Universiteit
Leiden
The Netherlands

Aspects of the analysis of cell imagery: from shape to understanding

Li, C.

Citation
Li, C. (2024, June 27). *Aspects of the analysis of cell imagery: from shape to understanding*. Retrieved from https://hdl.handle.net/1887/3765419

Version:	Publisher's Version
License:	Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from:	https://hdl.handle.net/1887/3765419

Note: To cite this publication please use the final published version (if applicable).

Chapter 4

An Automated Cell Tracking Pipeline for the Analysis of Neutrophil Dynamics

This chapter is based on the following publications:
C, Li., W, W.C. Yiu., W, Hu., L, Cao., H, P. Spaink., F, J. Verbeek., An Automated Cell Tracking Pipeline for the Analysis of Neutrophil Dynamics. **Prepared for submission**.

W, Hu., L, van Steijn., **C, Li.**, F, J. Verbeek., L, Cao., R, M.H. Merks., H, P. Spaink., A Novel Function of TLR2 and MyD88 in the Regulation of Leukocyte Cell Migration Behavior During Wounding in Zebrafish Larvae. *Frontiers in Cell and Developmental Biology*. Vol. 9, 2021.

Abstract:

Neutrophils play a key role in the innate immune system. They act as the primary line of defense when bacteria, viruses or other harmful foreign particles invade the immune system. Accurate measurement of movements of neutrophils including velocity, direction and displacement, are crucial to study the regulation of cell migration behaviour. Cell tracking is a key technology to realize the quantification of these measurements. In this paper, we developed a pipeline, including cell segmentation, cell motion tracking between two frames and trajectory linkage, to realize tracking of the cell. Our starting point was to collect time-lapse sequences of neutrophils using a confocal microscope. We pre-processed each frame in the time-lapse sequence so as to improve the image quality by denoising, smoothing and contrast enhancement. Subsequently, a deep learning model, i.e. U-Net, was used to segment cells in each image frame. U-Net was used again to track the cells between two adjacent frames by calculating the score matrices representing the posterior probability of linkage. Moreover, an extended Viterbi algorithm was applied to find optimal trajectories based on score matrices obtained from U-Net. Results demonstrate that our pipeline outperforms the other state-of-the-art methods. It has a great potential to be applied in other cell migration studies.

4.1 Introduction

Acute inflammation could be identified by the monitoring of neutrophil migration, which can be caused by invading pathogens [121]. Neutrophils are one of the crucial immune cells. The migration pattern of neutrophils reflects the process *in vivo* of the innate defense mechanism against invading pathogens. However, the underlying mechanisms of neutrophil migration are not yet completely understood. In the field of biology and medicine [122][123], microscopy techniques enabling observing at high spatial and temporal resolution have contributed to the advancement of the analysis of the dynamics of the cellular processes. As the first line of defense against harmful pathogens, the exploration of movements of living neutrophils, *in vivo*, in the spatial-temporal domain can contribute to a better understanding of their behaviours and mechanisms. Cell tracking is the key technology to realize the quantitative analysis of neutrophil migration. It aims to identify the trajectory of each cell in a time-lapse sequence. However, due to the complexity of cell behaviour this field still lacks accurate and automated algorithms. The manual tracking of cells is a time-consuming and challenging task. Especially with the introduction of high throughput screening, it becomes impossible to conduct manual tracking for a large number of image datasets. Therefore, developing automatic and powerful cell-tracking approaches is essential.

In biomedical research, cell tracking necessarily is preceded by cell detection and based on an approach with either single-cell tracking or multi-cell tracking [124][125][126][127].

The main idea of tracking by detection is to segment cells from each frame first and then associate the cells frame by frame. Cell segmentation algorithms are designed to localize single cell object in each frame. Initially, the majority of existing approaches are based on different strategies such as thresholding, watershed, and deep learning neural networks [128][129][130][131], etc. The Otsu adaptive thresholding method [132] is one of the thresholding strategies, which can find an optimal value by maximizing the variance between foreground and background, so as to distinguish them.

The watershed algorithm described by Vincent [133] calculates the gradient magnitude of the image to identify potential region boundaries, and markers are selected and placed on the image. The watershed transform simulates a flooding process where water is poured into the valleys, gradually filling them up. When water from different regions meet at the same point, dams are formed, which represent segment boundaries. This guide to partitioning an image into distinct regions or objects.

The aforementioned rule-based approaches are somewhat reliant on human intervention for the fine-tuning of parameters. Recently, supervised deep learning methods become popular due to their learning-based capacity and do achieve good results in the field of cell segmentation. U-net [134] is a very successful deep learning model for segmenting

biomedical images due to its light-weighted and well-performed architecture. Subsequently, different improved versions of U-Net such as Res-UNet [135] and Dense-UNet [136] have been constructed and applied on biomedical segmentation tasks. U-Net is originally a semantic segmentation method that focuses on the separation between foreground and background. It could not identify each object individually if objects are colliding or overlapping. Cell segmentation, as a pre-step of tracking, is expected to separate each single cell from the image. This can be achieved by instance segmentation. A network with one encoder and two decoder paths, was designed based on U-Net structure, along with a watershed post-processing, to perform the instance cell segmentation [125]. It has proven to achieve top performance in the IEEE ISBI 2020 Cell Tracking Challenge. All of this related research confirms the power of the U-Net architecture for biomedical image segmentation tasks.

Once cell segmentation is performed, the next step is the cell association. It determines the linkages between cells on a frame-by-frame basis. Classic linkage methods include nearest neighbour and linear programming [131][137]. The nearest neighbour method links the cells to the nearest cell on adjacent frames based on the characteristics of cells such as intensity, shape [138], motion etc. For instance, Yan [126] proposed an algorithm named Kernel Density Estimation Mean Shift (KDE) as a linkage solution. It started by converting the initial object into density models and recursively update the mean shift factor based on a local density in the consecutive frames until a stationary location is reached. The cell closest to the stationary point was chosen as the candidate. However, a simple nearest neighbour linkage method could not solve the more complex cell behaviour well; i.e. events such as cell appearing, disappearing, merging, and splitting during the movements. Linear programming algorithms proved to successfully solve these problems [124][135][137]. In particular, the Viterbi algorithm, a classic linear programming algorithm. It is a global method to link cell trajectories based on the probabilistic functions or scoring functions obtained from feature similarities of cells. The excellent performance of the Viterbi linkage algorithm has been reported in the past years [124][139][140].

The main idea of the aforementioned methods is to extract the handcrafted cell similarities followed by a linkage method. In recent years, deep learning methods become popular for cell tracking tasks [127][141][142][143]. There are two ways to realize cell tracking using deep learning methods. One is a two-step approach that a deep learning model learns the cell migration patterns from the ground truth data, which replaces the process of handcrafted cell similarity extraction. Subsequently, a linkage method is constructed. Lugagne et al. [127] proposed a U-Net structure for learning the cell movement patterns from an *E.coli* dataset. From the predictions of the tracking model, a set of score matrices, including the probability of each pair of cells among all consecutive frames, were calculated. A nearest-neighbour

linkage method was used to associate the cells over the frames, by which a sound performance was achieved. The other way is a one-off training process. A deep learning model is trained to simultaneously learn the cell movement patterns and also the linkages. Christian et al. [141] presented a novel recurrent fully convolutional network for tracking on ISBI Cell Tracking Challenge. The network was trained on a dataset including both segmentation and tracking ground truth data for each frame. Cells in the public dataset have associated instance IDs which are used to annotate the trajectory of each instance cell. Marzieh et al. [142] proposed a framework based on a novel multi-channel feature learning model to track cells. This model can learn the spatial and temporal features simultaneously over time-lapse sequences from the ground truth data. They built a ground truth dataset by manually cropping individual cells of any consecutive frames to form image pairs. The pairs of images were annotated as true if they are the same cell in two adjacent frames, otherwise, it was annotated as false. In related work, simultaneous cell tracking in spatial and temporal feature space with a deep learning framework is gradually becoming a trend but still has its limitations. This is because the preparation of a large volume of ground truth data to train a deep learning model is very laborious. This is especially the case for the complex time-lapse sequences such as those focussing on neutrophil movement as subject in our research. Therefore, a two-step tracking strategy is most commonly used in the existing literature and is also preferred for our study.

Furthermore, recently many standalone tools were developed in support of cell tracking tasks, such as Baxter [140], TrackMate [144], PhagoSight [145], TrackPad [146], and CellProfiler [147]. In general, these open-source and interactive tools integrate many functionalities including cell segmentation, tracking, visualization, manual correction, and analysis. These tools have advanced the field of cell tracking for different cell types, however, do not fit all types. In our neutrophil dataset, complex cell events are encountered such as appearing, disappearing, merging, and splitting. Existing tools have difficulties to track these cell accurately.

Based on the motivations, for our research, we designed a cell tracking pipeline to track the complex migration dynamics of neutrophils. The proposed pipeline comprises three steps. For the first step of cell segmentation, different segmentation models were investigated. U-Net model was recognized as the most suitable model to segment neutrophils from images. Subsequently, a two-step association strategy was applied. The scoring function of cell similarities was obtained by another U-Net-based tracking model. It learned the migration patterns between adjacent frames from the ground truth data. Lastly, an extended Viterbi linkage algorithm was proposed to solve the complex linkage problem. Compared with the other state-of-the-art methods [126][127][148], our algorithm achieved the lowest mean value of 0.010 of Falsely Identified Tracker (*FIT*) and 0.173 of Falsely Identified Object

(*FIO*), together with the highest Track Purity (*TP*) value of 0.763 and a second-best Object Purity (*OP*) of 0.299 for our neutrophil dataset.

The contributions of our research are summarized as follows: (1) We built a ground truth dataset of neutrophil migration for the zebrafish model, which provides a potential chance for other researchers to investigate similar programs; (2) We developed a pipeline to solve the neutrophil tracking challenges, which integrates cell segmentation, cell motion tracking, and trajectory linkage; (3) We designed a creative Viterbi algorithm for trajectory linkage. Different heuristics were formulated for specific migration patterns of neutrophils, which aims to solve complex tracking problems.

4.2 Materials and Methods

The framework of our pipeline is shown in Fig. 4.1. At first, a pre-processing was done to improve the image quality of the time-lapse sequence. Two U-Net-based deep learning models were applied respectively for segmentation and tracking. After training the tracking model, the predictions of the possible cell location on the next frame were generated. Finally, the extended Viterbi algorithm was performed to link the trajectories based on the predictions. The trajectories were visualized through a time-lapse sequence. More methodology details were introduced in the following subsections.

4.2.1 Image Capturing and Pre-processing

The neutrophil time-lapse sequences are captured from the tail of a zebrafish after an induced tail-wounding. All the experiments with zebrafish followed the international guidelines specified by the EU Animal Protection Directive 2010/63/EU. The zebrafish in the experiment were 3 days post-fertilization (dpf) and the tail wounding was conducted [121]. The wounded tail area of specific samples was imaged using a Leica TCS SP8 confocal microscope (Leica Microsystems) with a 10× objective (N.A. 0.40). Neutrophils, localized within an area of 200µm from the wounding edge toward the body trunk, were counted as recruited cells. Under the microscope, the zebrafish tails were scanned from top to bottom. A total 8-layer 3D stack at each time point was captured. The image stack size is $512 \times 512 \times 8$. The layer interval is 5µm-6µm, while the thickness of the tail is 35µm-42µm. For each sample, a 2-hour time-lapse sequence was captured, with a time interval of one minute. Thus, there are 120 frames for each time-lapse sequence. Each time-lapse sequence contains around 10 to 40 cells. We captured 20 time-lapse sequences in total.

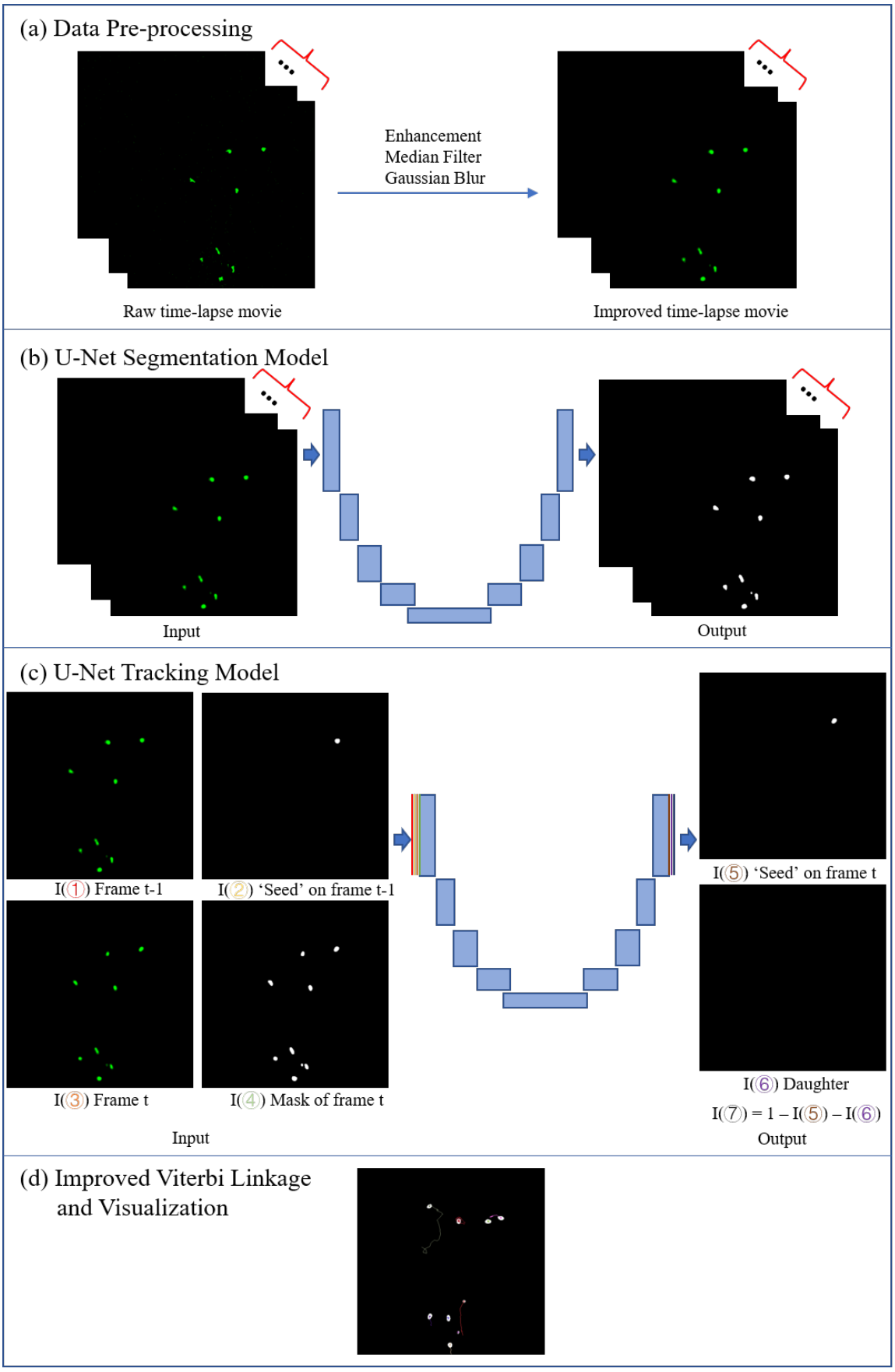

Fig. 4.1 The pipeline of cell tracking. (a) Data pre-processing. (b) U-Net segmentation model. (c) U-Net tracking model. I(①)I(②)I(③)I(④) represent the four inputs and I(⑤)I(⑥)I(⑦) represent the three outputs of the model. I(⑦) is calculated by the reversion of the other two outputs I(⑤) and I(⑥). (d) Viterbi linkage and visualization.

The raw data we obtained are thus 3D + t time-lapse sequences. Due to the relative undersampling in Z-axis direction, for this paper, we intend to focus on solving tracking problem for neutrophils in 2D space and to evaluate how well a 2D cell tracking method can solve the problem. To that end, a maximum intensity projection was applied to the 8-layer image stack so as to derive a 2D projection data at each time point. In Fig. 4.2 (a), the first projected frame of one sequence is depicted. The intensity is relatively low and the cells are not clearly visible. Therefore, image pre-processing steps were applied to improve the image quality. First, we enhanced the image contrast to highlight the cell; cf. Fig. 4.2 (b). However, this also enhanced the background noise. Therefore, to denoise a median filter was used; cf. Fig. 4.2 (c). Subsequently, we chose a Gaussian blur to smooth the cell surface area; cf. Fig. 4.2 (d). The corresponding magnified area in the red box of Fig. 4.2 (a)(b)(c)(d), are shown in Fig. 4.2 (e)(f)(g)(h) respectively, so as to clearly present the image details.

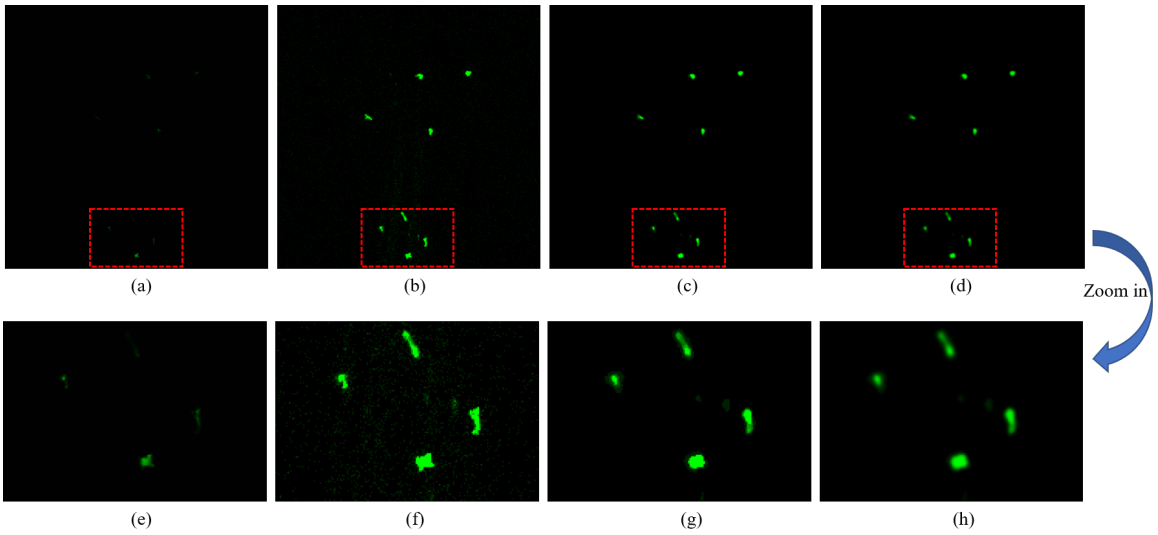

Fig. 4.2 Data pre-processing. (a) One frame of the raw data. (b) Image contrast enhancement. (c) Image denoising with median filter. (d) Image smooth with Gaussian blur. (e)(f)(g)(h) are the corresponding zoom-in areas of the red boxes in (a)(b)(c)(d), respectively.

4.2.2 Cell Segmentation

4.2.2.1 Ground Truth Labelling for Segmentation

Annotated images should be prepared. They are used for the stages of both training and evaluation of segmentation models. We used AnnotatorJ [149] to label each cell. This is a semiautomatic tool to find an approximate cell mask by applying U-Net-based pre-segmentation. It follows a manual correction to precisely create the masks of individual cells. Users can create their own annotated dataset based on different options: binary mask annotation and instance mask annotation. In our study, we created 240 images of instance

mask annotations. Binary mask annotations can be obtained by thresholding. Both binary and instance mask annotations were used to train the U-Net semantic segmentation model and instance segmentation model, respectively, as well as evaluate the segmentation performance.

4.2.2.2 Segmentation Models

A good segmentation model is expected to identify each cell, without missing cells, over-segmentation, and under-segmentation. In our study, we intend to find an accurate segmentation model for the neutrophil image dataset. Four models were built and compared.

First, we implemented a U-Net model (U-Net 1). It has a similar structure to the original U-Net model [134] but without an overlap-tile strategy. Both the input and output sizes of the network were set to 512 × 512, which is the same as the raw image size. U-Net 1 was trained over 10 epochs of 300 steps on the Tensorflow platform on a dedicated server equipped with two NVidia GeForce GTX 2070/8 GB GPUs. The batch size was set to 1 due to the large image size and the limitation of GPU memory. Data augmentation was applied in order to increase the number of training sets and avoid overfitting of the network. In our case, width and height shift, vertical and horizontal flip were used to augment our dataset because these transforms do not change the cell morphology. For our dataset, the pixel number of the foreground (positive sample) and the background (negative sample) are unbalanced. The negative samples occupy the most area of the image. The consequence is that no candidate cells would be predicted and generated by the network. Thus, selecting a suitable loss function plays a key role during training. Dice loss function and focal loss function [150][151] have demonstrated to be very effective in solving class unbalanced problems and thus were compared these two in our experiments. Subsequently, the original U-Net model (U-Net 2) [134] was implemented. We intend to compare both nets. The differences between U-Net 1 and U-Net 2 are: U-Net 2 works with an overlap-tile strategy. The input and output sizes are 512 × 512 and 388 × 388, respectively. The output images need to be resized to 512 × 512 so as to get the same size images as our raw data.

In addition, an instance U-Net segmentation model (U-Net 3) [125] was adopted. The structure of U-Net 3 has one encoder path and two decoder paths which aim to do instance-based segmentation task. This model was involved to test the possibility of using instance-based segmentation model to solve the under-segmentation problems encountered in the two U-Net-based semantic segmentation models. In addition to the deep learning models, a rule-based Watershed Masked Clustering (WMC) segmentation algorithm [152] was included since it has been used with a similar cell tracking task.

4.2.2.3 Evaluations for Segmentation

In order to evaluate the performance of the segmentation models, we collected five commonly-used evaluation metrics [153]. The first one is the Jaccard index, which is defined as:

$$IoU(T,E) = \frac{T \cap E}{T \cup E} \tag{4.1}$$

T represents the ground truth image, E represents the predicted segmentation image. The metric relies on the computation of Intersection-over-Union (IoU) between a pair of objects on T and E, respectively. IoU measures pixel-wise foreground overlap between them.

The second commonly-used evaluation metric is the $F1$ score. We measured the cell number of True Positive (TP), False Negative (FN), and False Positive (FP). $F1$ score is computed by:

$$F1 = \frac{2TP}{2TP + FP + FN} \tag{4.2}$$

Where TP represents the corresponding pair of objects on both T and E. FP represents the objects which are predicted on E while not annotated on T. FN is the objects which are missed by the segmentation model compared with T.

FN is considered as a third evaluation criterion individually. Because cell missing is not expected in the tracking task. It causes a loss of target and a wrong connection of trajectory. Besides IoU, $F1$ score, and FN, we also counted the number of under-segmented and over-segmented cells. These two measurements reflect the ability of the segmentation model to solve the under-segmentation and over-segmentation problems.

In principle, a better performance is achieved with a higher IoU and $F1$ score, together with a lower cell number of FN, under-segmentation, and over-segmentation. In practice, however, all of these metrics are interrelated. It depends on the value of threshold t that sets the predicted probability of each pixel above t as foreground and below t as background. If t is too high, the number of pixels that are recognized as the foreground decreases. It causes the predicted cells to have a much smaller surface area, resulting in a lower IoU value, a higher number of over-segmented cells, and a lower number of under-segmented cells. In addition, some small cells are missed, which caused FN to increase further. Similarly, if t is too low, we got results on the contrary. In our experiment, we set $t = 0.1$ in order to keep a balance among all of these evaluations.

4.2.3 Cell Tracking

4.2.3.1 Ground Truth Labelling for Tracking

For a supervised deep learning model for tracking, the ground truth dataset is required. In support of our study so as to obtain such dataset, a manual tracking was performed by experts [121] using the TrackMate plugin [144]. For each time-lapse sequence, the expert labeled 3 to 9 trajectories. In total, there are 110 trajectories from all 20 time-lapse sequences. The (x, y) coordinates of each trajectory were recorded, which were used to localize the cells' positions in the time-lapse sequence.

4.2.3.2 U-Net for Tracking and Evaluations

A U-Net-based deep learning model was adopted to learn the features of cell motion from adjacent frames of the ground truth trajectories.

The U-Net-based model was designed for tracking *E.coli* cells trapped in mother machine microfluidic chambers [127]. In this case, the cell movements were constrained in the vertical direction. It is a simplified cell tracking problem. We intend to apply and extend this idea to a more complicated case. The U-Net architecture has four inputs and three outputs (cf. Fig. 4.1 (c)). So, for every cell at every time point of the ground truth trajectory, there are four inputs and three outputs. The four inputs include the current frame, the segmentation mask of the current frame, the previous frame, and the 'seed' cell that we want to track from the previous frame. The three outputs include the 'seed' cell on the current frame, the potential daughter cell on the current frame if a division just happened and the mask image of the reversion of the other two output images. For our dataset, we generated the training set including input and output images from ground truth trajectories based on the recorded coordinates. Cell division events do not occur in our case, thus all of the daughter images are empty.

In this study, we divided all 110 ground truth trajectories into training, validation and test sets based on a ratio 8:1:1. We compared the performance of three tracking models on test set. The first one was the pretrained model which was already trained on yeast cells [127]. This model was trained over 400 epochs of 250 steps per epoch with a batch size of 5 and a learning rate of $1*10^{-4}$. We used this model to evaluate the performance of test set directly. Moreover, we trained the other two models using TensorFlow on GPUs. One concerned a model from scratch. All the parameters of the network are initialized randomly and we trained this model based on our neutrophil training set. This model was trained for 20 epochs with mini-batches of size 1 and 250 steps per epoch and a learning rate of $1*10^{-5}$. The other model concerned a fine-tuned model. We fine-tuned the pretrained model on our training set, so as to update the parameters of the network. This network was trained for 20 epochs with a

batch size of 1 and a learning rate of $1*10^{-5}$. All of these hyperparameters were set based on experimental fine-tuning and empirical knowledge.

After we trained these three U-Net models, we evaluated the performance of each model by counting the number of incorrectly predicted cells on the test set.

4.2.3.3 Extended Viterbi Linkage Method

After the training, the images of predictions were generated from the tracking models. The score matrices were computed from these predictions. The pixels attributed to the cell in each predicted image were matched to the pixels in the segmentation mask of the corresponding frame. The overlap area of each pair of cells on the two images was calculated and a score matrix for this specific time point was generated. Fig. 4.3 shows three simple examples of score matrices among four consecutive frames. The number of rows represents the number of cells in frame t, and the number of columns represents the number of cells in frame $t+1$. The values in the matrix represent the score between each pair of cells of the two adjacent frames. For each time-lapse sequence in our study, it has 120 frames in total. Therefore, we can obtain 119 adjacent score matrices for each time-lapse sequence.

Frame t + 1

Frame t	0	1	2	3	4
0	0.9	0	0	0	0
1	0	0.1	0.7	0	0
2	0	0.8	0.2	0	0
3	0	0	0	0.7	0.1

(a)

Frame t + 2

Frame t+1	0	1	2	3	4
0	0.9	0	0	0	0
1	0	0.1	0.7	0	0
2	0	0.8	0.2	0	0
3	0	0	0	0.7	0.1
4	0	0	0	0	0

(b)

Frame t + 3

Frame t+2	0	1	2	3	4
0	0.9	0	0	0	0
1	0	0.1	0.7	0	0
2	0	0.8	0.2	0	0
3	0	0	0	0.7	0.1
4	0	0	0	0	0

(c)

Fig. 4.3 Score matrices. (a) Matrix between frame t and $t+1$. There are four cells in frame t and five cells in frame $t+1$. (b) Matrix between frame $t+1$ and $t+2$. (c) Matrix between frame $t+2$ and $t+3$. For each value in the matrix represents the computed score of each pair of cells on adjacent frames.

Based on these score matrices, our aim was to find the optimal path for each tracked cell that achieved the highest accumulated score over the whole time-lapse sequence. A linkage method was required subsequently.

Behavioural Patterns of neutrophil migration in the time-lapse sequence Before introducing the linkage method, identifying the possible behavioural patterns of neutrophils can help us design a customized strategy for each pattern. We have observed and identified the following patterns of neutrophil movement: (pattern 1) migration over all frames in the

sequence; (pattern 2) newly appearing cells; (pattern 3) disappearing cells; and (pattern 4,5) merge and split of cells. These patterns are depicted in Fig. 4.4. For our 120-frame time-lapse sequence, very few neutrophils are always in the field of view of the images (pattern 1). In most cases, cells go in/out of view at some time point, which are defined as appearing and disappearing of cells. Merge and split patterns are the key problems that we aim to solve with some strategies. In this study, an extended Viterbi linkage method was proposed. The use of Viterbi algorithm was motivated by the power of the global track linkage and its superior performance in Muscle Stem Cells (MuSCs) and myoblasts cell tracking problems in [140]. The optimal trajectory for each cell is ensured by solving a dynamic programming problem over a state space diagram.

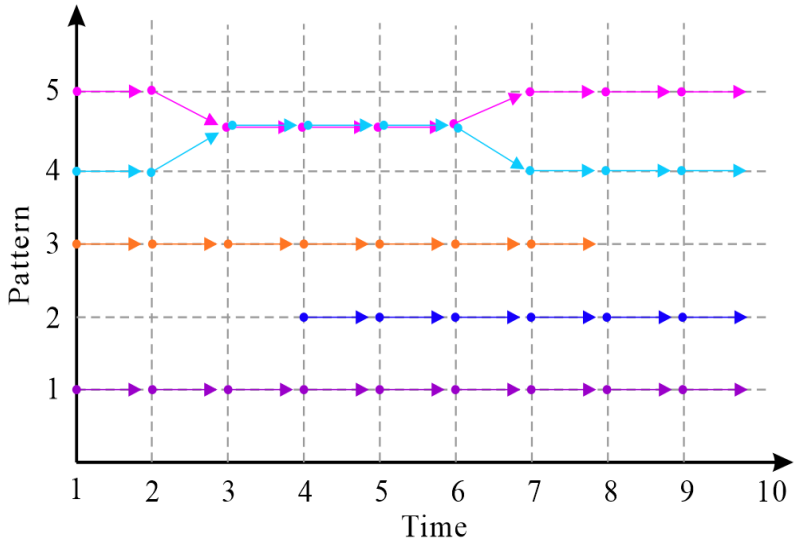

Fig. 4.4 The cell behavioural patterns in neutrophil time-lapse sequences. Pattern 1: migration over all frames; Pattern 2: newly appearing; Pattern 3: disappearing; Pattern 4 and 5: merge and split.

Viterbi Basic Rules The basic logic of the Viterbi algorithm comprises two phases, including a routing phase (forward) and a best track retrieval phase (backward). A routing phase starts from each cell on the first frame, iteratively, to ensure the optimal path based on the accumulated score on each node of the state space diagram. Fig. 4.5 is an example of the state space diagram. It is a fully connected network. It shows all cell nodes of four consecutive frames in Fig. 4.3. Suppose these four frames are the first four frames in a time-lapse sequence, thus t is equal to 1. We first looped each cell node (from $A0$ to $A3$) in frame 1 to find the optimal path through all frames. The values in score matrices correspond to the weights on the edges between each pair of nodes. Therefore, we can calculate the

accumulated score for each node. For example, we obtained the accumulated score of all the possible paths from node $A0$ to $C0$ via column B. The path $A0$-$B0$-$C0$ got the highest score of 0.72 (0.9×0.8). The other paths $A0$-Bi-$C0$ (i is from 1 to 4) reached a score of 0. We only maintained the best path to the next step and other paths were discarded in order to reduce the complexity of the algorithm. Similarly, from node $A0$ to $C1$, the optimal path is $A0$-$B0$-$C1$ with the highest score of 0.09 (0.9×0.1). From $A0$ to $C2$, $C3$, $C4$, the best paths are $A0$-Bi-$C2$ (0), $A0$-Bi-$C3$ (0), $A0$-Bi-$C4$ (0), respectively. Bi means no matter which B node the paths pass, the score of all paths was 0. Then we calculated the score from node $A0$ to all the nodes on column D based on the formerly remained best paths via columns B and C. These best paths are: $A0$-$B0$-$C0$-$D0$ (0.9×0.8×0.85), $A0$-$B0$-$C1$-$D1$ (0.9×0.1×0.2), $A0$-$B0$-$C1$-$D2$ (0.9×0.1×0.7), $A0$-$B0$-Ci-$D3$ (0). Ci represents no matter which C node the paths pass, the score of all paths is 0. Suppose column D is the last frame of the sequence, then the process enters the best track retrieval phase when the routing process reaches the last frame. Among all nodes D, $D0$ was the node with the highest score (0.9×0.8×0.85) and the path $D0$-$C0$-$B0$-$A0$ was retrieved as the optimal path. In the end, we obtained the optimal path from $A0$ to $D0$, which was shown in green nodes and edges in Fig. 4.5. In this way, other trajectories started from $A1$, $A2$ and $A3$ were also obtained and highlighted in different colours.

After all the cell trajectories in frame 1 were found, the cells that not passed by any of those trajectories in the next frame were treated as the newly appearing cells. In Fig. 4.5, $B4$ is a new cell appearing in frame 2, the same procedure was applied to find the best trajectory. For the cell which is disappearing, the corresponding value in the score matrix between itself and all the candidate cells in the next frame is very low. Here, we set a truncated threshold $T_{truncated}$. The score of a cell that is lower than $T_{truncated}$ was treated as a disappeared cell, referred to as the cell out of the field of view.

Dealing with Cells Merge/Split One drawback of the basic Viterbi algorithm is that it cannot deal with merge/split problems. Once the cell trajectories merged due to the collision or occlusion of cells at the same time point, the trajectories never split because they shared the same score matrices after the merge happened and ended up with the same optimal trajectory. This circumstance occurs as Viterbi treats each track to be independent from one another. However, in fact, the calculation of an individual cell trajectory is largely affected by the surrounding tracks. Therefore, to solve this problem, an extended Viterbi algorithm was developed to incorporate the surrounding tracks. To that end, three rules were set on top of the basic Viterbi algorithm to manage the merge and split scenarios among cell trajectories.

Rule 1: an alternative path routing is developed. It enables the track routing to merge and split with two specific conditions 1 and 2.

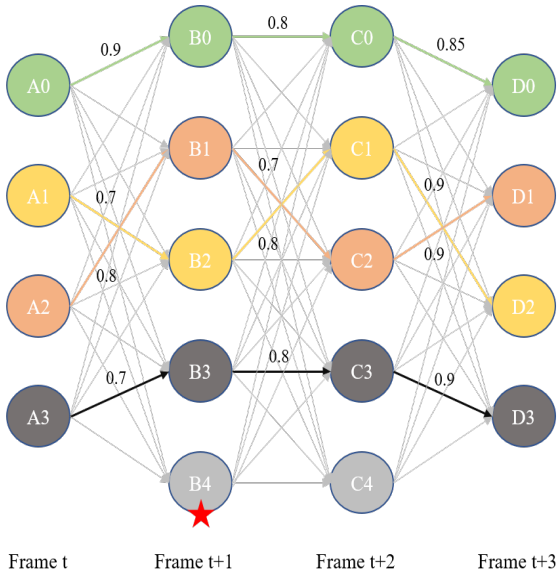

Fig. 4.5 State space diagram of four consecutive frames. The different four colours represent four trajectories which derived from basic Viterbi algorithm. This state space diagram also corresponds to the score matrices in Fig. 4.3. The red star represents a new start of the other trajectory in the loop.

Rule 2: a threshold T is created to support the alternative path routing. It provides a reference for merge/split decisions.

In the routing phase, the best node to frame $t-1$ could possibly be occupied by one or more cell tracks. In such circumstances, connection to that node would only be allowed if:

Condition 1: The total score of the routing track is above the threshold T up to that layer, or;

Condition 2: All occupied cell tracks' scores on the specific node are below the threshold T up to that layer.

These rules are also applied to the last layer in the best track retrieval phase.

The initial value of the threshold is an adjustable hyperparameter $T_initial$. The threshold value has to be discounted for every additional frame because the connection score has a decreasing trend from probability multiplication when routing to each new frame. The discount rate is calculated with the following equation.

$$DiscountThreshold = T_{initial}{}^{frame_number-1} \tag{4.3}$$

In Fig. 4.6 these concepts are illustrated.

Suppose the track of Cell 1 is settled in the red line in Fig. 4.6 and a Cell 2 was considered to be linked. When considering node connections from frame 3 to 2, the upper node in frame

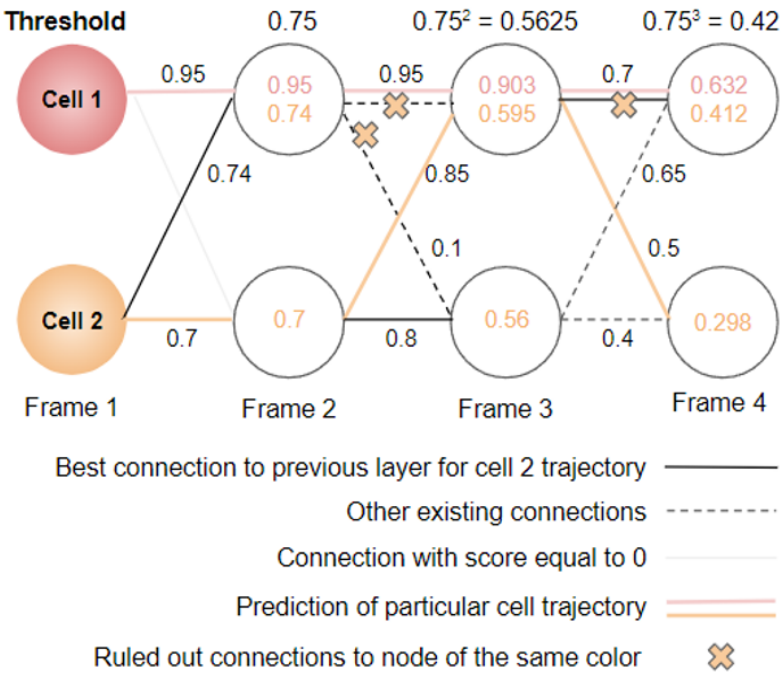

Fig. 4.6 Alternative path selection example.

2 is originally the best node to be connected for orange cell track, but it is occupied by the red cell track with a score higher than threshold. On the other hand, the routing track score of orange is 0.74 which is lower than threshold 0.75. Thus, connection to that node would be ruled out and a connection to the lower node in frame 2 (the second best connection) is established. This is illustrated by the solid orange line from frame 3 to 2 in Fig. 4.6. In the best track retrieval phase, which starts from frame 4, although the upper node has a higher overall score of 0.412 (0.412>0.298), it is lower than the discount threshold 0.42. It does not meet **Condition 1**. The score of 0.632 in the upper node of frame 4 in the red cell track is higher than 0.42, which is not meet **Condition 2**. Therefore, the lower node is chosen as the end node of the track (score 0.298) and a split event in frame 3 is detected for the two cell tracks.

Rule 3: a reroute strategy is constructed. During the process of routing, a circumstance occurs where the later routed track is a much better match compared to an occupied node in an earlier track. The reroute strategy allows an earlier routed cell track to be reset. To exemplify, let us consider the yellow Cell 3 is being routed after the orange cell, and the best track of the yellow cell is determined as shown in Fig. 4.7. It shares the second node with the orange cell track in frame 2 at the beginning. Nevertheless, only the yellow track score is above the threshold (0.9>0.75), which meets **Condition 1**. Therefore, the orange cell track is sent to reroute as it is occupying a node that is more suitable for the yellow track. While

rerouting the orange track, it is connected to node 3 in frame 2 since the first two nodes are more matched to the red and yellow tracks respectively. The orange and yellow tracks are sharing the same node in the next frame on the grounds that both of them are above the threshold of frame 3. In the last frame, they share the same node again because both of their scores are below the threshold of frame 4.

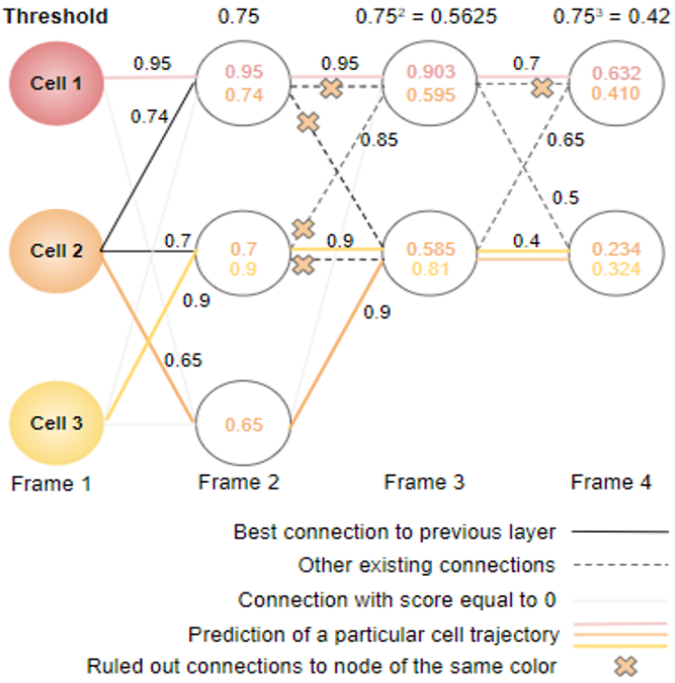

Fig. 4.7 The orange route is being rerouted due to a collision with the yellow track. The new route connects to a different node in frame 2.

Dealing with Cells Get into/out of View In Fig. 4.8, a situation is depicted where a new Cell 4 is being detected at frame 3, which will be routed largely similar to the rules mentioned except with a start frame discount. A start frame discount is a hyperparameter. Here it was set to 0.8 as an example. It is applied during score calculation because cells starting in later frames have fewer multiplications than track begins at an earlier frame. For instance, Cell 1 has two multiplications up to frame 4 (0.95×0.95×0.7). In order for Cell 4 to have a fair comparison, we set the hyperparameter to two multiplications (0.9×0.8×0.8) as well.

The cell trajectory terminates in three conditions. The first one is when it reaches the last frame of the time-lapse sequence. The second condition is when the score in the next frame is all zero. The third condition is when the next frame does not have an available cell due to nodes owned by other trajectories and the merge criteria is not fulfilled.

Additional Occupation Rules In the process of rerouting, a track that starts in a later frame (Cell 6) may occupy the rerouting path of a track beginning from an earlier frame (Cell

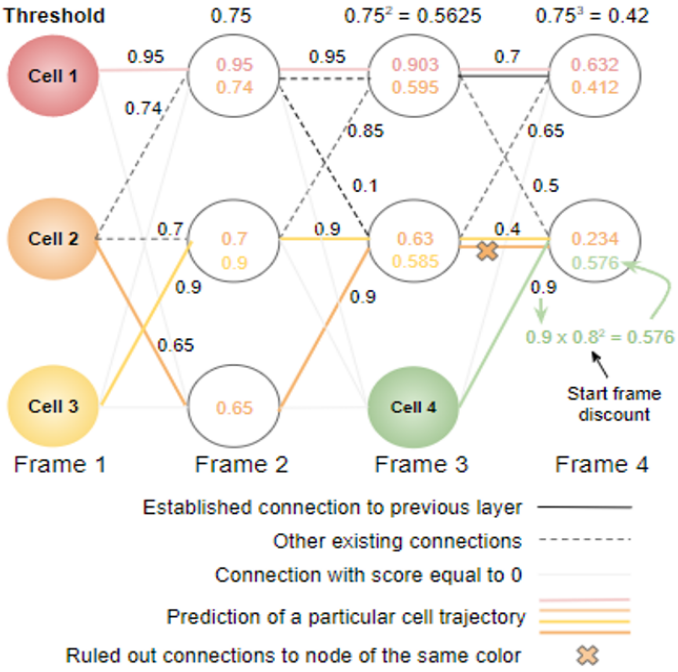

Fig. 4.8 A new cell (green) is detected in frame 3. It goes through a start frame discount for its connection, subsequently, takes over the lower node from yellow in frame 4.

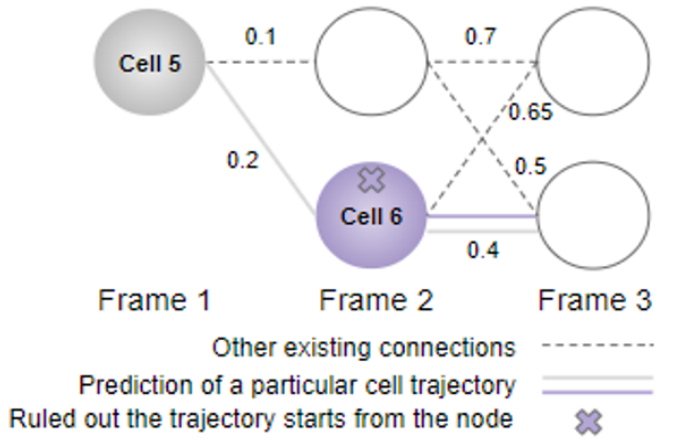

Fig. 4.9 Cell 5 established a connection to the beginning of Cell 6 during reroute. Cell 6 is then removed.

5) as shown in Fig. 4.9. In such case, the track that starts earlier would take over the later one even if it has a low connection score, the later track will be removed from track results. This is because the cell in the later trajectory no longer satisfies the definition of a new cell as it is owned by an existing trajectory. This situation only occurs in the reroute phase.

4.2.3.4 Evaluations for Linkage Trajectories

After implementing the extended Viterbi algorithm for each time-lapse sequence, all of the predicted cell trajectories of each sequence were saved, the same as the ground truth trajectories. Evaluation metrics were used to measure the performance of the linkage methods. Four measures [154] have been selected to evaluate how well a tracker identifies objects. These are:

Track purity (*TP*): a measure of the degree to which the predicted tracks ($\in$) follow ground truth objects (*GT*). It is the ratio of frames that $\in$ correctly identifies *GT* to the total number of frames $\in$ exists.

Object purity (*OP*): is a measure of the degree to which the ground truth objects (*GT*) are followed by predicted tracks ($\in$). It is the ratio of frames that *GT* is correctly identified by $\in$ to the total number of frames *GT* exists.

Falsely Identified Tracker (*FIT*): a measure of the degree to which the *GT* are tracked by the incorrect $\in$.

Falsely Identified Object (*FIO*): a measure of how often the $\in$ is tracking a different cell than the *GT* it was matched to.

In addition, the average length of tracks was computed.

Average length of tracks: the ratio of the total length of all predicted tracks to the number of predicted tracks.

4.3 Results and Discussion

We extracted results from each part of the pipeline: cell segmentation, cell motion tracking, and trajectory linkage, respectively.

4.3.1 Segmentation Results

240 ground truth segmentation images were divided into train-validation set (192 images) and test set (48 images) based on the ratio of 8:2. 48 images were used to test the segmentation model as the "unseen" data. The train-validation set was split to training set (153 images) and validation set (39 images) in a ratio of 8:2. Table 4.1 shows the performance of four segmentation models on 48 test images. The U-Net 1 was the structure that finally used in our study. It achieved 0.912 of *IoU* and 0.985 of *F1* score, which outperformed the other methods compared. U-Net 2 was the original U-Net model [134]. The input and output sizes of this model were 572×572 and 388×388, respectively. Thus, the output masks of this model should be resized to 512×512 as the original image size. During this up-sampling

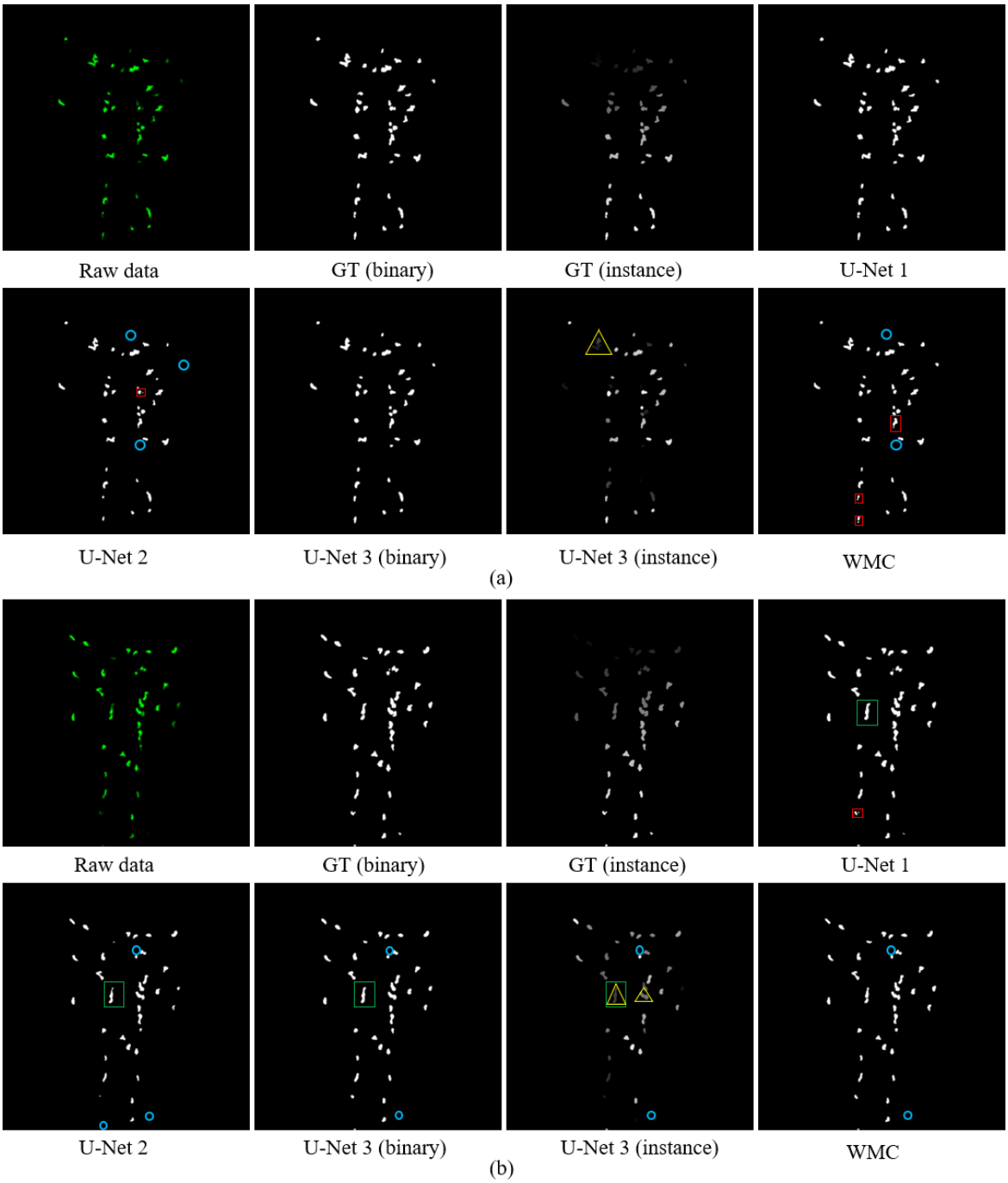

Fig. 4.10 (a) and (b) show two examples of segmentation results. It shows raw data, ground truth of binary mask and instance mask, respectively. The five segmentation results (U-Net 1, U-Net 2, U-Net 3 (binary), U-Net 3 (instance), Watershed) are shown. The red rectangular boxes represent the over-segmentation samples. The green rectangles represent the under-segmentation samples. The blue circles represent the *FN*. The yellow triangles represent the samples that are expected to separate by the instance segmentation model, however, not work so well yet.

resizing process, some mask information was not well preserved. This caused the predicted cells to have a smaller surface area compared with U-Net 1. It resulted in a lower *IoU* score of 0.697. Some small cells were missed as well which resulted in a high number of *FN*. U-Net 2 had less under-segmented cells and more over-segmented cells than U-Net 1 due to its smaller surface area. For U-Net 3 model, the binary masks were obtained first and a post-processing based on watershed transform was applied to obtain the instance masks. In the experiments, both the binary masks and the instance masks on test set were obtained and used to compute the evaluation metrics. The results in Table 4.1 show that, compared to U-Net 1, U-Net 3 achieved a lower *IoU* but a comparable *F1* score. U-Net 3 (binary mask) achieved a small number of under-segmented and over-segmented cells but with a little bit higher *FN* while U-Net 3 (instance mask) achieved higher values of *FN*, under-segmentation and over-segmentation. The experiments show that the complex instance-based segmentation model does not outperform much compared with a simple U-Net model. Moreover, it took 6-7 times more computation time than U-Net 1. We fine-tuned the parameters of the WMC segmentation algorithm for our neutrophil dataset, it yielded an acceptable accuracy of 0.725 of *IoU*, 0.976 of *F1* score, and 34 *FN* numbers. Besides, the WMC algorithm has only one under-segmented cell, demonstrating that it performs well in separating the touched cells. However, it has a relatively high number of 26 over-segmented cells compared with other models and has the lowest efficiency.

Table 4.1 The performance of five segmentation models on the test set.

Models	***IoU***	***F1* score**	***FN***	**Merge**	**Split**	**Time(s)**
U-Net 1	0.912	0.985	8	18	5	64.00
U-Net 2 [134]	0.697	0.956	53	6	7	313.47
U-Net 3 (binary) [125]	0.695	0.984	15	12	4	402.50
U-Net 3 (instance) [125]	0.682	0.985	16	28	12	402.50
Watershed [152]	0.725	0.976	34	1	26	702.58

Fig. 4.10 shows the two examples of the different segmentation results. U-Net 1 obtained the most closed segmentation results compared to the ground truth. The surface area of the predicted cells in U-Net 2 is much smaller which leads to more *FN* and over-segmented samples. In Fig. 4.10, this is indicated by blue circles and red rectangles. The WMC algorithm also resulted in more *FN* and over-segmented objects. This corresponds to the results as presented in Table 4.1. U-Net 3 (binary) was the most comparable model to U-Net 1, however, with a lower computational efficiency. U-Net 3 (instance) was expected to

separate the touched cells. This worked well in some cases. But sometimes it failed as shown with yellow triangles in Fig. 4.10.

All of these segmentation results have pros and cons. We selected the one which performed better using less computation time for the cell tracking task in the next step. Based on the analysis above, U-Net 1 was selected as the segmentation model in our pipeline.

4.3.2 Tracking Results

In order to predict the most possible location of each candidate cell on test set, three models including a pretrained model, a model from scratch and a fine-tuned model were trained. The predicted candidate cell was matched to both original image and segmentation mask to find if they match the corresponding region of interest (ROI) of the cell. If so, it was counted as a correct prediction; otherwise, a wrong prediction. In this process, the original image helps to provide the intensity value and segmentation mask provides the location of the cell. Both of them are necessary because sometimes there is more than one cell ROI predicted from the output of the model. In this case, we selected the ROI with the highest average intensity as the final localized cell. Subsequently, the total number of cells (predictions) on each of the ground truth trajectories in the test set was also counted. The error rate is calculated by erroneous predictions divided by the total number of predictions.

Table 4.2 The predicted errors of each tracking model on the test set.

Models	**The number of erroneous predictions**	**The number of predictions**	**Error rate**
Model from scratch	572	11641	0.049
Pretrained model	781	11641	0.067
Fine-tuned model	578	11641	0.050

Table 4.2 shows the error rate of the three models. In the total number of 11614 predictions, there were 572, 781 and 578 cells mis-predicted for model from scratch, pretrained model and fine-tuned model, respectively. From Table 4.2, the pretrained model produced the highest error rate of 0.067. This seems to be reasonable as this model was trained on an *E.coli* cells dataset, which is a cell type totally different from neutrophils. The model from scratch and the fine-tuned model obtained a comparable error rate whereby the model from scratch has the lowest one. In this study, we selected the outputs of fine-tuned model to do the next step of cell linkage rather than the model from scratch. Because, in the follow-up

experiments, we found that the tracking trajectories that were obtained on the basis of the model from scratch have many more linkage errors. The ideal situation is that there should be only one predicted cell in each output image due to the fact that our ground truth trajectories have no splitting annotations. However, the model trained from scratch predicts more than one cell in most output images. The extra predicted cells do not increase the error rate but it does affect the result of the Viterbi linkage process. Clearly, given our data, the fine-tuned model is much more robust than the model from scratch. Therefore, it is chosen for our study.

4.3.3 Linkage Results

For a 120-frame time-lapse sequence, a total number of 119 score matrices were obtained. The score was calculated based on the predictions of the tracking model and was defined as the overlapping area between the predicted cell on the current frame with each cell on the previous frame. Based on the score matrices, we performed our extended Viterbi algorithm, as well as several different linkage methods to compare the performance. The first one is the DeLTA linkage method [127]. DeLTA linkage is a straightforward method to link the trajectories. It starts from every cell in the first frame, to find the candidate cell in the next frame with the highest score value. If the values of two candidate cells are the same, it chooses to link the first candidate cell directly while the other cell can start a new trajectory. Each cell node can only be passed once by one trajectory. The second involved linkage method is the Hungarian algorithm [148]. The Hungarian algorithm aims to find the optimal path in the score matrices by solving an assignment problem. For the linkage process, it starts from each cell in the first frame to link the candidate cell frame by frame, until no candidate cell can be linked. The end condition depends on the configured threshold. In the experiment, the threshold was set to 0.001. Each frame was looped and the unpassed cell was treated as the start of a new trajectory. Comparing the aforementioned two linkage methods, the Viterbi algorithm is more powerful due to its concept of optimization of a global linkage. Viterbi optimizes the path by taking the relation of all frames into account, not only the adjacent frames. A basic Viterbi algorithm was also taken into our comparison in order to show the improvement of our extended Viterbi algorithm. In addition, a rule-based tracking algorithm named KDE with mean shift [126] was also involved, so as to demonstrate the outperformance of our pipeline incorporating a deep-learning-based tracking model and an extended Viterbi linkage method. Fig. 4.11 shows the performance comparison of the methods in our evaluation.

In Fig. 4.11, the box plots of the different linkage methods with five selected evaluation metrics on the test set are shown. The evaluation metrics *FIT*, *FIO*, *TP*, *OP* are normalized, in addition, average track length is computed. The normalized values represent the average

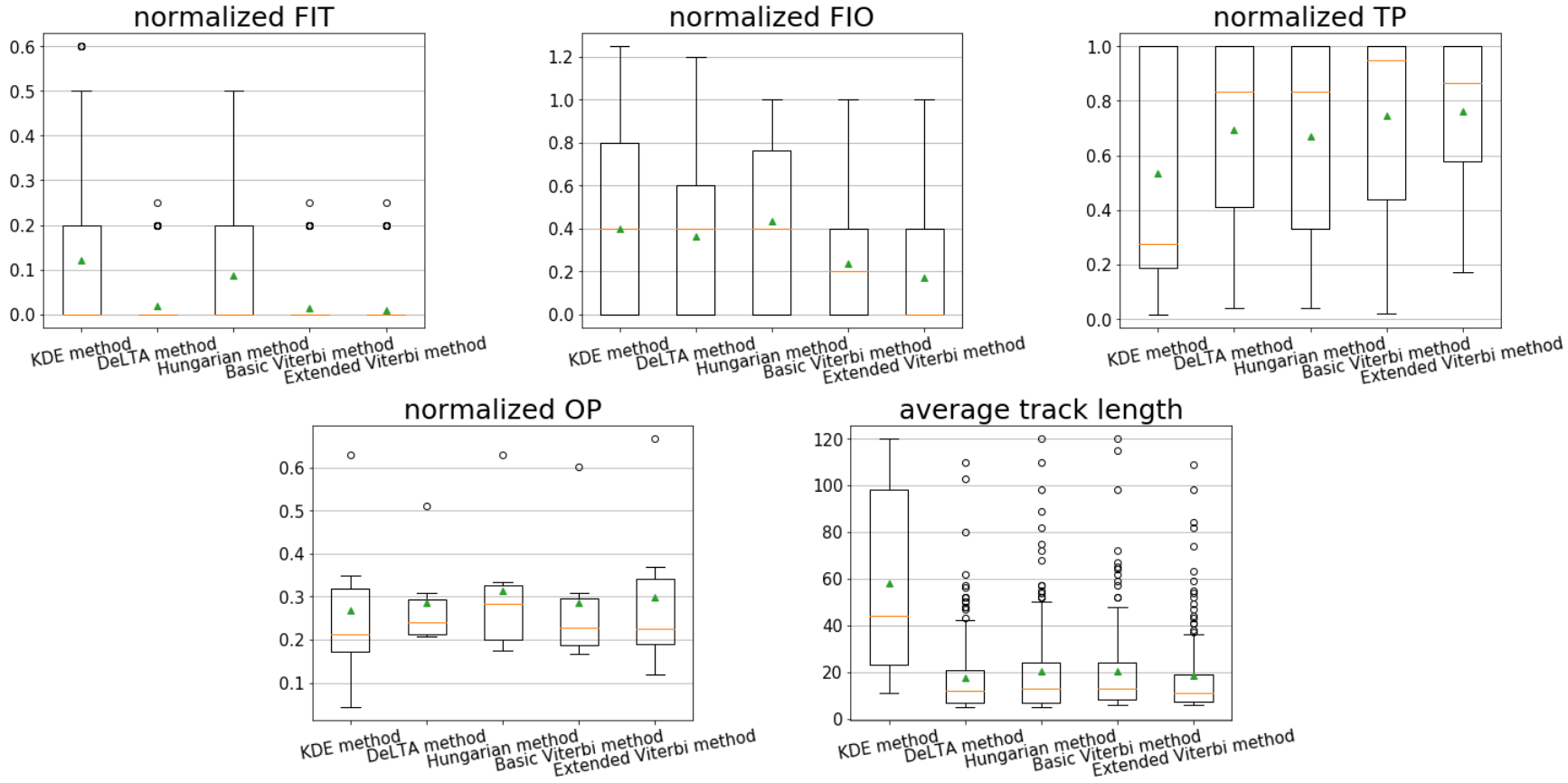

Fig. 4.11 The performance of different linkage methods with the selected evaluation metrics on the test set. The metrics are: falsely identified track (*FIT*), falsely identified object (*FIO*), track purity (*TP*), object purity (*OP*) and average predicted track length. In the boxplots, green triangle shows the mean values, the orange line is the median value, the box represents the first to third quartile of the data, the whiskers indicate the lower and upper extreme (1.5 times the interquartile range) and the black circles are the outliers.

number of errors (*FIT*, *FIO*) and purities (*TP*, *OP*) per frame per cell on the test set from the ground truth. In the box plot (cf. Fig. 4.11), the green triangle represents the mean value of each evaluation. The resulting values are presented in Table 4.3; these mean values correspond to the Fig. 4.11. From Table 4.3, our extended Viterbi achieved the lowest mean value of 0.010 of *FIT* and 0.173 of *FIO* of the methods in our comparison. For purity evaluations, the extended Viterbi also yielded the highest *TP* value of 0.763 as well as a second-best *OP* of 0.299. From the experiments, our extended Viterbi algorithm shows superior performance and produces a better cell tracking result.

Table 4.3 The performance of each linkage method. The footnote is the rank for all compared methods of each evaluation metric.

Algorithms	**Normalized *FIT***	**Normalized *FIO***	**Normalized *TP***	**Normalized *OP***	**Average track length**
KDE [126]	0.122	0.399	0.537	0.268	57.844
DeLTA [127]	0.020	0.365	0.694	0.285	17.520
Hungarian [148]	0.087	0.437	0.668	0.314	20.356
Basic Viterbi	0.015	0.237	0.747	0.286	20.158
Extended Viterbi	0.010[1]	0.173[1]	0.763[1]	0.299[2]	18.427[4]

The results in Fig. 4.11 show that compared with a rule-based KDE method, a deep learning-based tracking model together with any of the aforementioned linkage methods performs much better for our cell tracking task. KDE method obtained a highest *FIT* and a lowest *TP* and *OP*. It only outperformed 0.01 of *FIO* compared to the Hungarian algorithm. This is due to the features used in the KDE method; i.e. the kernel density of each cell. Only using kernel density is limited to distinguish the cell tracks on neutrophil dataset because of the complex morphology and migration behaviour of neutrophils. However, a deep learning model can learn more features of cell morphology and movement from the ground truth trajectories.

In Fig. 4.11, DeLTA, Hungarian, basic Viterbi and extended Viterbi methods show to achieve a comparable average track length. However, KDE algorithm creates a much longer average length than the other four methods. This is due to the fact that it does not solve the merge/split problems. Once the cells are merged, the trajectories of these cells are overlapping. It causes a longer track length of each cell trajectory which leads to a longer average track length. The original Viterbi algorithm exhibits the same problem. We applied

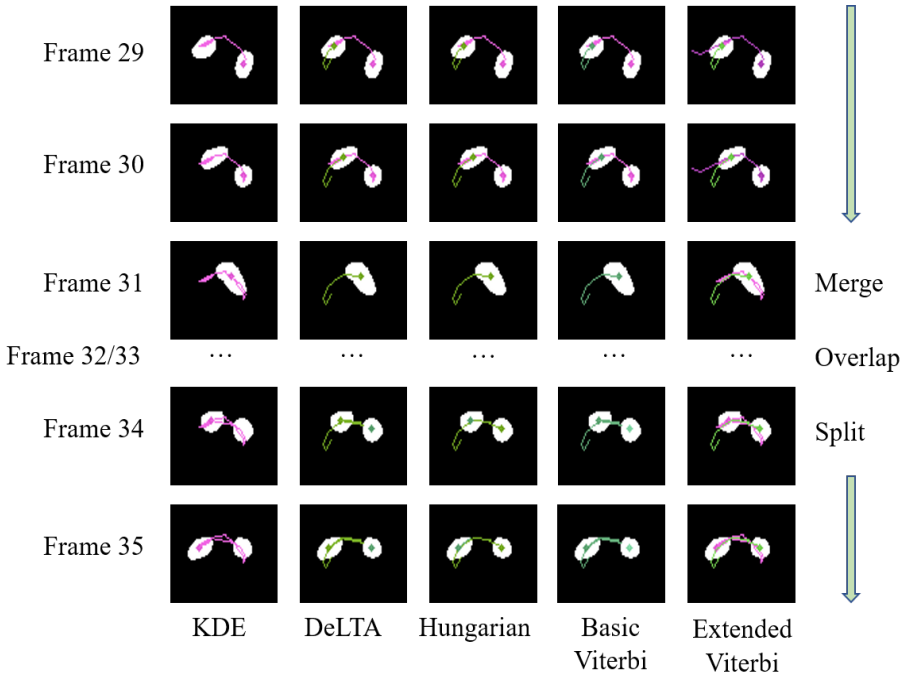

Fig. 4.12 A case of the comparison of how these methods solve the cell merge/split problem.

QR code 4.1 QR code 4.2 QR code 4.3

Fig. 4.13 Scan each QR code, different example time-lapse sequences of neutrophil tracking are given.

a post-processing for the basic Viterbi algorithm. The trajectory which has a lower score compared to the previous cell on the previous frame was truncated. The post-processing is helpful and makes the performance much higher. For our extended Viterbi algorithm, we designed different strategies to deal with the merge/split problems. The experiments show that the extended Viterbi algorithm further increases the performance.

A case of how the merge/split problem is solved by the five algorithms is depicted in Fig. 4.12. We present a series of consecutive frames between frames 29 to 35 of one particular time-lapse sequence. For the KDE algorithm, only the right cell, represented with a purple line, was tracked, the left cell was missed. For the DeLTA algorithm, the left cell, represented with a green line, was linked to the left one when the cells split in frame 34. This is not an expected direction. The right cell was treated as the start of a new cell trajectory. Same situation with the Basic Viterbi algorithm. For the Hungarian algorithm, the left cell, represented with a green line, was linked to the right cell after the split, which is an expected cell moving direction. For our extended Viterbi algorithm, the merge/split problem was solved. The left cell, represented with the green line is expected to move from left to right even after a collision, whereas the right cell, represented with the purple line, moves from right to left.

The trajectories of neutrophils are visualized subsequently and three examples are given in the following QR codes in Fig. 4.13. Scanning the QR codes and watching the time-lapse sequence.

4.3.4 Results on Other Datasets

In order to prove the robustness of our pipeline, two more experiments were conducted on datasets of two cell types. One type of data is downloaded from the cell tracking challenge [155]. It contains the stained nuclei of HL60 cells (Fluo_N2DH_SIM) and the HeLa cells stably expressing H2b-GFP (Fluo_N2DL_HeLa). The cells in this dataset move slowly and only express mitosis behaviour. The other type of data are MTLn3 plGFP cells [126][152]. This dataset consists of two groups. One is the experiment group, in which the cells move slowly without mitosis or any other behaviour. The other is the control group, in which the cells move faster but still without mitosis or any other behaviour.

Both of these cell data have simpler cell behaviour and slower movement compared to neutrophils. The length of the time-lapse sequence is much shorter than the 120-frame neutrophil sequences which facilitates the cell tracking task. The HL60 dataset has five time-lapse sequences in total with 65 frames per time-lapse sequence. The cell density is 10 to 40 per frame which is comparable to the neutrophil dataset. The MTLn3 dataset has two time-lapse sequences with 30 frames per sequence but with a very high cell density; about 80 to 100 cells per frame.

We retrained the U-Net 1 segmentation model and tracking model on the two new datasets, respectively. Subsequently, the same linkage methods were implemented. The performance results are shown in Fig. 4.14 and Fig. 4.15. Table 4.4 and Table 4.5 present the mean values (green triangles) corresponding to Fig. 4.14 and 4.15.

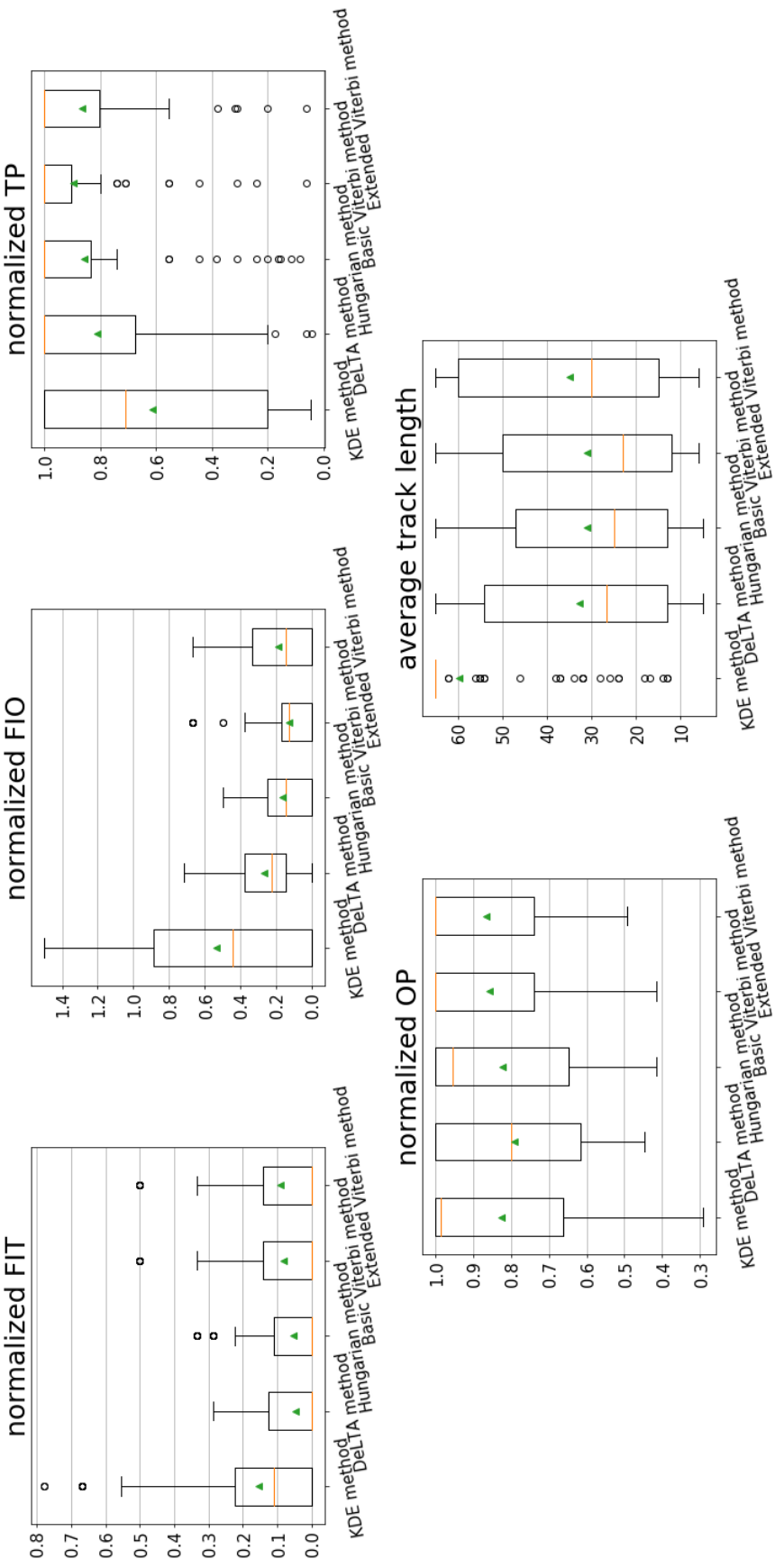

Fig. 4.14 The performance of different linkage methods with the selected evaluation metrics on the HL60/HeLa dataset. The metrics are: falsely identified track (*FIT*), falsely identified object (*FIO*), track purity (*TP*), object purity (*OP*) and average predicted track length. In the box plots, green triangle shows the mean values, the orange line is the median value, the box represents the first to third quartile of the data, the whiskers indicate the lower and upper extreme (1.5 times the interquartile range) and the black circles are the outliers.

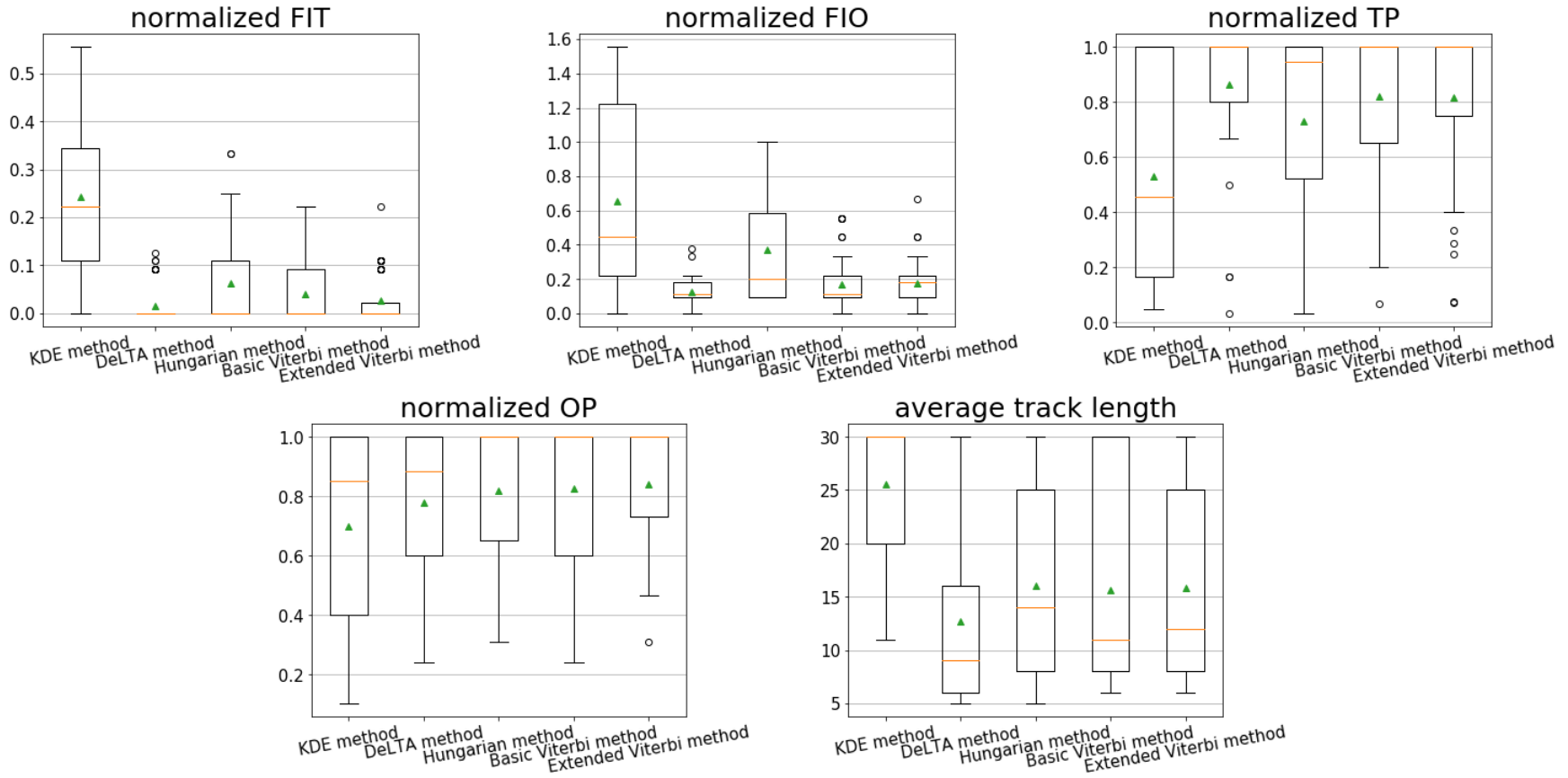

Fig. 4.15 The performance of different linkage methods with the selected evaluation metrics on the MTLn3 dataset. The metrics are: falsely identified track (*FIT*), falsely identified object (*FIO*), track purity (*TP*), object purity (*OP*) and average predicted track length. In the box plots, green triangle shows the mean values, the orange line is the median value, the box represents the first to third quartile of the data, the whiskers indicate the lower and upper extreme (1.5 times the interquartile range) and the black circles are the outliers.

Table 4.4 The performance of each tracking model on the HL60/HeLa dataset. The footnote is the rank for all compared methods of each evaluation metric.

Algorithms	Normalized *FIT*	Normalized *FIO*	Normalized *TP*	Normalized *OP*	Average track length
KDE [126]	0.154^{5}	0.537^{5}	0.612^{5}	0.825^{3}	59.801^{1}
DeLTA [127]	0.048^{1}	0.265^{4}	0.808^{4}	0.790^{5}	32.577^{3}
Hungarian [148]	0.053^{2}	0.165^{2}	0.857^{3}	0.823^{4}	30.867^{5}
Basic Viterbi	0.082^{3}	0.125^{1}	0.897^{1}	0.856^{2}	30.915^{4}
Extended Viterbi	0.091^{4}	0.186^{3}	0.864^{2}	0.866^{1}	34.866^{2}

Table 4.5 The performance of each tracking model on the MTLn3 dataset. The footnote is the rank for all compared methods of each evaluation metric.

Algorithms	Normalized *FIT*	Normalized *FIO*	Normalized *TP*	Normalized *OP*	Average track length
KDE [126]	0.243^{5}	0.658^{5}	0.532^{5}	0.700^{5}	25.509^{1}
DeLTA [127]	0.015^{1}	0.126^{1}	0.862^{1}	0.779^{4}	12.670^{5}
Hungarian [148]	0.062^{4}	0.373^{4}	0.732^{4}	0.819^{3}	16.010^{2}
Basic Viterbi	0.040^{3}	0.172^{2}	0.820^{2}	0.827^{2}	15.650^{4}
Extended Viterbi	0.028^{2}	0.175^{3}	0.818^{3}	0.841^{1}	15.796^{3}

For the HL60/HeLa dataset, the basic Viterbi algorithm realized the best score in *FIO*, *TP* and a second-best score of *OP* among these different methods. The extended Viterbi only achieved the best performance of *OP*. This experiment obviously demonstrated that our strategies work better on a cell dataset with more complex patterns such as neutrophils. For the cell dataset with simple patterns, the basic Viterbi is robust and outperforms the others. Compared with KDE method, deep learning methods still perform better than machine learning-based method which is consistent with the results on neutrophils.

For the MTLn3 dataset the performance of the two Viterbi-based algorithms is comparable with DeLTA and Hungarian methods. All of these four methods have their pros and cons, but still perform better than the KDE method. The extended Viterbi achieved the best score of *OP* whereas DeLTA obtained the best *FIT*, *FIO* and *TP*. The basic Viterbi yielded the second best *FIO*, *TP* and *OP*. The possible reasons that the Viterbi-based algorithms are not outperformed compared to the other two methods can be: (1) The experiment contains just two time-lapse sequences with 30 frames per sequence. This number of datasets is rather

limited for good training of a deep learning model. (2) Furthermore, the high cell density of the sequences easily causes linkage errors due to the global linkage way of the Viterbi algorithm. This is because the cells are very close to each other. The prediction would result in an overlap of two or more comparable adjacent cells causing the about same score in the matrix. It easily leads to an error trajectory in the global linkage process. Therefore, Viterbi-based algorithms are more robust on time-lapse sequences with a low or medium cell density instead of a very high density.

4.4 Conclusions

In this paper, we developed a pipeline incorporating cell segmentation, cell motion tracking between two frames, and trajectory linkage, to track the complex migration patterns of neutrophils in the time-lapse sequences. The pipeline integrates two U-Net models, to segment cells and learn the cell movement behaviour, respectively. The extended Viterbi algorithm can handle different patterns in cell migration such as appearing, disappearing, go in/out of the field of view, so as to more accurately link the cell trajectories. Compared with the existing state-of-art, our algorithm achieved superior performance and provides an effective approach to help understanding cell migration behaviour. In the next chapter, we will extend our work to 3D cell tracking using our neutrophil dataset. 3D data provide more spatial information in Z-axis and are expected to track the cell trajectories in a better way.