Cell Segmentation and Tracking Analysis

Zihang Wan, Xingyu Shen, Xinran Tang, Zeyu Kong, Jiagan Wang

***Abstract*—This report is mainly to develop an algorithm for tracking biological cells in time-lapse microscopy images. The proposed algorithm is composed by image processing, cell segmentation and tracking. The testing and training dataset we used is Microscopy Dataset that consists of four image sequences each from a separate time-lapse microscope. This paper uses subsequent quantitative matrices to analyze the results and accuracy of cell tracking.**

Keywords—cell tracking, watershed algorithm, segmentation, OTSU algorithm, binarization, centroid distance, shape

I. INTRODUCTION

In recent years, time-lapse microscopy plays an increasingly important role in cell biology. However, since its image sequence is too large, a huge number of cells have to be tracked and analyzed manually. Therefore, the subsequent analysis and tracking of moving cells, which is essential for the study of cell in cell biology, has always been a common and challenging topic. In this group project, python-OpenCV and related modules will be used to complete the project.

The purpose of our report is to analyze the cell by developing the algorithm to solve the problems of image processing, cell segmentation and cell tracking. In addition, the example graph of cells is shown in figure 1.

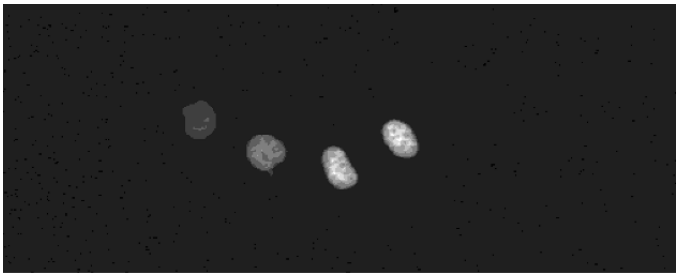

Fig. 1: Example of cells

The image dataset consists of four image sequences from a separate time-lapse microscopy recording, and all images are 16-bits/pixel. For the sequences 01 and 02, there are limited ground truths in GT folders which contain both segmentation and tracking annotations in the SEG subfolder and TRA subfolder respectively.

In Task 1, as for cell segmentation, we analyze the OTSU algorithm, Meanshift algorithm and Watershed algorithm respectively in the report, and get the cell contours with different colors. Their performances will be discussed in detail later in part 5 Discussion. As for the cell tracking, at first, we decided to use deep learning to judge the cell division precisely. However, in our dataset, there are no complete ground truth labels provided in the GT folder, whereas deep learning algorithms have a great demand for ground truth labels, it is difficult for us to implement deep learning algorithms. Therefore, we gave up on deep learning and chose the traditional algorithm to implement the cell tracking. we judge the cell division by the centroid distance, area and displacement. Moreover, Elliptic Fourier Descriptors are used for analyzing biological shapes of cells to obtain similarity for cell tracking.

In Task 2, we also figure out the problems of finding the cell count, the average size of all the cells, the average displacement of the cells, and the number of cells which are in the process of dividing. These problems can help us to analyze the motion of the cells. These methods will be explained in part 3 Methods.

II. LITERATURE REVIEW

In cell biology, tracking cells in time-lapse microscopy images is a significant task, which is difficult to track by hand due to too many and irregular cells in the resulting image. In this paper, we will not mention all types of cells, but focus on the given data and relies on methods of computer vision to study cells recognition, cells moved, and cells divided. This section mainly introduces approaches of cell preprocessing, cell segmentation, and cell tracking.

A. Image Preprocessing

The research paper "Lung Nodule Detection using Segmentation Approach for Computed Tomography Scan Images" presented meteorology for per-processing of the given Lung CT Scan image. The use of filter to eliminate the salt and pepper noesis on the image and binary conversion for help to detect the tumours or nodules in CT scan image. Then, associated with morphological operations and inversion handling the image to Opening the image and make the foreground brighter. As a result, the use of opening of the image can remove any kind boundary of cells based on discontinuity.

B. Cell Segmentation

The research paper 'Cell segmentation with median filer and mathematical morphology operation' authored by D. Anoraganingrum [1]. This paper illustrates a part of research on an automated cell tracking that aim to determine the movement of representative cells to monitor the effects of certain pharmaceutical ingredients on them. The goal of this project is to improve the efficiency of cell segmenting by means of a

1. D. Anoraganingrum, for computing technologies, University of Brethemen, Bremen, Germany.
2. The achievement of Opening relies on operations of erosion before dealation. These two operations are derived from Mathematical Morphology.

median filter and morphological operations including erosion and dilation. According to the result of experiment, this method is suitable for most cells and the performance of this method is as better as several boundary detection methods including Laplacian of Gaussian, Sobel, and Prewitt.

The paper 'Red blood cell segmentation using masking and watershed algorithm: A preliminary study' proposals an approach for segmentation and automated counting on red blood cell. There are YCbCr colour [3] conversion, masking, morphological operator, and watershed algorithm involved in the method. Combination of YCbCr colour conversion and morphological operations to segment white blood cell and using it as a mask. Then, removing white blood cell from the image of blood cell and diminishing the tiny object associated with morphological operators. The result of segmentation relies on the marker of handled overlapping cells to control the watershed algorithm.

The article "Smart Markers for Watershed-Based Cell Segmentation" authored by C.F. Koyuncu et al [4]. This article analyzes a new method for segmentation on living cells from phase-contrast microscopy, which uses a set of 'smart marker' for a marker-controlled watershed algorithm. This approach relies on the visual characteristics of the cells and has a vital part in the identification of markers. The experimental results show that the method has a positive impact on the performance of the watershed algorithm compared with its counterparts, which can increase the efficiency of the algorithm.

C. Cell Tracking

The article 'Multiple Nuclei Tracking Using Integer Programming for Quantitate Cancer Cell Cycle Analysis' demonstrated that automated tracking method for quantitative cell cycle analysis. In this approach, the use of adjacent graphs to describe the spatial distribution of adjacent cores, and design new differential measures based on spatial distribution, nuclear morphology appearance, migration, and intensity information. After that, tracking the nucleus associated with programming and partition matching strategies, as well as new and different metrics. The experiment results showed that the accuracy for segmentation and tracking on HeLa cancer cells are 99.5% and 90.0% respectively.

The article "A Computer Program Package for Quantitative Evaluation of Biological Shapes Based on Elliptic Fourier Descriptors" authored by H. Iwata [5] introduced a method named SHAPE which is based on Elliptic Fourier Descriptors (EFD) [6] to evaluate then contour shapes of biology. Image processing, contour recording, component analysis, and visual program for shape transformation are contained in the package of EFDs. The main component obtained by the program can be applied for the shape eigenvalues observed in subsequent analysis.

III. Method

In this section we will discuss the main algorithm, and the database we used. There are three main tasks we are going to solve: Cell Segmentation, Cell Tracking and Cell Dividing.

In order to perform an accurate cell segmentation, we need to pre-process the dataset to extract the cells out of background. Also, we detect the cells that are going through dividing, so we will get noticed when new cells are produced. Then we can segment the cells for cell tracking across the images.

A. Preprocessing

Preprocessing can be mainly divided into four parts, including normalization, contrast stretching, Gaussian thresholding and morphological operations. This process is shown in the figure 2.

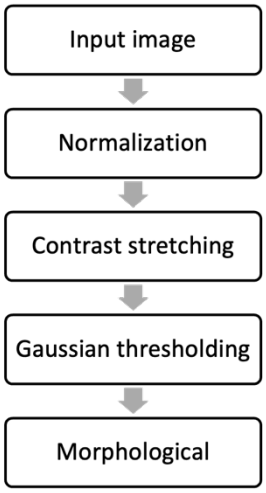

Fig. 2: Flowchat of the proprecessing process

At first, we make use of the OS module to access the required data and import the data into the work environment. Then, the functions in Python-OpenCV are required for reading the task data in grayscale graph for preparation. After that, we apply normalization on these graphs for the convenience of subsequent data processing and faster convergence of the program. The grayscale graph and normalized image are shown in the figure 3 and figure 4.

Fig. 3: Grayscale graph

Fig. 4: Grayscale image

The task data are biological cells in time-lapse microscopy images. As a result, images are better to be separated into foreground and the background. In this way, cells will be handled easier to observe and recognize. K.Hajdowska illustrated that the contrast of input images requires an increase

3. YCbCr colour, which is a type of color space and typically used for continuous image processing. Y is the luminance of color, then Cb and Cr are the concentration offset component of blue and red respectively.
4. C.F. Koyuncu, department of computer engineering, Bikent University, Ankara, Turkey.
5. H. Iwata, Associate Professor, The University of Tokyo.
6. Elliptical Fourier Descriptors, a mathematical approach which use the number of ellipses on the outline to quantify the online and contours in the image. This technique is widely used in Computer Vision and visual Pattern Matching.

in the top priority. Hence, the use of contrast stretching can convert the maximum and minimum gray value of the image into 255 and 0 respectively. Below figure 5 and 6 show a typical transformation function used for contrast stretching and the result image we got.

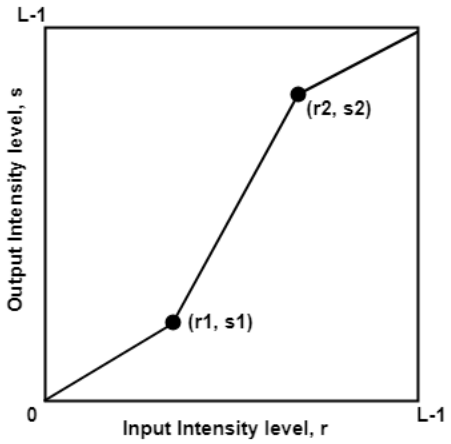

Fig. 5: Transformation function used for Contrast Stretching

Fig. 6: Image after constracting stretching

In this way, the contrast between the foreground and the background will be greater and cells become brighter and easier to observe. Apart from this, the use of Gaussian thresholding on the images are divided into two parts. One is using Gaussian filter to remove noise on, the other is converting images to binary images with set threshold values. In this way, the increase contrast of image can make cells brighter and easier to recognize. The Gaussian filter is defined as the following equation (1):

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-[\frac{x^2 + y^2}{2\sigma^2}]} \qquad (1)$$

The result image after going through Gaussian filter shows in figure 7.

Fig. 7: Image after constracting stretching

As some cells have holes in the middle or been partially cut off by the boundary, the shapes of cells are not well-formed. Hence, in our work, we use dilation in morphological to add tiny bright pixels at the boundary of cells to fill the gap. In this way, boundaries of cells become rounded and integrated. That is the whole part of preprocessing, and the next part is cell segmentation. The result of image after morphological operations shows in figure 8.

Fig. 8: Image after morphological operations

B. Cell Segmentation

We use watershed algorithm for cell segmentation instead of simple thresholding and contour detection. Since we are segmenting cells which contain lots of overlapping and touching between one and other, the watershed algorithm helps to separate the cells close to each other and mark them as individual cells. The cell segmentation process is shown in the figure 9.

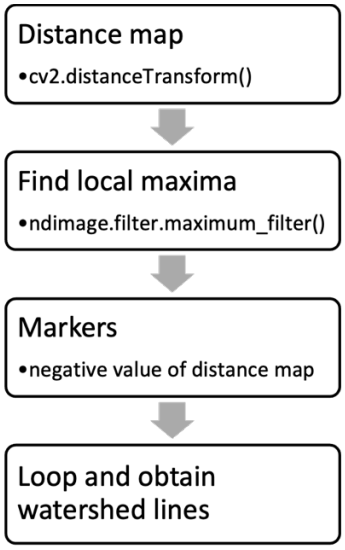

Fig. 9: The process of cell segmentation

When applying the watershed algorithm we must start with user-defined markers. In our case, we first generate a distance map of the enhanced images via distance transform function in Python-openCV. Then, we use maximum filter function to find the local maxima. Hence, the watershed algorithm will assume our markers represent the local minima in our distance map by taking the negative value of the distance. The equation (2) below defines the maximum filter.

$$I' = (\mathrm{u}, \mathrm{v}) \leftarrow \max\{I(u + i, v + j) | (i, j) \in R\} \qquad (2)$$

Based on these markers, the watershed algorithm treats pixels in our input image as local altitude. The algorithm creates floods in valleys, beginning from the markers and moving outwards, until the valleys of variable markers meet each other.

Pixels that have the same label value belong to the same object. We give each pixel value a unique label value and loop over them. There will be two cases: if the label is zero, we ignore it as we are examining the background; while it is not 0, we create a mask and set the pixels belonging to the current label to 255 (which is the white colour).

Moreover, we detect contours in the mask and extract the largest one. This contour represents the outline of an individual cell in the image, which is also known as watershed lines.

The result image after applying watershed algorithm is shown below as figure 10.

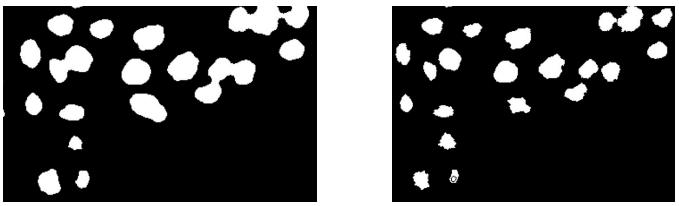

Fig. 10: Image before (left) and after (right) watershed algorithm

C. Cell Tracking

Step 1. Construct cell objects

To perform cell tracking, we firstly construct a cell object for all the cells in the given sequence of images, each with a unique id. These objects can later be used to store information such as centroid position, displacement, area, shape, previous position and whether the cell is in the process of dividing.

Step 2. Find Centroid Positions

In order to track the same cell across a sequence of frames, we firstly find centroid position of each cell using the cv2.moments() function. The calculation of the x and y axis of the centroid is given by the following equation (3):

$$C_x = \frac{M_{10}}{M_{00}}, C_y = \frac{M_{01}}{M_{00}} \quad (3)$$

Step 3. Calculate Euclidean Distances

Afterwards, we use the centroid positions to calculate the Euclidean distance between every pair of cells, with the formula (4):

$$Distance = \sqrt{[(x_2 - x_1)^2 + (y_2 - y_1)^2]} \quad (4)$$

where (x_1, y_1) is the centroid position of the first cell and (x_2, y_2) is the centroid position of the second cell.

We also chose 30 pixels as a limitation of the maximum distance we will consider, returning 1 for any distance larger than 30 and $[\frac{distance}{30}]$ for any other results to enhance performance.

Step 4. Find Shape & Calculate Shape Difference

Then we find the shape of each cell by passing them into the ***elliptic_fourier_descriptors()*** function. To calculate shape difference, we pass the ***elliptic_fourier_descriptors*** data of two cells into the helper function ***shape_difference()*** which calculates the maximum difference for the first 8 orders.

Step 5. Obtain Cell Similarity

Multiplying the distance and shape difference by a given ratio that sums up to 1, we obtain a combined similarity score. This allows us to adjust the amount of consideration for both distance and shape. In the case where we obtain very accurate distance measures but low accuracy shape measures, we can increase the ratio to maximise accuracy. For example, taking the ratio as 8:2. We also include the div factor for situations involve cell division. When a cell is in the process of dividing, we will not consider the shape difference and will change the div factor to 0. This is because the cell shape changes largely when a cell is dividing and will decrease accuracy. The equation (5) we implemented shows below.

$$dist = a_1 * centroid\ distance + a_2 * shape\ difference * div \quad (5)$$

Step 6. Find Most Similar Cell

Calculating the similarity score pair by pair and sorting the obtained scores in ascending order, we can easily find the cell in frame (n + 1) that is most similar to the current cell in frame n. Once we have found the most similar cell, we update its id in frame (n + 1) to be the same as frame n and record its previous position for calculation of the displacement between two frames. For all other cells that we could not find a match for, we generate a new id for them.

Step 7. Colour Cells & Draw Trajectories

Once we have found all matching cells across the sequence of images and updated their ids, we can generate a list of random colours that is the length of total number of different cells. Colour the cells according to their ids with ***drawContours()*** function and marking their ids on the images with ***putText()*** function, we obtain a coloured image sequence. We can also show the path of cell motion using the ***cv2.line()*** function, by drawing a line between the current cell position and the cell position in every previous frames.

Step 8. Analyse Cell Motion

Using the information stored in the cell objects, we can easily loop over the list of cells and calculate the total number of cells, average cell area, average cell displacement and number of cells dividing.

D. Cell Division

To identify cells division, we use the ***fitEllipse()*** function to fit an ellipse to a given cell. We can obtain the length and width of a cell from the returned values. Since cells that are in the process of dividing will be more bar-like instead of circular, we consider cells with a larger length / width ratio. In this case, we consider any cells with length larger than twice as long as width in the process of dividing. ($\frac{MA}{ma} > 2$) We also record whether a cell is dividing in the cell objects with a Boolean value.

IV. Experimental Results

In this part, we will discuss the specific parameters of the project and show the results of our code execution.

A. Experimental setting

In image preprocessing part, Gaussian filtering is used to the image denoising, the kernel size we set is (5, 5). However, the images are also dark, so Contrast Stretching is used to enhance its contrast so that it maps to a larger grayscale space. The formula (6) shows below.

$$I(x,y) = \frac{I(x,y)-Imin}{Imax-Imin}(MAX - MIN) + MIN \qquad (6)$$

where $Imin$ is the minimum gray value of the original image and $Imax$ is the maximum gray value of the original image, MIN and MAX are the minimum and maximum gray values of the gray space to be stretched. The minimum and maximum gray values we set are 0 and 255 respectively.

In cell segmentation part, we use ***cv2.distanceTransform()*** to find the seed points of cell s to perform watershed algorithm. The parameters we set are distance type is 2 which means the distance type we used is the simple Euclidean distance and the size of the distance transform mask is 0. Then, we also use ***cv2.getStructuringElement()*** to get the structural elements that specifies shape and size, the shape we set is ellipse, the size is (5, 5).

B. Experimental results

Task 1:

In task 1-1, we need to segment all the cell and show their contours in the images with the unique colours. The results of task 1-1 is shown in figure 11.

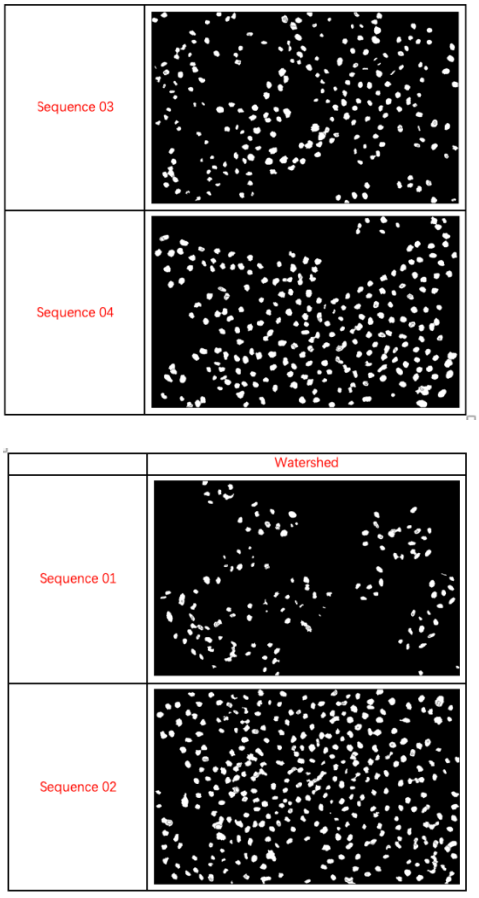

Fig. 11: Cell segmentation in sequences

In task 1-2, we need to track the cells over time and show their trajectories. The result should show the linear curve connecting the centroid points of the cells between the time when the cell first appeared and the current time points. The results of task 1-2 is shown in figure 12.

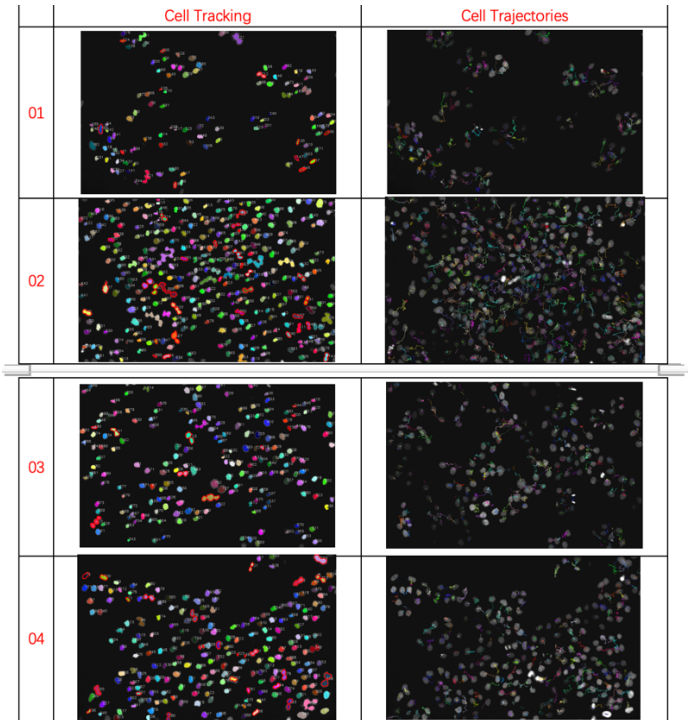

Fig. 12: Cell tracking in sequences

Task 2:

In task 2, we need to show the cell count, the average size and displacement of all cells and the number of cells that are in the process of dividing which is judged by the centroid distance and the cell features.

The cell count is shown in sequences in figure 13.

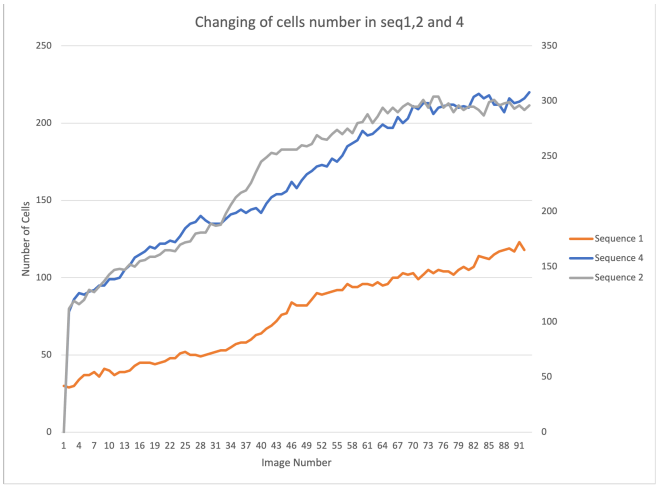

Fig. 13: Cell Count in Sequences

The average cell size in sequences is shown in figure 14.

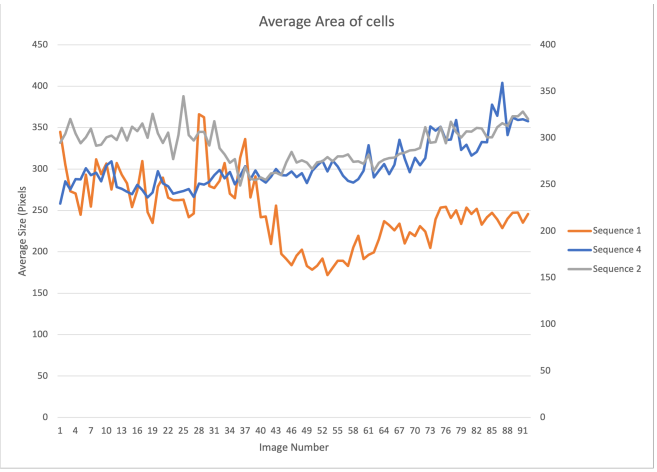

Fig. 14: The average cell size in sequences

The average cell displacement in sequences is in figure 15.

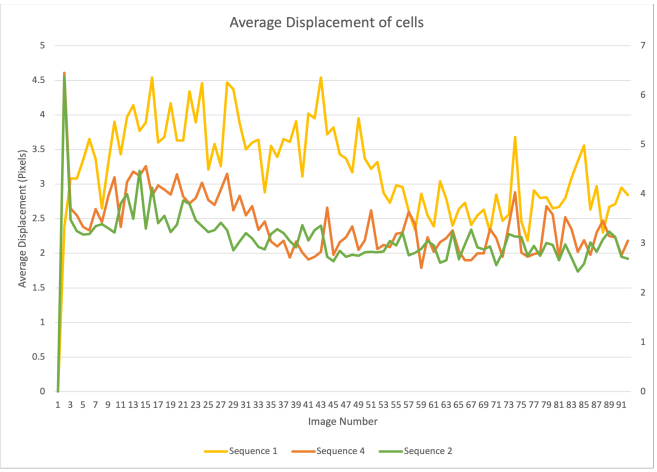

Fig. 15: The average cell displacement in sequences

Number of cells dividing in sequences in figure 16.

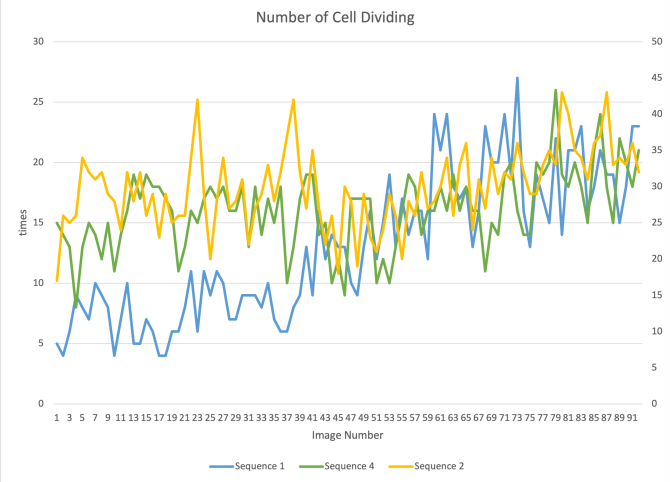

Fig. 16: The number of cell dividing in sequences

C. *Errors*

There are some issues in task 1-2.

In Sequence 01, some cell divisions are not recognized accurately, we think we can compare its shape across frames. Besides, daughter cells are not given 2 new IDs, one of them inherits the parent ID.

In Sequence 02, some cell divisions are not recognized accurately as well, we think we can compare its shape across frames. Besides, touching cells are recognized as a single cell, we think we could optimize the cell segmentation algorithm to improve the accuracy.

In Sequence 03, some cells are not recognized in certain frames, we think we could optimize the image processing algorithm to improve the accuracy.

In Sequence 04, some fast-moving cells into the image are not recognized, we think we could optimize the cell tracking algorithm to improve the accuracy.

Figure 17 below shows the sample analysis of the error mentioned above.

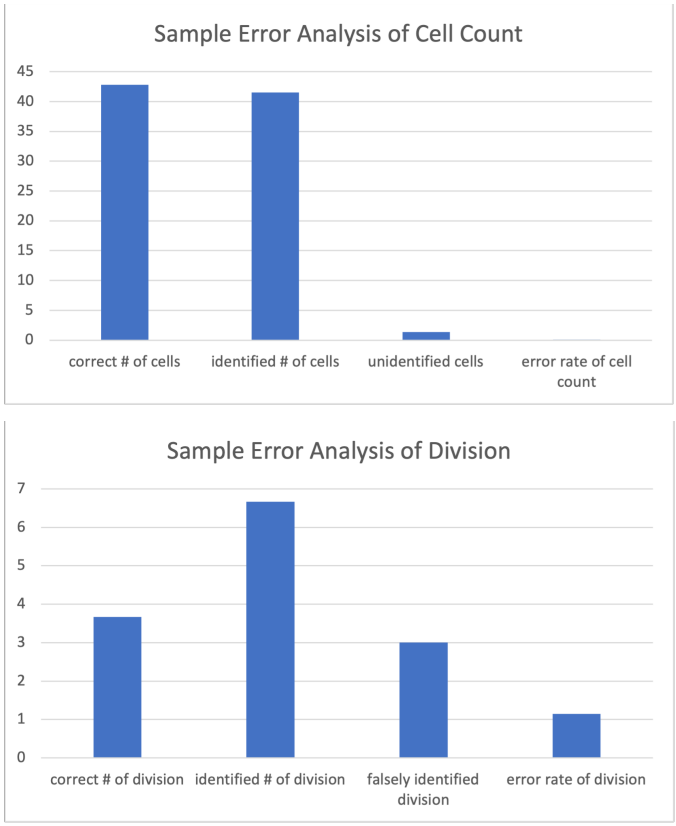

Fig. 17: The sample error analysis of cell count (top) and cell division (bottom)

Comparing the below two images in figure 18, we can see that the circled cell is not separated from the background in the watershed processed image. This is because the cell has a very light shade, and we can improve this by adjusting the threshold of image processing.

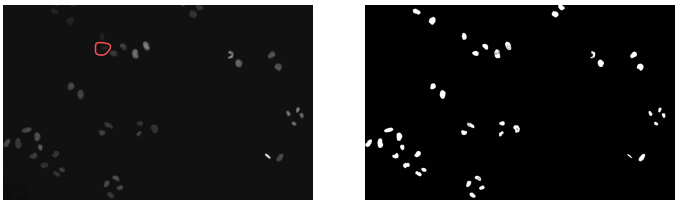

Fig. 18: Error in normalised image (left) and after watershed (right)

We can see in figure 19 that some longer cells that are not in the process of dividing are also circled. This is because watershed decreases the size of cells, and the identification of division purely relies on the shape of cells in current image. We can improve this by performing dilation after watershed to restore the shape of cells, as well as adding size difference between two frames into the consideration of cell division.

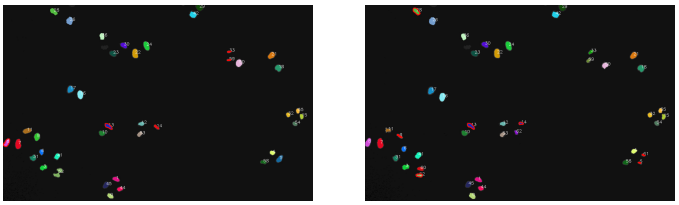

Fig. 19: Cell division identification result of image 13 (left) and 14 (right) in sequence 01

V. Discussion

A. Cell Segmentation

OTSU Algorithm:

All our group members have reached an agreement that a contrast stretching should be applied to make the border of cells clearer. But when it comes to the segmentation algorithm, we have some different opinions.

In the beginning we decided to use the Otsu algorithm, but it seemed to be working wrong. This method cannot observe some of the cells which are already not obvious in the original picture. Besides. It cannot consider two cells which are too closed to each other as two different cells (two cells in the right of the picture are considered as a single cell, the green one). With poor performance, this algorithm is finally abandoned. The figure 20 below shows the result images after we applied contrast-stretching and OTSU algorithm.

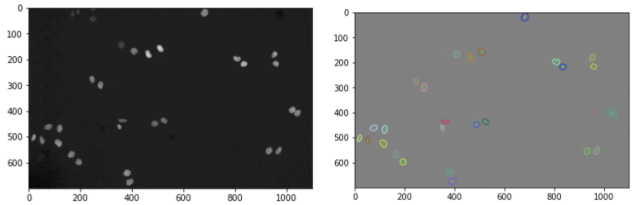

Fig. 20: Contrast-stretched picture(left) and after applying OTSU algorithm(right)

Meanshift Algorithm:

We also tried the Meanshift algorithm to segment the cell. First, we would like to use the method estimate bandwidth to get the proper bandwidth, but it takes a lot of time to run that cannot get the result, so we set the parameters bandwidth = 20.

However, there are some problems in Meanshift algorithm. It also takes too long time to run, we think the reasons may be the graph is so large, the feature space is so large that it takes a lot of computation. Besides, the bandwidth we set may be too small, resulting in slow convergence. In addition, the result of cell segmentation is also not ideal enough, due to the bandwidth setting. Therefore, we think Meanshift algorithm is not the optimal method. Moreover, the graph of the result is shown in the figure 21.

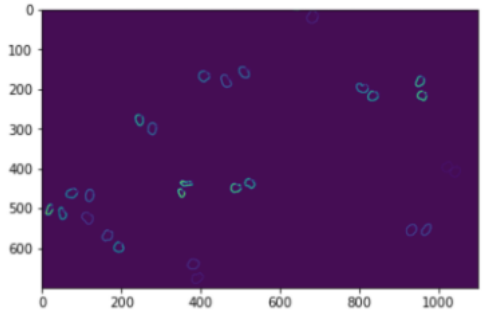

Fig. 21:Result of meanshift algorithm

After that, we turned to the watershed algorithm, which performs best and can separate closed cells clearly. So finally, we chose watershed algorithm to provide an accurate segmentation output for later processing.

B. Cell Division and Cell Tracking

In the beginning, we have considered using the position of cells and the area of cells to judge whether the two cells belonged to a single cell in the last frame.

For each cell in the current frame, we calculate its Euclidean distance to all the cells in the last frame. If the distance of two cells is less than x pixels (we used x=15 during that time), then these two cells will be separated into one group. Cells which are aligned into the same group are suspected to be the cells divided from the same original cell.

The cells in the same group will be judged for the second time to judge whether they come from the same original cell. The second judgement is that if these two cells' (both in the current frame) sum of distance to the original cell in the last frame is less than 30 pixels(we used y=30), and the ratio between sum of two areas and the area of the original cell is between lb~ub (we used lb=0.95 and ub=1.05), then the can make a conclusion that these two cells are the results of division of the same cells. The value of judgement parameters (x, y, lb, ub) are selected from the first divided pair of cells we observed, and this algorithm seemed to be working out correctly as shown in figure 22.

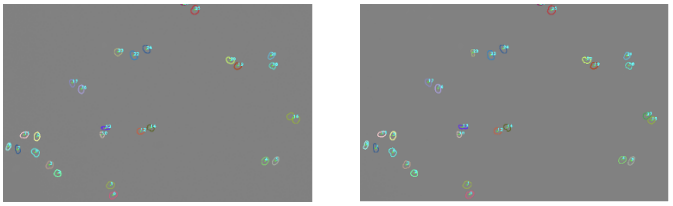

Fig. 22: The last pixel of image (left) and the current pixel (right)

It seemed that the “15” cell in the left divided into two different cells “15” and “27”, and the appearance of new index “27” means that these two cells are successfully considered as result of division of the same cells.

But after that, three mistakes come up: (1) The two cells should be “27” and “28”, since the two daughter cells should get a new ID. (2) Actually there’s no division in this place, and the misunderstanding origins from the OTSU method. (3) The parameter value selection is inappropriate.

The first mistake could be fixed, the second could be accepted, but the third one finally stopped our exploration of this algorithm, since the actual scene of cell division varies. We cannot find an appropriate x, y, lb and ub for every cell division. The daughter cell may be the same bigger than the original cell, and two daughter cells may quickly flee away, leaving a sum of distance larger than y. If the boundary is raised up, then more cells which are originally two different cells will be divided into the same group, making all the matters messed up. Figure 23 is a failure image output of this algorithm with cell tracking applied.

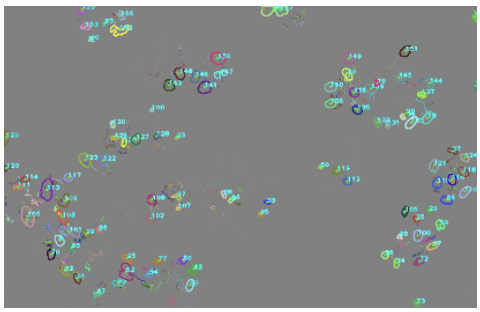

Fig. 23: The failure of exploration on cell division

After that, we updated this algorithm and formed our new cell division and tracking algorithm, which is described in previous chapters. This new algorithm also used the centroid position and areas and compared neighbourhood frames, but its functions perform much better. The results could be seen in the figure 24 below.

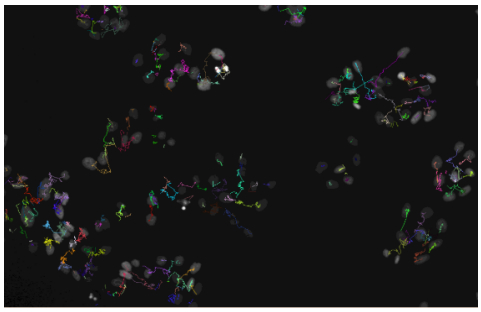

Fig. 24: The successful result of cell division and tracking

VI. CONCLUSION

In conclusion, we have completed the requirement for the image process, cell segmentation and cell tracking in the task of this group project. Therefore, the recognition part of the tracking of biological cell in time-lapse microscopy images has been implemented. However, there are still some problems in our projects. Our recognition accuracy is not accurate enough. For example, in some complex conditions, such as many cells, fast cell movement, and frequent entry and departure of cells in the field of vision, incomplete recognition of cell division may occur. The fundamental reason is that we use the distance of centroid point to judge cell division, and recognition errors may occur when cells move irregularly. We think we can still optimize the algorithm to improve the recognition accuracy.

Firstly, we can improve image preprocessing by adjusting thresholds so that light or small cells can also be recognized. We should also consider cells that move in and out of the frame by giving particular attention to cells that are half out of the edge. This can be realized by considering cell location and cell area. When a cell is close to an edge, we compare its area with the previous frame. If there is a sudden decrease in area, we consider the cell moving out of the image. If a new cell appears close to an edge with area increasing, we consider it moving in.

Secondly, using the current method, one daughter cell inherits the colour of the parent cell instead of a new colour. This is because we have not taken division into account the process of finding matches, as our accuracy for finding division is still quite low. In order to improve this, we may examine the shape of cells across frames and cell count to better determine whether cell division is happening.

Thirdly, on top of the distance and shape differences, we can add a third texture measure into the tracking process. By generating a co-occurrence matrix for each cell, we can compare the texture complexity difference of cells. Adding this factor to the calculation of cell similarity score could possibly improve the accuracy of cell tracking.

Besides, we do not choose machine learning because of the machine performance issues. Maybe we can use machine learning to improve the accuracy of cell division.

Overall, the above is all the content of our group project. We would like to thank for helping us in writing this report.

VII. CONTRIBUTION OF GROUP MEMBERS

Zeyu Kong completed the introduction, conclusion, experimental results, and partial discussion in the report, as well as presentation on the part of introduction and partial discussion and image processing part of the code.

Zihang Wan completed the main part of cell processing, the literate reviews and partial method in the report, as well as presentation on the method of preprocessing and partial discussion.

Xinran Tang mainly completed cell segmentation part of the code, presentation on watershed method and results evaluation parts, video editing of the presentation demonstration, as well as cell segmentation part and formatting of the report.

Xingyu Shen mainly completed the image preprocessing, cell tracking, division and motion analysis part of the code, cell tracking method and software demonstration in the presentation, video editing of the code demonstration as well as cell tracking, cell dividing method and improvements in conclusion of the report.

Jiagan Wang completed the conclusion part of the presentation and discussion of cell segmentation, division and tracking part in the report.

VIII. References

[1] D. Anoraganingrum, "Cell segmentation with median filter and mathematical morphology operation," Proceedings 10th International Conference on Image Analysis and Processing, 1999, pp. 1043-1046.

[2] J. M. Sharif, M. F. Miswan, M. A. Ngadi, M. S. H. Salam and M. M. bin Abdul Jamil, "Red blood cell segmentation using masking and watershed algorithm: A preliminary study," 2012 International Conference on Biomedical Engineering (ICoBE), 2012, pp. 258-262.

[3] C.F. Koyuncu, S. Arslan, I. Durmaz, R. Cetin-Atalay, C. Gunduz-Demir, "Smart Markers for Watershed-Based Cell Segmentation," PLoS ONE, 2012.

[4] F. Li, X. Zhou, J. Ma and S. T. C. Wong, "Multiple Nuclei Tracking Using Integer Programming for Quantitative Cancer Cell Cycle Analysis," in IEEE Transactions on Medical Imaging, vol. 29, pp. 96-105, January 2010.

[5] H. Iwata, Y. Ukai, "SHAPE: A Computer Program Package for Quantitative Evaluation of Biological Shapes Based on Elliptic Fourier Descriptors," Journal of Heredity, vol. 93, pp. 384–385, September 2002.

[6] H. Satra, A. Gupta, "Lung Nodule Detection using Segmentation Approach for Computed Tomography Scan Images," International Journal for Tesearch in Applied Science & Engineering Technology, vol. 9, pp. 1778-1790, September 2021.

[7] H. Singh, "Advanced Image Processing Using OpenCV," Practical Machine Learning and Image Processing, 2019.