Detecting and Tracking Cells using Network Flow Programming

Engin Türetken[1,2 * †] Xinchao Wang[1 †] Carlos Becker[1] Pascal Fua[1]

[1]Computer Vision Laboratory (EPFL), CH-1015 Lausanne, Switzerland
[2]Swiss Center for Electronic and Microtechnology (CSEM), Switzerland

Abstract

We propose a novel approach to automatically detecting and tracking cell populations in time-lapse images. Unlike earlier ones that rely on linking a predetermined and potentially under-complete set of detections, we generate an over-complete set of competing detection hypotheses. We then perform detection and tracking simultaneously by solving an integer program to find an optimal and consistent subset. This eliminates the need for heuristics to handle missed detections due to occlusions and complex morphology.

We demonstrate the effectiveness of our approach on a range of challenging image sequences consisting of clumped cells and show that it outperforms state-of-the-art techniques.

1. Introduction

Detecting and tracking cells over time is key to understanding cellular processes including division (mitosis), migration, and death (apoptosis). Modern microscopes produce vast image streams making manual tracking tedious and impractical. High-throughput automated systems are therefore increasingly in demand and several cell tracking competitions have recently been organized to attract Computer Vision researchers' interest and speed up progress [26, 34]. These competitions have shown that state-of-the-art methods are still error-prone due to occlusions, imaging noise, and complex cell morphology.

Cell tracking is an instance of the more generic multi-target tracking problem with the additional difficulties that cells, unlike for example pedestrians, can either divide or wither away and disappear in mid-sequence. In its generic form, the problem is often formulated as a two-step process that involves first detecting potential objects in individual frames and then linking these detections into complete trajectories. This approach is attractive because spurious detections, such as false positives due to imaging noise and artifacts, can be eliminated by imposing temporal consistency across many frames, which is more difficult to do in recursive approaches.

Many of the most successful algorithms to both people [12, 3, 39, 38] and cell tracking [21, 18, 32] follow this two-step approach by first running an object detector on each frame independently and building a graph whose nodes are the detections and edges connect pairs of them. They then find a subgraph that represents object trajectories by considering the whole graph at once. However, to handle missed detections, these methods often rely on heuristic procedures that offer no guarantee of optimality. This is of particular concern for cell tracking because detections are often unreliable due to the complex morphology of cell populations. For example, in Fig. 1, groups of cells that appear clumped together in some frames can only be told apart when considering the sequence as a whole, which current approaches to cell tracking rarely do.

In this paper, we therefore address the detection and tracking problems simultaneously by casting them as a network flow problem on an over-complete graph of potentially conflicting hypotheses. These hypotheses are competing explanations of the initial segmentations, in which the cells are often under-segmented, meaning that a single foreground region can correspond to many cells as shown in Fig. 2. Instead of selecting a single hypothesis for each such region, we build a spatio-temporal graph over all of them as depicted by Fig. 3, and find the globally optimal explanation by solving a Linear Integer Program (IP) within a small tolerance. This results in large graphs but eliminates the need for using heuristics to handle missed detections, as required by recent approaches [18, 32] that also rely on integer programming. Unlike these formulations, which contain multiple variable types and many constraints, we use only a single set of variables and three linear constraints that govern division, appearance, disappearance of cells and exclusion of conflicting hypotheses.

We show that this improves trajectories and yields superior detection performance on various datasets compared to the recent approaches of [32, 1, 24], which includes the technique that performed best on the above-mentioned cell-tracking challenges [26, 34].

*This work was supported in part by the Swiss National Science Foundation.
†Authors contributed equally.

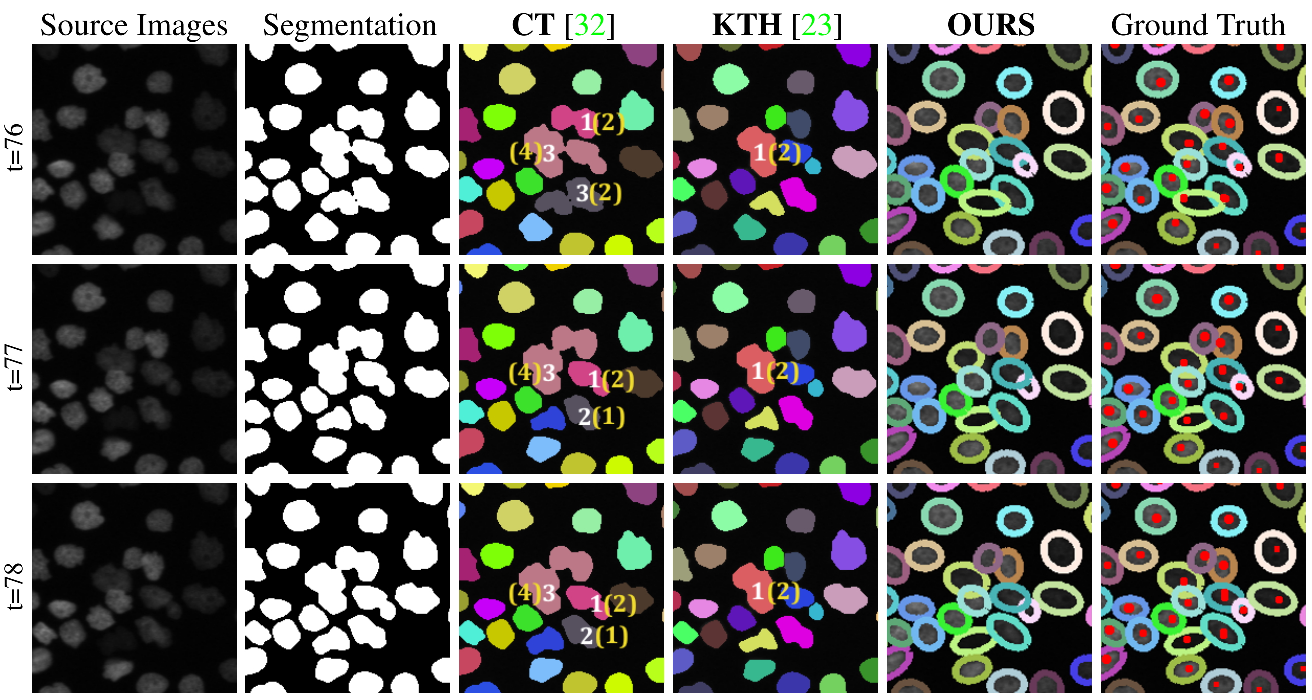


Figure 1: Three images from a typical sequence; the original segmentations produced by a pixel-based classifier; the results of [32](**CT**), [23](**KTH**), and our method (**OURS**); the manually annotated tracking ground truth in red dots and the optimal ellipse-tracks obtained using this ground truth. The track identities are encoded in colors. Our approach correctly tracks the cells in spite of long-term segmentation failures, and produce results that are very similar to the ground truth. In the **KTH** and **CT** columns, the white-colored numbers indicate the tracker-inferred numbers of cells that are contained by the segments beneath them, and the yellow-colored numbers show the ground truth. Best viewed in color.

2. Related Work

Current tracking approaches can be divided into *Tracking by Model Evolution* and *Tracking by Detection* [26]. We briefly discuss state-of-the-art representatives of these two classes below and refer the interested reader to the much more complete recent surveys [27, 26].

2.1. Tracking by Model Evolution

Most algorithms in this class simultaneously track and detect objects greedily from frame to frame. This means extrapolating results obtained in earlier frames to process the current one, which can be done at a low computational cost and is therefore fast in practice. Such methods have attracted attention both in the cell tracking field [9, 8, 7, 25] as well as in the more general object tracking one [41, 16, 28, 40, 11]. Common techniques in the cell tracking category involve evolving appearance or geometry models from one frame to the next, typically done using active contours [9, 8, 7, 25] or Gaussian Mixture Models [1]. Though these methods are attractive and mathematically sound, performance suffers from the fact that they only consider a restricted temporal context and therefore cannot guarantee consistency over a whole sequence.

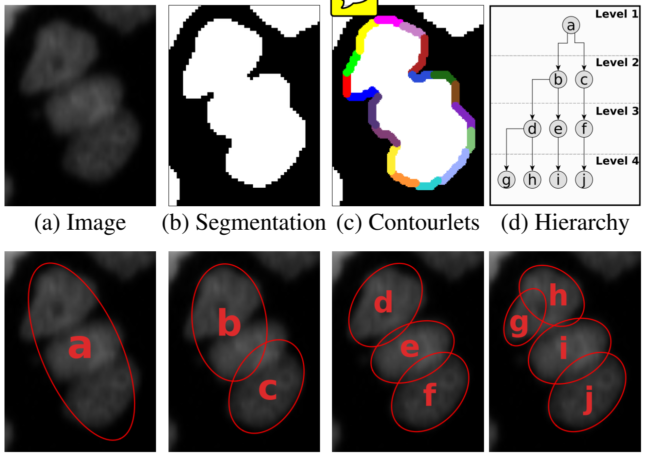
This limitation has been addressed by more global active contour methods [20, 29] that consider the whole spatio-temporal domain to segment the cells and recover parts of their trajectories. Although this provides improved robustness at the cost of increased computational burden, these approaches do not provide global optimality guarantees either.

2.2. Tracking by Detection

Approaches in this class have proved successful at both people [12, 5, 3, 39, 38] and cell tracking [21, 18, 32].

They involve first detecting the target objects in individual frames and then linking these detections to produce full trajectories. This is typically computationally more expensive than Tracking by Model Evolution. However, it also tends to be more robust because trajectories are computed by minimizing a global objective function that enforces consistency of appearance, disappearance, and division over time. This can be seen in the benchmark of [26] in which these methods tended to dominate.

One way to perform tracking-by-detection is to reason in the full spatio-temporal grid formed by stacking up all the spatial locations over time [4, 2, 30]. In the case of non-dividing objects such as pedestrians, this can be done



(a) Image (b) Segmentation (c) Contourlets (d) Hierarchy

(e) Level 1 (f) Level 2 (g) Level 3 (h) Level 4

Figure 2: Hierarchy of detection hypotheses. (a) An image region containing three HeLa cells clumped together. (b) Applying a pixel classifier results in under-segmentation, in which the three cells appear as a single connected component. (c) Automatically extracted contourlets for this component. Each one is overlaid in a different color. They are used to fit ellipses using a hierarchical agglomerative clustering algorithm. (d) The resulting hierarchy of 10 hypotheses. We show only the first four levels for simplicity. (e-h) Individual levels with one, two, three and four hypotheses. Best viewed in color.

in polynomial time [36]. In spite of the size of the graphs involved, this had made this approach very competitive and real-time implementations have been demonstrated [4]. However, the fact that cells can divide makes the resulting optimization problem NP-Hard, which is mainly why such approaches have not been explored in the cell tracking field. Instead, practical tracking-by-detection approaches typically rely on a small number of strong detections at individual frames to later form complete cell trajectories.

Therefore, the literature in this field has recently focused on solving two main challenges that are specific to cell-tracking, *(a)* cells can divide or die and disappear, and *(b)*, there is no guarantee that individual cells will be detected as separate entities in any given frame because two or more cells can clump together, producing under-segmentation errors, as seen in Fig. 2. In the following, we discuss recent efforts to overcome these key challenges.

Division and disappearance. While rule-based approaches have been used to handle division and disappearance, most recent ones formulate tracking as a global integer programming problem [18, 32]. This makes it possible to use priors for cell movement, division and disappearance between adjacent frames. One notable exception is the method of [23], which scored highest in one of the benchmarks [26], which sequentially adds trajectories to a cell lineage tree, using the Viterbi algorithm. Motion, division, and disappearance are encoded through a scoring function that quantifies how well the lineage tree explains the data.

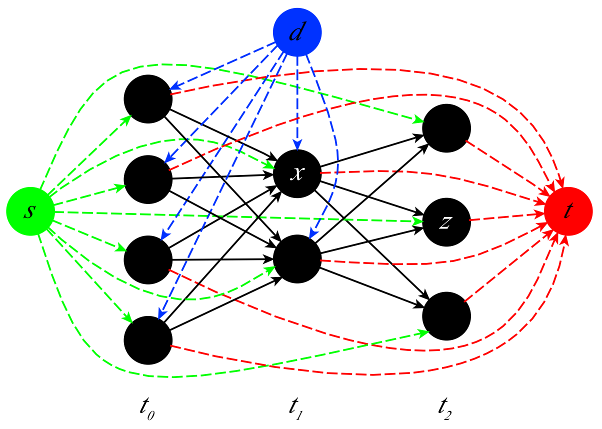
Clumped cells. Similarly, many heuristics have been proposed to solve the problem of clumped cells that cannot easily be told apart. One is to assume that clumped cells are unlikely to happen for a specific modality or that they do not pose a problem for the tracker [21, 18, 23]. While this is appropriate in some cases, it is clearly invalid for certain modalities such as the one shown in Fig. 1. A possible solution is to use semi-automated techniques [15] to interactively correct errors made by automated approaches, but this is time consuming for modalities that produce many clumped cells. Other heuristics involve splitting segmentations using the Radon and watershed transforms [9, 26]. Unfortunately, they are still relatively prone to over- and under-segmentation that complicate the tracking step. An ingenious approach is that of [32], which breaks the tracking step into two stages. It first finds trajectories by treating segmentations in each frame as clumps of one or more cells, with the exact number being initially unknown, and then uses a factor graph to resolve this ambiguity. However, because these two steps are performed independently, optimality is not guaranteed.

In the more general case of object detection, tracking groups of objects has been considered as a multiple-hypothesis problem, since there exist many plausible hypotheses to explain what is observed in individual frames. Though multiple-hypothesis tracking has been applied to people tracking [14, 19, 37, 22], to the best of our knowledge, it has not been explored in the context of cell tracking using integer programming.

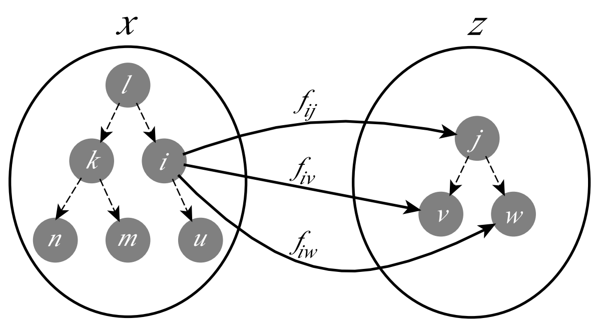
By contrast, our approach dispenses with such heuristics by allowing multiple competing interpretations of the data, choosing the globally optimal one among them by solving an integer program that enforces consistency over all frames.

3. Method

Our approach involves building a spatio-temporal graph of conflicting hypotheses in individual frames, and then finding the most likely trajectories in it. More specifically, we first produce a binary image of the underlying cell populations using a classifier trained on a few hand-annotated segmentations. For each connected component, we produce multiple detection hypotheses by hierarchically fitting varying number of ellipses to it. This results in a directed graph, such as the one depicted by Fig. 3. Its nodes are individual ellipses and its edges connect nearby ones in consecutive frames. Full trajectories can then be obtained by solving an integer program with a small number of constraints that exclude incompatible hypotheses and enforce consistency



(a)



(b)

Figure 3: Spatio-temporal graph of hypotheses for 3 consecutive time frames. (a) Each black circle is a hypervertex corresponding to a connected component of the segmentation and is connected to neighboring ones at the next time step. The special vertices s (source, in green), t (sink, in red) and d (division, in blue) allow respectively for cell appearance, disappearance and division. (b) Each hypervertex, such as x and z, contains a hierarchical set of hypotheses (vertices) such as those depicted by Fig. 2. These hypotheses are shown as gray circles and connected to nearby ones in the following frame via directed edges. We only show three of these edges to avoid clutter. Best viewed in color.

while allowing for cell-division, migration and death.

In the following, we first describe our approach to segmenting cell images and building hypotheses graphs from them. We then formulate the simultaneous detection and tracking problem on these graphs as a constrained network flow programming problem, and discuss how we compute the various energy terms of its objective function.

3.1. Building Hierarchy Graphs

Our algorithm, like those of [21, 18, 32], starts by segmenting cells using local image features. To this end, we first train the binary random forest pixel classifier of [33] for each evaluation dataset on a few partially annotated images. We use four different types of low-level features: pixel intensities, gradient and hessian values, and difference of Gaussians, all of which are computed by first Gaussian smoothing the input image with a range of sigma values.

Applying the resulting classifier to the full image sequences results in segmentations which often contain groups of clumped cells, such as the ones shown in Fig. 1. Therefore, each connected component of the segmentation potentially contains an a priori unknown number of cells.

We produce a hierarchy of conflicting detection hypotheses for each such component by fitting a varying number of ellipses to its contours. More specifically, we first identify all the contour points that are local maxima of curvature magnitude. This is done iteratively by selecting the maximum curvature points and suppressing their local neighborhoods. We then break the contour into short segments at these points, which yields a number of contourlets as shown in Fig 2(c). We cluster them in a hierarchical agglomerative fashion and fit ellipses to each resulting cluster using the non-iterative least squares approach of [6]. In all our experiments, we set the size of the suppression neighborhood to seven pixels because this is the minimum number of points required to reliably fit an ellipse using this approach.

Let C denote the set of all contourlet clusters for a connected component. Given a pair of clusters $C_i \in C$ and $C_j \in C$, we define their distance to be

$$\left[\sum_{C_l \in \{C_i \cup C_j\}} h(C_l, e) + \sum_{C_l \in C \setminus \{C_i \cup C_j\}} g(C_l, e)\right] c(e) \sqrt{\frac{1}{1 + ec^2(e)}}, \quad (1)$$

where e is the ellipse obtained by fitting to the points of $C_i \cup C_j$, and $c(e)$ and $ec(e)$ are its circumference and eccentricity respectively. $h(C_l, e)$ denotes the Hausdorff distance between the points of C_l and the ellipse e. The function $g(C_l, e)$ is defined in a similar way but without considering the points of C_l that are outside e.

The first term in the product captures image evidence along the contours of the entire connected component while the last two ones act as a shape regularizer to prevent implausible ellipse geometries from appearing in the solution. We use this distance measure to compute an ellipse hierarchy, such as the one of Fig. 2, for every connected component in the temporal sequence.

Given these over-complete hierarchy of detections, we then build a graph, whose vertices are the ellipses and the edges link pairs of them that belong to two spatially close connected components in consecutive frames. This is similar in spirit to recent graph-based approaches [18, 32], but with the key difference that our graphs have a hierarchical dimension that allows for finding the globally optimal detection hypotheses and trajectories in a single shot.

3.2. Network Flow Formalism

The procedure described above yields a directed graph $G' = (V', E')$, which we then augment with three distinguished vertices; namely the source s, the sink t and the division d as depicted by Fig. 3. We connect these three vertices to every other vertex in G' to allow accounting for cell appearance, disappearance and division in mid-sequence.

Let $G = (V, E)$ be the resulting graph obtained after the augmentation. We define a binary flow variable f_{ij} for each edge $e_{ij} \in E$ to indicate the presence of any one of the following cellular events:

- Cell migration from vertex i to vertex j,
- Appearance at vertex j, if $i = s$,
- Division at vertex j, if $i = d$,
- Disappearance at vertex i, if $j = t$.

Let $\mathbf{f}$ be the set of all f_{ij} flow variables and $\mathbf{F} = \{F_{ij}\}$ be the set of all corresponding hidden variables. Given an image sequence $\mathbf{I} = (\mathbf{I}^1, \ldots, \mathbf{I}^T)$ with T temporal frames and the corresponding graph G, we look for the optimal trajectories $\mathbf{f}^*$ in G as the solution of

$$\begin{aligned}
\mathbf{f}^* &= \operatorname*{argmax}_{\mathbf{f} \in \mathcal{F}} P(\mathbf{F} = \mathbf{f} \mid \mathbf{I}) && (2)\\
&\approx \operatorname*{argmax}_{\mathbf{f} \in \mathcal{F}} \prod_{e_{ij} \in E} P(F_{ij} = f_{ij} \mid \mathbf{I}^{t(i)}, \mathbf{I}^{t(j)}) && (3)\\
&= \operatorname*{argmax}_{\mathbf{f} \in \mathcal{F}} \prod_{e_{ij} \in E} P(F_{ij} = 1 \mid \mathbf{I}^{t(i)}, \mathbf{I}^{t(j)})^{f_{ij}} \times \\
&\qquad P(F_{ij} = 0 \mid \mathbf{I}^{t(i)}, \mathbf{I}^{t(j)})^{(1-f_{ij})} && (4)\\
&= \operatorname*{argmax}_{\mathbf{f} \in \mathcal{F}} \sum_{e_{ij} \in E} \log\left(\frac{P(F_{ij} = 1 \mid \mathbf{I}^{t(i)}, \mathbf{I}^{t(j)})}{P(F_{ij} = 0 \mid \mathbf{I}^{t(i)}, \mathbf{I}^{t(j)})}\right) f_{ij} && (5)\\
&= \operatorname*{argmax}_{\mathbf{f} \in \mathcal{F}} \sum_{e_{ij} \in E'} \log\left(\frac{\rho_{ij}}{1-\rho_{ij}}\right) f_{ij} + \sum_{j \in V} \log\left(\frac{\rho_a}{1-\rho_a}\right) f_{sj} \\
&\quad + \sum_{i \in V} \log\left(\frac{\rho_d}{1-\rho_d}\right) f_{it} + \sum_{j \in V} \log\left(\frac{\rho_j}{1-\rho_j}\right) f_{dj}\ , && (6)
\end{aligned}$$

where $\mathbf{I}^{t(i)}$ is the temporal frame containing vertex i, and $\mathcal{F}$ denote the set of all feasible cell trajectories, which satisfy linear constraints. In Eq. 3, we assume that the flow variables F_{ij} are conditionally independent given the evidence from consecutive frame pairs. Eqs. 4 and 5 are obtained by using the fact that the flow variables are binary and by taking the logarithm of the product. Finally, in Eq. 6, we split the sum into four parts corresponding to the four events mentioned above.

The appearance and disappearance probabilities, ρ_a and ρ_d, are computed simply by finding the relative frequency of these events in the ground truth cell lineages of the training sequences. On the other hand, the migration and the division probabilities, ρ_{ij} and ρ_j, are obtained using a classification approach as described in the next section.

We define three sets of linear constraints to model cell behavior and exclude conflicting detection hypotheses from the solution, which we describe in the following.

Conservation of Flow: We require the sum of the flows incoming to a vertex to be equal to the sum of the outgoing flows. This allows for all the four cellular events while incurring their respective costs given in Eq. 6.

$$\sum_{e_{ij} \in E'} f_{ij} + f_{sj} + f_{dj} = \sum_{e_{jk} \in E'} f_{jk} + f_{jt}\ , \qquad \forall j \in V'\ . \qquad (7)$$

Prerequisite for Division: We allow division to take place at a vertex $j \in V'$ only if there is a cell at that location. We write this as

$$\sum_{e_{ij} \in E'} f_{ij} + f_{sj} \geq f_{dj}\ , \qquad \forall j \in V'\ . \qquad (8)$$

Exclusion of Conflicting Hypotheses: Given a hierarchy tree of detections, we define an exclusion set S_l for each terminal vertex $l \in V'$ of this tree. For instance, in the example of Fig. 2(d), the four exclusion sets are $\{a, b, d, g\}$, $\{a, b, d, h\}$, $\{a, b, e, i\}$ and $\{a, c, f, j\}$. Let S be the collection of all such sets for all the connected components in the sequence. We disallow more than one vertex from each set to appear in the solution, and express this as

$$\sum_{\substack{j \in S_l,\\ e_{ij} \in E'}} f_{ij} + \sum_{j \in S_l} f_{sj} \leq 1, \qquad \forall S_l \in S\ . \qquad (9)$$

We solve the resulting integer programs within an optimality tolerance of $1e^{-3}$ using the branch-and-cut algorithm implemented in the Gurobi optimization library [13].

3.3. Cell Migration and Division Classifiers

Given two adjacent vertices $i, j \in V'$, and their associated ellipses e_i and e_j in consecutive time frames, we train a classifier to estimate the likelihood that both belong to the same cell. More specifically, we use a Gradient Boosted Tree (GBT) classifier [10] to learn a function $\varphi_{\text{migr}}(e_i, e_j) \in \mathbb{R}$, based on both appearance and geometry features including the distance between the ellipses, their eccentricities and degree of overlap, hierarchical fitting errors and ray features. Once $\varphi_{\text{migr}}(e_i, e_j)$ is learned, we apply Platt scaling [31] to compute the migration probability ρ_{ij} that the two ellipses belong to the same cell, and plug it into Eq. 6.

Similarly, the likelihood of ellipse e_j at time t dividing into two ellipses e_k and e_l at $t + 1$ is learned with another GBT classifier, trained on features such as the orientation and size differences among the ellipses. We present a detailed list of both the migration and division features in the supplementary material.

For prediction, we compute the division score for ellipse e_j at time t as

$$\varphi_{\text{div}}(e_j) = \max_{\substack{e_{jk} \in E',\, e_{jl} \in E': \\ k \neq l}} \varphi_{\text{div}}(e_j, e_k, e_l) \,, \tag{10}$$

where $\varphi_{\text{div}}(\cdot,\cdot,\cdot)$ is the scoring function learned by the classifier, and (e_k, e_l) is a pair of ellipses corresponding to two potential daughter cells at time $t+1$. We obtain the division probability ρ_j of Eq. 6 from $\varphi_{\text{div}}(e_j)$, again using Platt scaling.

4. Experiments

In this section, we first introduce the datasets and state-of-the-art baseline methods we use for evaluation purposes. We then demonstrate that our approach significantly outperforms these baselines, especially when the cells divide or are not well separated in the initial segmentations.

4.1. Test Sequences

We used 10 image sequences from three datasets of the cell tracking challenge [34]. They involve multiple cells that migrate, appear, disappear, and divide. Difficulties arise from low contrast with the background, complex cell morphology, and significant mutual overlap. We used the leave-one-out training and testing scheme within each dataset to train the classifiers of Section 3.3 as well as to learn the appearance and disappearance probabilities of Eq. 6.

- **HeLa Dataset:** It comprises two 92-frame sequences from the MitoCheck consortium. Cell divisions are frequent, which produces a dense population with severe occlusions.
- **SIM Dataset:** It comprises six 50- to 100-frame sequences. They simulate migrating and dividing nuclei on a flat surface.
- **GOWT Dataset:** It comprises two 92-frame sequences of mouse stem cells. Their appearance varies widely and some have low contrast against a noisy background.

4.2. Baselines

We compared our algorithm (**OURS**) against the following three state-of-the-art methods

- **Gaussian Mixture-based Tracker (GMM) [1]:** We run the Gaussian Mixture Models approach of [1], originally designed to track cell nuclei, whose code is publicly available. We manually tuned its parameters to the ones that yield the best results on each sequence.
- **KTH Cell Tracker (KTH) [23]:** The code is publicly available and has been reported to perform best in the Cell Tracking Challenge [34, 26]. We used the parameter settings optimized for each dataset and provided in the software package. By contrast, our own algorithm does not require any user-defined parameters.
- **Conservation Tracking (CT) [32]:** We run Ilastik V1.1.3 [35] and enabled the C-T functionality that implements the method of [32]. We used the default parameters provided with the tool to handle appearance, disappearance, division and transition weights. The **CT** algorithm, like ours, requires initial segmentations such as the ones shown in the second column of Fig.1. We used the same segmentations for both algorithms. We trained the division and the segment count classifiers of **CT** separately for each dataset on manually labeled cells.

To demonstrate the importance of individual components of our approach, we also ran simplified versions of **OURS** with various features turned off:

- **Classifier Only (OURS-CL):** We threshold the output of our migration and division classifiers at a probability of 0.5 and return the resulting ellipse detections.
- **Best Hierarchy Only (OURS-BH):** For each ellipse hierarchy tree, we only keep the level that yields the minimum fitting error, which we define in the supplementary material. We then run our IP optimization on the resulting graphs, which are smaller than the ones we normally use.
- **Linear Programming Relaxation (OURS-LP):** We relax the integrality constraint on the flow variables and solve the optimization problem of Eq. 6 using linear programming. We then round the resulting fractional flows to the nearest integer to produce the final solution.
- **No Conflict Set Constraint (OURS-NC)**: We remove the conflict set constraints of Eq. 9 and solve the resulting integer program as before.
- **Fixed Division Cost (OURS-FD)**: We set the division probability to a constant p_d, which we compute by finding the relative frequency of the division event in the training sequences.

4.3. Evaluation Metrics

We use precision, recall and the F-Measure, defined as the harmonic mean of precision and recall, to quantify the

		Division			Detection			Migration			MOTA
		Rec	Pre.	F-M.	Rec.	Pre.	F-M.	Rec.	Pre.	F-M.	
HeLa-1	**GMM**	0.56	0.43	0.48	N/A	N/A	N/A	0.92	0.98	0.95	0.82
	KTH	0.65	0.72	0.68	N/A	N/A	N/A	0.95	**0.99**	0.97	0.91
	CT	0.74	**0.79**	0.77	0.56	**0.89**	0.69	0.94	**0.99**	0.97	N/A
	OURS	**0.92**	**0.79**	**0.85**	**0.96**	0.83	**0.89**	**0.97**	**0.99**	**0.98**	**0.94**
HeLa-2	**GMM**	0.40	0.18	0.24	N/A	N/A	N/A	0.95	0.98	0.97	0.43
	KTH	0.65	0.72	0.68	N/A	N/A	N/A	0.94	**0.99**	**0.97**	**0.90**
	CT	0.76	0.81	0.78	0.73	0.63	0.67	0.94	**0.99**	0.96	N/A
	OURS	**0.86**	**0.83**	**0.84**	**0.86**	**0.78**	**0.82**	**0.96**	**0.99**	**0.97**	**0.90**
GOWT-2	**GMM**	0.0	0.0	0.0	N/A	N/A	N/A	0.16	0.79	0.26	0.02
	KTH	0.0	N/A	0.0	N/A	N/A	N/A	0.94	**1.0**	0.97	0.94
	CT	**1.0**	0.17	0.29	**1.0**	0.02	0.03	0.95	**1.0**	0.97	N/A
	OURS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.96**	**1.0**	**0.98**	**0.96**
SIM-4	**GMM**	0.25	0.33	0.29	N/A	N/A	N/A	0.91	0.94	0.92	0.81
	KTH	0.75	0.75	0.75	N/A	N/A	N/A	0.97	0.99	0.98	**0.96**
	CT	0.75	0.60	0.67	0.75	0.68	0.72	0.86	0.97	0.92	N/A
	OURS	**1.0**	**0.80**	**0.89**	**1.0**	**0.79**	**0.88**	**0.98**	**1.0**	**0.99**	**0.96**

Table 1: Comparison of our algorithm against state-of-the-art cell trackers. It yields a significant improvement on the division and detection accuracies and performs either on par or slightly better on the migration and MOTA scores.

algorithms' ability to detect cell division, detection, and migration events. We also use the more global Multiple Object Tracking Accuracy (MOTA) metric of [17] to evaluate the overall tracking accuracy for each sequence.

We follow the same evaluation methodology described in [32], which uses the connected components of the initial segmentations to compute the division, detection, and migration accuracies. More specifically, we consider a cell migration event to be successfully detected if the two connected components of the cell at t and $t+1$ are correctly identified. Similarly, we consider a division event to be successfully detected if it occurs at the correct time instant with the connected components of the parent and both daughter cells correctly determined. Finally, the detection event is said to be successfully identified if an algorithm infers the right number of cells within a connected component.

Unlike these measures, the MOTA metric is defined on individual cell tracks rather than connected components that potentially contain multiple clumped cells. It therefore provides a global picture of tracking performance.

4.4. Results

4.4.1 Comparing against the Baselines

We ran our algorithm and the baselines discussed above on all the test sequences introduced in Section 4.1. Table 1 summarizes the results for a representative subset and the remainder can be found in the supplementary material.

Some numbers are missing because the publicly-available implementation of **CT** [32] does not provide the identities of individual cells in under-segmentation cases. We therefore cannot extract the complete tracks required to compute the MOTA scores. Similarly, the KTH tracker [23] takes raw images as input and does not use the initial segmentations. Therefore, we cannot compute its accuracy for the detection event.

Table 1 shows that our tracker consistently yields a significant improvement on the division and detection events. Even on the migration events for which the baselines already perform very well, we do slightly better. However, because the division and detection events are rare compared to migrations, the significant improvements on the first two events only have a small impact on the MOTA scores.

An interesting case is that of **GMM**, which performs poorly on all three events. This could be explained by the fact that **GMM** relies on a simple hand-designed appearance and geometry model to detect individual cells. On the other hand, **KTH** assumes a more flexible appearance model but requires a higher number of parameters to be tuned for each sequence. These differences help explain why it performs better than **GMM**. Finally, **CT** employs a classification approach to segment the cells, resulting in more accurate detections. However, none of the baselines handle the under-segmentations in a global manner, and are outperformed by our approach, particularly for division and detection events.

Our tracker runs relatively fast even though we use a large number of variables. In the SIM-4 case, the number of flow variables is around 1.1 million but the integer programming optimization takes only 2 seconds. In the HeLa-2 case, the number of variables is around 8 million and the optimization takes about 360 seconds.

		Division			Detection			Migration			MOTA
		Rec	Pre.	F-M.	Rec.	Pre.	F-M.	Rec.	Pre.	F-M.	
HeLa-1	**OURS-CL**	0.95	0.06	0.11	N/A	N/A	N/A	0.98	0.47	0.64	N/A
	OURS-NC	0.48	0.14	0.22	0.0	0.0	0.0	0.96	0.93	0.94	-0.61
	OURS-FD	0.78	0.81	0.80	0.96	0.85	0.90	0.97	0.99	0.98	0.93
	OURS-BH	0.88	0.73	0.80	0.81	0.87	0.84	0.97	0.99	0.98	0.94
	OURS-LP	0.92	0.80	0.86	0.95	0.81	0.87	0.97	0.99	0.98	0.93
	OURS	0.92	0.79	0.85	0.96	0.83	0.89	0.97	0.99	0.98	0.94
HeLa-2	**OURS-CL**	0.91	0.10	0.18	N/A	N/A	N/A	0.92	0.60	0.72	N/A
	OURS-NC	0.73	0.31	0.43	0.06	0.02	0.02	0.95	0.96	0.95	0.35
	OURS-FD	0.77	0.82	0.80	0.84	0.78	0.81	0.96	0.98	0.97	0.90
	OURS-BH	0.84	0.77	0.81	0.78	0.77	0.77	0.95	0.99	0.97	0.90
	OURS-LP	0.85	0.83	0.84	0.84	0.78	0.81	0.95	0.99	0.97	0.90
	OURS	0.86	0.83	0.84	0.86	0.78	0.82	0.96	0.99	0.97	0.90
GOWT-2	**OURS-CL**	0.0	0.0	0.0	N/A	N/A	N/A	0.94	0.94	0.94	N/A
	OURS-NC	1.0	0.20	0.33	1.0	0.02	0.03	0.96	1.0	0.98	0.93
	OURS-FD	1.0	0.25	0.40	1.0	1.0	1.0	0.96	1.0	0.98	0.96
	OURS-BH	0.0	N/A	0.0	1.0	1.0	1.0	0.91	1.0	0.95	0.91
	OURS-LP	1.0	0.50	0.67	1.0	0.50	0.67	0.96	1.0	0.98	0.96
	OURS	1.0	1.0	1.0	1.0	1.0	1.0	0.96	1.0	0.98	0.96
SIM-4	**OURS-CL**	0.75	0.27	0.40	N/A	N/A	N/A	0.92	0.56	0.70	N/A
	OURS-NC	0.75	0.21	0.33	0.37	0.19	0.25	0.97	0.98	0.98	0.58
	OURS-FD	0.75	0.75	0.75	0.98	0.78	0.87	0.98	1.0	0.99	0.95
	OURS-BH	1.0	0.80	0.89	1.0	0.78	0.87	0.98	1.0	0.99	0.96
	OURS-LP	1.0	0.80	0.89	1.0	0.79	0.88	0.98	1.0	0.99	0.96
	OURS	1.0	0.80	0.89	1.0	0.79	0.88	0.98	1.0	0.99	0.96

Table 2: Tracking results with various features turned off. **OURS-CL** does not impose temporal consistency and suffers from low precision. **OURS-NC** imposes temporal consistency in the optimization but allows multiple conflicting hypotheses to appear in the solution, which yields a low precision. **OURS-FD** eliminates competing hypotheses but uses a fixed cost for the division event. **OURS-BH** performs non-maxima suppression on the hierarchical dimension and hence suffers from mis-detection errors. **OURS-LP** yields fractional flows and the rounding stage eliminates some cell tracks, which leads to a slight drop in performance. With all its features turned on, **OURS** achieves the best overall performance.

4.4.2 Evaluating the Importance of Various Components

To produce the results summarized by Table 1, we used our full approach as described in Section 3. In Table 2, we show what happens when we turn off some of its components to gauge their respective impacts.

OURS-CL relies on local classifier scores and does not impose temporal consistency. That is why, it produces a large number of spurious cell tracks. **OURS-NC** addresses this by imposing temporal consistency globally over the whole sequence but it allows multiple conflicting hypotheses to be active simultaneously. Therefore, it still suffers from spurious detections, which leads to low precision. **OURS-FD** disallows conflicting detections but relies on a fixed division probability, which is why it gives low division performance. **OURS-BH** uses division classifier costs but collapses the hierarchical dimension of our graphs and results in mis-detections. Finally, **OURS-LP** removes the integrality constraints on the flow variables. This gives a similar performance to **OURS** on most of the sequences suggesting that the integrality constraints are seldom helpful. However, in the case of HeLa-2, where division events are frequent, we observed that around 3% of the non-zero flow variables are fractional, which explains the 2% drop in recall compared to **OURS**.

5. Conclusion

We have introduced a novel approach to automatically detecting and tracking cell populations in time-lapse images. Unlike earlier approaches that rely on heuristics to handle mis-detections due to clumped cells and occlusions, our approach simultaneously tracks cells from an over-complete set of competing detection hypotheses by solving a single integer program. This results in more accurate trajectories and improved detection of mitosis events.

Furthermore, the formalism is very generic. In future work, we plan to apply it to people tracking and, in particular, modeling how groups can form and unform.

References

[1] F. Amat, W. Lemon, D. P. Mossing, K. McDole, Y. Wan, K. Branson, E. W. Myers, and P. J. Keller. Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data. *Nature methods*, 2014. 1, 2, 6

[2] A. Andriyenko and K. Schindler. Globally Optimal Multi-Target Tracking on a Hexagonal Lattice. In *ECCV*, pages 466–479, 2010. 2

[3] A. Andriyenko, K. Schindler, and S. Roth. Discrete-Continuous Optimization for Multi-Target Tracking. In *CVPR*, 2012. 1, 2

[4] J. Berclaz, F. Fleuret, E. Türetken, and P. Fua. Multiple Object Tracking Using K-Shortest Paths Optimization. *PAMI*, 33:1806–1819, September 2011. 2, 3

[5] R. Collins and P. Carr. Hybrid Stochastic / Deterministic Optimization for Tracking Sports Players and Pedestrians. In *ECCV*, 2014. 2

[6] D. K. Prasad and M. K. H. Leung and C. Quek. ElliFit: An Unconstrained, Non-Iterative, Least Squares-based Geometric Ellipse Fitting Method. *PR*, 46(5):1449–1465, 2013. 4

[7] R. Delgado-Gonzalo, N. Chenouard, and M. Unser. Fast parametric snakes for 3d microscopy. In *Biomedical Imaging (ISBI), 2012 9th IEEE International Symposium on*, pages 852–855, 2012. 2

[8] A. Dufour, R. Thibeaux, E. Labruyere, N. Guillen, and J.-C. Olivo-Marin. 3-d active meshes: fast discrete deformable models for cell tracking in 3-d time-lapse microscopy. *Image Processing, IEEE Transactions on*, 20(7):1925–1937, 2011. 2

[9] O. Dzyubachyk, W. van Cappellen, J. Essers, W. Niessen, and E. Meijering. Advanced level-set-based cell tracking in time-lapse fluorescence microscopy. *TMI*, 2010. 2, 3

[10] J. Friedman. Stochastic Gradient Boosting. *Computational Statistics & Data Analysis*, 2002. 5

[11] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough Forests for Object Detection, Tracking, and Action Recognition. *PAMI*, 2011. 2

[12] W. Ge and R. T. Collins. Multi-Target Data Association by Tracklets with Unsupervised Parameter Estimation. In *BMVC*, September 2008. 1, 2

[13] Gurobi. Gurobi Optimizer, 2012. http://www.gurobi.com/. 5

[14] M. Hofmann, D. Wolf, and G. Rigoll. Hypergraphs for Joint Multi-View Reconstruction and Multi-Object Tracking. In *CVPR*, pages 3650–3657, 2013. 3

[15] F. Jug, T. Pietzsch, D. Kainmüller, and G. Myers. Tracking by assignment facilitates data curation. In *IMIC Workshop, MICCAI*, 2014. 3

[16] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *PAMI*, 34(07), July 2012. 2

[17] R. Kasturi, D. Goldgof, P. Soundararajan, V. Manohar, J. Garofolo, M. Boonstra, V. Korzhova, and J. Zhang. Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol. *PAMI*, 31(2):319–336, February 2009. 6

[18] B. X. Kausler, M. Schiegg, B. Andres, M. Lindner, U. Koethe, H. Leitte, J. Wittbrodt, L. Hufnagel, and F. A. Hamprecht. A discrete chain graph model for 3d+ t cell tracking with high misdetection robustness. In *ECCV*, pages 144–157. 2012. 1, 2, 3, 4

[19] L. Leal-taixe, G. Pons-moll, and B. Rosenhahn. Branch-And-Price Global Optimization for Multi-View Multi-Target Tracking. In *CVPR*, 2012. 3

[20] H. Li, R. Sumner, and M. Pauly. Global Correspondence Optimization for Non-Rigid Registration of Depth Scans. In *Symposium on Geometry Processing*, pages 1421–1430, 2008. 2

[21] 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. *MIA*, 12(5):546–566, 2008. 1, 2, 3, 4

[22] M. Liem and D. Gavrila. Joint multi-person detection and tracking from overlapping cameras. *CVIU*, 128:36–50, 2014. 3

[23] K. Magnusson and J. Jalden. A batch algorithm using iterative application of the Viterbi algorithm to track cells and construct cell lineages. In *ISBI*, 2012. 2, 3, 6, 7

[24] K. Magnusson, J. Jalden, P. Gilbert, and H. Blau. Global linking of cell tracks using the Viterbi algorithm. *TMI*, 2014. 1

[25] M. Maška, O. Daněk, S. Garasa, A. Rouzaut, A. Munoz-Barrutia, and C. Ortiz-de Solorzano. Segmentation and shape tracking of whole fluorescent cells based on the chan-vese model. *TMI*, 32(6):995–1006, 2013. 2

[26] M. Maška, V. Ulman, D. Svoboda, P. Matula, P. Matula, C. Ederra, A. Urbiola, T. España, S. Venkatesan, D. M. Balak, et al. A Benchmark for Comparison of Cell Tracking Algorithms. *Bioinformatics*, 30(11):1609–1617, 2014. 1, 2, 3, 6

[27] E. Meijering, O. Dzyubachyk, and I. Smal. Methods for Cell and Particle Tracking. *Methods in Enzymology*, 504(9):183–200, 2012. 2

[28] S. Oron, A. Bar-Hillel, and S. Avidan. Extended Lucas-Kanade Tracking. In *ECCV*, 2014. 2

[29] D. Padfield, J. Rittscher, N. Thomas, and B. Roysam. Spatio-Temporal Cell Cycle Phase Analysis Using Level Sets and Fast Marching Methods. *MIA*, 13(1):143–155, 2009. 2

[30] H. Pirsiavash, D. Ramanan, and C. Fowlkes. Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects. In *CVPR*, June 2011. 2

[31] J. Platt. *Advances in Large Margin Classifiers*, chapter Probabilistic Outputs for SVMs and Comparisons to Regularized Likelihood Methods. MIT Press, 2000. 5

[32] M. Schiegg, P. Hanslovsky, B. Kausler, L. Hufnagel, and F. Hamprecht. Conservation Tracking. In *ICCV*, pages 2928–2935, Dec 2013. 1, 2, 3, 4, 6, 7

[33] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, and B. Schmid. Fiji: an open-source platform for biological-image analysis. *Nat. Methods*, 9(7):676–682, 2012. Code available at http://pacific.mpi-cbg.de. 4

[34] C. O. Solorzano, M. Kozubek, E. Meijering, and A. M. noz Barrutia. ISBI Cell Tracking Challenge, 2014. 1, 6

[35] C. Sommer, C. Straehle, U. Koethe, and F. Hamprecht. ilastik: Interactive Learning and Segmentation Toolkit. In *ISBI*, 2011. 6

[36] J. W. Suurballe. Disjoint Paths in a Network. *Networks*, 4:125–145, 1974. 3

[37] L. Wen, W. Li, J. Yan, Z. Lei, D. Yi, and S. Z. Li. Multiple target tracking based on undirected hierarchical relation hypergraph. In *CVPR*, 2014. 3

[38] C. Wojek, S. Walk, S. Roth, , K. Schindler, and B. Schiele. Monocular Visual Scene Understanding: Understanding Multi-Object Traffic Scenes. *PAMI*, 2013. 1, 2

[39] B. Yang and R. Nevatia. An Online Learned CRF Model for Multi-Target Tracking. In *CVPR*, 2012. 1, 2

[40] J. Zhang, S. Ma, and S. Sclaroff. MEEM: Robust Tracking via Multiple Experts Using Entropy Minimization. In *ECCV 2014*, 2014. 2

[41] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang. Fast Tracking via Dense Spatio-Temporal Context Learning. In *ECCV 2014*, 2014. 2