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

Li, C.

Chapter 5

A Feature Weighted Tracking Method for 3D Neutrophils in Time-lapse Microscopy

This chapter is based on the following publication:
C, Li., W, W.C. Yiu., W, Hu., L, Cao., F, J. Verbeek., A Feature Weighted Tracking Method for 3D Neutrophils in Time-lapse Microscopy. *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2022, pp. 2196-2202.

Abstract:

Neutrophils are one of the crucial immune cells. It plays a key role in the immune system defending the invasion of harmful particles, such as viruses and bacteria. The analysis of the dynamic process of neutrophil migration helps biologists to understand underlying mechanisms of neutrophils in response to wounding. However, accurate neutrophil tracking is still a challenging task in 3D space due to the complexity of cell morphology and behavior. In this study, we improved the quality of raw data by denoising and linear interpolation. A 3D U-Net was trained and used to detect the location of each cell in the time-lapse sequence. Subsequently, a feature weighted 3D tracking method was proposed. The experimental results show that our method performs well compared to the existing tracking algorithm. Our pipeline is also much more reproducible than other state-of-arts.

5.1 Introduction

The migration of leukocytes is a process of identifying the acute inflammation which could harm organisms. The function of leukocytes is to fight against the invading pathogens so as to protect organisms. Neutrophils are the first cells to rapidly respond to the site of inflammation and act as the first line of defense [121]. Research on the migration of neutrophils in living organism (*in vivo*) helps biologists to understand biological mechanisms better. For this kind of research, zebrafish larva is a popular model for studying cell functions due to their small size and transparency characteristics for screening and imaging [156]. In recent years, the microscopy techniques have greatly advanced in the field of biology and medicine. Multi-functional microscopy allows scientists to visualize the dynamic process of living cells in spatial and temporal resolution, both in brightfield and fluorescence modes [157][158]. Despite these advanced conditions, the accurate measurements of neutrophils' migration are still challenging, not only in 2D + T (x,y,t) space but also in 3D + T (x,y,z,t) space.

In general, cell tracking tasks are divided into two steps: cell segmentation to locate the position of each cell, and cell tracking to associate the trajectory based on cell similarities [124][127][159][160][161].

Segmentation algorithms have gone through a long developed process from threshold segmentation (e.g. Otsu [132]), watershed-based segmentation [152][141], to the deep learning segmentation methods (e.g. U-Net [134][162]). 3D U-Net is the popular segmentation model in the biomedical field. 3D U-Net segmentation model was first proposed in [163]. It was proved to achieve a good performance on kidney embryo segmentation. Although it is a deep learning model, it only requires small amount of annotated data for training. 3D U-Net reduces laborsome annotation work. However, unlike an individual kidney embryo, cells are often collided with each other. The watershed method was successfully used to help separate the touched cardiac cells in zebrafish as a post-processing [164].

Once the segmentation is performed, the next step is to associate the cell tracks frame by frame based on the similarities. Rule-based methods are studied the most in recent years[165][166]. A graph-based tracking algorithm was proposed in [165] which utilizes relative cell location information to associate the cell tracks. It achieved comparable performance to state-of-art methods in Cell Tracking Challenge 2019 and 2020. Rule-based methods are easier to tailor the features for specific datasets. Deep learning tracking methods extract features automatically. [164] proposed a deep learning-based software pipeline named 3DeeCellTracker for tracking. The datasets they used are cardiac cells in zebrafish and neurons in worm' brain. The characteristics of these datasets are that the cells move in the same pattern which can be simulated using random affine transformations. A simulated dataset was created by random affine transformations and used to train the deep learning

model. For neutrophils, the cell movement is much more complex and irregular. It is difficult to find a pattern in it. In [157], a 3D cell association learning network was designed and used for neutrophils tracking problem. It achieved a significant performance and solved the cell collision problem well. However, the deep learning models require a huge number of annotated data for training. The ground truth data were annotated by labeling each cell position and associating the cells over time manually. It is time-consuming and for other researchers, it is difficult to reproduce the process.

In our study, a 3D U-Net segmentation model and a rule-based tracking method were proposed that tailored to our captured neutrophil dataset. At first, the image pre-processing, containing enhancement, denoising, and smoothing, were used to improve the image quality. In this dataset, the size of 3D raw data is (512, 512, 8). Eight layers were scanned on Z-axis which makes the 3D data too thin to observe the movement on Z-axis. We did the linear interpolation to enlarge the dimension to (512, 512, 29). Secondly, a 3D U-Net segmentation model was trained and a watershed algorithm was applied to locate the positions of cells. Thirdly, we designed the features, incorporating cell distance, cell direction, and average cell movement. The final similarity scores calculated by weighting those features were used to improve the cell tracking. Subsequently, the Hungarian algorithm was used to associate the cell tracks at the end[1]. The results show that our feature-based method outperformed the graph-based method in [165]. In addition, Hungarian linkage performed better than a straightforward linkage method in [127].

5.2 Data Collection and Pre-processing

5.2.1 Data Capturing

The time-lapse dataset of neutrophil was captured by the experts. All the experiments were done with zebrafish and followed the international guidelines specified by the EU Animal Protection Directive 2010/63/EU. All the zebrafish were 3 days post fertilization (dpf) and the tail wounding was conducted with the protocol provided in [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 200 μm from the wounding edge toward the body trunk, shown in Fig. 5.1, were counted as recruited cells. Under the microscopy, the zebrafish tails were scanned from top to bottom. An 8-layer 3D stack at each time point was captured. The size of each stacked image is 512 × 512 × 8. The layer interval is 5∼6μm, along with the thickness of tail is 35∼42μm. For

[1] https://surfdrive.surf.nl/files/index.php/s/IQ1wEMn4lA7bB7x/download

each sample, a 2-hour time-lapse sequence was captured, the time interval is one minute, along with the 120 frames for each sequence. Then we obtained the time-lapse 3D data with the size of (512, 512, 8, 120), which corresponded to the axis (x, y, z, t). Each time-lapse sequence contains almost 10 to 40 cells. We captured 9 time-lapse sequences in total.

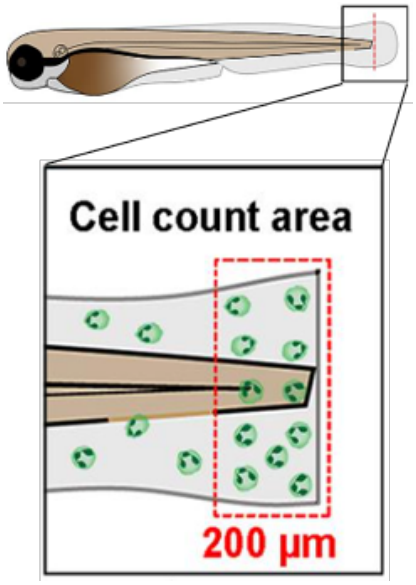

Fig. 5.1 The area of neutrophils recruited. The red line is the wounding edge in zebrafish tail. The red dashed box shows the area where neutrophils were counted as recruited neutrophils.

5.2.2 Data Pre-processing and Linear Interpolation

During the procedure of data capturing, there could be noises on the background that reduce the quality of images. Insufficient image brightness makes the cells in a lower intensity and not clearly visible as well. Therefore, several methods were applied to improve image quality. At first, we enhanced the image contrast to highlight the cell. However, noises at the background were enhanced as well. Thus, a median filter was used to denoise. Subsequently, we chose a Gaussian blur to smooth the cell surface. Fig. 5.2 shows the procedure of image pre-processing.

The 8 layers on Z-axis result in a very thin 3D bounding box. It is very difficult to observe the cell movement along the Z-axis. Due to the experimental design, the expert had to scan 3 channels (one brightfield channel, and two fluorescence channels) within one minute of the time interval. There was not enough time to scan more layers on the Z-axis. In order to observe the Z-axis movement of cells more clearly, we used linear interpolation twice [167] to increase the layers on the Z-axis from 8 to 15 for the first time, then to 29 for the second

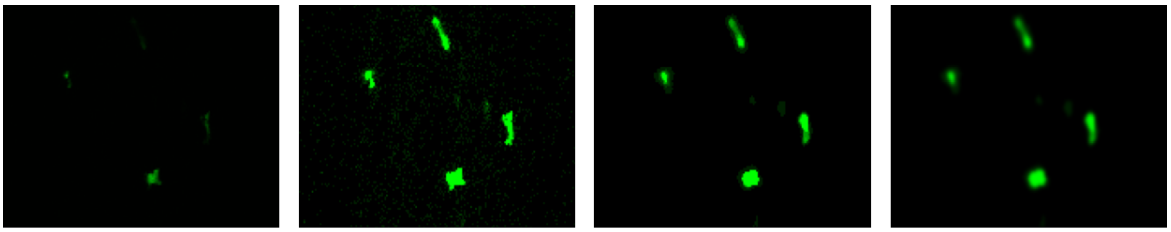

Fig. 5.2 Image preprocessing. (a) A region of raw data. (b) Image contrast enhancement. (c) Image denoising with median filter. (d) Image smooth with Gaussian blur.

time. Each interpolated layer was calculated as the average intensity of each two adjacent layers. We did not interpolate more times because the more layers the data have, the bigger the memory size, which is a burden for computing. We have kept a balance between data quality and data size.

5.3 3D U-Net Segmentation

The segmentation is a step to locate the position of each cell before tracking procedure. The 3D U-Net structure [164] was used in our study. Like the standard U-Net, it has a contracting path and an expansive path, which is a symmetric U-shaped structure. 3D U-Net is powerful for training a segmentation model with very little annotated images [163]. To prepare the annotated data, an open-source tool named Segmentor [168] was used for its efficiency and user-friendly interface. We annotated two 3D images with size of $512\times512\times8$, one for training and the other one for validation of the model. The same two interpolated 3D images with size of $512\times512\times29$ was annotated as well. We trained two segmentation models for different size images on a dedicated server equipped with two NVidia GeForce GTX 2070 with 8 GB GPUs using Linux Ubuntu operating system. In order to reduce the memory usage during training, we divided the large images ($512\times512\times8$, $512\times512\times29$) into small ones ($160\times160\times8$, $160\times160\times16$) as the input to train the models and then combine the sub-images together to form a whole images [164]. The dividing process help to enlarge the number of training images. Data augmentation was used to increase the training data as well. The output of the segmentation model is the probability of whether each voxel belongs to the cell region or not. The watershed segmentation in [164] was applied to separate each individual cell. It was used twice in the x-y plane and z-dimension, respectively, due to the data have differed resolution in X-Y axis and Z-axis.

5.4 Feature Weighted Tracking

Feature weighted algorithm contains several representative features generated from the properties of cell movements. They are: cell distance, cell direction, average cell movement distance. The similarity scores of each pair of cells on every two adjacent frames were calculated by weighted of those features. Based on the similarity scores, the Hungarian algorithm [148] was used to associate the related cells frame by frame.

5.4.1 Cell Distance

Cell distance D is defined as the distance from the center point $(x_{i,t}, y_{i,t}, z_{i,t})$ of a cell i on the frame t to the center point $(x_{j,t+1}, y_{j,t+1}, z_{j,t+1})$ of a candidate cell j on frame $t+1$. The distance is calculated based on Euclidean distance in (5.1).

$$D = \sqrt{(x_{i,t} - x_{j,t+1})^2 + (y_{i,t} - y_{j,t+1})^2 + (z_{i,t} - z_{j,t+1})^2} \tag{5.1}$$

The distance score was normalized by (5.2). A threshold T was given. It represents the maximum distance that the cell could move between two frames. If the distance is larger than T, the distance score is 0.

$$Distance\ Score = \max(0, (T - D)/T) \tag{5.2}$$

5.4.2 Cell Movement Direction

The direction of cell movement is an important factor that affects its next destination. Fig. 5.3 is a ground truth sample which shows a clue that the cell movement tends to stick to similar direction in a collision-split event.

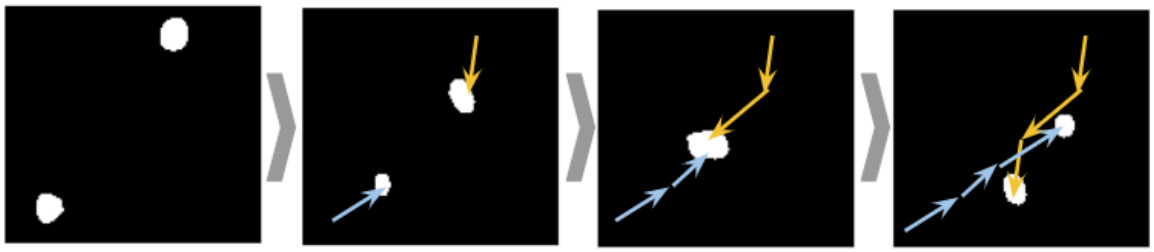

Fig. 5.3 Two cells that are moving from different directions, collide and then split according to their trend. Only one layer in Z-axis is shown.

The directional variation is more reliable when several prior frames are referenced. It can produce a sufficient direction variation of the cell movement. The number of the former

frames to be included is a hyperparameter which can be tuned. To calculate the direction variation, we first have to calculate the directional vector. Suppose there is a cell A on frame t, Fig. 5.4 displays how the directional vector is calculated by five frames before frame t in a 3D space with the coordinate of the Z-axis is zero.

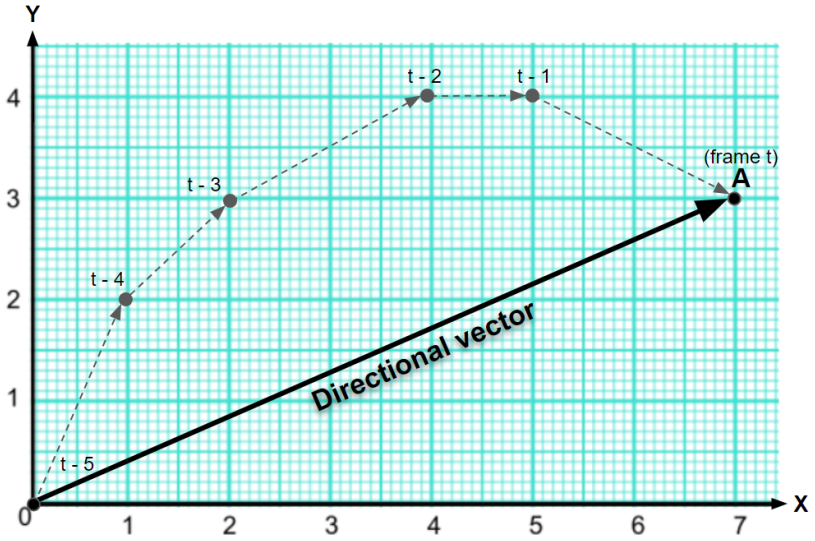

Fig. 5.4 Directional vector calculation from the last five frames of a cell track.

The start location of the cell on frame $t-5$ will be set to (0,0,0) in 3D space. The relative movement for each frame after is calculated and added. In this case, the relative coordinates of the previous five frames are (1,2,0)→(1,1,0)→(2,1,0)→(1,0,0)→(2,-1,0). The result of the directional vector is (7,3,0) computed by adding all the five relative vectors.

Next, the vectors from cell A to the surrounding candidate cells on frame $t+1$ were calculated. The angle difference between the directional vector a and each candidate vector b is calculated using the dot product equation in (5.3). An inverse cosine is applied on both sides to derive the degree α in (5.4).

$$a \cdot b = |a| \times |b| \times \cos\alpha \tag{5.3}$$

$$\alpha = \arccos\left(a \cdot b \;/\; (|a| \times |b|)\right) \tag{5.4}$$

The candidate cell that has a smaller degree to the directional vector will receive a higher score. The score will range between 0 (180°) to 1 (0°) in (5.5).

$$\textit{Degree Score} = (\cos\alpha + 1) \times 0.5 \tag{5.5}$$

5.4.3 Average Cell Movement Distance

Average cell movement distance is also an important factor to distinguish inactive and active cells. Cells that are inactive tend to linger around the proximate region, while cells that are more active tend to travel at a similar distance over each frame. Fig. 5.5 is a cell movement track extracted from a ground truth example which illustrates this scenario.

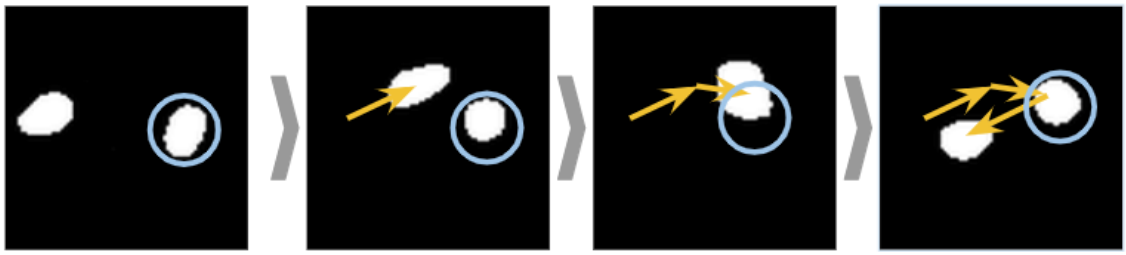

Fig. 5.5 Two cell tracks with different movement rates collide in frame 3 and splitted right after. The cell on the right remained inactive throughout the frames.

The cell track labeled with yellow arrows has a distant moving distance, whereas the cell track circled with light blue lingered in the nearby region. It can be seen that both cells maintained a similar movement distance over the 4 frames. The average movement is derived from (5.6). D_i is the distance of each pair of cells. n is the frame in which the cell moves.

$$Average\ Movement = \frac{1}{n}\sum_{i=1}^{n} D_i \tag{5.6}$$

The score is normalized to unity with (5.7). T is the same setting in (5.2).

$$Average\ Movement\ Score = Average\ Movement/T \tag{5.7}$$

5.4.4 Similarity Scores

The similarity scores are defined as the sum of all weighted feature scores in (5.8). The *feature_scores* is [*Distance Score*, *Degree Score*, *Average Movement Score*]. The influence of each feature can be adjusted through a 3 by 1 weight matrix. The weight matrix consists of 3 positive rational numbers, e.g., weight = [0.4, 0.3, 0.3].

$$Similarity\ Score = \sum_{i=1}^{n} weight(i) \times feature\_scores(i) \tag{5.8}$$

5.4.5 Hungarian Association

We derived all 119 of the similarity score matrices between each adjacent frame over the 120-frame time-lapse sequence. The scores in matrices were used as input for the Hungarian algorithm for constructing the cell associations. Hungarian is one of the linear programming algorithms that can be used to solve a profit maximization assignment problem [148] [169]. Fig. 5.6 (a) shows one example of similarity matrix between frame t and frame $t+1$. The number of rows and columns represents the cell numbers on frame t and frame $t+1$, respectively. The Hungarian algorithm requires a square matrix and returns an optimal path over the matrix. Hence, a dummy row with zero values was added as shown in Fig. 5.6 (b). If the number of rows is more than the number of columns, a dummy column is added.

The values in the matrix represent the similarity between all combinations of cell candidates on two adjacent frames. Therefore, we should maximize the sum of profits of all candidates using the Hungarian algorithm. In Fig. 5.6 (c), the nodes on the best path were underlined. The maximum sum of profit in the matrix is 0.9+0.7+0.8+0.7+0=3.1. The index pairs of each optimal position were returned as a list, such as [(0,0), (1,2), (2,1), (3,3), (4,4)]. Similarly, a new index list was obtained between frame $t+1$ and frame $t+2$. The cell tracks were formed by linking matched cell index frame by frame.

In addition, the candidate cells on frame $t+1$ that are associated by a dummy cell node on frame t are regarded as newly appeared cells. The candidates on frame t that are associated with dummy candidates on frame $t+1$ are regarded as disappeared cells.

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 + 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
4	0	0	0	0	0

(b)

Frame t + 1

Frame t	0	1	2	3	4
0	<u>0.9</u>	0	0	0	0
1	0	0.1	<u>0.7</u>	0	0
2	0	<u>0.8</u>	0.2	0	0
3	0	0	0	<u>0.7</u>	0.1
4	0	0	0	0	<u>0</u>

(c)

Fig. 5.6 The process of the Hungarian algorithm. (a) Raw matrix. (b) A dummy row was added. (c) The optimal path between two frames was found.

In this way, 119 index lists were obtained from these matrices. The final cell trajectories were associated based on the index lists.

5.5 Experimental Results

In order to evaluate the performance of our proposed method, the annotated ground truth and evaluation criteria are required.

5.5.1 Ground Truth Annotation

It is challenging and time-consuming to annotate cell trajectories in 3D + T space. In this study, the annotation was done in a combination way. At first, the 3D data were projected in 2D space. In a 2D time-lapse sequence, we recorded the cell center coordinates of the tracks over the cell movement. This procedure was done using 'Manual Tracking' plugin in ImageJ [69][170]. The next step is to map the 2D cell center coordinates on each track to the 3D + T sequence. Then the center point on the Z-axis of each cell was located and added manually. We annotated 20 ground truth trajectories over the raw dataset with different sizes of (512, 512, 8, 120) and (512, 512, 29, 120), respectively.

5.5.2 Evaluation Criteria

Four measures to evaluate how well a tracker identified objects, proposed by [154], were selected. They are:

Falsely Identified Tracker (*FIT*): a measure of the degree to which the ground truth objects (*GT*) are tracked by the incorrectly predicted tracks (ε).

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

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

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

The average length of tracks was computed as well.

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

5.5.3 Results

We did the experiments on both datasets with different sizes: (512, 512, 8, 120) and (512, 512, 29, 120). A graph-based tracking method proposed in [165] was compared. In [165], a

feature vector, consisting of cell volume, the total number of cell neighbors, and the average distance from all other cells, were obtained. Subsequently, the similarity between each pair of cells was calculated based on the feature vector.

In our algorithm, the parameters were configured as follows. The threshold T in (5.2)(5.7) was set to 35. It means if the distance between two cells is more than 35 pixels, they would not be considered the same cell. Thus, the *Distance Score* equals to 0. It was decided by observing the possible maximum movement distance. Once *Distance Score* equals 0, the other two scores were set to 0 as well. To compute cell movement direction, the number of prior frames was set to 6. To compute the average cell movement distance, the number of consecutive frames was set to 6. To calculate the similarity scores, the weight was set to [*Distance Score*, *Degree Score*, *Average Movement Score*]=[0.4, 0.3, 0.3]. All parameters were selected by tuning manually.

Based on the features we extracted, a Hungarian algorithm was applied to associate the cells frame by frame. In addition, a straightforward associated method in [127] was compared. The cell pairs with a high score in the matrix were linked, and the relation over frames was not taken into account. If there is a score in conflict between cells (e.g. two same scores in a row or a column), this method only chooses the first cell as the candidate cell rather than an optimal choice.

The performance comparison on the raw dataset with size of (512, 512, 8, 120) is shown in Fig. 5.7. In order to quantify the values clearly, Table 5.1 shows the mean values which correspond to Fig. 5.7. From Table 5.1, our algorithm reaches 0.145 of *FIO*, 0.845 of *TP*, 0.386 of *OP* and 20.238 of Average Track Length, respectively. It performs better than [165]. Only *FIT* is lower. Compared to [127], our algorithm also increased the performance by 0.050, 0.056, 0.039, and 3.654 of *FIO*, *TP*, *OP*, and Average Track Length, respectively.

Table 5.1 The Mean Value of Each Evaluation for Two Methods Comparison on the raw dataset with size of (512, 512, 8, 120).

Algorithms	***FIT***	***FIO***	***TP***	***OP***	**Average Track Length**	**Time** (s)
Algorithm in ref [165]	0.015	0.273	0.798	0.344	19.620	614.49
Algorithm in ref [127]	0.025	0.195	0.789	0.347	16.584	389.39
Our algorithm	0.045	0.145	0.845	0.386	20.238	358.96

Fig. 5.8 shows the performance of the three methods on the interpolated dataset. From the corresponded Table 5.2, we derived that both the false rates of our method are lower

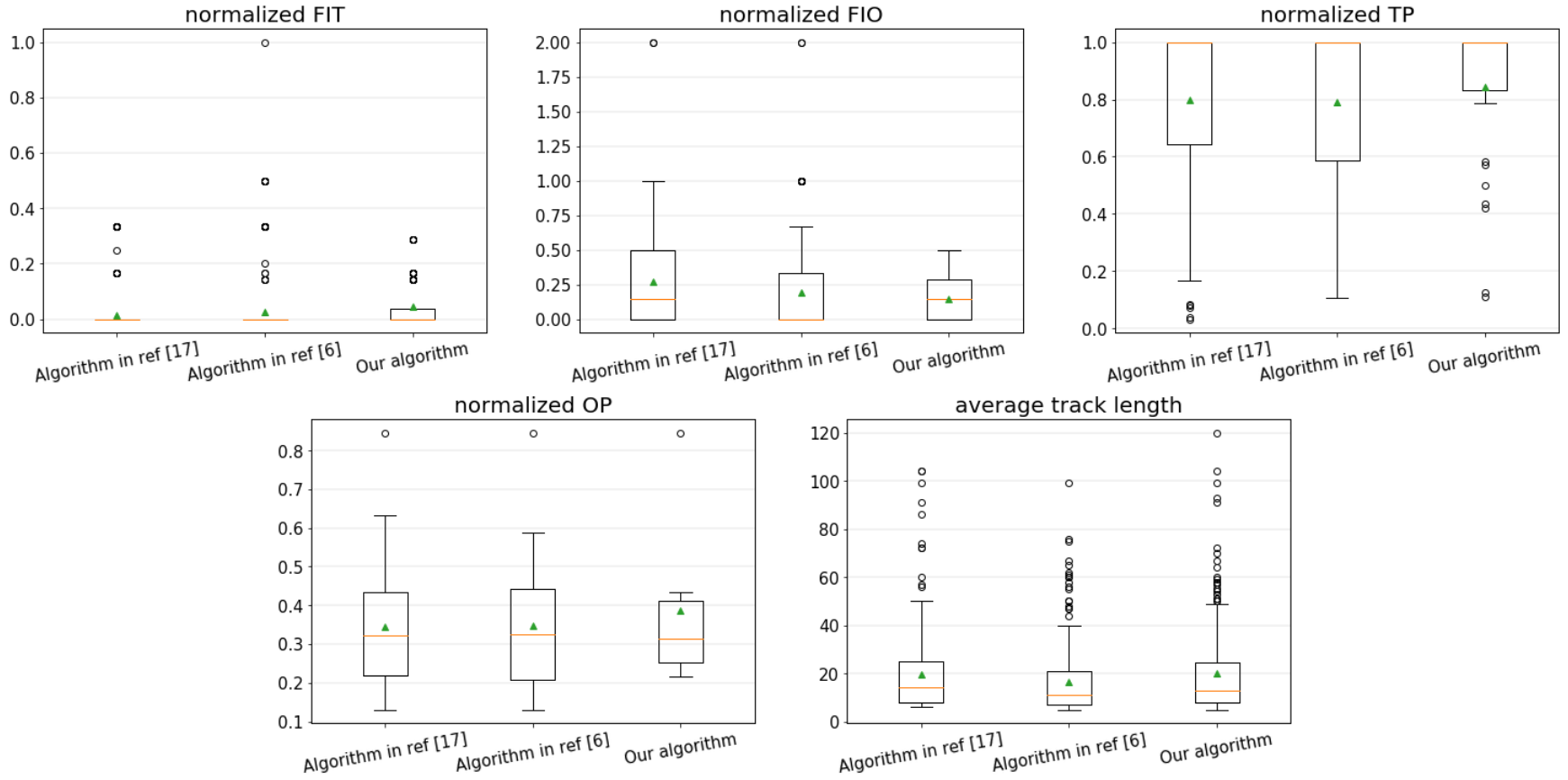

Fig. 5.7 The performance comparison on raw dataset with size of (512, 512, 8, 120). In the boxplots, green triangle shows the mean values, the orange line is the median value, the box represents the first to third quartile of the data, the whiskers show the lower and upper extreme (1.5 times the interquartile range) and the black circles are the outliers.

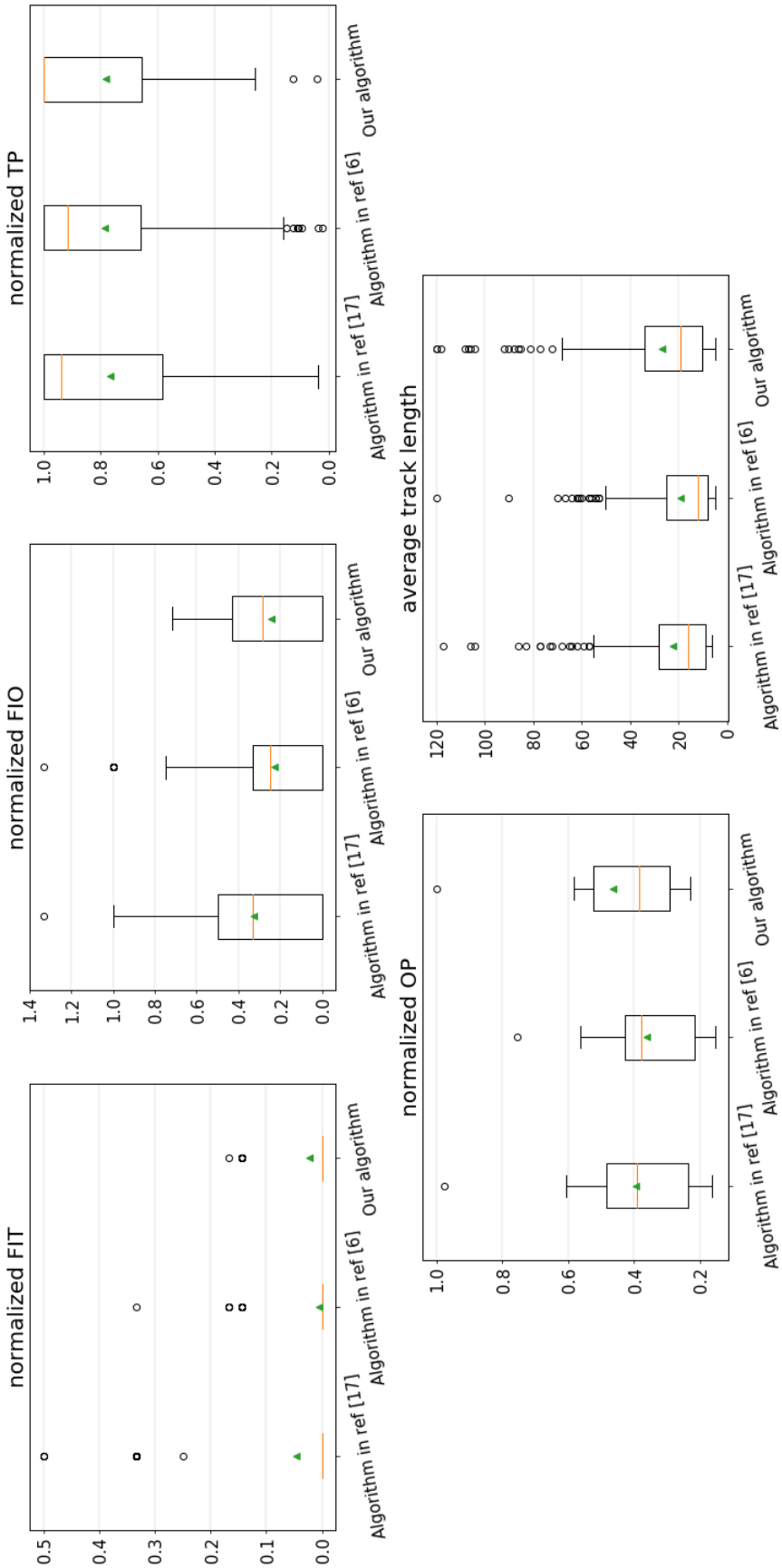

Fig. 5.8 The performance comparison on raw dataset with size of (512, 512, 29, 120). In the boxplots, green triangle shows the mean values, the orange line is the median value, the box represents the first to third quartile of the data, the whiskers show the lower and upper extreme (1.5 times the interquartile range) and the black circles are the outliers.

than [165]. *FIT* and *FIO* were decreased to 0.023 and 0.241, respectively. Furthermore, both purity rates (*TP* and *OP*) are higher than [165], along with 0.013, 0.071 were improved. Although [127] achieved the lowest false rate of 0.005 for *FIT* and 0.228 for *FIO*, our algorithm yielded a higher performance with *TP*, *OP* and average track length.

Table 5.2 The Mean Value of Each Evaluation for Two Methods Comparison on the interpolated dataset with size of (512, 512, 29, 120).

Algorithms	***FIT***	***FIO***	***TP***	***OP***	**Average Track Length**	**Time** (s)
Algorithm in ref [165]	0.046	0.326	0.767	0.393	22.490	2531.67
Algorithm in ref [127]	0.005	0.228	0.783	0.360	19.131	642.85
Our algorithm	0.023	0.241	0.780	0.464	26.633	619.64

The Average Track Length of the interpolated dataset is not only 4.143 longer than [165] but is also 6.395 longer than the result derived from the raw dataset. The reason is that the interpolation improves the representation of each 3D cell. Therefore, fewer cells were missed by segmentation procedure.

The computing costs of the two datasets were shown in Table 5.1 and 5.2. Our algorithm has lower computing time compared with a graph-based method in [165]. With the increase in the data size, graph-based method is much more computationally expensive while our feature-based algorithm works more efficient. The Hungarian method has a comparable computing costs compared with the simple linkage strategy in [127].

The experiments show that our method improved the performance of tracking on our neutrophil datasets compared to the available tracking algorithm in [165] and the simplistic associated method in [127].

5.6 Conclusions

This paper aims to develop a method to track neutrophils in 3D + T space accurately, so as to help biology experts to understand its movement behavior better. We started from data capturing and improved the low-quality raw data using image enhancement, denoising, and smoothing methods. A linear interpolation method was applied to increase the dimension along Z-axis to increase the sampling resolution. 3D U-Net segmentation models were trained and, together with a watershed algorithm, the cells' locations were detected in 3D space. A novel feature weighted tracking method was developed which is tailored to the

specific characteristics of our neutrophil dataset. The cell distance, cell movement direction, and average cell movement distance were three features selected according to the analysis of neutrophils' movement behavior. Based on these features, weighted similarity score matrices were generated and the Hungarian algorithm was used to associate cells frame by frame. The experimental results proved that our method outperformed the available state-of-art algorithm. The rule-based tracking method was selected because deep learning models require large ground truth data to fit in. Annotation in 3D + T space is still a challenge and there is not a user-friendly way to help labelling yet. Deep learning tracking models were proved to outperform the rule-based methods in state-of-arts [157][164]. However, it is difficult for other researchers to reproduce the algorithm if their work is not open-source. At present, the rule-based tracking methods are much more straightforward to follow. In the future, we would focus on improving annotation methods in 3D + T space as well as implementing tracking tasks with deep learning models.

Acknowledgment

This work was supported by the Chinese Scholarship Council through Leiden University.