Imperial College London
Department of Computing

Learning to Automatically Detect and Track Cells in Microscopic Imaging

by

Pedro Damian Kostelec

September 2014

Supervised by Ben Glocker

Submitted in part fulfilment of the requirements for the
MSc degree in Computer Science (Artificial Intelligence) of Imperial College London

Contents

6 Experimental results IN PROGRESS

In this chapter we quantitatively and qualitatively analyse the performance of the automatic cell detector and tracker. Although some evaluation of the performance of the detection method is performed by the original authors in [4] it is useful to see how the method performs on the studied datasets in order to understand how much of the tracking accuracy is lost due to cells missed by the detection module. First, in section 6.1 we evaluate the performance and computation time of the cell detector and in section 6.2 those of the cell tracker. Finally, in section 6.3, we explore the limitations of the methods and in section 6.4 summarize the results.

See the cell population tracking and linear construction with spationtemporal ocntet by Kang et al for a good results section

6.1 Cell detector DRAFT I

In this section we evaluate the performance of the automatic cell detection module. First, we introduce the performance metrics used to evaluate the accuracy of the cell detector. Then we present detection accuracy results. To evaluate the accuracy and generalizability of the detection module we perform two sets of experiments. First, we train the cell detector on a number of frames from each individual dataset, and measure the accuracy on the same dataset. Second, we train the detector on combinations of datasets in order to judge the performance degradation due to the learning on the more types of cells. Because of the varying size of the cells in the datasets, and the varying brightness of the cells, we expect that such a trained detector will perform poorer than when trained and tested on individual datasets, sometimes mistakenly detecting small artefacts in the background as cells. Finally, we compute the average detection time per frame for each dataset.

The aim of this research was to develop an automatic cell detection and tracking pipeline that would require as little manual work as possible. This implies that a balance between accuracy and amount of manual work had to be established. There is also an direct relationship between accuracy and computation time. In order to reduce the amount of manual work we aimed to configure the cell detection module such that it would perform well on all the tested datasets without any manual adjustments of parameters. The consequences of this decision are twofold:

1. The features computed on the candidate cell regions are the same for all datasets and have been presented in section 3.5. Although some datasets could be analysed faster or more accurately with a different subset of features, using the same features for all dataset eliminates the complicated feature selection process for the user and makes the system generalizable to a

large number of different cell types.

2. The parameters of the MSER detector should be adequately set to perform well on all datasets. This means that the MSER detector should be able to detect cells of varying size and contrast in the different datasets. The consequence of this limitation for datasets with large cells and some background noise is that a potentially much larger number of candidate regions will be detected than necessary. Since each candidate region has to be evaluated this results in an increased computation time.

We were able to identify features that compute in an acceptable time for all these datasets (see section 3.5). However, it should be noted that in the case of testing the detector on very large datasets with thousands of frames, some adjustments of the parameters could result in a significant reduction in computation time and increased accuracy.

6.1.1 Performance metrics DRAFT I

We measure the performance of the cell detector in terms of precision and recall. The metrics are defined in terms of:

True Positive instances (TP) are candidate cell regions that are manually annotated as cells and the detector successfully classified as cells.

False Positive instances (FP) are candidate cell regions that are not manually annotated as cells, but the detector incorrectly classified them as cells.

False Negative instances (FN) are candidate cell regions that are manually annotated as cells, but the detector incorrectly classified.

We then define precision as:

$$\mathrm{PRE} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}},$$

and recall (also known as sensitivity) as:

$$\mathrm{REC} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} .$$

6.1.2 Detection accuracy DRAFT I

As mentioned previously, we performed two different experiments to measure the performance of the cell detection module. The first experiment consisted of training and testing of the algorithm on the same dataset. The training set was 70% and the testing set 30% of the entire dataset. The training and testing datasets were created from images in randomized order from the entire dataset. This allowed us to measure, for each dataset, the maximum precision and recall values we could expect from the algorithm. In the second experiment the training was performed on combined datasets. The goal of this experiment was to observe how well the algorithm is able to generalize, and still return acceptable results.

Training and testing on individual datasets

The training was performed on 70% of the manually dot-annotated images from each dataset, and tested on the remaining 30%. Table 6.1 displays the computed precision and recall values.

Repeat the measurements with the best feature selection

Dataset	Precision	Recall
A	25.0	22.5
B	90.1	89.1
C	76.5	84.2
D	86.1	85.3
E	93.4	78.6

Table 6.1: Precision and recall values for the cell detector trained on each dataset individually.

The detector tested on datasets B to E achieved precision and recall values above 75. A manual comparison of the annotation confirms that the results are good, with few bad detections. Most of the differences in detection were caused by minor inconsistencies in annotations. The importance of consistent annotations was stressed in section 5.1.3.

Dataset A is an outlier with extremely low precision and recall values. This is likely caused by the specific characteristics of this image sequence, which we described in section 5.1.1. Briefly, between frame 17 and 18 there is an abrupt change of image clarity. The background from the threshold frame onwards have a texture that is very similar to that of the cells in the first 17 frames. This means that the negative candidate regions from frame 18 and onward clash with the positive candidate regions from the previous frames. The result is that the detector is unable to learn to discriminate cells from background.

These results show us the detector can correctly detect most of the (annotated) cells. Further manual reviews of the annotations would likely improve the performance, but this is not done here as it should have been tested on a separate validation dataset. Additionally, it is unlikely that future users of this detection method will always have perfect annotations available. In the next experiment, we will measured the performance of the detector when trained on a composite dataset. This will tell us whether it would be possible to learn a single, general detector and use it to test on a new, possibly unforeseen datasets.

Figures C.1 to C.5 in appendix C display a temporal view of the detected results in each datasets.

The vertical axis represents the consecutive frame number of the image sequence. The figures show that “cell tracks” are discernible, even if the number of outliers is significant. The detectors used to detect the cells on the entire dataset were trained on all the annotated frames.

Training on combined datasets

In this second experiment we were interested in measuring how well the algorithm is able to generalize when it is trained on a larger, combined dataset. For this purpose, several of the datasets were grouped, and the detector was trained to recognize cells of all types. The detector was trained on a random 70% of all annotated images in the combined dataset. It was then tested on a random 30% of annotated frames from each individual dataset separately. This means that sometimes the same frames could be used for training and testing. However, we also run combinations of datasets where one dataset is left out of training at each time. Testing on that dataset should then reveal if the algorithm generalizes well.

Table 6.2 summarizes the precision and recall values for all tested combined datasets. The column denoted γ contains testing performance as tested on 30% of the combined dataset. The values shown in bold correspond to the performance of testing on the dataset that was left out from the combined training dataset. The row denoted “Individual” indicates the results when the datasets where trained and tested individually (these are the same results as in table 6.1). The values shown in red indicate a decrease in precision/recall by at least 1 point compared to the results when trained and tested on individual datasets. Similarly, blue colour indicates an increase in precision/recall compared to the results on the row denoted “Individual”.

Repeat the measurements with the best feature selection

Dataset	Precision						Recall					
	γ	A	B	C	D	E	γ	A	B	C	D	E
Individual		25.0	90.1	76.5	86.1	93.4		22.5	89.1	84.2	85.3	78.6
ABCDE	74.5	49.2	76.0	85.2	69.2	94.5	79.9	58.3	90.8	57.1	98.9	79.7
ABCD_	69.7	49.2	77.0	85.8	69.9	**93.4**	80.0	58.3	90.3	58.7	98.4	**80.7**
ABC_E	72.8	45.9	75.0	82.6	**41.1**	87.2	83.2	65.0	96.4	73.6	**99.4**	89.8
AB_DE	75.5	47.5	75.8	**85.7**	73.1	96.5	73.1	50.8	88.9	**46.1**	97.1	68.3
A_CDE	75.9	41.3	**63.1**	85.5	71.2	94.6	70.4	58.3	**89.2**	57.0	99.0	80.2
_BCDE	86.3	**49.2**	76.4	85.9	71.8	94.4	76.4	**58.3**	89.5	56.0	97.4	75.7
BCD	82.8	**52.5**	78.0	87.5	72.7	**95.2**	74.5	**55.8**	89.5	54.7	97.4	**74.0**
_B_DE	85.9	**42.5**	80.3	**84.8**	72.1	97.7	82.6	**45.8**	91.2	**44.8**	97.4	66.3

Table 6.2: Precision and recall values for the cell detector trained on combined datasets. Values typed in bold indicate the testing datasets that were not included in the combined training dataset. Red/blue colour indicates a decrease/increase in performance compared to training and testing on each individual dataset by at least 1 precision/recall point.

The values in the table show us some significant insights. First of all, the recall values increased for most datasets when detecting cells using a detector trained with almost any combination of data. Using more data thus improves the recall. This means that in general the algorithm will detect a larger number of true positive instances, but also some false negatives. However, given the high recall value we can expect the number of false negatives to remain relatively small. For the tracking module it is easy to discard short, isolated detections. The high recall values indicate that we will likely detect most of the real cell trajectories.

Second, we notice that while most recall increase, precision tends to decrease in some datasets. This means that the number of false positives increased. As said beforehand, the cell tracking module can deal effectively with short, isolated false positives. However, if the number of false positives increases too much, the tracking module might detect cell trajectories where there are none (or correspond to background noise). However, given that in most examples the precision values did not fall below 70 we should still expect to achieve good tracking results.

The results on Dataset C went against these observations. Its recall values decreased significantly whenever a combined dataset was used to train the detector. This dataset contains many motion artefacts, and high variance of intensities of the cells, some of which smoothly blend into the background. This makes it very difficult to consistently annotate. It is possible that the inconsistencies in annotation make it perform in such a way when combined with other datasets.

Another interesting observation is that the precision and recall values for dataset A improved significantly compared to training the detector only on dataset A. We have previously presented some possible reasons for its poor performance when trained individually. The increased performance when using a cell detector trained on other datasets (even excluding dataset A itself) can be attributed to two things. First, the detector learns to detect a wider range of types of cells. Second, dataset A contains much fewer annotated cells than the other datasets (about five times less). It is possible that a detector trained on only dataset A overfits the annotations, and cannot generalize to detect cells in other frames.

The performance on dataset E remained almost constant, with very few exceptions (e.g. smaller recall value when the detector was trained on combined dataset containing datasets A, B, D and E).

In this section we aimed to understand whether a detector trained on a single, combined dataset could be used to detect cells in new, previously unseen images. The results have shown that a general detector will not perform well on *all* new unseen datasets. For example, we have measured a significant drop in precision values when testing on unseen datasets B and D, and a significant drop in recall values when testing on dataset C. However, in some cases training on a combined detector either improved the performance or reduced it insignificantly (such as in datasets A and E). What this tells us is that when presented with a new dataset it is worth trying to use a pre-trained detector and review its results. In some cases this could give as acceptable results, and manual annotation of the new dataset might be unnecessary.

It is worth keeping in mind that all five datasets analysed in this thesis show distinct characteristics (both in cell type and image clarity). In practice, datasets will often be similar and it might be sufficient to annotated a single dataset to train a detector that can be used on other similar samples.

6.1.3 Computations time DRAFT I

In order to positionally track cells in image sequence the computation time of extracting the cell position from each frame is just as important as the accurate identification of cells. Table 6.3 displays the average detection times per frame for each dataset. We also measured the average detection time per annotated cell in each dataset and the average ratio between the number of all candidate regions that were detected with the MSER detector and the number of annotated cells in each frame.

The detection was performed on a PC with an Intel(R) Core(TM) i7-2600 CPU with a clock frequency of 3.40GHz and 8GB RAM. The MATLAB version used to measure the detection speed was 8.1.0.604 (R2013a) running in Ubuntu Linux 13.04 x64. Althought the detector can be easily configured to use several workers to process the image sequences in parallel, the measurements were performed using a single worker, i.e. all images was processed sequentially.

Dataset	Time per frame [s]	Time per annotated cell [s]	Candidate-Annotation ratio
A	0.7458	0.4733	36:1
B	1.4466	0.1801	18:1
C	1.2730	0.1422	11:1
D	1.3607	0.9378	21:1
E	0.6273	0.2589	13:1

Table 6.3: Average computation times per frame and annotated cell.

The results show that we were able to optimize the cell detector to a point where it's speed is no longer an issue for many use cases. As a reminder, the original paper by Arteta *et. al.* [4] reports detection speeds of 30 seconds per 400-by-400 pixels image on an i7 CPU. Most of the computation time is spend by the MSER detector and the feature computation.

The measurements show that the detection speed is dependent primarily on the number of cells in each frame. Datasets B, C, and D contain 5 to 10 times more cells per frame than dataset A. Dataset E also contains 5 times more cells than dataset A, but it's computation time per frame was lower. This could be attributed to the smaller fraction of candidate regions that had to evaluated. In contrast, in dataset A which contains larger cells than the other datasets many more candidate cells were evaluated relative to the number of annotated cells. It is likely that, since the MSER detector was configured equally for all datasets, the MSER detector identified a larger number of small noise artefacts as cell candidates.

It might be useful to show the comparison with the old detector if I have the time

6.2 Cell tracker IN PROGRESS

After obtaining the cell detection results we must determine the number of trajectories and which detection results belong to which trajectory. In the analysis of the performance of the tracking system we assume that the detection results are already available. The cell detector had been trained on the annotated images from each dataset, and the detection of cells had been evaluated on the full image sequences. Similarly to the experiments run on the cell detection module, we also run two sets of experiments for the cell tracker. First, we measure the performance of the cell tracker after being trained on each dataset individually. Second, in order to measure the re-usability of a trained cell tracker to link cell detection results in any dataset, we train the cell tracking module on a combined dataset and evaluate its performance on previously unseen trajectories.

The aim of this research was to develop an automatic cell detection and tracking pipeline that would require as little manual work as possible. However, there are a few cases when an automated algorithm might not perform great, for example when testing on a new dataset when reusing a tracker trained on a different dataset. For this reason we present the user a set of four intuitive parameters that can be adjusted to modify the tracking results. We discussed the benefits of this configuration in section 4.6. When the parameters have been modified from their default values these will be written down next to the results.

Further, in order to evaluate the cell tracking module, we are interested in two things. First, how well do the generated trajectories fit the ground truth annotations. This measure is dependent on the quality of the cell tracker module as well as the cell detection module. If the cell detection module misses too many cells, it might be impossible for the tracking module to compensate for the errors. This is a measure of how well the automatic tracking pipeline works. However, we are also interested in how well the developed tracking module works independently of the cell detection module. This can give us an idea of how much of the performance is lost in the tracking module, given perfect detection results. For this reason, we will provide three values for each of the computed metrics:

1. Performance relative to the Ground Truth annotations (GT-P). This is a measure of how well the combination of the cell detector and tracker perform together.

2. Performance relative to *Detected Ground Truth annotations* (DGP-P). *Detected ground truth annotations* are obtained by automatically mapping the ground truth annotations to the cell detection results. For example, a ground truth annotated trajectory is mapped to detection results by finding for each annotated cell observation of the trajectory a detection result that is less than a certain number of pixels away (e.g. 10 pixels). This measure compares the *detected ground truth annotation* to the generated trajectories and gives a measure that is less dependent on the performance of the cell detection module.

3. Comparison of Ground Truth annotation and *Detected Ground Truth annotations* (GT-DGT-P). This measure does not measure the quality of the generated trajectories, but measures the performance lost due to errors in the cell detection module.

These performance evaluations measure the performance of the longest ground truth trajectories.

Annotating all the trajectories in the full datasets would require weeks of manual work and review to assert the correctness of the annotations. For this reason testing is limited to a certain number of long fully annotated trajectories. In each dataset we have identified and annotated a few long trajectories. The counts of the annotated trajectories together with their length is available in table 6.4. For the analysis of cell behaviour, we are mostly interested in these long trajectories, instead of the short trajectories caused by cells that appear just for a brief number of frames.

When finished annotating update the table

Dataset	Number of annotated trajectories	Length (number of frames) of annotated trajectories
TODO		

Table 6.4: TODO

All the provided measurements of the cell tracker performance have been obtained by cross validation. This allows us to measure how a trained tracker will generalize to an independent image sequence. Specifically, because the number of annotated trajectories is relatively small, we have used leave-one-out cross validation. This involves repeatedly partitioning the set of annotated trajectories into two subsets, training the tracker on the training subset, and measuring its performance on the testing subset. As the name suggests, in each round we train the tracker on all but one annotated trajectory, using the left-out trajectory for measuring the performance. This is repeated for each annotated trajectory and the results are averaged over all the rounds.

6.2.1 Performance metrics DRAFT I

To quantify the performance of the cell tracking module we use two metrics originating from radar tracking literature [29], recently used in other works [16, 23, 30]: *target effectiveness* and *track purity*.

Let us call a *target* a true cell trajectory as provided by manual ground truth annotations and *track* the generated trajectory by the cell tracker module. Then, for a target and track pair, target effectiveness can be defined as:

$$\text{Target Effectivenes} = \frac{100 \times \text{Number of frames where the target is correctly followed by the track}}{\text{Total number of frames of the target}}.$$

Note that the multiplication by 100 is simply to represent the output from 0 to 100 instead of 0 to 1. Target effectiveness therefore tells us how effectively a target if followed by a track. For each target we find the track that follows it correctly for the longest number of frames. The measure penalizes tracks that are initialized late or terminated early relative to the target. However, it does not penalize tracks for being initiated early, or terminated late. Therefore, we need an additional measure, track purity, which does just that. In the case of track purity the roles of track and target are inverted. It measures how well the target follows a track, and is defined as:

$$\text{Track Purity} = \frac{100 \times \text{Number of frames where the track is correctly followed by the target}}{\text{Total number of frames of the track}}.$$

This measure will penalize early initiated or late terminated tracks. It is similar to precision in that it penalizes false detections. We measure it by assigning to each track the target that follows it for the longest number of frames.

6.2.2 Tracking accuracy IN PROGRESS

As mentioned previously, we performed two different experiments to measure the performance of the cell tracking module. The first experiment consisted of training and testing of the algorithm on the same dataset. The performance evaluation was performed by leave-one-out cross validation where we repeatedly train the model on $n-1$ annotated trajectories and test it on the trajectory that was left out. This first experiment allowed us to measure, for each dataset, how well the tracker could link cells with a specific mode of motion. In the second experiment the training was performed on a combined dataset. The goal of this experiment was to observe how well the algorithm is able to learn from a large number of trajectories exhibiting different modes of motions and apply the learned model to track a previously unseen trajectory.

Training and testing on individual datasets NEW

The performance measurements were obtained by leave-one-out cross validation. Table 6.5 displays the computed target effectiveness and track purity values for each individual dataset.

Rerun the tracker and measure performance. These are only preliminal results

Dataset	Target Effectiveness			Track purity		
	GT-P	DGT-P	GT-DGT-P	**GT-P**	DGT-P	GT-DGT-P
B	**90.652**	92.463	98.041	**99.735**	99.735	100
C	**92.906**	93.385	99.463	**87.741**	87.712	100
D	**98.351**	98.542	99.794	**77.169**	77.169	100
E	**94.051**	94.051	100	**92.52**	92.52	100

Table 6.5: Target effectiveness and track purity for the cell tracker as measured on individual datasets.

Note that table table 6.5 does not include dataset A because the detection module was unable to detect cells in at least two trajectories. This prevented us from training the robust tracklets linker classifier due to the inability to generate negative training examples (i.e. examples of robust tracklets that should not be linked).

All the datasets were configured and trained to close gaps of up to 9 frames, with the parameters set to the default values of $\pi_{init} = 1$, $\pi_{term} = 1$, $\pi_{link} = 1$ and $\pi_{FP} = 1$. The exception is dataset E, where the tracker was configured to perform two runs, the first one closing gaps up to length 9, and the second one up to 20, and the parameters π_{init} and π_{term} were set to 3. This extra gap closing was required for this dataset, because the motion artefacts caused that the number of very short robust tracklets (of length 1 and 2) was very high. The robust tracklets and generated trajectories for this dataset can be seen in fig. 6.1. Since all robust tracklets of length 1 are simply eliminated, there were a few gaps longer than 9, which could be effectively closed in the second iteration.

Missing figure

example close gap 20 for dataset E

Figure 6.1: example close gap 20 for dataset E

All four dataset exceeded target effectiveness of 90. Dataset B achieved track purity of 99.7, dataset C of 87.7, dataset D of 77.2 and dataset E of 92.5. The lower track purity in dataset E can be attributed to two related reason. First, dataset E contains a significant number of motion artefacts, arguably more than the other datasets. The second and more likely reason for the low target effectiveness is that several frames had been manually deleted from this dataset. In the original dataset more than 30% of the frames were completely unusable; no cells were clearly discernible. These frames had been simply deleted before any analysis was performed on them. The results is that the original cell trajectories were cut, and these cuts are clearly noticeable in a few places in the image sequence. Cell trajectories where the cut is clearly visible were annotated as 2 distinct trajectories. However, since the tracker wasn't aware of this deletion it, it linked these individual track segments, resulting in a lower track purity value. This can be clearly seen in fig. 6.2 which contains the most affected trajectories.

Missing figure

Add low purity cause image for dataset D

Figure 6.2: Add low purity cause image for dataset D

The comparison of ground truth annotations and *detected ground truth annotations* is very high in all datasets, which tells us that the cell detector has in general detected most of cells of the evaluated trajectories (note that the position of missing detections were completed by linearly interpolating the positions of nearby detections; the high values do not mean that were almost no false negatives).

The track purity for DG-DGP-P is always one, because due to the design of *detected ground truth annotations* it cannot occur that they ever are longer than the ground truth annotations.

Finally, the measurements confirm the expectation that the performance of the generated trajectories relative to the *detected ground truth annotations* is higher than their performance relative to ground truth annotations.

The generated trajectories for each of these datasets are available in fig. C.2 and **??????** in appendix **??**. These trajectories were obtained from trackers that were trained on all the annotated trajectories.

Add all generated trajectories to appendix

Training on a combined dataset NEW

The first experiment showed us the ability of the tracker to learn to correctly link trajectories from a single dataset. Often, most cells in the same dataset move in a similar mode of motion (e.g. mostly stagnant, slow translation in one direction, fast translation, frequent direction changes, etc). In this second experiment we will train the tracker on all the annotated trajectories from all five datasets together (at each round of the cross validation we only leave a single tracklet out of the training set for testing). We are interested in how well the tracker will be able to learn from this vast range of modes of motions, and whether it will be able to track a new cells that could exhibit any of these modes of motion.

An additional difficulty in setting up this experiments comes from the fact that the training and testing is performed on the longest tracklets of the entire dataset. Therefore, shorter image sequences are likely to be under-represented when training and testing on the dataset composed of all five datasets. Therefore the experiment was also evaluated on a combined dataset containing only the three shortest image sequences (A, B and C).

Dataset	Target Effectiveness			Track purity		
	GT-P	DGT-P	GT-DGT-P	**GT-P**	DGT-P	GT-DGT-P
ABCDE	**81.7**	83.0	97	**74**	73.8	1.00
ABCD						
ABC						

Table 6.6: TODO.

Warning that since dataset D which has the longest trajectories is also the worst its values overpower the rest.

This are only preliminary results

Discuss a bit

6.2.3 Computation time NEW

When testing the tracker on individual datasets we also measured the time required to link all the robust tracklets into trajectories. The results are shown in table 6.7.

The detection was performed on a PC with an Intel(R) Core(TM) i7-2600 CPU with a clock frequency of 3.40GHz and 8GB RAM. The MATLAB version used to measure the tracking time was 8.1.0.604 (R2013a) running in Ubuntu Linux 13.04 x64.

Dataset	Total time [seconds]	Time per frame [seconds]	Number of frames per second
B	2.6401	0.0378	26.4550
C	9.7935	0.0777	12.8700
D	19.256	0.0511	19.5695
E	5.7784	0.0298	33.5570

Table 6.7: TODO.

The average time per frame over all the datasets is 0.049 seconds, which is equivalent to processing 23.1 frames per second. This short computation time is negligible next to the time taken by the cell detection module. The benefit of this is that the user can play with the parameters π_{init}, π_{term}, π_{link} and π_{FP} and regenerate the trajectories in just a few seconds.

The slowest part of the cell tracking module is the computation of spatio-temporal features, especially the unoptimized code that computes the Gaussian broadening features (see section 4.7.2).

6.3 Limitations and areas of improvement NEW

Answer: what, why, how to improve in future

Testing the tracker on the studied datasets has given good results as seen in fig. 6.3 which shows all generated trajectories of length at least 10 for dataset B.

Figure 6.3: datasetBlongerthan10

As we have shown in fig. 6.1 it is also able to efficiently close very large gaps. The tracker attempts to close gaps up to the maximum length it is trained to do. Since a classifier is used to determine whether two tracklets should be linked, in general equal linking likelihoods can be computed for tracklets that are e.g. 2 frames apart than tracks that are 7 frames apart. If we try to close very long gaps at once, e.g. 30 frames some errors could occur. For example, if a cell trajectory is composed of three segments (robust tracklets) where the middle one is of length less than 30, the tracker might decide to link the first and third segments of the trajectory, making the middle segment a false positive or a short tracklet, which would be wrong. To work around this problem the closing of gaps can be performed iteratively, e.g. first for gaps of length 10, then 20 and then 30. This solution works fine, but can still introduce some smaller errors. An alternative implementation of the system could use a regressor instead of a classifier such that small caps are given a larger precedence over long gaps. However, this method require to identify a suitable way to train the regressor automatically.

The tracker is able to deal with some limited jiggling of frames. However, as we have seen in fig. 6.2 excessive jiggling (in this case cause by the manual deletion of several frames) can cause the tracker to perform poorly. This issue could be resolved with a pre-processing step that would stabilize the images based on the background information present in them before inserting them to the cell tracker.

Finally, let's point out that the tests performed on the tracker only measured the performance of the longest trajectories. The reason for this is to that careful annotation and review of all the trajectories in the datasets would be too time consuming and would likely include some not-insignificant amount of incorrectly linked cells.

ELimintae Kalmna filter notice if I decide to implement it

Finally, sometimes the connections between robust tracklets is worse than expected, as seen in fig. 6.4. This could be solved by computing more advanced spatio-temporal features. For example, extrapolating pairs of candidate linking tracks to their midpoints by means of a Kalman filter and computing the euclidean distance between the extrapolated tracklets head and tail.

Figure 6.4: datasetwithoutliersfromtrajectory

6.4 Summary NEW

In this chapter we have demonstrated the performance of the cell detection and tracking modules. Several experiments on individual and combined datasets were performed to measure how well the algorithms are able to generalize to new independent datasets.

The cell detector module was able to correctly detect a large number of cells in the input images in all but one dataset. The cell tracker module was then able to use these detection results to generated cell trajectories with good results. The limitations and good points of the tracker are also described and demonstrated.

Additionally, we have measure the computation time of both the detector and tracker. We have shown that our system is able to process on average 53 frames per minute (96% of the time spent by the cell detector). Most of the time is spend in the extraction of extremal regions and feature computation of the detection module. A considerably smaller amount of time is spent in the cell tracker module, mostly in the unoptimized spatio-temporal feature computation.

Appendices

Bibliography

[1] P. K. Elzbieta Kolaczkowska, "Neutrophil recruitment and function in health and inflammation," 2013. 7

[2] J. Pillay, I. den Braber, N. Vrisekoop, L. M. Kwast, R. J. de Boer, J. A. M. Borghans, K. Tesselaar, and L. Koenderman, "In vivo labeling with 2h2o reveals a human neutrophil lifespan of 5.4 days," *Blood*, vol. 116, no. 4, pp. 625–627, 2010. 7

[3] P. S. Tofts, T. Chevassut, M. Cutajar, N. G. Dowell, and A. M. Peters, "Doubts concerning the recently reported human neutrophil lifespan of 5.4 days," *Blood*, vol. 117, no. 22, pp. 6050–6052, 2011. 7

[4] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect cells using non-extremal regions," in *Proceedings of the 15th International Conference on Medical Image Computing and Computer-Assisted Intervention - Volume Part I*, MICCAI'12, (Berlin, Heidelberg), pp. 348–356, Springer-Verlag, 2012. 8, 9, 11, 17, 18, 19, 20, 21, 22, 46, 51, 57

[5] Y. Chen, K. Biddell, A. Sun, P. Relue, and J. Johnson, "An automatic cell counting method for optical images," in *[Engineering in Medicine and Biology, 1999. 21st Annual Conference and the 1999 Annual Fall Meetring of the Biomedical Engineering Society] BMES/EMBS Conference, 1999. Proceedings of the First Joint*, vol. 2, pp. 819 vol.2–, Oct 1999. 10

[6] X. Chen, X. Zhou, and S.-C. Wong, "Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy," *Biomedical Engineering, IEEE Transactions on*, vol. 53, pp. 762–766, April 2006. 10, 13

[7] L. Vincent, "Morphological grayscale reconstruction in image analysis: applications and efficient algorithms," *Image Processing, IEEE Transactions on*, vol. 2, pp. 176–201, Apr 1993. 10

[8] J. Serra, *Image Analysis and Mathematical Morphology.* Orlando, FL, USA: Academic Press, Inc., 1983. 10

[9] D. Mukherjee, N. Ray, and S. Acton, "Level set analysis for leukocyte detection and tracking," *Image Processing, IEEE Transactions on*, vol. 13, pp. 562–572, April 2004. 11, 13

[10] C. Tang, Y. Wang, and Y. Cui, "Tracking of active cells based on kalman filter in time lapse of image sequences of neuron stem cells." 11, 14

[11] D. Xu and L. Ma., "Segmentation of image sequences of neuron stem cells based on level-set

algorithm combined with local gray threshold.," Master's thesis, Harbin Engineering University, 2010. 11

[12] C. Arteta, V. S. Lempitsky, J. A. Noble, and A. Zisserman, "Learning to detect partially overlapping instances.," in *CVPR*, pp. 3230–3237, IEEE, 2013. 11, 12, 19

[13] J. Matas, O. Chum, M. Urban, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of the British Machine Vision Conference*, pp. 36.1–36.10, BMVA Press, 2002. doi:10.5244/C.16.36. 11

[14] T. Joachims, T. Finley, and C.-N. J. Yu, "Cutting-plane training of structural svms," *Mach. Learn.*, vol. 77, pp. 27–59, Oct. 2009. 11

[15] R. Bise, T. Kanade, Z. Yin, and S. il Huh, "Automatic cell tracking applied to analysis of cell migration in wound healing assay," in *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, pp. 6174–6179, Aug 2011. 12, 25

[16] S. Huh, *Toward an Automated System for the Analysis of Cell Behavior: Cellular Event Detection and Cell Tracking in Time-lapse Live Cell Microscopy.* PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, March 2013. 12, 13, 53

[17] D. House, M. Walker, Z. Wu, J. Wong, and M. Betke, "Tracking of cell populations to understand their spatio-temporal behavior in response to physical stimuli," in *Computer Vision and Pattern Recognition Workshops, 2009. CVPR Workshops 2009. IEEE Computer Society Conference on*, pp. 186–193, June 2009. 13

[18] B. Xu, M. Lu, P. Zhu, Q. Chen, and X. Wang, "Multiple cell tracking using ant estimator," in *Control, Automation and Information Sciences (ICCAIS), 2012 International Conference on*, pp. 13–17, Nov 2012. 14

[19] K. Li and T. Kanade, "Cell population tracking and lineage construction using multiple-model dynamics filters and spatiotemporal optimization," in *Proceedings of the 2nd International Workshop on Microscopic Image Analysis with Applications in Biology (MIAAB)*, September 2007. 14

[20] A. Massoudi, D. Semenovich, and A. Sowmya, "Cell tracking and mitosis detection using splitting flow networks in phase-contrast imaging," in *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, pp. 5310–5313, Aug 2012. 14

[21] L. Zhang, Y. Li, and R. Nevatia, "Global data association for multi-object tracking using network flows," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008. 15, 28

[22] C. Huang, B. Wu, and R. Nevatia, "Robust object tracking by hierarchical association of detection responses," in *Computer Vision - ECCV 2008* (D. Forsyth, P. Torr, and A. Zisserman,

eds.), vol. 5303 of *Lecture Notes in Computer Science*, pp. 788–801, Springer Berlin Heidelberg, 2008. 15, 28

[23] R. Bise, Z. Yin, and T. Kanade, “Reliable cell tracking by global data association.,” in *ISBI*, pp. 1004–1010, IEEE, 2011. 15, 25, 28, 31, 53, 57

[24] H. Kuhn, “The hungarian method for the assignment problem,” *Naval Research Logistics Quarterly*, vol. 2, pp. 83–97, 1955. 15

[25] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and Vision Computing*, vol. 22, no. 10, pp. 761 – 767, 2004. British Machine Vision Computing 2002. 18

[26] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, “Support vector machine learning for interdependent and structured output spaces,” in *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML ’04, (New York, NY, USA), pp. 104–, ACM, 2004. 20

[27] K. Li, E. D. Miller, M. Chen, T. Kanade, L. E. Weiss, and P. G. Campbell, “Cell population tracking and lineage construction with spatiotemporal context,” *Medical Image Analysis*, vol. 12, no. 5, pp. 546 – 566, 2008. Special issue on the 10th international conference on medical imaging and computer assisted intervention - {MICCAI} 2007. 25, 59

[28] M. Looney, E. Thornton, D. Sen, W. Lamm, R. Glenny, and M. Krummel, “Stabilized imaging of immune surveillance in the mouse lung.,” *Nature Methods*, vol. 8, no. 5, pp. 91–6, 2011-01-01 00:00:00.0. 37

[29] S. Blackman and R. Popoli, *Design and Analysis of Modern Tracking Systems*. Artech House radar library, Artech House, 1999. 53

[30] S.-i. S. Eom, K. D. F. Elmer, B. Ryoma, and K. Takeo, “Tracking of hematopoietic stem cells in microscopy images for lineage determination,” *J Latex Class Files*, vol. 6, no. 01-09, p. 9, 2007. 53