COMP9517 Group Project Report

Yanyun WU
z5191449

Yue ZHANG
z5232113

Rifu SU
z5232173

Jianheng QIAN
z5275335

Ruoxi BAO
z5314772

***Abstract*—Microscopy has provided a unique observe complex multi-cellular processes, such as cell motion and cell division. Segmenting and tracking cells from time-lapsed micrographs requires each cell to be identified and tracked through sequential microscopic snapshots. Although recent endeavours to automate this process have improved cell detection rates, manual identification of the same cells is still the most reliable technique. However, when datasets become very large, it is time-consuming and challenging for researchers to investigate all cells manually. This article presents an automated method for cell segmentation, tracking and motion analysis with reliability compared to the hand-operated way.**

Keywords—cell segmentation, cell tracking, cell motion analysis, time-lapsed micrographs

I. INTRODUCTION

In modern biology research, studying how cells move, divide, and interact in cell populations is a series of fundamental questions. Biologists can use advanced electron microscopy imaging technology (time-lapse microscope images) to track biological cells and generate image sequences to solve these problems. However, usually, these images sequences contain a large number of microscopy images of cells. It is cumbersome and time-consuming of manual tracking methods that are used to process these images. In addition, when the number of cells is enormous, accuracy is challenging to guarantee. It is almost impossible to achieve in the process. Hence, it is necessary to use computer vision methods for the automatic tracking of biological cells. Achieving automatic tracking of cells can make the process more accurate and efficient.

In this project, we propose to use computer vision methods to automatically segment and track cells and perform subsequent quantitative analyses of cell movement in time-lapse microscope images.

The microscope image dataset analysed in this project is composed of images in four standard files and images in two GT folders (Ground truth). Tif images with image sizes below 1M in files other than GT files can be read directly. Generally, cell tracking will train machine learning or neural network models (supervised methods) and then use the model to predict. However, we used supervised methods to achieve the result of this task is unsatisfactory. The number of images (with label) in the GT folder is too rare to train a high accuracy model. Therefore, we still use the traditional algorithm to complete this task.

This project can be divided into segmentation and tracking in task1 and analysis of cell motion in task2. In task1, firstly, read the image, then check the image range. After that, we need to make a contrast stretch because the image contrast is small. Finally, remove the noise of the images by erosion and dilation by using thresholding, and then save the images. The next step is segmentation that we use Watershed. Finally, track all the cells over time and show their trajectories as overlays. For tracking, the specific first step is to find the centre of the cells, label it and save the diagram, and then loop through all the frames, recognise the same cell, label it, and label the trajectories simultaneously. In task2, to count the number of cells in an image, we only need to fetch the length of the list that contains matched cells. To calculate the average size(pixels) of every image, we ignore cells on the boundary of the image. After that is dividing cells. Cv2.fitEllipse(), ellipse fitting function, used to detect cell shape. The role is to determine whether the cell is currently dividing. Find the dividing cell and mark it with the cv2.circle function. Check if there is a cell of a similar size and state, such that we define it as a parent cell. Cv2.fitEllipse() and get_displacement() functions are utilised to obtain the shape and distance, which are also in use to determine whether to find a new dividing cell. Find the divided cell and mark it with the cv2.circle function.

II. LITERATURE REVIEW

Automatic cell segmentation and tracking have been indispensable in living cell studies, demanding high accuracy and efficacy in cell detection, processing, and further analysis. Along with the development of computer vision techniques, much research has been done in this field over the years.

In time-lapse microscopy images, one of the essential aspects of modern cell image experiments is automatically tracking and analysing the motion of cells. Nowadays, more research about cell detection and tracking applies cell features-driven models, segmentation-driven methods, and machine learning techniques. To deploy analysis of images, it is crucial to have a sketch of the datasets at first, then try to extract information and features that they contain. In addition, utilising open-source tools as well as commercial ones could bring existing image processing algorithms to another level.

As for cell segmentation, Tomas Vicar et al. [1] mentioned several necessary image processing steps due to the transparent nature of cells. The authors entailed that traditional contrast enhancement could introduce various artifacts like halos, shade-off and non-uniform shadow-cast artifacts. To suppress such artifacts, they proposed more procedures to improve segmentation performance. The authors first performed an analysis with commercial 'all-in-one' tools, and the only algorithm that outperforms others is Fogbank, which is versatile and easy to use. However, the 'all-in-one' tool has its limitations for data without labels, particularly for the presence of contrast-enhancing artifacts in microscopic images. Therefore, they suggested extending the segmentation process into four steps: image reconstruction, background segmentation, cell detection and segmentation tailored to various microscopic techniques. In the image reconstruction step, the authors showed that a simple morphological top-hat filter could deal with most cases instead of complicated strategies like level-set-based approaches. In the next phase of workflow, they compared several methods for foreground-

background segmentation, where simple thresholding proved to be the most efficient strategy for cell segmentation. Whereas J. Yan et al. [2] applied watershed and hybrid merging techniques for cell segmentation and achieved a decent result.

Once cell segmentation is accomplished, the next stage is to detect individual cells and track their motions. Automatic cell tracking comes with different challenges from image acquisition technique difference, complex cellular topology, and uneven activity of the cell [3]. J. Yan et al. [2] utilised the distance and size of cells for tracking, but this method may cause errors in cell detection and tracking due to the lack of knowledge about the cell size and properties. To overcome such challenges, Dewan, M.A.A. et al. [4] proposed to assign a unique ID to each cell to improve cell tracking and movement analysis. For implementation, each of the cells is identified by its ID number and one frame number, where it locates. Based on that, the authors categorised their cells into four types. In this way, it is easier to track cells in fixed patterns, and they achieved a good result as expected by employing this method. Considering that our dataset is quite different from [2] and [4], we decide to apply the distance and ID markers from the two papers for cell tracking.

This paper develops a method for automated cell segmentation and tracking while addressing the challenges above. We utilise distance to measure cells, which facilitates our method to identify cells during tracking accurately. Furthermore, a tree structure to detect the cell centroid is employed, along with an ellipse fit function to determine whether a cell is dividing. All these factors help increase the accuracy of the method for cell segmentation, tracking and motion analysis presented in the paper.

III. METHODS

The methods applied in this project consists of four main steps: image pre-processing, cell segmentation, cell tracking and cell motion analysis. The image pre-processing step provides necessary image enhancements for the following segmentation process. The cell segmentation step finds contours of cells and segments the cells in different images. The cell tracking step is to record the trajectory of every cell for each image in a sequence. With all the means we gain from the above three steps, we can facilitate the process of cell motion analysis, such as counting cells and determining cell divisions.

The datasets of this project are petty, and it brings challenges to keep as many cells as possible after segmentation and maintain the precision of tracking cells simultaneously. In order to improve the cell segmentation performance, we have compared several methods such as Watershed and traditional morphology methods in the image pre-processing step. In the step of segmenting cells, we encounter the challenge of determining an appropriate threshold value of cell size for segmentation to keep as many cells as possible and ensure cell tracking performance. In the cell tracking step, obstacles are raised as finding correct cells correspondence between images in a sequence because the distance is the only feature in our method to identify whether one cell is a mother cell or a daughter cell after cell divisions. On top of that, in the cell motion analysis step, it is necessary to employ topology features to detect cell division. With the four steps that we design, we have built up a decent cell activity monitor system.

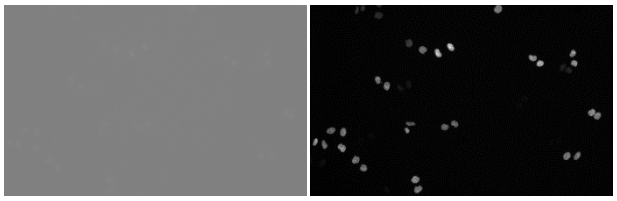
A. *Image Pre-processing*

There are several reasons to process the original images before other implementations properly. One reason is that the image pre-processing is to increase the contrast of the original microscopy images and enhance the contours of the cells.

Meanwhile, there are noises in images, and they will significantly interfere with the experimental results if we do not get rid of them. Hence, we need to reduce as much noise in images as possible. After reading the original image as a 16-bit image, we have checked the image properties by examining both the largest and the smallest pixel values in images. The inputs are grey images. The pixel value for the first frame in Sequence 01 is from 32995 to 34996. In order to improve the contrast of the image, we use the Contrast Stretching method. The contrast in an image measures the range of intensity values and is the difference between the maximum and minimum pixel values. The contrast range of an 8-bit image is from 0 to 255. Anything less than that means the image has lower contrast than possible. Assume that "I" is the original input image and "O" is the output image. Let "a" and "b" be the minimum and maximum pixel values allowed (for an 8-bit image, that means a = 0 and b = 255) and let "c" and "d" be the minimum and maximum pixel values in "I". Then the following function produces the contrast stretched image "O":

$$O(x,y) = (I(x,y) - c)\left(\frac{b-a}{d-c}\right) + a) \qquad (1)$$

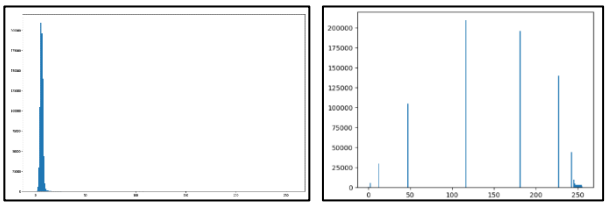
We transform it from RGB to grey and stretch it to an 8-bit image. The original image is dusty, and we can hardly see any cells. After stretching, it is easy to tell that the images' contrast is enhanced, making cells more identifiable than before. Fig. 1(a) is the original input, and Fig. 1(b) is the image after stretching.



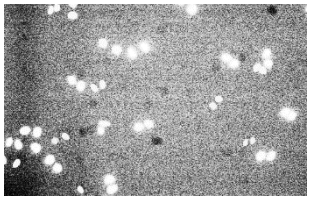
(a) (b)

Figure. 1: Comparison between before and after pre-processing. (a) Original image. (b) Image after stretching.

The brightness of most cells is improved, but some cells are still not clear enough. We attempt to use histogram equalisation to improve the brightness of such cells. Fig. 2(c) shows the histogram of Fig. 1(b), and we can tell that they are very centralised, and it is feasible to perform a histogram equalisation. The histogram after equalisation is Fig. 2(d), and the distribution is visually better. However, even though the contrast is further enhanced, this process also amplifies the noises in the image, as shown in Fig. 2(e). Therefore, we did not introduce the histogram equalisation method for image pre-processing in the end.



(c) (d)



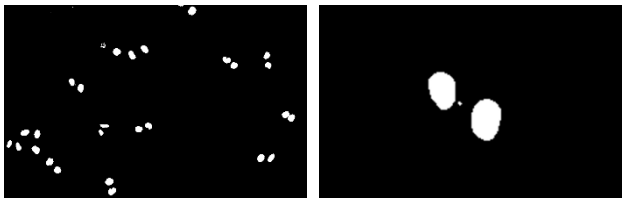
(e)

Figure. 2: Comparison between before and after histogram equalisation (c) Original histogram. (d) Histogram after histogram equalisation. (e) Image after histogram equalisation processing.

B. Cell Segmentation

After the pre-processing, we proceed to segment cells. The cell segmentation step is to detect contours of all cells and only keep cells that we think are valid for the following experiments. It is also worth addressing that finding a proper measure for determining what cells we should keep is challenging. We expect to eliminate the cells which are incomplete around the border are or too small.

It is noticeable that the image after stretching is still not clear enough to filter out the cells from the background. To find a solution to this problem, we start attempts by finding an approximate estimate of the cells. The cells are the only objects that we care about in the image. The original image not only includes the target object, the cells in other words, but also the background and noises. We desire to extract the target object from the multi-value digital image directly. For this purpose, we introduce the Otsu binarisation method to improve the image clarity further. After binarization, we have the output shown in Fig. 3(a).



(a) (b)

Figure. 3: Cell segmentation examples. (a) Image after Otsu processing. (b) White noises in the image.

Furthermore, we would like to eliminate any little white noise, as shown in Fig. 3(b) from Fig. 3(a). To accomplish this, we deploy the traditional morphology method. In this way, we will perform an erosion implementation followed by a dilation implementation.

The working mechanism of erosion is just like soil erosion. This method will erode the boundaries of the foreground object and keep the foreground in white. One of the arguments is kernel size, which will slide through the image (as in 2D convolution). However, when setting kernel size to 1, whether the pixel value in the original image is 1 or 0, it will always be detected as 1.

On implementing erosion to the binary image, the value of the output pixel is the maximum pixel value of all pixels in the neighbourhood, which makes the object more visible and fills minor voids in the objective cells. Then the dilation's output pixel is the minimum pixel value of all pixels in the neighbourhood.

Dilation is the opposite of erosion. Under the kernel sliding, if it finds at least one "1" in pixel, all pixels will be "1". According to this concept, implementing this method will enhance the white region or extract more features in the foreground.

In general, the utilisation of erosion follows by dilation since erosion can remove white noise and shrink the object, while the practice of dilation is to connect our separated object. After using the traditional morphology method to remove noises and small objects, only substantial objects remain in images. In the end, we have clear foreground and background successfully separated.

We also attempt to deploy the Watershed method for cell segmentation.

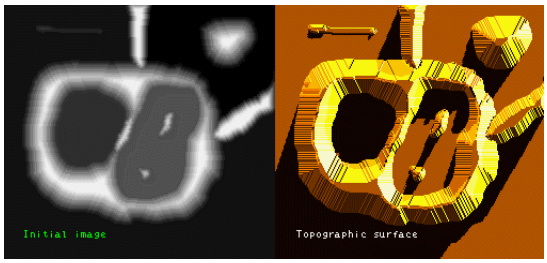
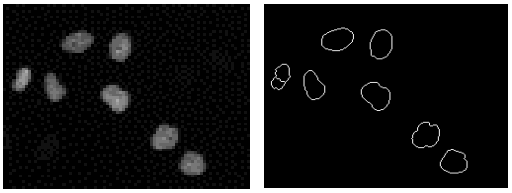


Figure. 4: Visualising the Watershed: The left image can be topographically represented as the image on the right. [5]

The Watershed method is a transformation that aims to segment the regions of interest in a grayscale image. It treats the image as a topographic map, with the intensity of each pixel representing the height. For instance, dark areas are considered to be 'lower' and act as troughs, whereas bright areas are 'higher' and act as hills or mountain ridges. This method is advantageous when two regions of interest are close to each other; that is, their edges touch.

The watershed algorithm will start from the Otsu thresholding and opening as we did. Then we extract the sure foreground by distance transform and connected components. Then call the cv2.watershed. Nevertheless, the results there is over-segmentation with Watershed. Except for that, there is a slight difference between the two algorithms. The cells generated by Watershed may have some improvement on the cell shape detection. As both algorithms are using Otsu thresholding, the numbers of the cells detected is almost the same. We choose the traditional Otsu algorithm to avoid the cell numbers increasing caused by over-segmentation.

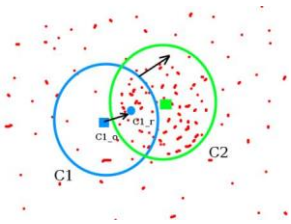


(a) (b)

Figure. 5: Comparison between Watershed and Otsu method. (a) Image after Watershed processing. (b) Image after Otsu processing.

Fig. 5(a) is the watershed output; image Fig. 5(b) is the traditional Otsu output. We can see that the cell on the left was identified as two cells.

On top of that, we set up experiments with the MeanShift method as well.



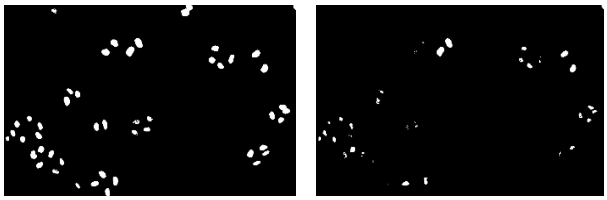
(c)

Figure. 5(c): MeanShift illustration. [6]

MeanShift is a clustering algorithm that assigns pixels to clusters by iteratively shifting points towards the modes in the feature space, where a mode is a position with the locally highest number of data points (highest density).

Fig. 5(c) is the conception of MeanShift. The blue circle is the original region. As you can see, the centre point of this window is C1_r, so we need to slide the blue circle toward C1_r. By the iteration of moving, we can get the dense pixel distribution circle(C2).

We deployed the MeanShift in the first try, which generated the output Fig.5 (b). As the bandwidth is different for each image, the fixed value would give a worse performance in some frames. So we add the estimate_bandwidth from sklearn before the MeanShift, such that the bandwidth could adjust by itself. After adding the estimate function, we have Fig.5(e), which identifies more cells compared with simply introducing the MeanShift. But the output is still not quite ideal as we want. Fig. 5(c) is generated with our algorithm, and Fig. 5(e) and Fig.5(e) is generated with MeanShift. Results show that the accuracy of the MeanShift algorithm is lower, and the cells identified are not very complete. Besides this, MeanShift is more time-consuming compared with Otsu thresholding.



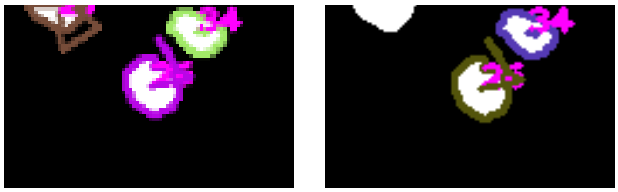
(d) (e)

(f)

Figure. 5: Comparison between Otsu and MeanShift method. (d) Image after Otsu processing. (e) Image after MeanShift processing without estimate bandwidth. (f) Image after MeanShift processing with estimate bandwidth.

By comparing traditional Otsu thresholding with Watershed and MeanShift, although Watershed has better segment integrity of the cells, it is also easy to over-segment. The number of cells recognised by MeanShift is not good enough, and the segmented integrity is not satisfactory. So we choose the traditional Otsu method for the cell segmentation.

Next, we remove the cells on edge. As the cell compromised by the border cannot be counted in, we use clear_boarder (From skimage.segmentation) to clear the labels. As we can see, after cleaning, the cells' contours will not show up.



(g) (h)

Figure. 6: Comparison between edge cells before and after border cleaning. (g) Image after before cleaning. (h) Image after cleaning.

Then we find the contours from threshold images and label them with different colours and a unique id. For the contours, we have already finished the binary image, and we can use the OpenCV function cv2.findcontour to get the contours of cells in different images. The retrieval mode we choose is cv.RETR_TREE, this parameter will help us ultimately establish the hierarchical structure of contours. We also need a parameter which is cv2. CV_CHAIN_APPROX_SIMPLE, this will only save the inflection point information of the contour, save all the points at the inflection point of the contours into the contour's vectors, and the information points on the straight line between the inflection point and the inflection point are not retained. As motioned before, we do not need some cells that are too small, so in this step, we can calculate the area of the cells according to the contours and only keep the ones with areas larger than 30, for it is a visible size. Besides the area of cells, we can calculate the centroid of each cell according to the contours.

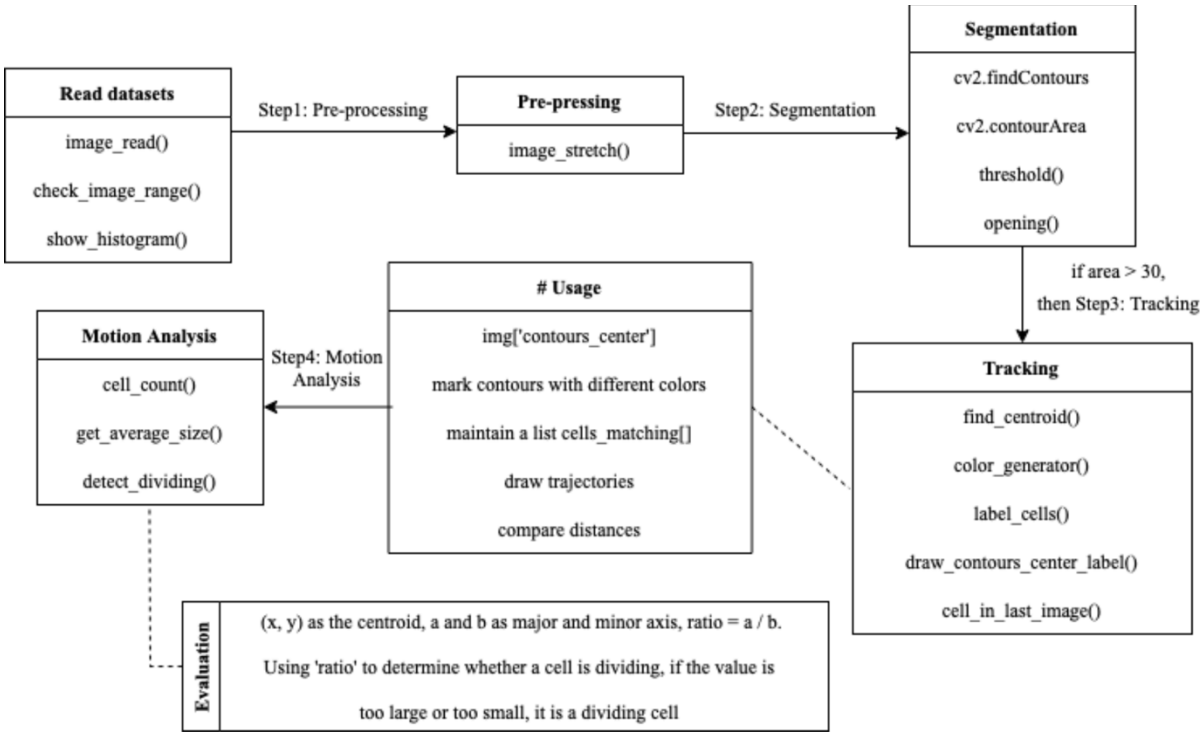


Figure. 7: Workflow of methods implementation

From contours, we can get the centroid of each cell, according to the formula:

$$m_{00} = \sum_{x=-r,y=-r}^{r} I(x,y) \quad (2)$$

$$m_{01} = \sum_{x=-r,y=-r}^{r} y \times I(x,y) \quad (3)$$

$$m_{10} = \sum_{x=-r,y=-r}^{r} x \times I(x,y) \quad (4)$$

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}) \quad (5)$$

Regarding the pixel of the image as a density function $f(x,y)$, the expectation E of the pixel is calculated, which is the moment of the image (moment of origin). We can use cv2.moments to get the moment. According to the formula, we can calculate the centroid C of the contour of the cell.

C. Cell Tracking

The cell tracking step utilises the centroids of cells to record coordinates of cells in the frame, then mark each cell with a unique ID also generate different colours for their contours. We use a random algorithm to generate the label colour of each cell and ensure that the colours of all cells are different. Then we would like to label the cells on each picture with a numerical number. However, we encounter an obstacle, which we must ensure that the same cell on each picture will always exist. At the same time, if a new cell is newly appearing in the following image, we desire to give it a new label, which will not collide with other cells before. Hence, we need a method to detect whether a cell is a new cell. We use an iterative method to update the cell in each snapshot. Since the cells are moving, we could calculate the distance between the two pictures. The formula is as follows:

$$dist = \sqrt[2]{(a_x - b_x)^2 + \left(a_y - b_y\right)^2} \quad (6)$$

where "a" and "b" are centres of the cells, "x" and "y" are the location of the centre in the image. The unit is in pixels.

Set a parameter called DIST, which is the value of the moving distance between two cells. We can recognise that they are the same cell in two images. Otherwise, we assign a new label and colour to that cell.

The next task is to track the trajectory of each cell and map out the state of activity. Suppose that we are going to track a specific point on the boundary of a cell. If the cell rotates and other irregular activities happen, our tracking will be compromised. For this issue, we can use the centroid point that we obtained for each cell before. Use centroid point as our tracking point and draw the trajectory according to the movement distance of the centroid of each cell. We also adopt an iterative update method. This is similar to we estimate whether a cell is a new cell or not. After estimation, if it is a new cell, we will build a list to track the trajectory and take the centre of mass as the first element; Otherwise, we can add the position of the centre of mass of this cell to our list at this time. After we finish all the iterations, we only need to move according to the coordinates of the centroid of each cell to draw its moving trajectory.

D. Cell Motion Analysis

There are several tasks in the cell motion analysis section, which are respectively calculating the number of cells, cell average size of all cells, the average displacement of all cells and determining whether a cell is dividing. In order to count the number of cells in an image, we only need to fetch the length of the list that contains matched cells. To get the average size of cells in a snapshot, the program should ignore cells on the boundaries of the image, where such cells may not be complete. In this way, we only utilise eighty percentage of every image for this calculation while the performance is not compromised.

However, the most important two parts in this section are cell movements and cell division. The cell movement between two frames can be justified by the distance and shape. We keep updating the position of the cells in the current frame and

compare it with the last frame. If the cells in the current frame have the Euclidean distance of less than 21 pixels, we will mark them as the same cell from the last frame. The new cell will inherit the ID and colour from the old cell. The larger distance would be mismarking the unrelated cells nearby. The smaller distance would fail to recognise the cells. The other way to recognise the cell is by shape. As we observed, the cells' shapes are incomplete, as shown in Fig. 8(a) and Fig. 8(b), which are consecutive frames taken from Sequence01, the potential misjudgement could be increased. Therefore, we only take the distance as the criteria for recognising.

(a) (b)

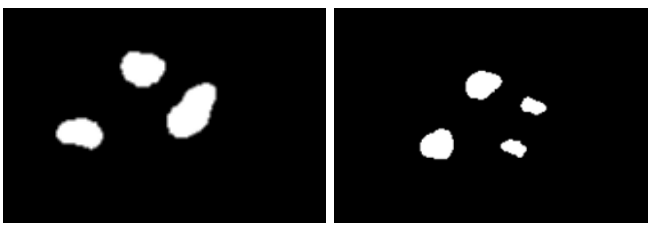
Figure. 8: Incomplete cell shapes.

With the cell recognised, we can calculate the average displacement of the cells in the current frame with the formula:

$$mean_{displacement} = \frac{\sum movement_i}{n} \quad (7)$$

where *n* is the number of the cells moved in the current frame and movement is the Euclidean distance the same cell moved between two frames.

As we have already identified the same cells from the last frame, we could differentiate the cells that prompt. We take those cells and find the nearest cells around them. The increased number of the cells may be detected because the cell is dividing or the brightness of the cell increases suddenly. Fig. 9(d) shows the newly divided cells. We can observe that these cells are close in the distance and similar in morphology. Therefore, by making the distance and morphology the criteria, we can remove the error in detecting new cells caused by cell brightness increasing. For each new cell, we check if there exist the cells within a Euclidean distance of 30 pixels. If it exists, we treat new cells and the cells nearby as ellipses. As the cells are dividing are similar ellipse with the almost same direction, we generate the mask for the cell by cv2.fitEllipse, which will give the ellipse a ratio (the longer edge of the ellipse divides the shorter edge) and the angle of rotation. As long as the cells' ratio differences are within 0.3 and the angle difference within 25 degrees, we label the pair as dividing. After the dividing cells are identified, we check if it is still dividing in the next frame. If the ellipses' ratio is bigger than 1.5, we would decide that it is not in dividing and remove the label from it.



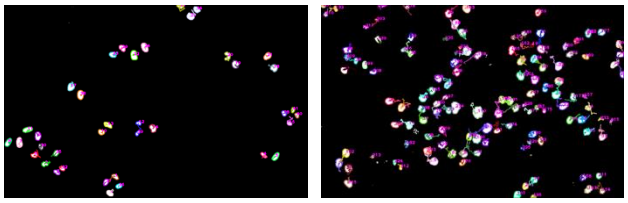
(c) (d)

Figure. 9: Before and after cell division.

(c) Image before cell division. (d) Image after cell division.

IV. Experimental Results

The experimental datasets consist of four sequences, where two of them have ground truths and the other do not. Each of the sequences contains 92 time-lapsed images. In this project, we have used 368 time-lapsed images in total. Therefore, the 220 time-lapsed images in GT folders were not used because we did not use supervised methods. Based on the methods that we implemented, the following are the implementation of our methods and the regarding experimental results.

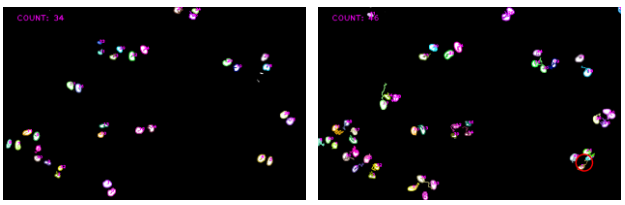


(a) (b)

Figure. 10: The result of task 1-1 and task 1-2. (a) cells with unique colours and IDs. (b) Cell trajectories

As we mentioned above, we only keep cells with the area larger than 30, so in Fig. 10(a), this is the result of cell segmentation after image pre-processing. We can see that cells are well-segmented according to our methods and their contours are shown in the image as overlays. Moreover, the areas of the cells are visually large enough to meet our benchmark. In addition, we have assigned unique colours to each of the cells.

In the image Fig. 10(b), we have drawn trajectories of all the cells over time. The trajectory of a cell is a piecewise linear curve connecting the centroid positions of the cell, from the time when the cell first appeared up to the current time point. For each cell, draw its trajectory using the same colour as the contours for visual consistency. That is, for each image in a sequence, the program should show for each cell with its trajectory up to that time point.



(c) (d)

Figure. 11: Results of task 3-1 and task 3-2,3. (c) The cell count (the number of cells) in the image. (d) Drawing a thick red circle around dividing cells.

As shown in Fig. 11(c), we obtain the number of cells in the image by calculating the length of a list of matched cells and printing the number on the upper left corner.

For image Fig. 11(d), this is how we detect dividing cells. We draw a bold red circle around dividing cells. Also, if a parent cell divides, the two daughter cells should each get a new colour and an ID.

```
--------------------
Image 27
Number of cells: 46
Average displacement of cells: 4.0
Average size of cells: 41869
Number of cells are dividing(pairs): 1
--------------------
```

(e)

Figure 12: The statistic information of all tasks.

Fig. 12(e) lists the statistical information of our tasks in this project. From the previous image to the current image in the sequence. The displacement of a cell can be estimated by taking the Euclidean distance (in pixels) between the centroid positions of the cell (already computed in Task 1-2) from one image to the next.

After removing some cells on the boundary of the image that is not completely contained in the image, the program will output the average size (in pixels) of all the cells in the image. On top of that, the program will output the number of cells that are in the process of dividing.

V. Discussion

We encountered many challenges for this project, and therefore we have also tried many methods to compare their efficacy. Ultimately, we choose the methods that can produce the best results for this project. For example, the results of using the watershed method and not using this method are almost the same. This is because the identifiable number of cell divisions is almost determined when the binary image is generated. Another example is the mathematical morphology method, which we compared with the convolutional neural network (CNN) method. The experimental results show that the accuracy of CNN is not as high as that of the mathematical morphology method for this project because the datasets included in this project only has 220 time-lapsed images with labels in GT folders. In other words, the amount of data is too small to train a high-precision model. If the size of the dataset is on a large scale, CNN will outperform this method with no doubt.

There are also some limitations of our project. The first is the segmentation errors. Around fifty percent of the original cells were left to do the tracking after segmentation. The result was decent, while the accuracy can be further improved. A method to increase the accuracy is that the ambiguity of cell tracking can be reduced with fewer image segmentation errors. Besides, suppose there are cells with overlapping areas or close distances. In that case, our tracking results will also become inaccurate, mainly because using distance as the single feature is not enough for the large-scale cell population in one frame. One solution to this error is that in addition to using a single distance as a feature, we can also add another feature, shape. As we analysed above, the accuracy of cell tracking massively depends on the segmentation result. If we can keep more cells in the frame, the trajectories of cells will be more comprehensive.

As for future work, during the exploring of relevant papers in the image detecting field, we find that convolutional neural networks (CNN) are widely used in classification tasks, and the output result is the class label of the entire image.

Compared to the traditional CNN net, UNet is a pixel-level classification structure, and the output is the category of each pixel, the pixels of different categories will display different colors. UNet is often used in biomedical images,

Therefore, Olaf et al. [7] trained a convolutional neural network that uses a sliding window to provide the surrounding area of the pixel (patch) as an input to predict the class label of each pixel.

This network has two advantages:

(1) The output result can locate the position of the target category.

(2) Since the input training data is patch, this is equivalent to data enhancement, thus solving the problem of a small number of biomedical images.

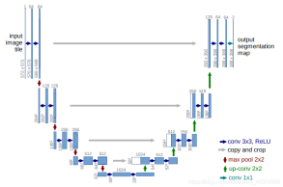


Figure 13: UNet Structure [7].

The left side of Fig. 13 is the feature extraction network: use convolutional layer and pooling

The right side of Fig. 13 is a feature fusion network: use the feature map generated by upsampling and the left feature map for concatenating operation. The pooling layer will lose image information and reduce image resolution and is permanent. It has some impact on image segmentation tasks but has little effect on image classification tasks. Upsampling can include advanced abstract features. The low-resolution image becomes high-resolution while retaining the high-level abstract features, and then concatenates with the low-level surface feature high-resolution image on the left)

Finally, two convolution operations are performed to generate a feature map, and then two convolution kernels with a size of 1*1 are used for classification to obtain the last two heatmaps. For example, the first one represents the score of the first category. The second sheet represents the second type of score heatmap, which is then used as the input of the softmax function to calculate the softmax with a relatively large probability and then perform loss and backpropagation calculations.

Since this model can combine floor and ceiling details and specialise in these complex/large grayscale range/ and boundary not clear cells image. So maybe this is a proper method to increase the detecting rate of cell segmentation.

VI. Conclusion

This article presents the implementation of image pre-processing, cell segmentation, tracking, and motion analysis. Due to the similarity of the dataset compositions, the same algorithms are applied to all the sequences throughout the four phases. In this way, we can utilise the same benchmarks on different sequences. However, we also encountered challenges like finding methods that suit this project's datasets and evaluating cell division. We have implemented several methods to compare their efficacy for the image pre-processing and cell segmentation steps, such as the mathematical morphology method, Watershed, convolutional neural network (CNN). The mathematical morphology method outperforms CNN for the small size of datasets while producing similar results as Watershed. We can notice that the cell density increases from sequence one to sequence four so

that there are more overlapping and uneven cell edges. In order to overcome this obstacle, we detect contours of all the cells, which generates a tree structure, and then only keep the cells with a contour area greater than 30. Since this method is applied to a single cell, even if one cell contour is uncommon, the system can still have a decent segmentation performance. Therefore, the application of tree structure dramatically improves the system's efficiency and enhances the robustness and descriptiveness of the cell features.

To track cells after the segmentation step, we first find the centroid of each cell, then assign a unique identification number to each of them along with a distinguished colour for their contours. For every cell, we compare their coordinates in every two adjacent images by utilising Euclidean distance, which is very efficient to keep the trajectories of every cell.

As for cell motion analysis, we employ a function to detect the shape of cells, whether they are still ellipses. If a cell is relatively too large or too small to an extent, we consider it as a parent cell. Otherwise, it is a child cell. In the end, we also use a red circle to mark the cells which are still dividing.

VII. Contribution of Group Members

- *Yanyun WU* (z5191449):

Participating the code writing. Make some minor addition to the method section in the report.

- *Yue Zhang* (z5232113):

Collaborated with Rifu SU to composite the report and the demo slides, responsible for literature review, summary of methods, and conclusion. Highly involved in integrating content and formats for the report.

- *Rifu SU* (z5232173):

In cooperation with Yue Zhang to composite the report and the demo slides, responsible for introduction, minor addition to the methods, and discussion. Participate in the content and format integration of report and demonstration.

- *Jianheng Qian* (z5275335):

Collaborated with Yanyun WU to implement the task1 image pre-processing, marking,major response for the result part of report and involved in method.

- *Ruoxi Bao* (z5314772):

Responsible for writing the code of task 2, with help of Yanyun WU,determines the methods and tools used to detect cell division, completed the method content and framework of the report and presentation parts.

As the results presented above, the methods used in this project have produced a decent performance, while there are still limitations. On the one hand, we segment almost half of the cells out of the original images. Whereas the accuracy of cell tracking depends on the segmentation performance, we could record trajectories of more cells if we could improve the segmentation method. On the other hand, we only deploy distance as the feature to track cells, and clashes might happen when there are too many cells in one area or overlapping, so that distance of several close cells can be the same.

Automated cell segmentation and tracking techniques are still developing, and our project has provided a benchmark for this type of cell and such microscopy. In future work, we expect more features can be introduced to do the tracking work, and researchers can also use other methods to improve the segmentation performance. The benchmark produced in this project can be tested and validated on more complex datasets of other cell types or other microscopies.

VIII. References

[1] Vicar, T., Balvan, J., Jaros, J., Jug, F., Kolar, R., Masarik, M. and Gumulec, J., 2019. Cell segmentation methods for label-free contrast microscopy: review and comprehensive comparison. BMC bioinformatics, 20(1), pp.1-25.

[2] Yan, J., Zhou, X., Yang, Q., Liu, N., Cheng, Q. and Wong, S.T., 2006, October. An effective system for optical microscopy cell image segmentation, tracking and cell phase identification. In 2006 International Conference on Image Processing (pp. 1917-1920). IEEE.

[3] Debeir, O., Van Ham, P., Kiss, R. and Decaestecker, C., 2005. Tracking of migrating cells under phase-contrast video microscopy with combined mean-shift processes. IEEE transactions on medical imaging, 24(6), pp.697-711.

[4] Dewan, M.A.A., Ahmad, M.O. and Swamy, M.N.S., 2011. Tracking biological cells in time-lapse microscopy: An adaptive technique combining motion and topological features. IEEE Transactions on Biomedical Engineering, 58(6), pp.1637-1647.

[5] S. Beucher, "The Watershed Transformation page", People.cmm.minesparis.psl.eu, 2010. [Online]. Available: https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html. [Accessed: 19- Nov- 2021].

[6] Doxygen, "MeanShift and Camshift", openCV.org, 2021. [Online]. Available: https://docs.opencv.org/3.4/d7/d00/tutorial_MeanShift.html. [Accessed: 19- Nov- 2021].

[7] O. Ronneberger, P. Fischer, and T. Brox, 'U-Net: Convolutional Networks for Biomedical Image Segmentation', arXiv:1505.04597 [cs], May 2015, Accessed: Nov. 19, 2021. [Online]. Available: http://arxiv.org/abs/1505.04597