Computer Vision 2020 Term 2 Group Project Report

Group D: Luyuan He, Rifang Li, Long Long, Ran Ji

I. Introduction

Using information technology to automate the process of tracking and analysing biological cells is becoming one of the most common and crucial computer version tasks in the field of biology because it is a necessity for both the definition of biological processes and the diagnosis of diseases [1]. Biological cells are often recorded using microscopes that can capture videos in a period. Then these videos are transferred into time-lapse image frames. Traditionally, these frames are analysed manually, however, to obtain robust results requires monitoring a large number of cells over a long period. Often it is difficult and tedious for human observers to follow it along and provide reliable results [2]. Therefore, there is a tremendous need for automating the process and is gaining increased interest from research. This paper presents our method for the group project, which is similar to the above-described tasks, of the Computer Version course offered by the University of New South Wales [3]. Our group is assigned to perform three main tasks on three different sets of biological cells in time-lapse microscopy images. The image datasets used are downloaded from the international Cell Tracking Challenge (CTC) [4] and they are *DIC-C2DH-HeLa*, *Fluo-N2DL-HeLa*, and *PhC-C2DL-PSC*. For the first task, our job is to detect and track all the cells in each image, draw bounding boxes, and their path for each cell. This first segmentation step is very important because its success rate affects the correctness of the following steps. Task two ask us to look for mitosis events and highlight them. These events are particularly useful in the automated study of cells [1]. The last task requests us to calculate and analyze the motion of the user-selected cell and find its confinement ratio.

II. Literature Review

Enormous researches have been made into automating the process of tracking and analyzing biological cells and considerable techniques have been developed. The most critical step in this process is to identify and segment each cell correctly. One of the most widely used segmentation strategies is the thresholding method followed by seeded watershed [6][7], thanks to its performance, user-friendliness, and computational capacity. Another common cell-detection strategy is based on h-maxima transformation and has better results on certain types of cells [8]. Recently, deep learning-based methods have drawn the most attention because they typically perform well [9][10], however, they require a larger amount of data and computational power.

III. Methods

Based on the three tasks, we proposed a segmentation-based method for this group project. It includes three major modules: detection, tracking, and analysis, as shown in Fig.1. In the detection module, the program first reads the input frame images from files and normalize them. Then it tries to highlight and detect each cell in each frame image. During this step, different operations are used for each dataset, because of the huge differences between them, especially between *DIC-C2DH-HeLa* dataset and the other two datasets. This enables the segmentation of each cell, a binary thresholding method with different parameters was used for all three datasets. After it, a rectangle bounding box is drawn around each separated cell in each frame image. The next module is the tracking module. Based on their bounding box, a centroid is calculated for all of the cells. A unique identifying number is assigned to each of these centroids and the history of their coordinates is recorded. Then the distance of each cell travelled is calculated. This module draws a line from its original coordinate to its next coordinate, up until the current time frame. The next step is to detect cell divisions. While cells are dividing, the total number of cells will be increased. Methods based on distance detection and shape detection are used to detect which original cells the new cells corresponding to. The final module is the analysis module. In this module, the user can choose any cell at any time frame image, then four feature values about cell motion will be displayed. The four features are: speed of the cell at that time point, total distance travelled up to that time point, net distance travelled up to that time point, and confinement ratio of the cell motion.

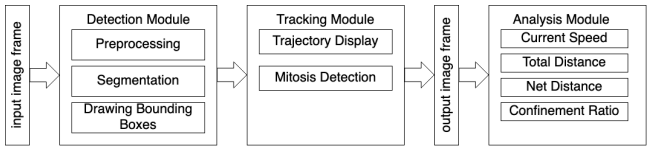


Fig. 1. Framework of our method

A. Detection Module

Accuracy of tracking is highly dependent on the accuracy of cell detection in a segmentation-driven method like ours. Also, accurate cell localization is crucial in tracking to extract reliable features for the cell analysis module [8]. To achieve the above two goals, our detection module is designed in detail below.

1) Preprocessing: The cell images presented in our datasets are captured by microscopes. Naturally, the cells not clearly illuminated from their surrounding artifacts and background. Different datasets also create different features on their frame image. Hence, we decide to do a data normalization first. The normalization process changes the ranges of pixel intensity values so that all the images are consistent in the same pixel range. We tried different normalization functions including histogram equalization, contrast limited adaptive histogram equalization (CLAHE), and median scaling. We used CLAHE for our normalization. This method is mainly practical for the image histogram which is not single because the images have multiple peaks. After normalization, further operations are required before

the segmentation step for *DIC-C2DH-HeLa* and *Fluo-N2DL-HeLa* datasets as described in detail below.

a) DIC-C2DH-HeLa: This dataset contains frame images with the following characteristics: cells are clustered together, the contrast of cell boundaries is very low, and there are lots of noises. It is still reminded as a challenge after we tried many methods in combinations of morphological and topological methods. After some researches, we decide to use the pre-trained deep learning model introduced in [9]. In this method, it uses two convolutional neural networks (CNNs), one for cell marker pixels, and another one is used to predict the image foreground then it uses a watershed algorithm to segment the cells. Only using the first CNN is sufficient for our tasks. The architecture of its CNNs was inspired by u-net [11] and modified for performance reasonings. A weighted cross-entropy loss function inspired by [11] was modified and then used to control the process of training the above CNNs. It is calculated as:

$$L(p, y) = -\frac{\sum_{q\in\Omega} w(q)\log\left(p_{y(q)}(q)\right)}{\sum_{q\in\Omega} w(q)}, \quad (1)$$

where p is the network prediction with respect to a reference output y. The pixel weight, w, the function is unique for every training sample. It is determined by the following equation:

$$w(q) = \left[1 + a\sum_{\phi\in\Phi}\max(d - ||q, \phi||, 0)\right] \cdot b, \quad (2)$$

where the Euclidian distance from q to the closest pixel in ϕ is represented by $||q, \phi||$. Both of the CNNs were trained from scratch using training procedures that follow the standard practice and experience of similar approaches [12][13]. We store the resulted markers prediction images and feed them into the segmentation step.

b) Fluo-N2DL-HeLa: This dataset contains images with a light background and light cell objects. To illuminate the cells, we create a new image using the following procedure. First, the minimum value *min* and the maximum value *max* is calculated then the value *m* by 255 divided the difference between *max* and *min*. Then each pixel in the original image subtracts the *min.* Each pixel of the result images is then multiplied by *m* as shown in the following equation:

$$p_{out} = (p_{in} - min) * \frac{255}{max - min}, \quad (3)$$

where p_{out} is the pixels of the resulting image and p_{in} is the pixels of the original image.

c) PhC-C2DL-PSC: No further operations for this dataset.

2) Segmentation: Binary thresholding is used for segmentation step. Because all of the dataset images are grayscale images, we can replace each pixel's value with either 0 which is black or 255 which is white. Like the name suggested, a constant threshold value, *T* is used. For each pixel in the image, we compare the image intensity of that pixel with the value *T*, if it is smaller than *T*, we replace that pixel to a black pixel, otherwise, we replace that pixel to a white pixel. Ideally, this will extract the cell objects as white and the background as black.

3) Drawing Bounding Boxes: As per the project specification [3], we are asked to draw a bounding box around each cell on every frame images. Because we have already converted our frame images into binary images where cell objects are white and backgrounds are black as the results of the segmentation step. Functions using algorithms introduced in [14] are applied to retrieve contours of each binary image. The contours are a list of point sets, we then draw bounding rectangles around each point set. Because some small background noises, we calculate the area of the bounding rectangles we just drew and only choose those lay within the range of the areas of our cell. Because the average cell size of each dataset is different, we choose different offset values here as well.

B. Tracking Module

Tracking and detecting cell divisions are difficult because our datasets are recorded living biological systems. In such a system, cells move around with a wide range of movement characteristics and they often interact so closely with each other [8]. We developed the following centroid based module that tries to mitigate the impacts described above.

1) Trajectory presentation: This step displays the history travel path up until the current frame. In order to manage easily, we create a struct class called *tracker* and create an instance of it every time program is run. First, we calculate and draw a centroid for each cell based on its bounding box. Then register these centroids and their coordinates to the *tracker* instance. Next, we assign a unique identifying number for each centroid and record it as also. For each new frame, the previous frame's recorded data in the *tracker* instance are pulled, compared with the data from the current frame. Now for each centroid in the previous data, we compute the distance between it and each centroid in the current data. The smallest the value would be chosen, it is the closest point and we treat it as the same centroid that moved. The new centroid coordinate is stored in the list of that centroid history positions. Starting at the first element in that list, which is the original point, a line is drawn, up until the current frame. If the total number of centroids is bigger than the previous one, we use the following mitosis detection step is to detect if there was a mitosis event. If the total number of centroids is smaller than the previous one, we perform our normal steps first and discard those unmatched. This occurs because of detection errors.

2) Mitosis Detection: If the total number of cells in a frame image is larger than its previous frame, mitosis events may happen. We are asked to catch these events and change the colour of the bounding box for those cells are dividing. The main problem at this stage is that because of the limitation of detection accuracy, it is easy to cause tracking points to disappear. After the tracking point disappeared, we had to ask the *tracker* instance to immediately delete the information about this point from its history recording. Otherwise, the *tracker* instance will treat the new cell as a reappearing old cell, which may lead to misjudgment of mitosis. Because the *tracker* instance relies on the ordered dictionary struct to store centroids and centroids history. We created another two lists called *key_list* and *pre_key_list*,

which store the keys in the centroids dictionary of the current and previous frame in order. Search backwards to find out which values of *key_list* are larger than the max value in *pre_key_list*. These values are considered as the keys of the candidate of new cells in the centroids dictionary. We all know that mitosis produces two cells, one is regarded as a new cell by the tracker, and the other is regarded as the original cell. In the end, the pair of cells that successfully matched with each other can be certainly determined as new cells from mitosis. This method effectively avoids various problems caused by detection errors. For example, as can be seen from the left image of Fig.2, only one new cell appears in the picture, and there are no paired cells around it. So there's a high probability that this cell is the result of a detection error rather than mitosis. In contrast, in the right image of Fig.2, each new cell found a matching cell nearby, so they were seen as a product of mitosis.

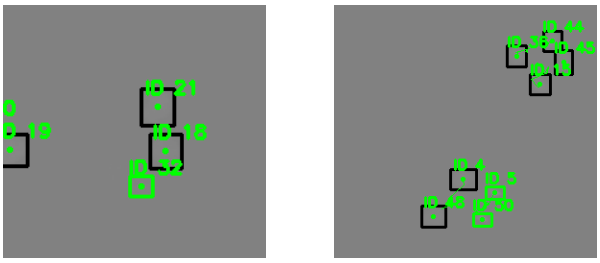


Fig. 2. Mitosis example

3) Analysis Module: We calculate four features of cell motion in this module. The first one is the speed of the cell at the current time point. Because we already know the coordinate of each cell centroid and the coordinate of its previous position, we use the Euclidean distance to calculate the total distance in pixel each cell travelled in one frame. Use total distance divide by one we get the current speed of each cell in pixels/frame unit. The total distance travelled each frame for each cell is recorded. The second one is the total distance travelled up to the current time point. The distance travelled each frame is added together for each cell. This way we can get the total distances in pixel the cell moved. The third one is the net distance up to the current time point. It is calculated by the Euclidean distance between the cell's coordinates in the earliest time point of its trajectory and the coordinates in the current time point. This is the way we can get much distance linearly the cell moved. Moreover, using the total distance travelled by the cell up to the current time point divides the net distance travelled up to the current time point, we can get the confinement ratio of the cell motion. After those calculations, the results of them usually are irrational numbers or infinite loop decimals which are very long and hard to display. Hence, for convincing reasons, the output result is only displayed in two decimal places. After all, we implement a function to listen to the mouse event. Users could select each cell and the four features calculated above would be displayed on the terminal.

IV. Experimental Setup

A. Datasets

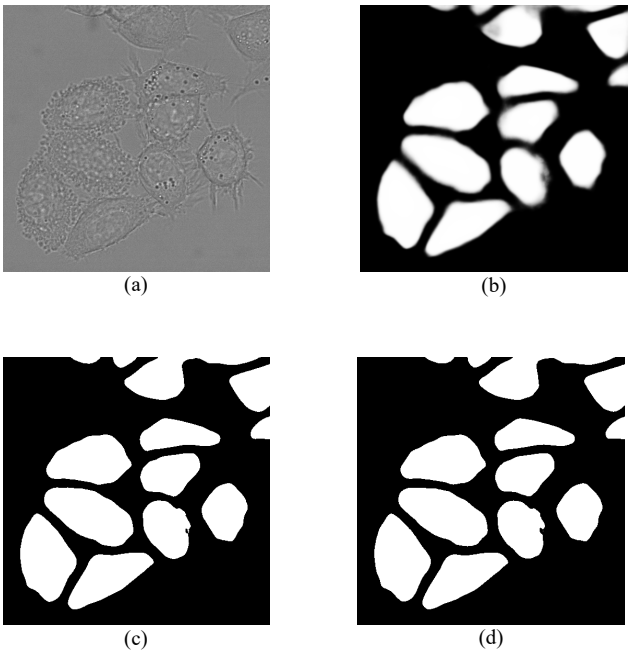
Three datasets are used for this project. Four sequences of image frames are in each of the datasets. The first dataset, *DIC-C2DH-HeLa*, contains Hela cells and was captured by Dr G. van Cappellen in Erasmus Medical Center of Rotterdam, the Netherlands, using phase-contrast microscopy [5]. Each frame in its sequences is a 512 × 512 grayscale image. There are 84 frames in the first two sequences and 115 frames in the latter two. A sample image can be seen on Fig.1. The second dataset, *Fluo-N2DL-HeLa*, contains Hela cells and was provided by Mitocheck consortium, using fluorescence microscopy [5]. Each frame in its sequences is an 1100 × 700 grayscale image. There are 92 frames in all four sequences. A sample image can be seen on Fig.2. The last dataset, *PhC-C2DL-PSC*, contains pancreatic stem cells and was captured by Dr T. Becker in Fraunhofer Institution for Marine Biotechnology of Lübeck, Germany, using phase-contrast microscopy [5]. Each frame in its sequences is a 720 × 576 grayscale image. There are 426 frames in the first two sequences and 300 frames in the latter two. A sample image can be seen on Fig.3.

B. Working Environment

We perform and experiment our method on an Intel 8-Core i9 2.3 GHz processor with 16 GB of RAM. PyCharm Professional 2020.2 is used as the main coding environment. Our code is tested on Python 3.7 with *opencv-python* 4.2.0.34, *tensorflow* 2.0.0 and *Keras* 2.3.1 installed.

C. Hyperparameter Settings

We have done several adjustments and experiments on the hyperparameter of the thresholding functions, *T*, for all three datasets. For dataset *DIC-C2DH-HeLa*, we tried 100, 110, 127, 200 as shown in Fig. 1. As we can see that because the deep learning model did a good job of detecting each cell. Different values of *T* do not have many differences. We choose to use 127 as the *T* value.

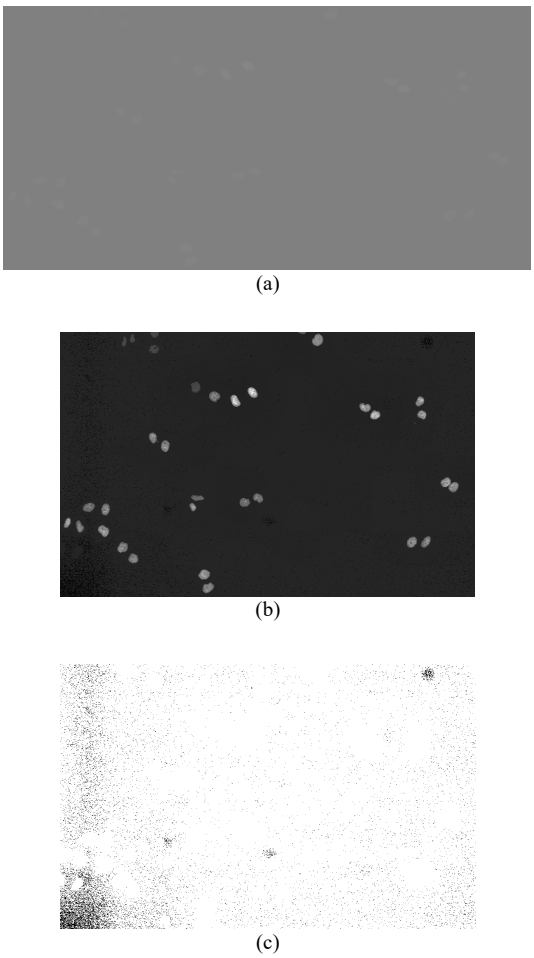


(a) (b) (c) (d)

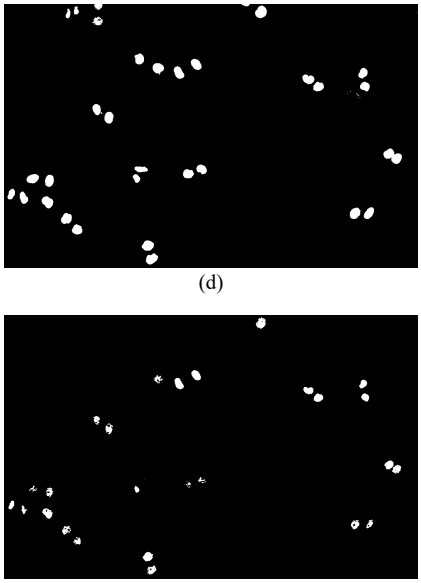
(e) (f)

Fig. 3. Illustration of *DIC-C2DH-HeLa* dataset and thresholding segmentation. (a) A sample image frame from this dataset. (b) Image after preprocessing step. (c) Thresholded image using T=100. (d) Thresholded image using T=110. (e) Thresholded image using T=127. (f) Thresholded image using T=200.

For dataset *Fluo-N2DL-HeLa* we tried three different values of *T* for thresholding functions and the result is shown in Fig. 2. We tried 20, 37 and 127. We choose 37 for this dataset.



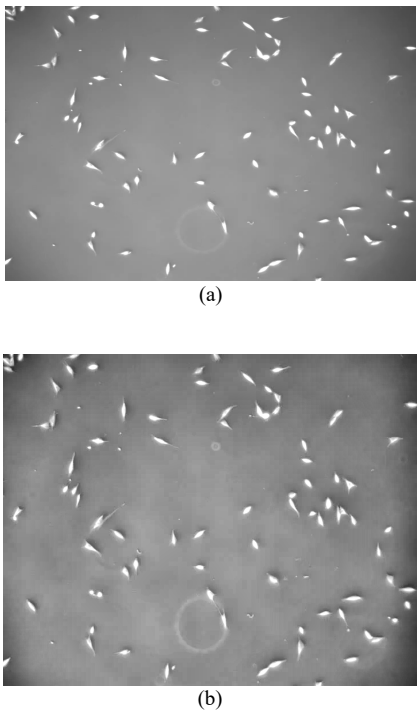
(a) (b) (c)



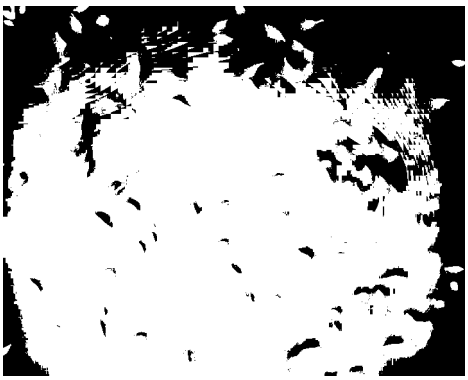
(d) (e)

Fig. 4. Illustration of *Fluo-N2DL-HeLa* dataset and thresholding segmentation. (a) A sample image frame from this dataset. (b) Image after preprocessing step. (c) Thresholded image using T=20. (d) Thresholded image using T=37. (e) Thresholded image using T=127.

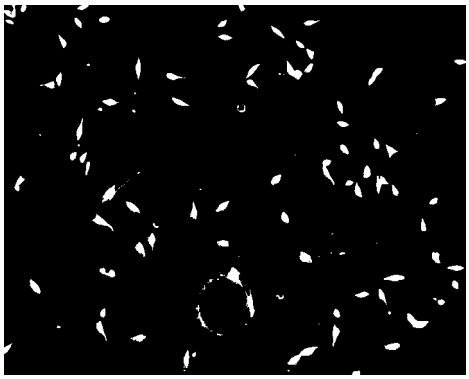
The final dataset is the *PhC-C2DL-PSC* dataset. We tried three values for *T* of the thresholding function. We tried 127,170 and 200. We decide to use 170 because the 200 choice seems to lose too many details. The results are shown in Fig. 3.



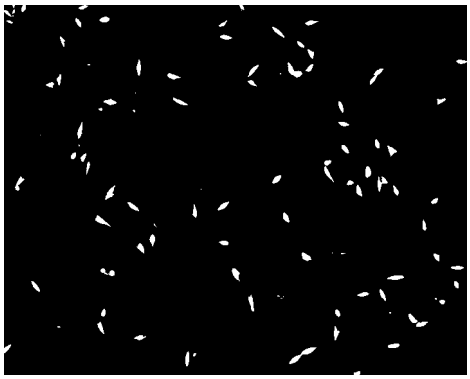
(a) (b)



(c)



(d)



(e)

Fig. 5. Illustration of *PhC-C2DL-PSC* dataset and thresholding segmentation. (a) A sample image frame from this dataset. (b) Image after preprocessing step. (c) Thresholded image using T=127. (d) Thresholded image using T=170. (e) Thresholded image using T=200.

V. Results and Discussion

Because there were no labelled datasets nor ground truth datasets offered for all the images, we evaluate the three datasets manually. The first 10 frames of each dataset's first sequences are evaluated.

A. Evaluation of Segmentation

To evaluate the performance of the detection module. We manually look at each image frames of the first 10 in each dataset, we assume our observations are correct, then the number of manually detected cells are recorded. Then we compare it with the results of our program. Because errors are not only caused by fail to detect the cells, other errors, such as drawing rectangles on areas that do not have cells also needs to be addressed. Hence, we use four parameters for our evaluation. The first one is a true positive (*TP*) which means our program believes there is a cell and we observe it as also. The second one is false positive (*FP*) which means our program believes there is a cell and we do not find it. The third one is a false negative (*FN*) which means our program missed a cell that we have observed. From these parameters, precision and recall scores could be calculated. Precision is the accuracy of the positive predictions and it can be calculated by *TP/(TP+FP)* while recall is the ratio of positive instances that are correctly found, and it can be calculated by *TP/(TP+FN)*. For convenience reasons, instead of using two more parameters, we combine precision and recall into a single parameter which would be the fourth parameter we use, called F_1 score and it is calculated as follows:

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall}. \quad (4)$$

The details of dataset names, total frame evaluated, the number of cells detected manually, and the values of *TP, FP, FN,* F_1 correspondingly are displayed on Table I.

TABLE I. Cell Segmentation Evaluation

Data	Total Frames	Detected Manually	TP	FP	FN	F_1 (%)
DIC-C2DH-HeLa	10	110	102	10	6	92
Fluo-N2DL-HeLa	10	370	345	86	45	84
PhC-C2DL-PSC	10	885	798	480	105	73
Total	30	1365	1245	576	156	77

B. Evaluation of Tracking

Similar to the evaluation of the tracking module, we manually observe the first ten image frames and record their path and cell division events. We use accuracy as the measurement for this evaluation. The details of dataset names, total frame evaluated along with the accuracy of a trajectory detection and mitosis detection can be found in Table II.

TABLE II. Accuracy of Tracking

Data	Total Frames	Trajectory Detection	Mitosis Detection
DIC-C2DH-HeLa	10	84%	80%
Fluo-N2DL-HeLa	10	88%	86%
PhC-C2DL-PSC	10	78%	74%

C. Error Analysis

During *DIC-C2DH-HeLa* dataset segmentation attempts, almost all the traditional algorithms could not effectively segment these cells. Before using the pre-

trained neural network, we once tried several self-written algorithms that failed, and we want to present and analyze here.

1) The first attempt: Empirically, cells are surrounded by visible dividing lines. After the processing of improving the contrast, this algorithm analyzes the obvious grey level change between adjacent squares one by one with squares of (n*n) pixels. It is assumed that the initial state is outside the cell. When the grey level change exceeds the threshold value for the first time, the cell is considered to have entered the cell. The judgment of leaving the inside of the cell is more complicated because there are also uneven tissues inside the cell, so after the change of grey level, it needs to keep relatively stable grey level for a longer distance before it can be considered to have left the inside of the cell. However, this is where the algorithm fails. The cells are so tightly bonded, there is little 'outside' space between the two cells. No matter how we adjust the parameters, the image we get is always unusable. The failure picture is shown as Fig.6.

2) The second attempt: This one is more sophisticated. In our opinion, without using CNN, the only thing that can distinguish a cell from a cell in *DIC-C2DH-HeLa* dataset is its tissue density, in other words, the disorder of its regions. The cytoplasm and cell boundaries always have more small tissue, the nucleus is relatively flat, and the area without cells is the flattest. Hence, we tried to use the regional grey standard deviation method to determine the location of the nucleus. In order not to miss any tissue structure, we adopted a strategy of per-pixel calculation. After gaussian blur, the standard deviation of the pixel grey within a box (the size of the box is an adjustable parameter) around each pixel is calculated, and the results are stored in a list. In other words, this list stores the standard deviation of the area around this pixel.
When the calculation is done, copy and sort the list and find the median. The original list is used to determine the location of the pixel that specific value represented.
Then it can be assumed that the values in the list close to the median represent the pixels in the picture with a moderate degree of chaos, that is, the location of the nucleus. The method failed because it was too sophisticated. The size of the calculation box, including the ratio of length to width, the level of Gaussian ambiguity, and the choice of the median interval will strongly influence the final result, and the robustness is extremely low. This work is more like what CNN does than manual adjustment. Fig.7 is what we can get by adjusting parameters manually.

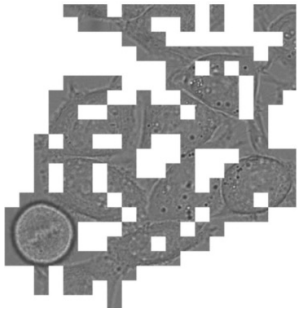


Fig. 6. The result from the first failed attempt.

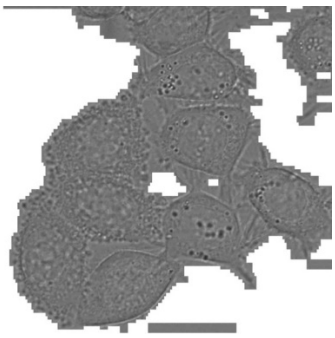


Fig. 7. The result from the second failed attempt

VI. Conclusion

The method we used in this project may be not that perfect, but at least we spent a lot of time trying different methods and found this method is very effective for the three different types of cell. This method increases the accuracy of cell detection and cell tracking. Though it is hard to distinguish when the cell and cell stick together in *DIC-C-DH-HeLa*, we still tried our best to increase the accuracy. We researched online and found the deep learning pre-trained model we are currently using and the detection rate for that dataset increased dramatically. The main problem for this project is that the limitation of detection accuracy would cause tracking points to disappear. After the tracking point disappears, the tracker should immediately delete the information about this point from centroid and centroid history. Otherwise, the tracker will treat the new cell as a reappearing old cell, which may lead to misjudgment of mitosis. This project is an amazing practice for this course, it combines all the features of the computer vision and OpenCV which was learned in this team. From basic image processing, threshold and morphological processing, gradient processing, edge detection, tracking, motion and a part of deep learning. In the future, we recommend the use of deep learning on the segmentation part.

VII. Contribution of Group Members

Generally, because getting great segmentation results of each dataset is the most import part and the first part of this project, four of the group members have tried many methods to get the best performance. Specifically, the contributions of each group member are detailed below on Table III while Ran Ji did not make any effective contributions on programming nor writing the report.

TABLE III. Contributions of Group Members

	Detection	**Tracking**	**Analysis**	**Report**
DIC-C2DH-HeLa	Luyuan He	Luyuan He and Rifang Li	Long Long and Luyuan He	Luyuan He, Rifang Li and Long Long
Fluo-N2DL-HeLa	Rifang Li			
PhC-C2DL-PSC	Rifang Li			

REFERENCES

[1] E. Meijering, O. Dzyubachyk, I. Smal, and W. A. van Cappellen, "Tracking in cell and developmental biology," Semin. Cell and Dev. Biol., vol. 20, no. 8, pp. 894–902, Oct. 2009.

[2] A. J. Hand, T. Sun, D. C. Barber, D.R. Hose, and S. Macneil, "Automated tracking of migrating cells in phase-contrast video microscopy sequences using image registration," Microscopy, vol. 234, pp. 62–79, 2009.

[3] E. Meijering, "COMP9517: Computer Vision 2020 Term 2 Group Project Specification", University of New South Wales, Sydney,NSW, Australia, 2020.

[4] V. Ulman et al. An objective comparison of cell-tracking algorithms. Nature Methods, vol. 14, no. 2, pp. 1141-1152, December 2017.

[5] "2D+Time Datasets", Accessed on: Aug. 7, 2020. [Online]. Available: http://celltrackingchallenge.net/2d-datasets/

[6] Beucher S, Lantuejoul C. Use of watersheds in contour detection. In: Proceedings of the International Workshop on Image Processing; 1979.

[7] Wählby C, Sintorn I‐M, Erlandsson F, Borgefors G, Bengtsson E. Combining intensity, edge and shape information for 2D and 3D segmentation of cell nuclei in tissue sections. J Microsc 2004;215:67‐76.

[8] M. A. A. Dewan, M. O. Ahmad, and M. N. S. Swamy, "Tracking Biological Cells in Time-Lapse Microscopy: An Adaptive Technique Combining Motion and Topological Features," in IEEE Transactions on Biomedical Engineering, vol. 58, no. 6, pp. 1637-1647, June 2011, doi: 10.1109/TBME.2011.2109001

[9] F.Lux, P. Matula, "Cell Segmentation by Combining Marker-Controlled Watershed and Deep Learning", arXiv:2004.01607 [eess.IV] https://arxiv.org/abs/2004.01607

[10] E. Moen et al. Deep learning for cellular image analysis. Nature Methods, vol. 16, no. 12, pp. 1233-1246, December 2019

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Im- age Segmentation," in Medical Image Computing and Computer-Assisted Intervention (MICCAI), vol. 9351 of LNCS, pp. 234–241. Springer, 2015.

[12] Christian Payer, Darko Stern, Marlies Feiner, Horst Bischof, and Martin Urschler, "Segmenting and Tracking Cell Instances with Cosine Embeddings and Recurrent Hourglass Networks," Medical Image Analysis, vol. 57, pp. 106–119, 2019.

[13] Fabian Isensee, Jens Petersen, Andre Klein, David Zim- merer, Paul F. Jaeger, Simon Kohl, Jakob Wasserthal, Gregor Koehler, Tobias Norajitra, Sebastian Wirkert, and Klaus H. Maier-Hein, "nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation," Informatik aktuell, p. 22, 2019.

[14] Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)