# Cell Tracking using Convolutional Neural Networks

Anton Jackson-Smith
Stanford
acjs@stanford.edu

## Abstract

*Automated cell segmentation and tracking is an ongoing challenge in biology, with better technology facilitating higher throughput and clearer data. We build a system for tracking fluorescent cell nuclei in microscopy data, using hierarchical visual tracking on convolutional neural networks. We instrument an existing cell-segmentation CNN to provide feature data for cell tracking, in addition to training our own network, RecogNet, to provide higher-quality features online. The tracker reaches 60% precision at a 20 pixel threshold, and we present several avenues for further work in the future.*

## 1. Introduction

A key challenge in biotechnology is data collection: sensitively gathering information on the internal composition and dynamics of single cells. Historically, these measurements were made in bulk, with the resulting data representing population averages across millions of cells. This approach provides information about composition, but conceals small-scale dynamics and variation between cells. More recently, single-cell analysis, aided by technologies such as single-cell sequencing, improved imaging, and microfluidics, has allowed for characterisation of total composition, as well as dynamics, in individual cells. However, the technique has a different limitation: data collection and analysis is time consuming, and statistical power is limited by the number of cells that can be individually measured by a human researcher.

High-throughput single cell analysis provides a solution to these limitations, providing the sensitivity and specificity of single-cell measurements with the statistical power of bulk measurements. However, while cells can be imaged and characterised in unprecedented numbers, data analysis (particularly image segmentation and cell tracking) is still a manual and time-consuming process. While tools exist to aid this process, even modern analysis pipelines require significant input: pre-filtering of data, hand-tuning of algorithms, and post-analysis curation of the output data. Human effort is required linearly with cell and image count. An automated pipeline which could segment and track cell nuclei with little or no intervention would greatly increase availability of data, as well as the productivity of biologists.

*DeepCell* [11] endeavours to solve this problem. The software uses a convolutional neural network, trained on various cell types, to automatically segment cells within images as part of a larger analytical pipeline. However, the network does not yet track individual cells over time, a particularly important task for gathering time-series data. This project aims to add cell tracking capabilities to the DeepCell software package.

Tracking individual cells over time is generally that of object tracking, but with several complications particular to the domain. Rather than tracking a single object across a confused background, the system must track a large number of cell, on the order of hundreds per field of view, across a background which is otherwise clean. Any solution must distinguish a number of very-similar objects from one another, either through subtle visual differences, motion inference, or other methods. Although cells exist in a generally two-dimensional plane, they can overlap and move over one another. Unique to the cell tracking problem, cells can divide.

This work focuses on a specific subset of cell tracking: tracing fluorescently-labelled cell nuclei. Following the approach by Ma et al [8], we leverage the complex features of deep convolutional neural networks, combined with the real-time speeds of kernelized correlation filters for tracking. This method provides a number of advantages. Where an existing CNN is used, we realise the efficiencies of training only one network to perform multiple tasks, as well as the advantage of pre-computing feature maps during segmentation that can later be used for tracking. By tracking cells individually, rather than globally-optimizing for minimal translocation, we obtain the flexibility to detect cell events such as division or death, and restart or terminate tracking.

Overall, we apply hierarchical visual feature tracking in two modes. First, we use the existing DeepCell segmentation network to generate segmentation information as well

as tracking feature maps. Secondly, we build and train a bespoke CNN, specifically for the task of distinguishing cell nuclei, and then use this network to generate tracking feature maps in real-time. The tracking framework learns a correlation filter for each tracked cell, training on features extracted by the convolutional networks. Features at deeper levels are propagated up to shallower levels, combining accurate tracking using deep semantics with the spatial resolution of the early filters.

## 2. Related Work

The field of cell segmentation is not new: Gauthier and Levine reported an improved method for segmenting live whole-cell images in 1995 [4], using a two-step thresholding approach. It is interesting to note that, while the technology for imaging and analysis has certainly improved, the challenge the authors sought to solve is still outstanding. A decade later, Chen et al reported a multi-step method based on global thresholding for segmentation, followed by a watershed algorithm to further separate overlapping nuclei. Several rules were then used to reconcile over-separated fragments into nuclei. Tracking is then performed using a set of rules: a cell is presumed linked to its nearest neighbour in the subsequent frame, unless another constraint (such as that two cells cannot have the same identity at $t + 1$) is violated. The authors report approximately 94% accuracy at tracking nuclei through a capture sequence.

*CellTrack* is a software package published by Sacan et al in 2008. [10] The software provides ready access to a number of segmentation and tracking algorithms, but still requires human intervention in the processing pipeline. The authors also developed a 'snake' active contour method for the task of cell segmentation. Template-matching and optical flow deformation are used to calculate deformation across timesteps, and the energy of the 'snake' contour is minimised to refine the cell outline.

A common cell-tracking approach based on pre-segmentation of time series data is based on the linear assignment problem, described by Jaqaman et al [7] in the context of particle tracking within cells. In brief, particles (or cells) are connected locally by 'tracklets', linking positions from one timestep to another while satisfying constraints (particles cannot duplicate) and minimising a cost equation. The optimisation to find the set of tracklets linking two timesteps is spatially global, but temporally local, applying only to the present timestep. Next, a temporally-global optimisation is performed to link tracklets into complete trajectories across time. In general, this will only require connecting one tracklet to another, but the cost function allows for tracklets to be reconfigured where it minimises the overall cost of the trajectories. The method facilitates tracking particle merges and divisions. Overall, solving the linear assigment problem in this manner is effective where well-segmented data exists, however cell overlaps are still problematic. The present state-of-the-art in cell tracking is based on hand-curated filters and rules; however object tracking generally is embracing machine-learning approaches.

Hong et al describe a pre-trained CNN combined with an online Support Vector Machine to track objects. [6] Target-specific saliency maps are constructed through in order to improve localisation accuracy. The map is combined with a generatively-constructed appearance model to locate targets through Bayesian filtering. Dundar et al train a two-layer network in both supervised and unsupervised forms, feeding the convolutional layers into a "Radial Basis Function Network" to predict a confidence map of the desired object's location. [3] Confidence output would be a useful mechanism to augment a linear-assignment tracker, but the authors report mixed accuracy, and it is unclear if the approach would generalise to novel datasets.

Ma et al outline the approach we take in this paper. [8] Rather than requiring a full round of forward and back propagation, the authors train correlation filters over features extracted from multiple levels of an existing deep convolutional nnetwork. The algorithm leverages the semantic sensitivity of the top layers of the network to identify targets, combined with the spatial sensitivity of the low layers for localisation, but avoids repeating a full pass through the network during tracking. At each timestep the filter is used to locate its target within a search window. The window is then recentered, the filter is updated, and the search continues. Filter calculations are performed in Fourier space, dramatically increasing the speed of match processing. [1]

## 3. Methods and Implementation

The tracking system has three components: CellTrack, the hierarchical tracker to perform correlation-filter tracking based on extracted features; DeepCell, for semantically segmenting cell images and producing feature maps; and RecogNet, a custom neural-network specifically trained to distinguish between cell nuclei, for online generation of feature maps for the tracker. While we originally sought to use DeepCell alone due to its success at cell segmentation and extensive training time, it has several limitations, prompting the development of RecogNet.

## 4. CellTrack

### 4.1. Acquisition

The hierarchical tracker follows the design of Ma et al. [8] The tracker uses the 'kernel trick', computing circularised kernels in Fourier space, to increase performance. [5] At each tracked cell, a search window is located over the cell in the first frame of the input. When running using DeepCell, this frame is the set of neural network features

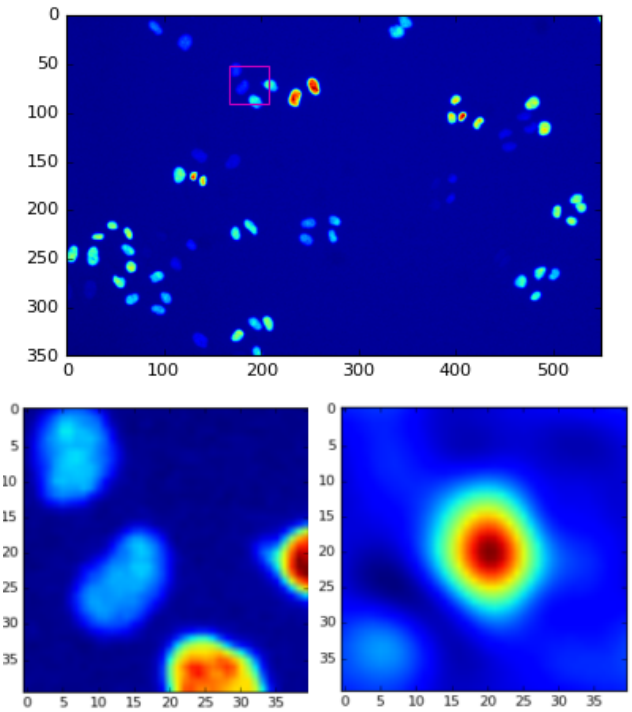

Figure 1. The live visualization interface for CellTrack. (Top) A global view of the frame analysed, with a bounding box identifying the search window containing the tracked cell; (left) a close up of the search window containing the tracked cell; (right) the weighted correlation filter response to the content of the search window.

at a configurable number of layers, output during segmentation. When running using RecogNet, the frame is a raw image, and the features are computed by the CNN online.

Once the search window is positioned, a suite of correlation filters are learned by minimising the following equation:

$$w^* = argmin \sum_{m,n} \|w \cdot x_{m,n} - y(m,n)\|^2 + \lambda \|w\|_2^2$$

where $w^*$ is the learned filter, $x_{m,n}$ is the search window, for all circular rotations $(m,n) \in \{0,1,...,M-1\} \times \{0,1,...,N-1\}$, and $y(m,n)$ is a Gaussian function centered at the unshifted window (ie, $(m,n) = (0,0)$). In Fourier space, this minimisation can be performed by calculating the following:

$$W^d = \frac{Y \odot \bar{X}^d}{\sum_{i=1}^{D} X^i \odot \bar{X}^i + \lambda}$$

where uppercase variables represent the lowercase variables transformed into Fourier space, $X^d$ is feature map $d$ in the layer, and $\bar{X}$ is the complex conjugate of $X$. We train filters on a number of layers of the selected convolutional network.

### 4.2. Tracking

At each tracking timestep step, the correlation filters are matched features within a search window $z$ overlaid on the last known location of the target. Matches are computed in Fourier space using the equation:

$$f_l = \mathcal{F}^{-1}(\sum_{d=1}^{D} W^d \odot \bar{Z}^d)$$

The deeper filters are then propagated to higher levels in a weighted manner:

$$f_{l-1}(m,n) + \gamma_l f_l(m,n)$$

The estimated new location is found by taking

$$\arg\max_{m,n} f_0$$

and the search window is realigned to the new location.

### 4.3. Update

The correlation filters for each feature layer are then updated using a moving average controlled by a learning rate hyperparameter, $\eta$. In our implementation, $\eta = 0.01$. For efficiency, this update is also be performed in Fourier space (not shown).

### 4.4. Input/Output

CellTrack takes as input a segmented image indicating the locations of tracking targets in the first frame of a time-series sequence, as well as a list of either raw images (when using RecogNet), or feature maps output by DeepCell. Cells can be selected for tracking, or all segmented cells can be tracked across the time series. Live output of the tracking is available for debugging or investigation (Figure 1).

Each CellTrack run outputs a list containing the learned filters for each cell, as well as a history of its centroid positions at each timestep over the run. This data can then be used to annotate segmented but unlabelled images, or to directly map cell trajectories.

## 5. DeepCell

DeepCell is a deep convolutional neural network designed to segment cells. [11] The model has been trained on several different training sets, allowing for segmentation of cell nuclei, bacteria, and whole eukaryotic cells under light microscopy. The network has five convolutional layers, followed by a 200-neuron fully-connected layer feeding into three output neurons. One output indicates whether a target

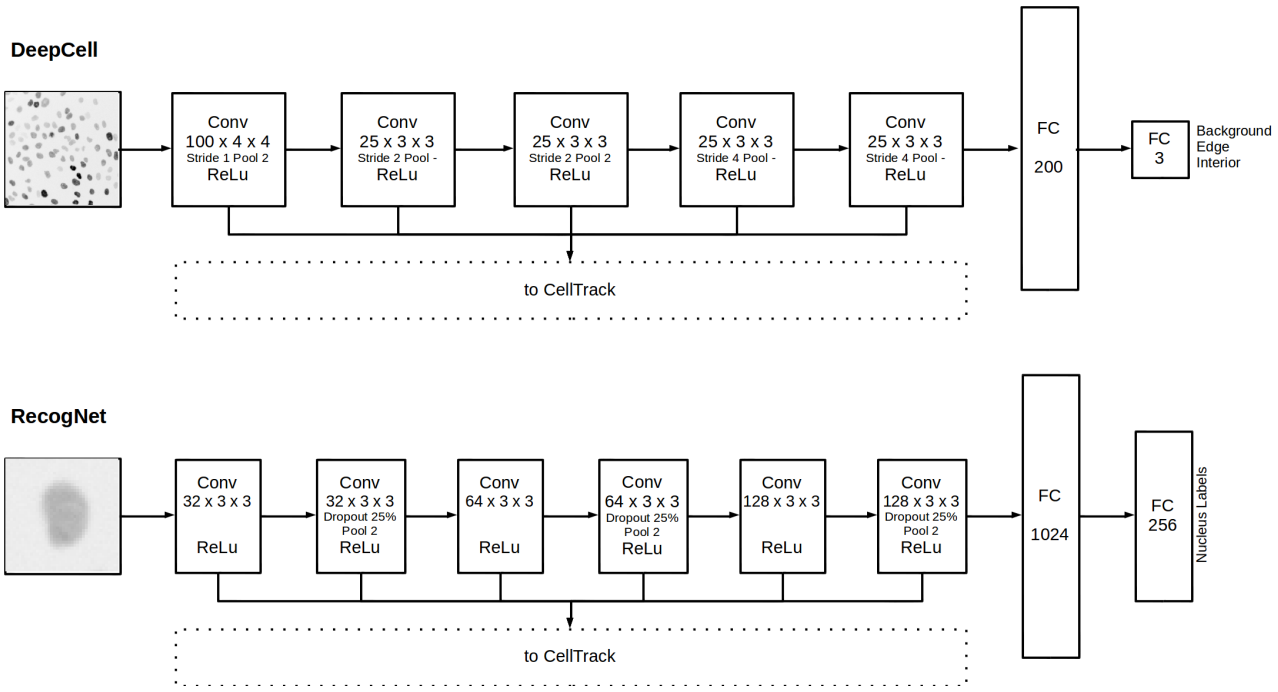


Figure 2. Network designs for DeepCell and RecogNet.

pixel is cell interior, one indicates boundary, and the other background. The full network is effectively convolved over an input image to segment it at the pixel level. The network is implemented in Theano, and a sub-tensor optimisation prevents a full recalculation per pixel. The network's overall output is three masks (nucleus, boundary, background), composed of the output of an ensemble of five trained models.

We modified DeepCell to simultaneously extract feature maps from the convolutional layers during segmentation, and write them out to disk. Maps from each layer are composited to form full-size feature mappings, of the same shape as the final segmentation output. As a result of DeepCell's segmentation strategy and downsampling during convolution this results in a segmented image and feature maps 30 pixels smaller on each dimension than the original. Cells in the 15 pixel boundary are ignored during tracking. The subtensor design of DeepCell makes these transformations more difficult than they would be otherwise, as the pool layers and strided convolutions have somewhat counterintuitive effects on map dimensions.

Overall, DeepCell now produces a set of masks and feature maps, per frame of microscopy input. After hyperparameter optimization, we build tracking filters using layers 1 through 4 of the network, with the layers weighted $(1.0, 0.5, 0.2, 0.2)$ respectively.

## 6. RecogNet

We decided to also implement our own CNN, due to the limitations of using DeepCell for feature recognition—particularly its poor ability to discriminate between cells (see Results and Discussion, below). RecogNet is a deep convolutional neural network trained specifically to discriminate between different cell nuclei. The network is implemented in Keras on Theano, and is composed of six 3x3 convolutional layers with ReLu activation, three maxpool layers, a 1024-neuron fully-connected layer, and a 256-category softmax output. Dropout is implemented at various levels of the network. The network runs on GPU, and is instrumented to output feature maps from each convolutional layer in addition to (or instead of) label classification scores. Although the network is deeper and contains more filters than DeepCell, it runs significantly faster due to the vastly reduced input size: 40x40, versus 426x330 for DeepCell on our data.

Our goal was not so much to perform a classification task as it was to optimise the convolutional layers to produce feature maps suitable for inter-nucleus discrimination. In order to achieve this, however, we cast the optimization problem as one of categorization. Input to the network consists of 1x40x40 (greyscale) images of centered nuclei, equivalent to the 40x40 search windows used by CellTrack. We have multiple images of the same nucleus, captured over time, annotated in our tracking-training database. We select 256 nuclei, and train the network to classify each nucleus at each

timestep as its correct ID.

Again, this is not to perform identification using the CNN. First, the problem is remarkably challenging: nuclei change appearance dramatically over time, particularly during cell division. Second, the training would not be generalisable to other cell tracking: our network would only be able to effectively separate nuclei it had seen before. The intensive training required is not efficiently repeatable for every new tracking task. Finally, the category layout based on arbitrary nucleus ID is an inappropriate means to classify such similar objects; some kind of learned embedding may provide better results, as similar nuclei could be classified closer in multidimensional space. The classification task does force the higher CNN layers to be able to discriminate fine differences between nuclei: the precise feature mappings we need for accurate and generalisable tracking using CellTrack.

RecogNet was trained for 20 epochs, using 13 500 training and 1 500 validation samples. The data is dynamically loaded and segmented from raw microscopy images and segmentation data: segmentation data is used to center a bounding box around the nucleus in the raw image. The data is mean-centered across the training set, and normalised to $\pm$ 1. We retain approximately 4000 additional samples as test-set data.

Validation loss decreased from approximately 4 to 0.76 over the course of training, resulting in 0.83% validation accuracy (0.91% train); remarkable, given the above. Visualisation of the learned weights shows a transition from cell-like filters at the shallow levels, to abstract patterns at deep levels: as we would expect and in contrast to DeepCell.

### 6.1. Dataset

We used two datasets for development and validation of CellTrack. The Covert Lab at Stanford has a collection of time-series fluorescent nuclei images gathered during the course of experiments. This data has been segmented and tracked through automatic pipelines, with hand curation of parameters and results. While the data is not entirely accurate, the experiments have produced an enormous amount of data. A single fluorescence experiment produces $\tilde{1}20$ frames of $\tilde{9}5$ fields of view, each containing $\tilde{5}00$ cells, for more than 5 000 000 individual cell images, or 50 000 cell tracks. Data from the 20150812-Pos5 was used to train RecogNet, and to develop and test CellTrack. The data is of fluorescently-labelled fibroblast cells.

While hand-annotated cell tracking dataset of significant size are rare (as a result of the difficulty of producing them), the *Biomedical Imaging Cell Tracking Challenge* produces and distributes annotated datasets for their annual cell-tracking competition.[9] Past years' datasets are available online. We selected the N2DL-HeLa dataset to use as an independent test set of tracker accuracy. Although the cells are HeLa cancer-line cells, the nuclei have similar morphology and movement profiles as the Covert lab dataset. The nuclei in the dataset do divide more frequently; a factor contributing to low test performance. The dataset has two timeseries of 91 frames, each containing $\tilde{5}0$ cells. Each series has hand-segmented masks for certain frames, and hand-annotated ID masks for every frame.

### 6.2. Validation

Wu et al. describe a series of benchmarks appropriate for measuring tracking performance. [12] They define a precision metric, where precision is the percentange of predictions made by the tracker within a threshold Euclidian distance of the ground truth for that prediction. A precision curve is plotted based on the calculated precision across a range of thresholds. Their success metric measures the overlap between the predicted object bounding box and ground truth. We do not use this measure, given that bounding box prediction is based on the output of segmentation once we've predicted the identity segment's centroid. We use the precision metric to quantitate the performance of our tracker under various conditions.

Tracking was performed on both the 20150812-Pos5 dataset (200 cells), and on N2DL-HeLa-1 (approx 50 cells). We first identified ground-truth ids corresponding to our tracked cells by sampling the acquisition coordinates for each cell in the ground-truth data. Once we had labels for each cell, we built a trajectory of $(x, y)$ coordinates by taking the centroid position of the labelled cell at each timestep. Where a track disappears from the ground truth data (such as a cell death or division), we cease comparison with predicted values. We then compared the computed ground truth trajectory with that predicted by CellTrack, calculating precision as above.

## 7. Results and Discussion

We tested the tracking system using both DeepCell features, and RecogNet. DeepCell performs reasonably on the dataset, reaching 60% accuracy rapidly, levelling off, and then further improving as the threshold is relaxed (Figure 3). Precision is approximately at 60% at the 20 pixel threshold taken as representative by Wu et al.[12] While this is a promising start, it does not meet the high accuracies reported by Chen and Jaqaman [2, 7].

Qualitative diagnosis, by following individual cells as they are tracked, suggests a major cause of inaccuracy. First, where another cell enters the search window, the correlation filter tends to see the other cell as a possible match, "jumping" to it if the correlation exceeds that of the actual cell in magnitude. While this problem could be ameliorated by applying a momentum term to the moving search window, the larger problem is that the correlation filter, even

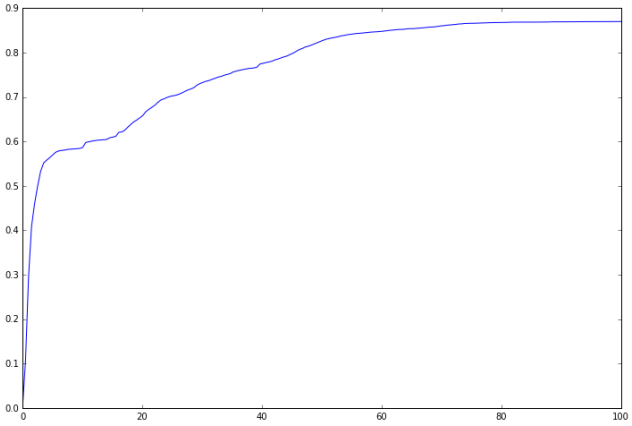


Figure 3. Precision curve for CellTrack using DeepCell features on 200 cells from the 20150812-Pos5 dataset. X axis is threshold in pixels, Y axis is the percentage of predicted points within threshold pixels of ground truth.

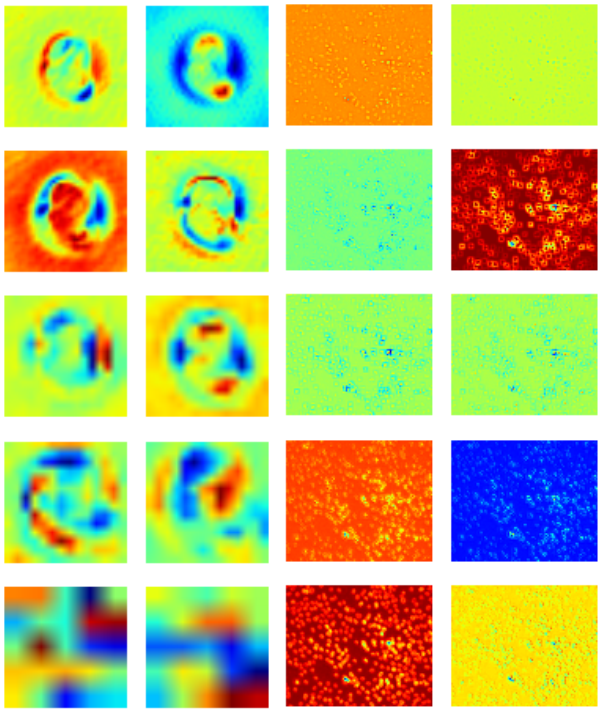

Figure 4. Comparison of learned filters between RecogNet and DeepCell. The left two columns are RecogNet, right are DeepCell. Levels one through five are top to bottom. Two filters were chosen per Net/Layer combination.

trained on CNN features, cannot distinguish cell nuclei. We visualized the deep filters in DeepCell, finding that they still preserved relatively intact spatial information, rather than the abstract patterns normally seen in CNNs (Figure 4). We hypothesised that because DeepCell only needs to predict whether a given pixel is nucleus or background, optimisation does not select for discrimination between nuclei. It

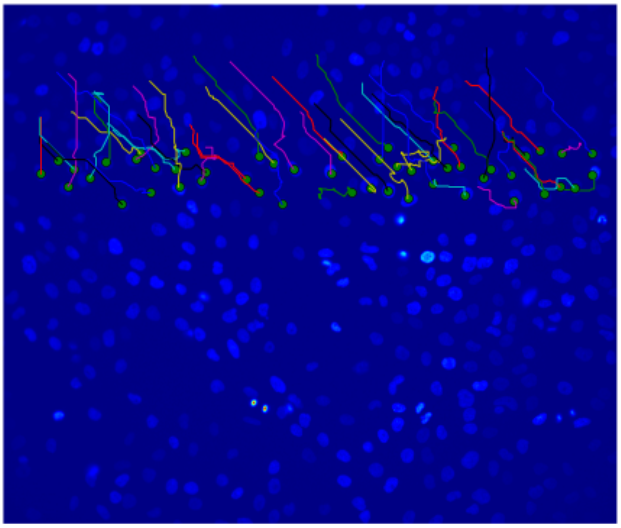

Figure 5. Cell trajectories predicted by RecogNet. Note the persistent bias toward the top-left.

was this hypothesis that led to the design of RecogNet.

RecogNet is significantly less effective at tracking nuclei, with precision below 10%. An analysis of the predicted trajectories on both datasets shows a persistent tracking bias toward the origin (Figure 5). Moreover, the correlation filters maintain a persistent positive at the dead-center of the search window, in addition to appearing insensitive to other content in the window. Optimising the filter weight values, learning rate, and layers used for correlation has not changed this behaviour, but it may be that weights from the network need normalisation, or that another underlying problem exists. We believe that RecogNet holds better potential for improving accuracy than DeepCell, but a larger rewrite of the CellTrack framework may be required.

RecogNet also has the potential to solve the problem of identifying cell division, so that tracking can retarget to follow daughter cells, as well as prediction of other metadata states: cell death, overlaps, etc. Providing this data would allow downstream analysis to take the states into account, such as ignoring signal fluorescence channels after a cell dies or where there is interference. The accuracy reached by the classification net, which included identifying dividing cells, indicates that the network possesses the representational power match these states. At each step of cell tracking, the "cell metadata" classification could be computed alongside the feature maps used for the tracking itself.

There are other trade-offs to the choice between DeepCell and RecogNet. DeepCell's stored feature maps save the computational effort of recalculating features at each (and, during tracking, often arbitrary) positions, but require a large amount of disk: on the order of 1 GB per frame for the full stack of layers. For our reference time series of 120 frames, storage and the overhead of loading approximately

100GB of data is challenging. This overhead makes DeepCell slower than RecogNet when tracking small numbers of cells, but when tracking large numbers DeepCell has the advantage as RecogNet computes features for every cell at every frame. Room for efficiency improvements does exist, particularly bunding all search windows for a given frame so that they can be processed on the GPU at once.

Another possibility for improving tracking accuracy would be to combine hierarchical tracking approach with other methods. For instance, the weighted correlation filter at each search window could provide a cost to a linear-assignment problem, augmenting its ability to accurately track cells.

## 8. Conclusion

While the hierarchical tracking approach shows promise, this implementation has further to come in terms of performance and robustness. Accurate hands-off cell tracking remains a challenge, but modern machine-learning approaches will certainly have something to offer as technology improves. One limitation is the lack of high-quality annotated data to use for training and validation, but reasonable alternatives can be found using a combination of automation and manual data cleanup. Overall, we hope to continue this work, improving both accuracy and classification speed.

## References


[1] D. S. Bolme, J. R. Beveridge, B. a. Draper, and Y. M. L. Y. M. Lui. Visual object tracking using adaptive correlation filters. *Computer Vision and Pattern Recognition CVPR 2010 IEEE Conference on*, pages 2544–2550, 2010.

[2] X. Chen, X. Zhou, and S. T. C. Wong. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. *IEEE transactions on bio-medical engineering*, 53(4):762–6, apr 2006.

[3] A. Dundar, J. Bates, C. Farabet, and E. Culurciello. Tracking with deep neural networks. In *2013 47th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–5. IEEE, mar 2013.

[4] D. Gauthier and M. Levine. Live cell image segmentation. *IEEE Transactions on Biomedical Engineering*, 42(1):1–12, 1995.

[5] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-Speed Tracking with Kernelized Correlation Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, apr 2014.

[6] S. Hong, T. You, S. Kwak, and B. Han. Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network. feb 2015.

[7] K. Jaqaman, D. Loerke, M. Mettlen, H. Kuwata, S. Grinstein, S. L. Schmid, and G. Danuser. Robust single-particle tracking in live-cell time-lapse sequences. *Nature methods*, 5(8):695–702, aug 2008.

[8] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang. Hierarchical Convolutional Features for Visual Tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3074–3082, 2015.

[9] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. W. Balak, P. Karas, T. Bolcková, M. Streitová, C. Carthel, S. Coraluppi, N. Harder, K. Rohr, K. E. G. Magnusson, J. Jaldén, H. M. Blau, O. Dzyubachyk, P. Kížek, G. M. Hagen, D. Pastor-Escuredo, D. Jimenez-Carretero, M. J. Ledesma-Carbayo, A. Muñoz-Barrutia, E. Meijering, M. Kozubek, and C. Ortiz-de Solorzano. A benchmark for comparison of cell tracking algorithms. *Bioinformatics (Oxford, England)*, 30(11):1609–17, jun 2014.

[10] A. Sacan, H. Ferhatosmanoglu, and H. Coskun. CellTrack: an open-source software for cell tracking and motility analysis. *Bioinformatics (Oxford, England)*, 24(14):1647–9, jul 2008.

[11] D. van Valen and M. W. Covert. DeepCell. *unpublished*, 2016.

[12] Y. Wu, J. Lim, and M.-H. Yang. Online Object Tracking: A Benchmark. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2411–2418. IEEE, jun 2013.