Ideally thinking, we can add an intensive calculation method in our automated program. For example, add a tracking region on the green and red channel respectively, so that we can focus on the signal like a manual selection circle. However, the direct-tracking of the fluorescence is much more complex. Because the green and red fluorescences are alternatively occurring on a separate image, makes it easy to lose the tracking information. Hence, tracking the cell in a bright filed is the best choice in our imaging data. Although the main approach of the calculation is slightly different, they are comparable because the fluorescences signals are stochastically distributed within the entire cell.

III .4.2 Results comparation:

The multi-term Gaussian model of these curve fitting is given by *Equation 4* [5] (a for amplitude, b for peak centre, c for peak width), in a two-term Gaussian (n=2, i=1,2), in a three-term Gaussian (n=3, i=1,2,3). Three-term are required if there is a second peak trend to appear. For example, the data collected from the automated method (the red curve) is better fitted with a three-term Gaussian model, because the curve is trending to rise again at the final frames.

$$f(x) = \sum_{i=1}^{n} a_i \cdot e^{[-(\frac{x - b_i}{c_i})^2]} \tag{4}$$

The compared curve-fitting results of the automated and manual method for red and green fluorescences, has been shown in Figure 19 and 20 respectively. Generally, the automated curves are more close to the general level (the changing of the peak value is more gentle and the 'curve mount' is more placid), while the manual method shows a more obvious peak region. Detailed compare, in red channel analysis, the peak centres of the main peak (b_1) are very close (all around 25), which indicate the automated method is accurate to calculate the densities of the fluorescences. The peak amplitude (a_1) is similar (all around 0.7), which means the data strength of the signal is suitable. The peak widths (c_1) in the automated analysis are generally wider than manual work, but not effective to find the gating information. In green channel analysis, the automated curve needs a more obvious peak value because the peak amplitude is much lower than the manual method. In summary, the automated analysis is quite accurate to fit the data, especially for the red channel signal. The problem in green channel is that the peak region is not obvious. This demerit can be fixed by estimating the strong signal region in the green channel, to highlight the peaks. (More data analysis results are plotted in Figure 21)

[5] Note in the multi-term Gaussian model, the first term (a_1, b_1, c_1) describes the behaviour of the main peak and the second term (a_2, b_2, c_2) represents a general tendency of the data. The third term (a_3, b_3, c_3) is added in case there is a potential second peak.

Results of comparing the automatic analysis and manual selection method (red fluorescence):

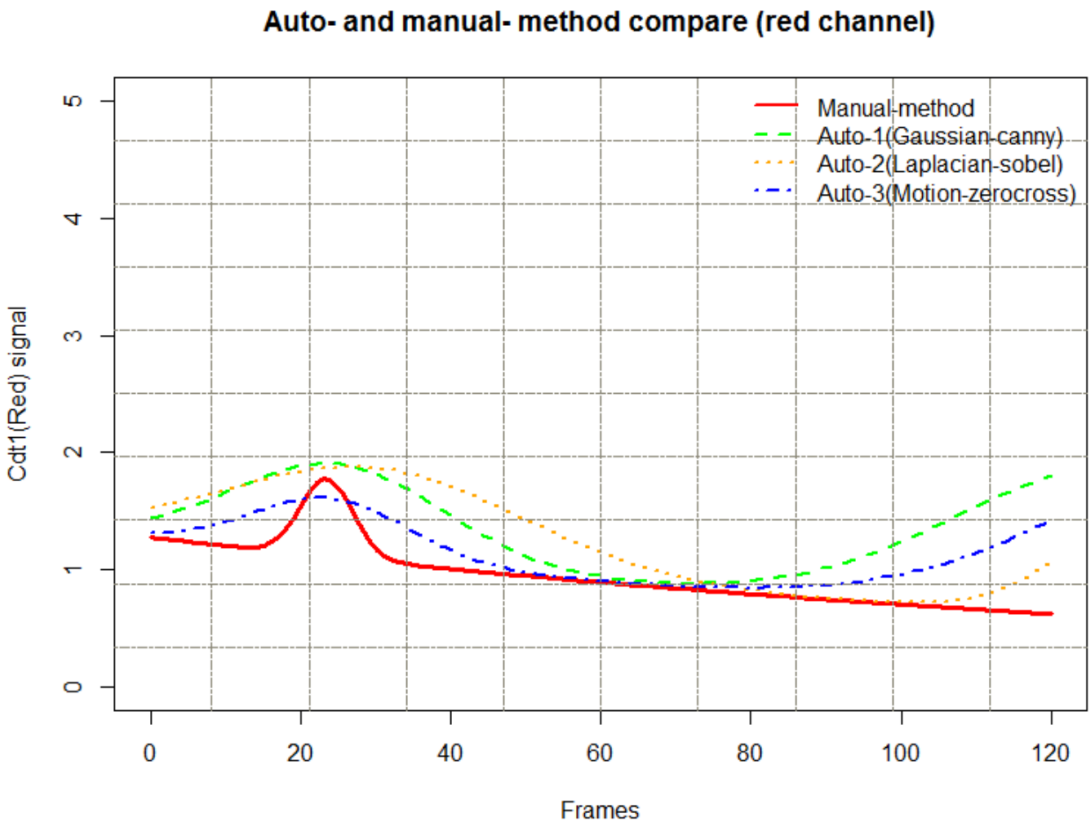

Figure 19: Results of comparing the auto- and manual- method (red channel signal). The peak regions of the red signal are corresponding well with each other.

Gaussian curve fitting [6]: $f(x) = a_1 \cdot e^{-(\frac{x-b_1}{c_1})^2} + a_2 \cdot e^{-(\frac{x-b_2}{c_2})^2} + [a_3 \cdot e^{-(\frac{x-b_3}{c_3})^2}]$

Table 1: Parameters of two-term Gaussian smoothed function (red curve fitting)

	Describe	Manual	Gaussian canny	Laplacian sobel	motion zerocross
a1	Amplitude1	0.72	0.71	0.80	0.69
b1	Peak centre1	23.41	24.64	25.83	24.12
c1	Peak width1	4.94	10.20	18.21	7.52
a2	Amplitude2	5e+11	5e+11	5e+11	5e+11
b2	Peak centre2	-8925	-7832	-7289	-8631
c2	Peak width2	1732	1672	1813	1728
a3	Amplitude3	/	0.87	0.92	0.89
b3	Peak centre3	/	139.22	154.41	142.53
c3	Peak width3	/	23.43	18.14	20.43

[6]The third term in manual curve is not provided, because there is no second peak in this example

Results of comparing the automatic analysis and manual selection method (green fluorescence):

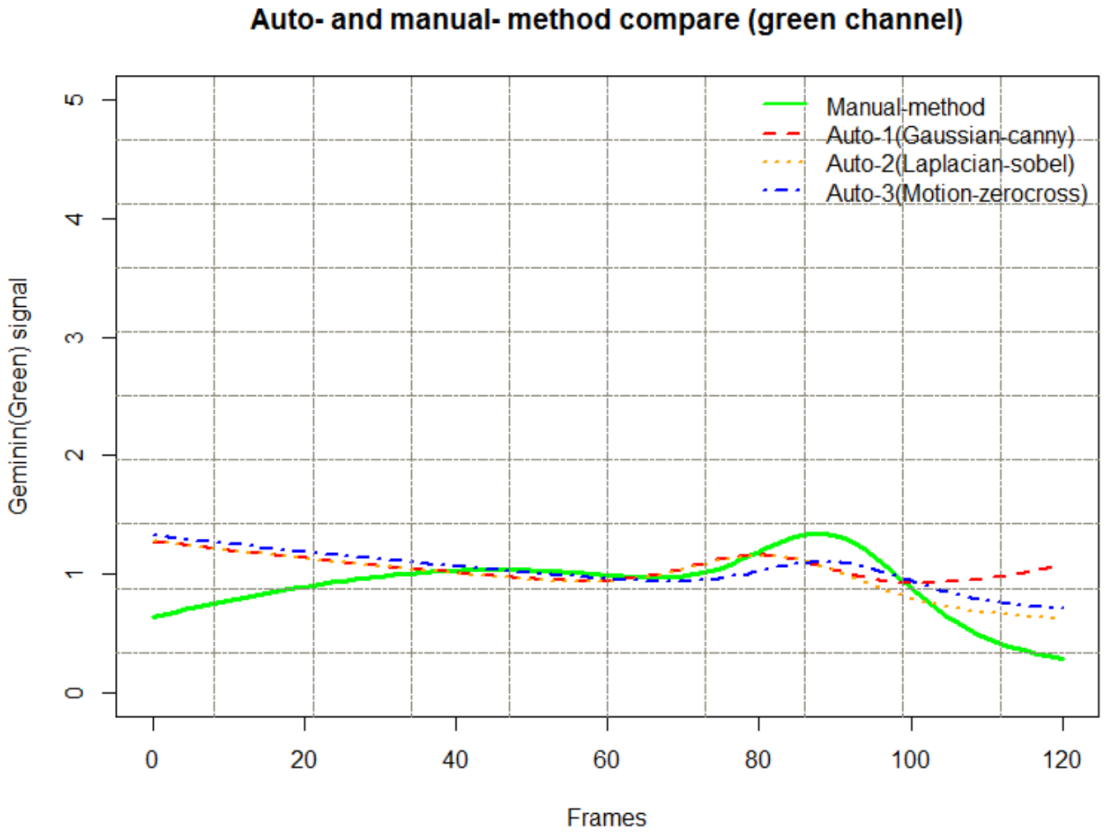

Figure 20: Results of comparing the auto- and manual- method (green channel signal). The general outputs are close, but manual approach shows more obvious peak regions.

Gaussian curve fitting [7]: $f(x) = a_1 \cdot e^{-(\frac{x-b_1}{c_1})^2} + a_2 \cdot e^{-(\frac{x-b_2}{c_2})^2} + a_3 \cdot e^{-(\frac{x-b_3}{c_3})^2}$

Table 2: Parameters of two-term Gaussian smoothed function (green curve fitting)

	Describe	Manual	Gaussian canny	Laplacian sobel	motion zerocross
a1	Amplitude1	0.67	0.31	0.37	0.28
b1	Peak centre1	89.55	80.21	82.32	88.11
c1	Peak width1	13.38	15.23	17.98	14.82
a2	Amplitude2	4e +13	6e +13	6e +13	6e +13
b2	Peak centre2	-3132	-4322	-5432	-5425
c2	Peak width3	2144	1443	1475	1632
a3	Amplitude3	1.03	2.21	2.16	2.08
b3	Peak centre3	45.49	168.34	178.12	159.20
c3	Peak width3	65.44	83.23	98.44	76.58

[7] The peak centre of first and third term in manual curve is close, hence two gaussian-peaks are merged. That is also the reason we set up different thresholding method for green signal. Detailed discussed in 'Gating threshold' (**Section III .2.3**)

Results of comparing the automatic analysis and manual selection method (additional data analysis)[8]:

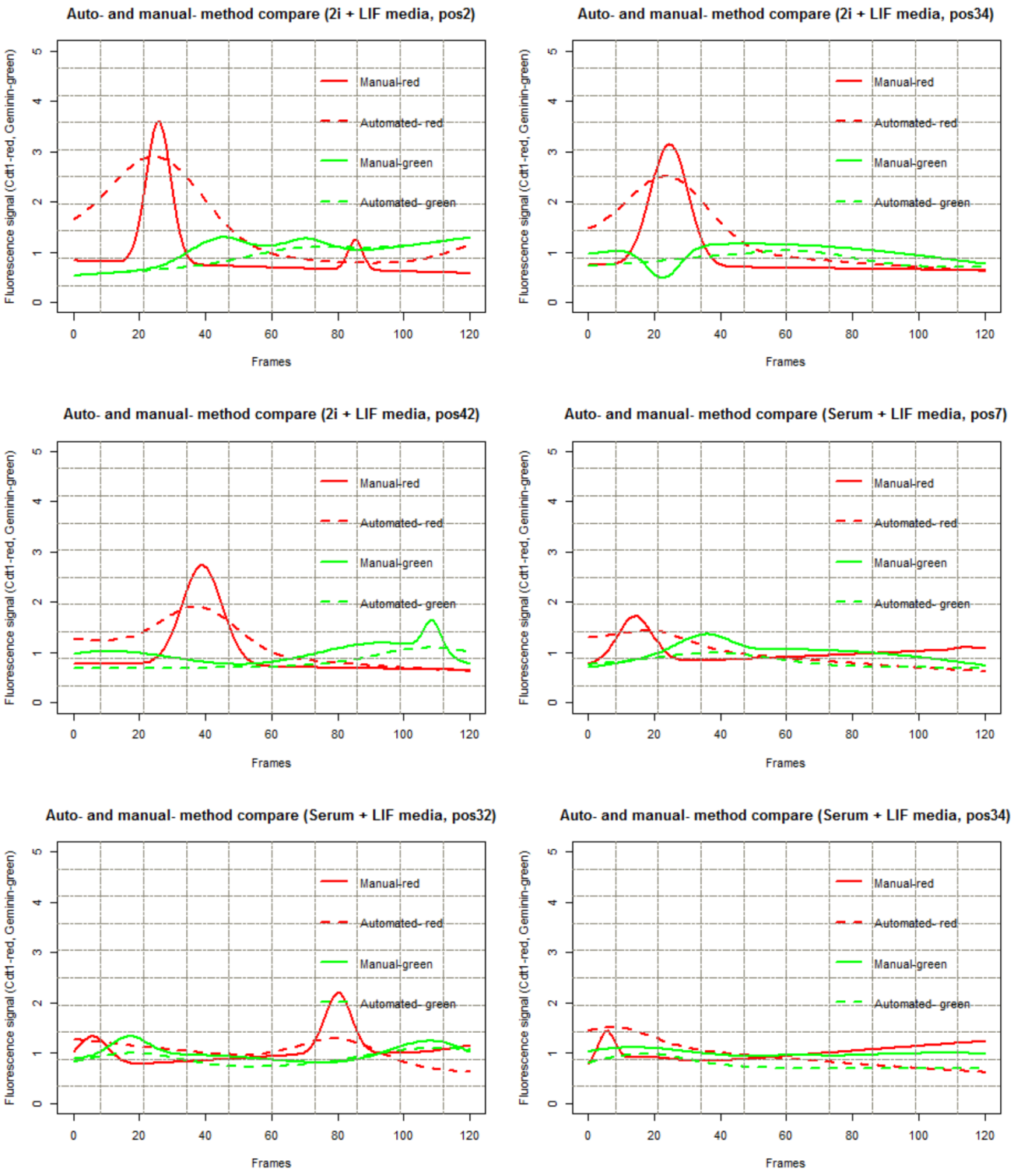

Figure 21: Results of comparing the auto- and manual- method (2i and serum + LIF media)

[8]The original imaging data is provided by W.Zhai (final year PhD in our group) for her experiment to compare two culture conditions for cell growth, 2i and serum is named by the cell media+ LIF (Leukaemia inhi ory factor).

IV Discussion:

IV .1 Green channel problem - calculating intensive local maximum to enhance the peak region

We notice that the peak region of the automated method in the green signal is not obvious because they are too close to the background level. In this case, the manual method works much better to find a peak region for its intensive calculation. There is a coming ideal to enhance the peak region of green signal in the automated method. Instead of calculating the mean of the sum within the entire mask, we can draw a data map of the green signal within the cell mask. And then, we find the local maximum value on the data map using a smaller area. We use the specific area to calculate the mean of the sum within that particular region.

IV .2 Cell cluster segmentation - using edge-highlighted method separate individual cell in a cluster

In this report, we are focusing on the cell segmentation applied on cells that are growing isolated from other cells. But in many cases, the cells are sticking together (to form a cell cluster) during the experiment. The automated mask-building segmentation is an edge-based segmentation. This method is good at detecting the general 'contour' of a cell or a cell group. However, it is not powerful to solve the problem in a cell cluster. It becomes a big challenge to analysis the fluorescence signal under the cell mask when we are required to select an individual cell from the cell cluster. We are considering some possible solution to divide an individual cell from the cell clusters. The watershed method is recommended but it is non-ideal in our routine for problems of over-segmentation. Hence we tried an edge-highlighted method (depicted in Figure 22). The auto-segmented mask is generated by our code, but it is covering the entire cell cluster. Here we draw an additional mask for an edge-highlighted method, this mask is specifically working for detecting the general 'contour' of individual cells. Then we apply these two masks together to get a combined mask so that we can further separate the individual cell under the entire mask. However, this method also highlighted the inactive area in the cell cluster which contains many dead cells. The dead cell problem is also discussed in this section. Further analysis is required to evaluate this method.

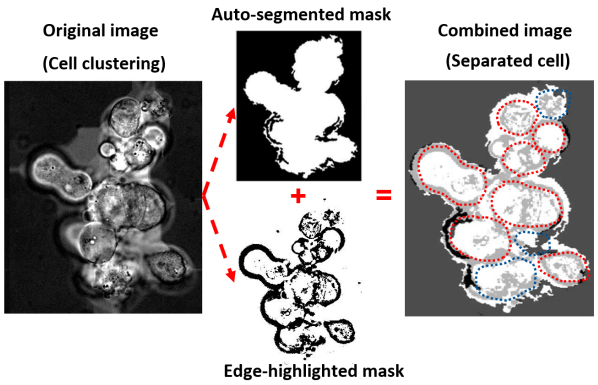

Figure 22: Process of adding edge-highlighted mask to separate individual cell in a cell cluster

IV .3 Dead cell recognition - building a machine learning mask database

The dead cell problem is another big challenge we are facing to over come. In current algorithm, the cell mask is not clever enough to distinguish the active cell and dead cell. It is not a severe problem if the cell keeps healthy. But in a real experiment, the cell is much likely to be dead after a long-time testing. Normally, the tested cell is not live longer than two days even given the best condition and treatment. However, the fluorescence signal may remain in the cell body even the cell is dead. If we use the mask to calculate the fluorescence density, we will include the dead fluorescence which we really want to avoid.

We discover the phenomenon that the active cells always keep moving and changing their shapes, while the dead cell is much more stable to stay at the previous location. If we can memorize this behaviour, it will be much helpful to figure out the active cells and dead cells. We are planning to build a memory database to store the possible behaviour of the active cell. The reason we use the adaptive square mask is to reduce the disk memory and speed up the memorising, as the

active cell may have thousands of different transformations. Oppositely thinking, because the status of the dead cell is much more stable compared to the active cell. Instead of memorising the active cell, why not to memorise the behaviour of a dead cell. The database to memorise the dead cell will be much smaller. And then, we can use the auto-generated marks from our code to subtract the dead cell region matching with the database, to get the wanted region of the active cell.

V Conclusions and future work:

V .1 Conclusion

In this project, we design an automated program to analyse the imaging data from the FUCCI experiment. The program helps us extract the relevant parameters from the analysis to characterise the cell phases according to the red and green fluorescence signals. The results are evaluated by comparing against the manual approach. Many combination methods are investigated to build a better-fitted mask for individual cell segmentation. The combination algorithms of Gaussian-canny, Laplacian-sobel and motion-zerocross are selected for further comparison. The accuracy of the automated image analysis is tested against the manual approach. The results show the automated program is successful in finding the peak region in the fluorescence density calculations, especially for the red channel signals. The multiple-term Gaussian models fit well with the green and red fluorescence signal extracted from the imaging data. The model serves to find the gate information from a cell cycle. In summary, building the automatic image analysis system helps us efficiently extract the relevant parameters from a large amount of data to successfully characterise the cell phase with FUCCI expression.

V .2 Future work

Several points can be improved in our automated program in the future. First, the calculation method of the green fluorescences signal can be more intensive by finding a local maximum density. Although the results gained from the red signal analysis are good-fitted with the manual work, the intensive calculation method is required to find a more obvious peak. The auto-segmented mask is good at individual cell analysis, but the method of separating an individual cell from the cell cluster still needs to be further evaluated. The idea of building a machine learning database to recognize the active cell and the dead cell has been discussed, but there is a long way to go to finally apply the algorithm to our imaging data analysis. Last but not least, we are considering to develop the code to be more humanized and generalized so that it can be used in many other experiments.

VI Acknowledgements:

I would like to express my gratitude to my supervisor, Prof. Pietro Cicuta, for his expert guidance and extraordinary support during the time I worked on this project. I am also grateful to Weichao Zhai, the final year PhD student in our BSS group, for her help and advice, especially for providing the imaging data in the part of extracting the cell cycle parameters with FUCCI expression. Also thanks to Dr. Kedar Natarajan (European Bioinformatics Institute) for advising us the gating information method of the cell cycle.

References

[1] H. Kurokawa A. Sakaue-Sawano et al. Visualizing spatiotemporal dynamics of multicellular cell-cycle progression. *Cell 132, 487-498*, 2008.

[2] D. Blasiak and W. Chan. Motion filter vector quantization. *IEEE Xplore, 1522-4880*, 2002.

[3] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679698*, 1986.

[4] P. Bourillot Y. Tapponnier D. Coronado, M.Godet et al. A short g1 phase is an intrinsic determinant of nave embryonic stem cell pluripotency. *Stem Cell Research: 10, 118-13*, 2013.

[5] LS. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing, vol 4, no. 3, pp 248-260*, 1975.

[6] G. Tong G. Guo, M. Huss et al. Resolution of cell fate decisions revealed by single-cell gene expression analysis from zygote to blastocyst. *Develop. Cell: 18, 675-685*, 2010.

[7] R. Menafra H. Marks, T. Kalkan et al. The transcriptional and epigenomic foundations of ground state pluripotency. *Cell: 149, 590-604*, 2012.

[8] R.A. Haddad and A.N. Akansu. A class of fast gaussian binomial filters for speech and image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 39, pp 723-727*, 1991.

[9] E. Frise J. Schindelin et al. Fiji: an open-source platform for biological-image analysis. *Nature methods 9(7): 676-682, PMID 22743772*, 2012.

[10] S. Sarkar M. D. Heath. A robust visual method of assessing the relative performance of edge-detection algorithm. *IEEE transctions on pattern analysis and machine intelligence. Vol. 19, NO. 12*, 1997.

[11] E. Mijering. Cell segmentation: 50 years down the road. *IEEE Signal Processing Magazine, pp. 140-145*, 2012.

[12] Ostu N.A. Threshold selection method from gray. *IEEE Trans on Systems. Man d Cybe etics, SMC - 9,9(1):62-66*, 1979.

[13] Mark S. Nixon and A. S. Aguado. Feature extraction and image processing. *Academic Press*, 2008, p. 88.

[14] S. Pauklin and L. Vallier. The cell-cycle state of stem cells determines cell fate propensity. *Cell: 155(1), 135-147*, 2013.

[15] I. Pitas and A. N. Venetsanopoulos. Nonlinear digital filters: Principles and applications. *Kluwer Academic*, 1990.

[16] J.M.S. Prewitt. "object enhancement and extraction" in "picture processing and psychopictorics". *Academic Press*, 1970.

[17] B. Reinius Q. Deng, D. Ramskld and R. Sandberg. Single-cell rna-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science: 343, 193-196*, 2014.

[18] B. Yu S. Chang and M. Vetlerti. Adaptive wavelet threshold for image de-noising and compression. *IEEE Trans Image Processing, 9(9):1532-1546*, 2000.

[19] F. Rudolf J. Stelling S. Dimopoulos, C. E. Mayer. Accurate cell segmentation in microscopy images using membrane patterns. *Bioimage informatics, p. 26442651*, 2014.

[20] E.Rodriguez Y.Lin S.Koh, P.Mascalchi et al. A quantitative fastfucci assay defines cell cycle dynamics at a single-cell level. *The Company of Biologists Ltd: 130, 512-520*, 2017.

[21] R. Weeks. Fundamentals of electronic image processing. *IEEE Press*, 1996.

[22] N. Zielke and B.A. Edgar. Fucci sensors: powerful new tools for analysis of cell proliferation. *Dev Biol: 4, 469-487*, 2015.

VII Appendix I: Program code and scripts

VII .1 Automatic image analysis- Matlab code

The main code - part 1

The first part provides the basic construction of our customized automatic image analysis system. The general process of the main program including initialization, region selection, mask building, fluorescences measure.

```
clc;
clear all;
close all;

%Change the path here to selected the interested folder
%The '.movie' files are stored in the folder named by
%the date of the experiment in ordered.
dir_path = '/Users/JiamingYU/Desktop/Project_jy380/data/Pos_3_2i_14Dec2016';
cellname = 'scan0_petri0_position003.14Dec2016_14.47.35'; %start file name
%dir_res = [dir_path '/output/analy_cell' num2str(cellname)];

startframe = 1; %set begin searching place

%% initializations
%searching the folder to get all file names
[FileName_start,PathName,~] = uigetfile([dir_path '/*.movie'],'select_file')
    ;
ind_after_scan = strfind(FileName_start,'_');
petri_position = FileName_start(ind_after_scan(1)+1:ind_after_scan(1)+18);

FileNames = dir([PathName '/*' petri_position '*.movie']);
for i = 1:numel(FileNames)
    ind_after_scan = strfind(FileNames(i).name,'_');
    runs(i) = str2double(FileNames(i).name(5:ind_after_scan(1)-1));
end
[~,I_sort]=sort(runs);
SelectedNames = {FileNames(I_sort).name}; %Searching the file name

%record the total number of '.movie' file in this folder
endframe = numel(FileNames);

%% initialize the recording variables
T = endframe-startframe+1;
Red = zeros(T,1);
Red_s = zeros(T,1);
Green = zeros(T,1);
Green_s = zeros(T,1);

%% Start reading in the first movie
```

```
sn = SelectedNames(startframe);
f_name = [dir_path '/' sn{:}];
movie = moviereader(f_name);
imgray1 = movie.read(1);
figure; %show the read in image
imshow(imgray1,[],'InitialMagnification',100);

%% Select region of interest (ROI)
%contain the interested cell
disp '--- select a rectangle around the cell ----------'
rect=fix(getrect);
rectangle('Position',rect,'edgecolor','w','Linewidth',1);
close all

%% Set background level
disp '--- select background ----------'
figure;
imshow(imgray1,[],'InitialMagnification',100);
bgr=fix(getrect);
close all
imc = imcrop(imgray1,rect);
disp '--- click the inside of the cell ----------'
figure
imshow(imc,[])
new_rectC = fix(getrect);
C1x = round(new_rectC(1)+new_rectC(3)/2);
C1y = round(new_rectC(2)+new_rectC(4)/2);
close
oldmask = zeros(size(imc));
oldmask(C1y, C1x) = 1;

%% apply to the whole movie folder
for i = startframe : endframe
        sn = SelectedNames(i);
        f_name = [dir_path '/' sn{:}];
        movie = moviereader(f_name);

        mT = uint16(zeros([size(imgray1),5]));

        for k = 1:5
            mT(:,:,k) = movie.read(k); %read in bright field image
        end
        imgray1 = mT(:,:,1);
        mT_c = mT(rect(2):rect(2)+rect(4),rect(1):rect(1)+rect(3),:);
        bgT_c = mT(bgr(2):bgr(2)+bgr(4),bgr(1):bgr(1)+bgr(3),:);
        imc = imcrop(imgray1,rect); %image cropped
        mT_c1 = mT_c(:,:,1); %select first channel

```

```
87          CellMask = mask_canny(mT_c1); %select an algorithm
88          %can be adjusted and connected to part2
89          %CellMask = mask_pattern(mT_c1,10);
90
91          bwbigMask = false(size(imgray1));
92          bwbigMask(rect(2):rect(2)+rect(4),rect(1):rect(1)+rect(3)) =
                CellMask;
93
94          %Channel 6 is captured under no lighting to serve as a background
                calibration for the green signal (channel 7), which records the
                signal for 'mAG-Geminin'. Channel 8 is captured under no lighting
                 to serve as a background calibration for the red signal (channel
                 9), which records the signal for 'mKO2-Cdt1'. We subtract the
                background channels 6 and 8 from channels 7 and 9, respectively,
                to reduce the effects of the environment on our data, improving
                the accuracy of our calculations.
95          Im_green = movie.read(6)-movie.read(7);
96          Im_red = movie.read(8)-movie.read(9);
97
98          %background level in green channel
99          bg_G = mean(mean(Im_green(bgr(2):bgr(2)+bgr(4),bgr(1):bgr(1)+bgr(3))
                ));
100         %background level in red channel
101         bg_R = mean(mean(Im_red(bgr(2):bgr(2)+bgr(4),bgr(1):bgr(1)+bgr(3))))
                ;
102
103         % get fluorescent signal
104         Green(i) = mean(mean(bwbigMask.*double(Im_green)))-bg_G;
105         Red(i) = mean(mean(bwbigMask.*double(Im_red)))-bg_R;
106         Green_s(i) = sum(sum(bwbigMask.*double(Im_green)));
107         Red_s(i) = sum(sum(bwbigMask.*double(Im_red)));
108
109         %figure, imshow(CellMask), title('segmented image');
110         s = sprintf('%03d', i);
111         %namef = [s '.png'];
112         fig = figure;
113         imshow(imc,[],'InitialMagnification',100)
114         alphamask(CellMask, [1 1 0], 0.2);
115         print(fig,s,'-dpng');
116         hold off;
117         close all;
118         %imwrite(fig,namef);
119         %close all;
120         %savesegmimg(bwbigMask, imgray1, i, dir_res, rect, bgr)
121 end
122
123 %% plot the green and red fluorescent signal curve
124 green_sum = (Green(startframe : endframe))';
```

```
125 red_sum = (Red(startframe : endframe))';
126
127 figure; % figure of red and green curve
128 %plot red
129 red_x = 1:length(red_sum);
130 red_y = red_sum/mean(red_sum);
131 %red_y= red_sum./Red';
132 red_f = fit(red_x.',red_y.','gauss2'); %apply gaussian fitting
133 %red_f = fit(red_x.',red_y.','gauss3');
134 plot(red_f,'r-',red_x,red_y); %plot the Gaussian fitting curve
135 %line(red_x,red_y,'Color','r') %plot the orignal data (in case to compare)
136 xlim([0 length(red_sum)]);
137 %ylim([0 2]);
138 grid on;
139 hold on;
140
141 %plot green
142 green_x =1:length(green_sum);
143 green_y = green_sum/mean(green_sum);
144 %green_y= green_sum./Green';
145 green_f = fit(green_x.',green_y.','gauss2');
146 %green_f = fit(green_x.',green_y.','gauss3');
147 plot(green_f,'g',green_x,green_y);
148 %line(green_x,green_y,'Color','g');
149 hold on;
150
151 %% Used to show the green and red channel signal when it is necessary
152 % imgray6 = movie.read(6);
153 % imgray7 = movie.read(7);
154 % imgray8 = movie.read(8);
155 % imgray9 = movie.read(9);
156 % imgray_green = imgray6-imgray7;
157 % imgray_red = imgray8-imgray9;
158 %
159 % imgray_green_uint8Image = uint8(255 * mat2gray(imgray_green));
160 % imgray_red_uint8Image = uint8(255 * mat2gray(imgray_red));
161 %
162 % figure;
163 % %imshow(imgray6,[],'InitialMagnification',100);
164 % imshow(imgray_green_uint8Image,[],'InitialMagnification',100);
165 % %imshow(imgray8-imgray9,[],'InitialMagnification',100);
166 %
167 % %figure;
168 % %imshow(imgray_green_uint8Image(rect(3):rect(1),rect(2):rect(4)));
169 % rect1=fix(getrect);
170 % rectangle('Position',rect1,'edgecolor','w','Linewidth',1);
171 % close all
172 %
```

```
% imc1 = imcrop(imgray_green_uint8Image,rect1);
% imc2 = imcrop(imgray_red_uint8Image,rect1);
%
% figure
% imshow(imc1,[])
% figure
% imshow(imc2,[])
```

The main code - part 2

The second part mainly works on applying different filtering and edge detection algorithms to get binary gradient images (BGIs). Prepare the BGIs to generate the cell masks.

```

%% Read in region of interest (ROI)
%preview of the select region
bgT_c1 = bgT_c(:,:,1);
mT_c1 = mT_c(:,:,1);

mT_green = mT_c(:,:,6);
mT_red = mT_c(:,:,8);
%% show the ROI cropped image
figure(1)
imshow(mT_green,[],'InitialMagnification',100)
figure(2)
imshow(mT_red,[],'InitialMagnification',100)

%% The green and red signal channel
mT_green_uint8Image = uint8(255 * mat2gray(mT_green));
mT_red_uint8Image = uint8(255 * mat2gray(mT_red));
%display general mask on green signal
g = fspecial('gaussian');
mT_green_gaussian = imfilter(mT_green_uint8Image,g,'replicate');
T_gray = graythresh(mT_green_gaussian);
mT_green_gaussian1 = im2bw(mT_green_gaussian,T_gray);
mT_green_gaussian2 = imfill(mT_green_gaussian1,'holes');
figure,imshowpair(mT_green_gaussian1,mT_green_gaussian2,'montage')
%display general mask on red signal
g = fspecial('gaussian');
mT_red_gaussian = imfilter(mT_red_uint8Image,g,'replicate');
T_gray = graythresh(mT_red_gaussian);
mT_red_gaussian1 = im2bw(mT_red_gaussian,T_gray);
mT_red_gaussian2 = imfill(mT_red_gaussian1,'holes');
figure,imshowpair(mT_red_gaussian1,mT_red_gaussian2,'montage')

%% The bright field channel
mT_c1_uint8Image = uint8(255 * mat2gray(mT_c1));
imshow(mT_c1_uint8Image,[],'InitialMagnification',100)
```

```
%close all;

%% Start apply different filtering algrithms
%gaussian filter
g = fspecial('gaussian');
mT_c1_gaussian = imfilter(mT_c1_uint8Image,g,'replicate');
T_gray = graythresh(mT_c1_gaussian);
mT_c1_gaussian1 = im2bw(mT_c1_gaussian,T_gray);
mT_c1_gaussian2 = imfill(mT_c1_gaussian1,'holes');
figure,imshowpair(mT_c1_gaussian1,mT_c1_gaussian2,'montage')
%mean (average) filter
g = fspecial('average');
mT_c1_average = imfilter(mT_c1_uint8Image,g,'replicate');
%motion filter
g = fspecial('motion');
mT_c1_motion = imfilter(mT_c1_uint8Image,g,'replicate');
T_gray = graythresh(mT_c1_motion);
mT_c1_motion1 = im2bw(mT_c1_motion,T_gray);
mT_c1_motion2 = imfill(mT_c1_motion1,'holes');
figure,imshowpair(mT_c1_motion1,mT_c1_motion2,'montage')
%disk filter
g = fspecial('disk');
mT_c1_disk = imfilter(mT_c1_uint8Image,g,'replicate');
%laplacian filter
g = fspecial('laplacian');
mT_c1_laplacian = imfilter(mT_c1_uint8Image,g,'replicate');
T_gray = graythresh(mT_c1_laplacian);
mT_c1_laplacian1 = im2bw(mT_c1_laplacian,T_gray);
mT_c1_laplacian2 = imfill(mT_c1_laplacian1,'holes');
figure,imshowpair(mT_c1_laplacian1,mT_c1_laplacian2,'montage')
%log filter
g = fspecial('log');
mT_c1_log = imfilter(mT_c1_uint8Image,g,'replicate');
%sobel filter
g = fspecial('sobel');
mT_c1_sobel = imfilter(mT_c1_uint8Image,g,'replicate');
T_gray = graythresh(mT_c1_sobel);
mT_c1_sobel1 = im2bw(mT_c1_sobel,T_gray);
mT_c1_sobel2 = imfill(mT_c1_sobel1,'holes');
figure,imshowpair(mT_c1_sobel1,mT_c1_sobel2,'montage')
%prewitt filter
g = fspecial('prewitt');
mT_c1_prewitt = imfilter(mT_c1_uint8Image,g,'replicate');

%show the filtered ROI image
figure,imshowpair(mT_c1_gaussian,mT_c1_average,'montage')
text(0,15,'Guassian and average filter','Color','white','FontSize',14);
figure,imshowpair(mT_c1_motion,mT_c1_disk,'montage')
```

```
text(0,15,'Motion and disk filter','Color','white','FontSize',14);
figure,imshowpair(mT_c1_laplacian,mT_c1_log,'montage')
text(0,15,'Laplacian and log filter','Color','white','FontSize',14);
figure,imshowpair(mT_c1_sobel,mT_c1_prewitt,'montage')
text(0,15,'Sobel and prewitt filter','Color','white','FontSize',14);


%% Apply different edge detection algorithms to different filtered ROI
close all;
%gaussian filtered ROI
filter_edge(mT_c1_gaussian);
%average(Mean) filtered ROI
filter_edge(mT_c1_average);
%motion filtered ROI
filter_edge(mT_c1_motion);
%disk filtered ROI
filter_edge(mT_c1_disk);
%laplacian filtered ROI
filter_edge(mT_c1_laplacian);
%log filtered ROI
filter_edge(mT_c1_log);
%sobel filtered ROI
filter_edge(mT_c1_sobel);
%prewitt filtered ROI
filter_edge(mT_c1_prewitt);
```

The main code - part 3

The third part used to investigate an individual cell analysis in a single frame.

```

%% Analysis an individual cell
%num = 1;
%rect=[1093,630,292,312]; %[xmin ymin width height]
%bgr=[510,126,174,198]; %testing location
%get the coordinate position of the tracking cells from 'TrackMate' plugin (ImageJ) to roughly record the suitable position of the ROI.
R_cor = csvread(Trackmate_out1.csv) %read in the estimated location for ROI
B_cor = csvread(Trackmate_out2.csv) %read in the estimated location for background
rect= R_cor(num); %Updating
bgr=B_cor(num);

imc = imcrop(imgray1,rect); %cropping
mT = uint16(zeros([size(imgray1),5]));
for k = 1:9
    mT(:,:,k) = movie.read(k);
end
```

```
bgT_c = mT(bgr(2):bgr(2)+bgr(4),bgr(1):bgr(1)+bgr(3),:);
mT_c = mT(rect(2):rect(2)+rect(4),rect(1):rect(1)+rect(3),:);

%% Now start to set the parameter according to the selected background
%preview of the select region
bgT_c1 = bgT_c(:,:,1);
mT_c1 = mT_c(:,:,1);
% figure(1)
% imshow(bgT_c1,[],'InitialMagnification',100)
% figure(2)
% imshow(mT_c1,[],'InitialMagnification',100)

%% The cropped image is converted into an eight-bits version
mT_c1_uint8Image = uint8(255 * mat2gray(mT_c1));
%imshow(mT_c1_uint8Image,[],'InitialMagnification',100)
close all;

%% use different filter to detect the edge of the image
% Sobel filter and Canny Filter
BW1 = edge(mT_c1,'sobel');
BW2 = edge(mT_c1,'canny');
BW_canny_sobel = edge(BW2,'sobel');
figure('position',[400 200 800 800]);
subplot(3,2,1)
imshowpair(BW1,BW2,'montage')
title('Sobel Filter,Canny Filter');
BW_sobel_canny = edge(BW1,'canny');
subplot(3,2,2)
imshowpair(BW_sobel_canny,BW_canny_sobel,'montage')
title('Sobel and Canny combination Filter');

%% Prewitt filter and Roberts Filter
BW3 = edge(mT_c1,'Prewitt');
BW4 = edge(mT_c1,'Roberts');
BW_prewitt_roberts = edge(BW3,'Roberts');
BW_roberts_prewitt = edge(BW4,'Prewitt');
%figure;
subplot(3,2,3)
imshowpair(BW3,BW4,'montage')
title('Prewitt Filter, Roberts Filter');
subplot(3,2,4)
imshowpair(BW_prewitt_roberts,BW_roberts_prewitt,'montage')
title('Prewitt and Roberts combination Filter');

%% Log filter and Zero-cross Filter
BW5 = edge(mT_c1,'log'); %Laplacian of Gaussian
BW6 = edge(mT_c1,'zerocross');
BW_log_zerocross = edge(BW5,'zerocross');
```

```
BW_zerocross_log = edge(BW6,'log');
%figure;
subplot(3,2,5)
imshowpair(BW5,BW6,'montage')
title('log Filter, zerocross Filter');
subplot(3,2,6)
imshowpair(BW_log_zerocross,BW_zerocross_log,'montage')
title('log and zerocross combination Filter');


%% segement using dialation methods (based on 'Canny Filter')
%use strel function to erasion and dilation of the image
se90 = strel('line', 6, 90);
se0 = strel('line', 6, 0);
BWsdil = imdilate(BW2, [se90 se0]);
figure('position',[100 200 400 400]);
subplot(2,2,1), imshow(BWsdil), title('dilated gradient mask');
%fill the hole on the dilation image
BWdfill = imfill(BWsdil, 'holes');
subplot(2,2,2), imshow(BWdfill);
title('binary image with filled holes');
%'imclearborder' to select the mian cell
BWnobord = imclearborder(BWdfill, 4);
subplot(2,2,3), imshow(BWnobord), title('cleared border image');
% smoothing (Create morphological structuring element (STREL))
seD = strel('diamond',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);
subplot(2,2,4), imshow(BWfinal), title('segmented image');


%% Segmentation using threshold value
g = fspecial('gaussian');
mT_c1_it1 = imfilter(mT_c1_uint8Image,g,'replicate');
figure;
subplot(2,2,1), imshowpair(mT_c1_uint8Image,mT_c1_it1,'montage')
subplot(2,2,2), imhist(mT_c1_it1); %see histogram data
T_gray = graythresh(mT_c1_it1); %auto-threshold
mT_c1_it2 = im2bw(mT_c1_it1,T_gray);
subplot(2,2,3), imshow(mT_c1_it2);
mT_c1_it2_fill = imfill(mT_c1_it2,'holes');
subplot(2,2,4), imshow(mT_c1_it2_fill);


%% Segmentation using adaptive method
%10 by 10 blocks
mT_c1_it3 = blkproc(mT_c1_it1,[10 10], @adaptt);
figure;
```

```
subplot(3,2,1), imshow(mT_c1_it3);
title('Apply adaptive 10 by 10 box');
mT_c1_it1_sobel = edge(mT_c1_it3,'sobel');
mT_c1_it1_canny = edge(mT_c1_it3,'canny');
subplot(3,2,2), imshowpair(mT_c1_it1_sobel,mT_c1_it1_canny,'montage')
title('Sobel Filter,Canny Filter');
%20 by 20 blocks
mT_c1_it3 = blkproc(mT_c1_it1,[20 20], @adaptt);
subplot(3,2,3), imshow(mT_c1_it3);
title('Apply adaptive 20 by 20 box');
mT_c1_it1_sobel = edge(mT_c1_it3,'sobel');
mT_c1_it1_canny = edge(mT_c1_it3,'canny');
subplot(3,2,4), imshowpair(mT_c1_it1_sobel,mT_c1_it1_canny,'montage')
title('Sobel Filter,Canny Filter');
%30 by 30 blocks
mT_c1_it3 = blkproc(mT_c1_it1,[30 30], @adaptt);
subplot(3,2,5), imshow(mT_c1_it3);
title('Apply adaptive 30 by 30 box');
mT_c1_it1_sobel = edge(mT_c1_it3,'sobel');
mT_c1_it1_canny = edge(mT_c1_it3,'canny');
subplot(3,2,6), imshowpair(mT_c1_it1_sobel,mT_c1_it1_canny,'montage')
title('Sobel Filter,Canny Filter');

%% Segmentation using Watershed segmentation
mT_c1_it4 = imtophat(mT_c1_it1, strel('disk',50));
mT_c1_it5 = imadjust(mT_c1_it4);
figure;
subplot(3,1,1),imshowpair(mT_c1_it4,mT_c1_it5,'montage');
T_level = graythresh(mT_c1_it5);
mT_c1_BW = im2bw(mT_c1_it5,T_level);
mT_c1_C = ~mT_c1_BW;
subplot(3,1,2), imshowpair(mT_c1_BW,mT_c1_C,'montage');
mT_c1_D = -bwdist(mT_c1_C);
mT_c1_D(mT_c1_C) = -Inf;
mT_c1_L = watershed(mT_c1_D);
%figure, imshow(mT_c1_L);
mT_c1_wi = label2rgb(mT_c1_L,'hot','w');
%subplot(3,1,2), imshow(mT_c1_wi);
subplot(3,1,3), imshowpair(mT_c1_it1,mT_c1_wi,'montage');

%% calculate mean and sum of red and green signals
%CellMask = sege_canny( mT_c1 );
bwbigMask = false(size(imgray1));
bwbigMask(rect(2):rect(2)+rect(4),rect(1):rect(1)+rect(3)) = BWfinal;
Im_green = movie.read(6)-movie.read(7);
Im_red = movie.read(8)-movie.read(9);
%back ground
bg_G = mean(mean(Im_green(bgr(2):bgr(2)+bgr(4),bgr(1):bgr(1)+bgr(3))));
```

```
bg_R = mean(mean(Im_red(bgr(2):bgr(2)+bgr(4),bgr(1):bgr(1)+bgr(3))));
% get fluorescent signal
Green_mean = mean(mean(bwbigMask.*double(Im_green)))-bg_G;
Red_mean = mean(mean(bwbigMask.*double(Im_red)))-bg_R;
Green_sum = sum(sum(bwbigMask.*double(Im_green)));
Red_sum = sum(sum(bwbigMask.*double(Im_red)));
```

The function code - filter edge function:

The *filter edge* function used to compare the outputs by applying different edge detection algorithms from the filtered ROI image to provide BGIs, including Sobel, Canny, Prewitt, Robert, Zero-cross etc.

```
function output_edge = filter_edge( mT_c1_test )
%edge detection algorithms

    % Sobel algorithm and Canny algorithm
    BW1 = edge(mT_c1_test,'sobel');
    BW2 = edge(mT_c1_test,'canny');
    BW_canny_sobel = edge(BW2,'sobel');
    figure('position',[400 200 800 800]);
    subplot(3,1,1)     %figure plot
    imshowpair(~BW1,~BW2,'montage')
    title('Sobel Edge Detection ,Canny Edge Detection');

    % Prewitt algorithm and Roberts algorithm
    BW3 = edge(mT_c1_test,'Prewitt');
    BW4 = edge(mT_c1_test,'Roberts');
    BW_prewitt_roberts = edge(BW3,'Roberts');
    BW_roberts_prewitt = edge(BW4,'Prewitt');

    %figure;
    subplot(3,1,2)
    imshowpair(~BW3,~BW4,'montage')
    title('Prewitt Edge Detection , Roberts Edge Detection');

    % Log algorithm and Zero-cross algorithm
    BW5 = edge(mT_c1_test,'log'); %Laplacian of Gaussian
    BW6 = edge(mT_c1_test,'zerocross');
    BW_log_zerocross = edge(BW5,'zerocross');
    BW_zerocross_log = edge(BW6,'log');

    %figure;
    subplot(3,1,3)
    imshowpair(~BW5,~BW6,'montage')
    title('log Edge Detection , zerocross Edge Detection');
end
```

The function code - alpha mask function:

The *alphamask* function works for drawing the cell mask on the bf channel of the original imaging data, the appearance of the mask can be adjusted by setting the *colour* parameter and *transparent* parameter. This function is based on Andrew Davis's scripts, 2012, adapted by J.Yu.

```
function hOVM = alphamask(bwMask, colour, transparency, axHandle)
% ALPHAMASK:  Overlay image with semi-transparent mask
%
% Overlays a semi-transparent mask over an image.  By default the
%   currently displayed figure is overlain.
% Options include overlay colour and opacity.
% Returns a handle to the overlay mask.
%
% Usage:
%   hOVM = alphamask(bwMask, [colour, transparency, axHandle])
%           bwMask: logical matrix representing mask
%           colour: vector of three rgb values in range [0, 1] (optional;
    default [0 0 1])
%     transparency: scalar in range [0, 1] representing overlay opacity (
    optional; default 0.6)
%         axHandle: handle to axes on which to operate (optional; default
    current axes)
%             hOVM: handle to overlay image is returned
%
% See also IMSHOW, CREATEMAS
% Check arguments
if ~exist('bwMask', 'var') || ~ismatrix(bwMask), error('bwMask matrix is a
    required argument'); end;
if ~exist('colour', 'var'), colour = [0 0 1]; end;
if ~exist('transparency', 'var'), transparency = 0.6; end;
if ~exist('axHandle', 'var'), axHandle = gca; end;
if ~isvector(colour) || ~isscalar(transparency) || ~ishandle(axHandle),
    error('One or more arguments is not in the correct form'); end;
maskRange = max(max(bwMask))-min(min(bwMask));
if maskRange ~= 1 && maskRange ~= 0, error('bwMask must consist only of the
    values 0 and 1'); end;

% Create colour image and overlay it
rgbI = cat(3, colour(1)*ones(size(bwMask)), colour(2)*ones(size(bwMask)),
    colour(3)*ones(size(bwMask)));
hold on,
hOVM = imshow(rgbI, 'Parent', axHandle);
set(hOVM, 'AlphaData', bwMask*transparency);        % use mask values as
    alpha channel of overlay
hold off;
```

The function code - mask pattern function:

The *mask pattern* function help to change the cell mask into an adaptive version mask, so that we can reduce the required disk to store the mask, also increase the calculation speed.

```
function CellMask_Pattern = mask_pattern( mT_c1,n ) %n is the block size
    %give a gaussian filter
    g = fspecial('gaussian');
    mT_c1_gau = imfilter(mT_c1,g,'replicate');

    %apply the adaptt daughter function
    BWs = blkproc(mT_c1_gau,[n n], @adaptt);

    BWdfill = imfill(BWs, 'holes');

    %'imclearborder' to select the mian cell
    BWnobord = imclearborder(BWdfill, 4);

    % smoothing (Create morphological structuring element (STREL))
    seD = strel('diamond',1);
    BWfinal = imerode(BWnobord,seD);
    BWfinal = imerode(BWfinal,seD);
    CellMask_Pattern = BWdfill;

end
```

The function code - adaptt function:

The *adaptt* function is a daughter function of the mask pattern function. Apply the adaptive method to a small region of pixel.

```
function[y] =adaptt(x)
    if std2(x) <8
        y = ones(size(x,1),size(x,2));
    else
        %y = im2bw(x, graythresh(x));
        y = zeros(size(x,1),size(x,2));
    end
```

The function code - scroll movie function:

The *scroll movie* function can help read and review the '.movie' file from the folder.

```
function [] = scroll_movie(filename)

if nargin < 1
    [filename, pathfile] = uigetfile('*.movie');
    filename = fullfile(pathfile,filename);
end
```

```
movieobj = moviereader(filename);
videofig(movieobj.NumberOfFrames, @redraw, movieobj, round(movieobj.
    FrameRate));
redraw(movieobj,1);

end
```

The function code - redraw function:

The *redraw* function uses to adjust the image fame from the '.movie' file.

```
function redraw(movieobj,frame)

IM = movieobj.read(frame);
im = IM;
% imshow(imadjust(im,stretchlim(im,0.0001)),[])
imshow(im,[])
set(gca,'units','normalized');
text(0,0,['frame ',num2str(frame)],'Color','r','Units','pixels','
    VerticalAlignment','bottom');

end
```

VII .2 Scripts in ImageJ

Convert customized '.movie' file into '.avi'

The '.movie' files are stored in the folder named by the date of the experiment in ordered. Because there are multiple stacks in each single frame, the separate videos for 'bf', 'green' and 'red' channel are unavailable to watch directly. We wrote a Macros scripts in ImageJ to automatic transfer the stacked '.movie' file into three separate channel videos- bf, green and red, here is the attached scripts.

```

//Turn on the batch processing
setBatchMode(true);

//set the store directory as a variable
directory = "S:\\wz271\\FucciCell\\270417\\petri0\\";

//get file names
fileNames = getFileList(directory);
//fileNums = getLength(fileNames)
fileNums = 40

for (i = 0; i< fileNums;i=i+1) {

        position_bf = substring(fileNames[i],0,11) +"\\bf000.tiff file=bf
            sort";
```

```
16          //position_red =  substring(fileNames[i],0,11) +"\\red000.tiff file=
               red sort";
17          //position_green =  substring(fileNames[i],0,11) +"\\green000.tiff
               file=green sort";
18
19          //set the iput file
20          input_open = "open=" + directory + position_bf;
21          //print(input_open);
22
23          //read in the image files
24          run("Image Sequence...", input_open);
25
26          //run("Brightness/Contrast...");
27          //run("Enhance Contrast", "saturated=0.35");
28
29          //convert and save the image file to the movie file (.avi)
30          output_save = "compression=JPEG frame=10 save=" + directory  +
               substring(fileNames[i],0,11)+"_bf.avi";
31          //print(output_save);
32
33          //save output movie in the same directory
34          run("AVI... ", output_save);
35
36          //close in case runout the memory
37          //close();
38          run("Close All");
39          }
```

VII .3 Scripts in R

Scripts in R are used to plot the results figures. The input files are the output files from the main program.

```
1  #=======================================
2  #==================Testing==============
3  #=======================================
4  setwd('/Users/JiamingYU/Desktop/SummerProject/pos_0/output/
      analy_cellscan0_petri0_position000.17May2016_12.38.36/')
5  #setwd('/Users/JiamingYU/Desktop/SummerProject/pos_1/output/')
6  #setwd('/Users/JiamingYU/Desktop/SummerProject/pos_2/output/')
7
8  data_green_sum = read.table('Green_sum.txt')
9  data_red_sum = read.table('Red_sum.txt')
10 data_green_mean = read.table('Green_mean.txt')
11 data_red_mean = read.table('Red_mean.txt')
12 plot(data_green_sum$V1/mean(data_green_sum$V1),col='green',type = 'l',
13        ylab='Relative value',xlab = 'Time index',lty=1,lwd=2)
14 lines(data_red_sum$V1/mean(data_red_sum$V1),col='red',lty=2,lwd=2)
15 grid (10,10, lty = 6, col = "cornsilk4")
```

```
16
17
18 #=============================================================
19 #==================Plot the matlab output data================
20 #=============================================================
21 library(R.matlab)
22 setwd('/Users/JiamingYU/Desktop/SummerProject/OutputData')
23
24 data_mat <- readMat('Analysis_data1.mat')
25 pdf('Analysis_data1_outputs.pdf',width=6,height=6,paper='special')
26 for (i in c(76:80)) {
27 test1 = as.vector(data_mat$output.variable[[i]])[[1]][1,]
28 test2 = as.vector(data_mat$output.variable[[i+10]])[[1]][1,]
29
30 plot(test1/mean(test1),col='green',type = 'l',
31      ylab='Relative value',xlab = 'Time index',lty=1,lwd=2,
32      main = paste('Test analysis data1, Position = ',i-70))
33 lines(test2/mean(test2),col='red',lty=2,lwd=2)
34 grid (10,10, lty = 6, col = "cornsilk4")
35 legend('topleft',c('Green sum','Red sum'),col=c('green','red'),lty= c(1,1),
     lwd=2,cex=1,bty = 'n')
36 }
37 dev.off()
38
39 data_mat <- readMat('Analysis_data2.mat')
40 pdf('Analysis_data2_outputs.pdf',width=6,height=6,paper='special')
41 for (i in c(77:88)) {
42   test1 = as.vector(data_mat$output.variable[[i]])[[1]][1,]
43   test2 = as.vector(data_mat$output.variable[[i+11]])[[1]][1,]
44   plot(test1/mean(test1),col='green',type = 'l',
45        ylab='Relative value',xlab = 'Time index',lty=1,lwd=2,
46        main = paste('Test analysis data2, Position = ',i-70),ylim = c(0,5))
47   lines(test2/mean(test2),col='red',lty=2,lwd=2)
48   grid (10,10, lty = 6, col = "cornsilk4")
49   legend('topleft',c('Green sum','Red sum'),col=c('green','red'),lty= c(1,1)
       ,lwd=2,cex=1,bty = 'n')
50 }
51 dev.off()
52
53
54
55 #=========================================
56 #=======red channel data==================
57 #=========================================
58 #Draw the compare of the differnent method (red signal)
59 N=120
60 x = seq(0, N, length=N)
61 library(R.matlab)
```

```
setwd('/Users/JiamingYU/Desktop/SummerProject/OutputRed')
data_red <- readMat('Gaussian_results_red.mat')

#the manual approach
a = as.vector(data_red$output.variable[[1]])[1]
b = as.vector(data_red$output.variable[[1]])[2]
c = as.vector(data_red$output.variable[[1]])[3]
f1_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)
plot(x,f1_red,ylim = range(0,5),type='l',col='red',pch=1,xlab ='Frames',
    ylab = 'Cdt1(Red) signal',lwd=3)
title(main = 'Auto- and manual- method compare (red channel)')
abline(v=23,lwd=1,col="red",lty=2)

#Auto mask1
a = as.vector(data_red$output.variable[[2]])[1]
b = as.vector(data_red$output.variable[[2]])[2]
c = as.vector(data_red$output.variable[[2]])[3]
f2_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)+a[3]*exp
    (-((x-b[3])/c[3])^2)
points(x,f2_red,ylim = range(0,3),type='l',col='green',pch=2,lty=2,lwd=2)
abline(v=b[1],lwd=1,col="green",lty=2)

#Auto mask2
a = as.vector(data_red$output.variable[[3]])[1]
b = as.vector(data_red$output.variable[[3]])[2]
c = as.vector(data_red$output.variable[[3]])[3]
f3_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)+a[3]*exp
    (-((x-b[3])/c[3])^2)
points(x,f3_red,ylim = range(0,3),type='l',col='orange',pch=3,lty=3,lwd=2)
abline(v=b[1],lwd=1,col="orange",lty=2)

#Auto mask3
a = as.vector(data_red$output.variable[[4]])[1]
b = as.vector(data_red$output.variable[[4]])[2]
c = as.vector(data_red$output.variable[[4]])[3]
f4_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)+a[3]*exp
    (-((x-b[3])/c[3])^2)
points(x,f4_red,ylim = range(0,3),type='l',col='blue',pch=4,lty=4,lwd=2)
abline(v=b[1],lwd=1,col="blue",lty=2)

grid (10,10, lty = 6, col = "cornsilk4")
legend('topright',c('Manual-method','Auto-1(Gaussian-canny)','Auto-2(
    Laplacian-sobel)','Auto-3(Motion-zerocross)'),col=c('red','green','orange
    ','blue'),
       lty= c(1,2,3,4),lwd=2,cex=1,bty = 'n')

#=========================================
#=======green channel data=================
```

```
#=========================================
#Draw the compare of the differnent method (green signal)
N=120
x = seq(0, N, length=N)
library(R.matlab)
setwd('/Users/JiamingYU/Desktop/SummerProject/OutputRed')
data_green <- readMat('Gaussian_results_green.mat')

#the manual approach
a = as.vector(data_green$output.variable[[1]])[1]
b = as.vector(data_green$output.variable[[1]])[2]
c = as.vector(data_green$output.variable[[1]])[3]
f1_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)
plot(x,f1_red,ylim = range(0,5),type='l',col='green',pch=1,xlab ='Frames',
    ylab = 'Geminin(Green) signal',lwd=3)
title(main = 'Auto- and manual- method compare (green channel)')

#Auto mask1
a = as.vector(data_green$output.variable[[2]])[1]
b = as.vector(data_green$output.variable[[2]])[2]
c = as.vector(data_green$output.variable[[2]])[3]
f2_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)+a[3]*exp
    (-((x-b[3])/c[3])^2)
points(x,f2_red,ylim = range(0,3),type='l',col='red',pch=2,lty=2,lwd=2)

#Auto mask2
a = as.vector(data_green$output.variable[[3]])[1]
b = as.vector(data_green$output.variable[[3]])[2]
c = as.vector(data_green$output.variable[[3]])[3]
f3_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)+a[3]*exp
    (-((x-b[3])/c[3])^2)
points(x,f3_red,ylim = range(0,3),type='l',col='orange',pch=3,lty=3,lwd=2)

#Auto mask3
a = as.vector(data_green$output.variable[[4]])[1]
b = as.vector(data_green$output.variable[[4]])[2]
c = as.vector(data_green$output.variable[[4]])[3]
f4_red = a[1]*exp(-((x-b[1])/c[1])^2)+a[2]*exp(-((x-b[2])/c[2])^2)+a[3]*exp
    (-((x-b[3])/c[3])^2)
points(x,f4_red,ylim = range(0,3),type='l',col='blue',pch=4,lty=4,lwd=2)

grid (10,10, lty = 6, col = "cornsilk4")
legend('topright',c('Manual-method','Auto-1(Gaussian-canny)','Auto-2(
    Laplacian-sobel)','Auto-3(Motion-zerocross)'),col=c('green','red','orange
    ','blue'),
       lty= c(1,2,3,4),lwd=2,cex=1,bty = 'n')
```

VIII Appendix II: Additional figures

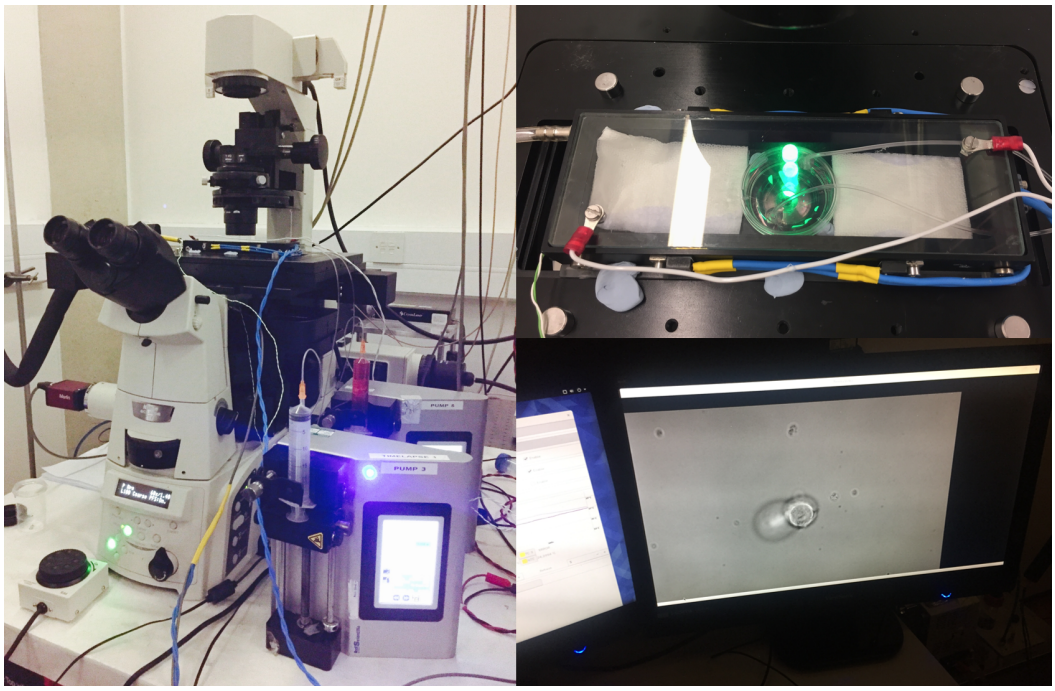

Figure 23: Photos for FUCCI experiment set up

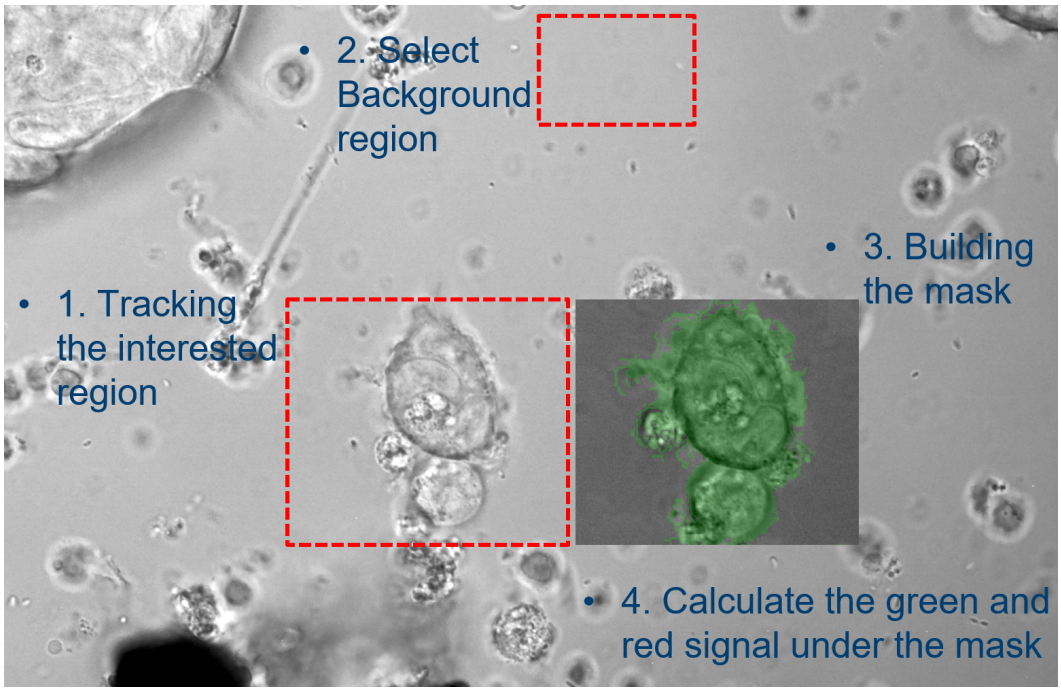

Figure 24: General process for automatic image analysis