Active Structured Learning for Cell Tracking: Algorithm, Framework, and Usability

Xinghua Lou*, Martin Schiegg, and Fred A. Hamprecht

***Abstract*—One distinguishing property of life is its temporal dynamics, and it is hence only natural that time lapse experiments play a crucial role in modern biomedical research areas such as signaling pathways, drug discovery or developmental biology. Such experiments yield a very large number of images that encode complex cellular activities, and reliable automated cell tracking emerges naturally as a prerequisite for further quantitative analysis. However, many existing cell tracking methods are restricted to using only a small number of features to allow for manual tweaking. In this paper, we propose a novel cell tracking approach that embraces a powerful machine learning technique to optimize the tracking parameters based on user annotated tracks. Our approach replaces the tedious parameter tuning with parameter learning and allows for the use of a much richer set of complex tracking features, which in turn affords superior prediction accuracy. Furthermore, we developed an active learning approach for efficient training data retrieval, which reduces the annotation effort to only 17%. In practical terms, our approach allows life science researchers to inject their expertise in a more intuitive and direct manner. This process is further facilitated by using a glyph visualization technique for ground truth annotation and validation. Evaluation and comparison on several publicly available benchmark sequences show significant performance improvement over recently reported approaches. Code and software tools are provided to the public.**

***Index Terms*—Active learning, cell tracking, glyph visualization, integer linear programming, machine learning, mitosis detection, structured prediction, tracking features.**

I. Introduction

EVEN today, cell tracking remains a challenging topic that is under active study [2], [3], particularly in the context of high-content screening [4] and cell lineage reconstruction [5]–[7]. However, manual cell tracking is still predominant in practice [8] and life science researchers need tools that are accurate, configurable and yet require limited manual input [9]. Unlike tracking people or cars in videos [10], cellular image sequences usually contain large numbers (up to thousands) of objects which may be densely packed and be indiscernible by appearance (Fig. 1E). Also, due to physical or experimental limits, the temporal resolution may be so low that cells appear to be "jumping" between frames, showing rare spatial overlap (Fig. 1A). In addition, the underlying cellular activities may be diverse, including cell division (Fig. 1C and D), cell death, large deformation (Fig. 1B), leaving/entering the field-of-view, etc. Taken together, the estimation of inter-frame correspondences of cells becomes a very difficult task. Last but not least, the final result of a tracking-by-assignment depends on the previous cell segmentation/detection step. Despite many encouraging advances [11]–[13], segmentation/detection errors are inevitable and will propagate to tracking errors. This can cause misinterpretation, e.g., an over-segmentation can easily be mistaken for a cell division (Fig. 1F).

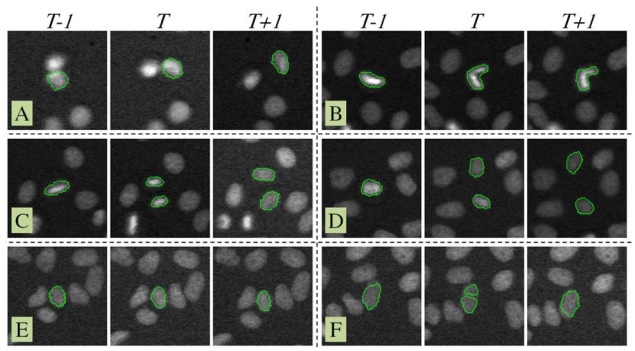

Fig. 1. Challenges in cell tracking. A: Relatively large displacement. B: Large deformation. C and D: Heterogeneous cell division appearance. E: Dense and touching cells. F: Segmentation error (over-segmentation). Three consecutive frames are shown and the boundary of the cells of interest are highlighted.

A. Taxonomy of Cell Tracking Methods

Existing cell tracking methods can broadly be categorized as deformable models, stochastic filtering and object assignment. Deformable models combine detection, segmentation, and tracking by initializing a set of models (e.g., active contours) in the first frame and updating them in subsequent frames [14]–[16]. Large displacements are difficult to capture with this class of techniques and are better handled by state space models, e.g., in the guise of stochastic filtering. The latter also allows for sophisticated observation models [17]. Stochastic filtering builds on a solid statistical foundation, but is often limited in practice due to its high computational demands. Object

Manuscript received August 02, 2013; revised December 11, 2013; accepted December 13, 2013. Date of publication January 02, 2014; date of current version March 31, 2014. This work was supported in part by the CellNetworks Cluster (EXC81), in part by FORSYS ViroQuant (0313923), in part by SB-Cancer, in part by DFG (GRK 1653), and in part by the Enable fund of the University of Heidelberg. This paper was presented in part at the 25th Annual Conference on Neural Information Processing Systems (NIPS) 2011. *Asterisk indicates corresponding author.*

*X. Lou was with Heidelberg Collaboratory for Image Processing (HCI), Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, 69115 Heidelberg, Germany. He is now with the Computational Biology Center of Memorial Sloan-Kettering Cancer Center, New York, NY 10065 USA (e-mail: xinghua.lou@iwr.uni-heidelberg.de).

M. Schiegg and F. A. Hamprecht are with Heidelberg Collaboratory for Image Processing (HCI), Interdisciplinary Center for Scientific Computing (IWR), University of Heidelberg, 69115 Heidelberg, Germany (e-mail: martin.schiegg@iwr.uni-heidelberg.de; fred.hamprecht@iwr.uni-heidelberg.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMI.2013.2296937

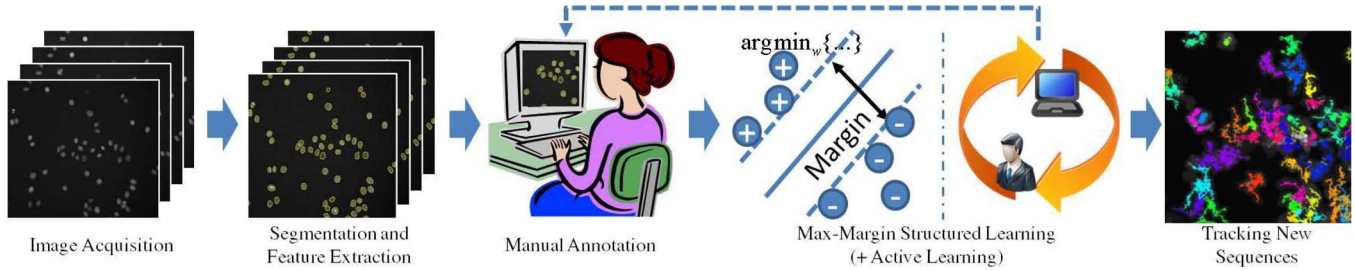

Fig. 2. Overview of our learning based approach. After some segmentation and tracking feature extraction, we take user's track annotation to train a cell tracking model. Active learning is also integrated to reduce the annotation effort. The trained model is then applied to new sequences.

assignment methods make it harder to represent object properties, but they scale well [3], [18], [19], allowing the tracking of thousands of cells in 3-D [20]. For the remainder of this paper, we concentrate on object assignment, or tracking-by-assignment, methods.

B. Machine Learning for Object Tracking

All of the above approaches contain energy terms whose parameters may be tedious or difficult to adjust. Recently, great efforts have been made to develop better energy terms with the help of machine learning techniques, mostly in the context of pedestrian tracking and traffic monitoring. Specifically, for tracking-by-assignment, this was first accomplished by casting tracking as a local affinity prediction problem with either offline [21] or online learning [22]–[24], weakly supervised learning with imperfect oracles [25], manifold appearance model learning [26], or ranking [27], [28]. However, these local methods fail to capture the very important dependency among assignments, hence the resulting local affinities do not necessarily guarantee a better global assignment [29]. To address this limitation, [30] extended the RankBoost method from [28] to rank global assignments represented as a conditional random field (CRF). To reduce the annotation cost, active learning was also applied, e.g., in the context of video data labeling [31].

C. Advantages of Learning Based Tracking

We believe that a learning based approach has the following advantages over approaches without learning. Firstly, learning allows users to inject their prior knowledge in a form or language that is natural to them, namely direct cell-to-cell assignments between frames. This is more intuitive than specifying numerical values for parameters whose meaning, or effect, may remain obscure (as in parametric tracking approaches). Secondly, with learning, adding more annotations allows for systematical improvement of the model. Finally, it enables the use of very high-dimensional features. Though conventional grid parameter search is effective for models with low-dimensional features, it becomes inapplicable for high-dimensional ones for being too expensive (exponentially many combinations) and suboptimal (discretization of the parameter space).

D. Our Contributions

In comparison to previous cell tracking systems, our method is special in that it builds on the concept of learning from user annotated tracks. Our major contributions are as follows. We first present an extended formulation of the object assignment models that takes many complex tracking features and events into account. This generalization improves the expressiveness of the model, but also increases the number of parameters. We hence, secondly, propose to use max-margin structured learning to automatically learn optimum parameters from a training set, and hence profit fully from this richer description. Thirdly, to reduce the high cost of ground truth annotation, we further developed an active learning approach for efficient training data retrieval. To the best of our knowledge, this is the first active learning method for cell tracking. Finally, we share our considerations for a cell tracking framework and its usability, including design of tracking features, a novel visualization technique for annotation and validation, and a C++/MATLAB software toolbox for the community.

E. Paper Organization

The rest of the paper is organized as follows. In Section II, we present the technical details of our tracking model and the learning algorithms (max-margin and active). We then share our considerations for feature design, visualization and software packages in Section III, followed by results in Section IV. Finally, discussion and conclusions are provided in Sections V and VI, respectively.

II. Algorithms

A. Tracking Model: Events, Hypotheses, and Scoring

We assume that a previous detection and segmentation step has identified object candidates in all frames (see Fig. 2). We set out to find that set of object assignments that best explains these observations. To this end, we admit the following set $\boldsymbol{E}$ of standard events [3], [18]: a cell can *move* or *divide* and it can *appear* or *disappear*. In addition, we allow two cells to (seemingly) *merge*, to account for occlusion or under-segmentation; and a cell can (seemingly) *split*, to allow for the lifting of occlusions, or for over-segmentation. These additional hypotheses are useful to account for the errors that typically occur in the detection and segmentation step in crowded or noisy data. The distinction between division and split is reasonable given that typical fluorescence stains endow the anaphase with a distinctive appearance.

We first define important notations in Table I. Given sets of detected objects $\{\boldsymbol{C}, \boldsymbol{C}'\}$ from two subsequent frames, there is a multitude of possible assignment hypotheses (see Fig. 3). We have two tasks: firstly, to allow only consistent assignments (e.g., making sure that each cell in the second frame is accounted

TABLE I
NOTATIONS FOR TRACKING MODEL

Notation	Definition
$\boldsymbol{x}$	Input data, detected objects from a pair of subsequent frames
$\boldsymbol{C}, \boldsymbol{C}'$	Set of objects from the first/second frame
$\mathcal{P}(\boldsymbol{C})$	$\mathcal{P}(\boldsymbol{C}) := \boldsymbol{C} \cup (\boldsymbol{C} \otimes \boldsymbol{C})$, union of $\boldsymbol{C}$ and all ordered pairs of objects in $\boldsymbol{C}$
$\boldsymbol{E}$	Set of all possible events
$\boldsymbol{z}$	Tracking result, assignments of objects
$\boldsymbol{f}^e_{c,c'}$	Feature vector for the assignment of $c \in \mathcal{P}(\boldsymbol{C})$ and $c' \in \mathcal{P}(\boldsymbol{C}')$ as event e
$\boldsymbol{w}^e$	Parameter vector for event e
$z^e_{c,c'}$	Binary indicator variable for the assignment of $c \in \mathcal{P}(\boldsymbol{C})$ and $c' \in \mathcal{P}(\boldsymbol{C}')$ as event e

TABLE II
NOTATIONS FOR LEARNING

Notation	Definition
$\boldsymbol{X}$	Set of training data
$\boldsymbol{Z}^*$	Set of ground truth tracking annotations
$\lambda, \Omega(\boldsymbol{w})$	Regularization strength and regularization function
ξ_n	Slack variable for sample n
$\mathcal{Z}_n$	Space of all possible tracking solutions (structured) for sample n

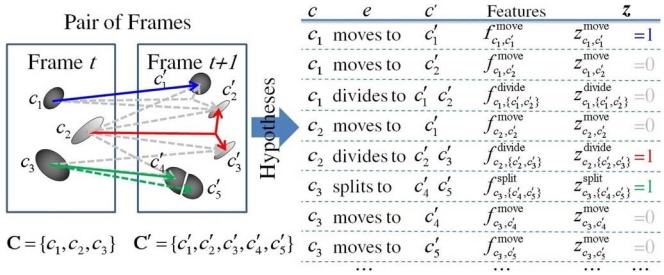

Fig. 3. Toy example: two sets of object candidates, and a small subset of the possible assignment hypotheses. One particular interpretation of the scene is indicated by colored arrows (left) or equivalently by a configuration of binary indicator variables $\boldsymbol{z}$ (right-most column in table). Some rejected hypotheses are shown in gray (best viewed in color).

for only once); and secondly to identify, among the multitude of consistent hypotheses, the one that is most compatible with the observations, and with what we have learned from the training data.

We express this compatibility of the assignment between $c \in \mathcal{P}(\boldsymbol{C})$ and $c' \in \mathcal{P}(\boldsymbol{C}')$ by event $e \in \boldsymbol{E}$ as an inner product $\langle \boldsymbol{f}^e_{c,c'}, \boldsymbol{w}^e \rangle$. Here, $\boldsymbol{f}^e_{c,c'}$ is a feature vector that characterizes the discrepancy (if any) between object candidates c and c'; and $\boldsymbol{w}^e$ is a parameter vector that encodes everything we have learned from the training data. Summing over all object candidates in either of the frames and over all types of events gives the following compatibility function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w}) := \sum_{e \in \boldsymbol{E}} \sum_{c \in \mathcal{P}(\boldsymbol{C})} \sum_{c' \in \mathcal{P}(\boldsymbol{C}')} \left\langle \boldsymbol{f}^e_{c,c'}, \boldsymbol{w}^e \right\rangle z^e_{c,c'} \quad (1)$$

where $\boldsymbol{z} = \{z^e_{c,c'}\}$ is a set of binary variables whose element indicates whether a hypothesis is accepted. The compatibility function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w})$ states how well a set of accepted hypotheses $\boldsymbol{z}$ matches the observations $\boldsymbol{f}(\boldsymbol{x})$ computed from the raw data $\boldsymbol{x}$, given the knowledge $\boldsymbol{w}$ from the training set.

Equation (1) is equivalent to

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w}) := \boldsymbol{w}' \Phi(\boldsymbol{x}, \boldsymbol{z}) \quad (2)$$

where $\boldsymbol{w}$ is the concatenation of event-specific parameter $(\boldsymbol{w}^{\text{move}}, \boldsymbol{w}^{\text{divide}}, \ldots)$ and $\Phi(\boldsymbol{x}, \boldsymbol{z})$ is the concatenation of event-specific features summed up over all activated events, which is referred to as joint feature vector [32]

$$\Phi(\boldsymbol{x}, \boldsymbol{z}) = \begin{bmatrix} \sum_{c \in \mathcal{P}(\boldsymbol{C})} \sum_{c' \in \mathcal{P}(\boldsymbol{C}')} \boldsymbol{f}^{\text{move}}_{c,c'} z^{\text{move}}_{c,c'} \\ \sum_{c \in \mathcal{P}(\boldsymbol{C})} \sum_{c' \in \mathcal{P}(\boldsymbol{C}')} \boldsymbol{f}^{\text{divide}}_{c,c'} z^{\text{divide}}_{c,c'} \\ \ldots \end{bmatrix}. \quad (3)$$

Furthermore, the selection of $\boldsymbol{z}$ is subject to consistency requirements: each candidate in the first frame must have a single fate, and each candidate from the second frame a unique past. That is, for hypotheses associated with the same candidate, only one of them can be accepted. Formally, we have $\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}$, a space that satisfies the following constraints:

$$\forall c' \in \mathcal{P}(\boldsymbol{C}'), \sum_{e \in \boldsymbol{E}} \sum_{c \in \mathcal{P}(\boldsymbol{C})} z^e_{c,c'} = 1, (\text{consistency}) \quad (4)$$

$$\forall c \in \mathcal{P}(\boldsymbol{C}), \sum_{e \in \boldsymbol{E}} \sum_{c' \in \mathcal{P}(\boldsymbol{C}')} z^e_{c,c'} = 1, (\text{consistency})$$
$$\forall e \in \boldsymbol{E}, c \in \mathcal{P}(\boldsymbol{C}), c' \in \mathcal{P}(\boldsymbol{C}'), z^e_{c,c'} \in \{0, 1\}. (\text{booleanity}). \quad (5)$$

As an important technical detail, note that $\mathcal{P}(\boldsymbol{C}) := \boldsymbol{C} \cup (\boldsymbol{C} \otimes \boldsymbol{C})$ is a set comprising each object candidate, as well as all ordered pairs of object candidates from a frame. For the example in Fig. 3, $\mathcal{P}(\boldsymbol{C}) = \{c_1, c_2, c_3, \{c_1, c_2\}, \{c_1, c_3\}, \{c_2, c_3\}\}$. This allows us to conveniently subsume cell divisions, splits and mergers in the above equation.

To this end, model inference refers to finding the best $\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}$ that *maximizes* the compatibility function $\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w})$

$$\hat{\boldsymbol{z}} = \arg\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w}). \quad (6)$$

The remaining tasks, discussed next, are how to learn the parameters $\boldsymbol{w}$ from the training data; given these, how to find the best possible $\boldsymbol{z}$; and finding useful features.

B. Structured Max-Margin Parameter Learning

1) Learning to Track by Risk Minimization: In learning the parameters automatically from a training set, we pursue two goals: first, to go beyond manual parameter tweaking in obtaining the best possible performance; and second, to make the process as facile as possible for the user. This is under the assumption that most experimentalists find it easier to specify what a correct tracking should look like, rather than what value a more-or-less obscure parameter should have.

Given N training frame pairs $\boldsymbol{X} = \{\boldsymbol{x}_n\}$ and their correct assignments $\boldsymbol{Z}^* = \{\boldsymbol{z}^*_n\}$, $n = 1, \ldots, N$, the best set of parameters is the optimizer of

$$\arg\min_{\boldsymbol{w}} \mathcal{R}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{Z}^*) + \lambda \Omega(\boldsymbol{w}). \quad (7)$$

Here, $\mathcal{R}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{Z}^*)$ measures the empirical loss of the current parametrization $\boldsymbol{w}$ given the training data $\boldsymbol{X}, \boldsymbol{Z}^*$. To prevent overfitting to the training data, this is complemented by the regularizer $\Omega(\boldsymbol{w})$ that favors parsimonious models. We use L_1 or L_2 regularization ($\Omega(\boldsymbol{w}) = \|\boldsymbol{w}\|_p^p/p$, $p = \{1, 2\}$), i.e., a measure of the length of the parameter vector $\boldsymbol{w}$. The L_2 norm is often used for its numerical efficiency, while the L_1 norm is popular thanks to its potential to induce sparse solutions (i.e., some parameters can become zero). The empirical loss is given by $\mathcal{R}(\boldsymbol{w}; \boldsymbol{X}, \boldsymbol{Z}^*) = (1/N) \sum_{i=1}^{N} \Delta(\boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n(\boldsymbol{w}; \boldsymbol{x}_n))$. Here $\Delta(\boldsymbol{z}^*, \hat{\boldsymbol{z}})$ is a loss function that measures the discrepancy between a true assignment $\boldsymbol{z}^*$ and a prediction by specifying the fraction of missed events with respect to the ground truth

$$\Delta(\boldsymbol{z}^*, \hat{\boldsymbol{z}}) = \frac{1}{|\boldsymbol{z}^*|} \sum_{e \in \boldsymbol{E}} \sum_{c \in \mathcal{P}(\boldsymbol{C})} \sum_{c' \in \mathcal{P}(\boldsymbol{C}')} z_{c,c'}^{*e} \left(1 - \hat{z}_{c,c'}^{e}\right). \quad (8)$$

This decomposable function allows for easy exact inference when solving (9) [33]. Note that this loss only penalizes false negatives. We do not explicitly penalize false positives, because they are already interconnected through the consistency constraints in (4) and (5).

2) Max-Margin Structured Learning: Importantly, note that under constraints (4) and (5) both the input (objects from a pair of frames) and output (assignments between objects) in this learning problem are *structured*—a set of variables that are interdependent. We hence resort to max-margin structured learning [32] to exploit the structure and dependency within the assignment hypotheses. In comparison to other aforementioned learning methods, structured learning allows us to directly learn the global affinity measure, avoid generating many artificial false assignment samples, and drop any assumptions on the signs of the features. Structured learning has been successfully applied to many complex real world problems such as word/sequence alignment, graph matching, and image segmentation [34].

In particular, we attempt to find the decision boundary that maximizes the margin between the correct assignment $\boldsymbol{z}_n^*$ and the closest runner-up solution. An equivalent formulation is the condition that the score of $\boldsymbol{z}_n^*$ be greater than that of any other solution by some margin. To prevent overfitting, one can relax this constraint by introducing slack variables ξ_n, which finally yields the following objective function for the max-margin structured learning problem from (7)

$$\begin{aligned} \underset{\boldsymbol{w}, \boldsymbol{\xi} \geq \boldsymbol{0}}{\arg\min} \quad & \frac{1}{N} \sum_{n=1}^{N} \xi_n + \lambda \Omega(\boldsymbol{w}) \\ \text{s. t.} \quad & \forall n, \forall \hat{\boldsymbol{z}}_n \in \boldsymbol{\mathcal{Z}}_n : \\ & \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{z}_n^*; \boldsymbol{w}) - \mathcal{L}(\boldsymbol{x}_n, \hat{\boldsymbol{z}}_n; \boldsymbol{w}) \geq \Delta(\boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n) - \xi_n \end{aligned} \quad (9)$$

where $\boldsymbol{\mathcal{Z}}_n$ is the set of possible consistent assignments and using $\Delta(\boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n)$ instead of a fixed margin is known as "margin-rescaling" [32]. Intuitively, it pushes the decision boundary further away from the "bad" solutions with high losses. Note that the loss in (9) is $\xi_n = |\Delta(\boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n) - \mathcal{L}(\boldsymbol{x}_n, \boldsymbol{z}_n^*; \boldsymbol{w}) + \mathcal{L}(\boldsymbol{x}_n, \hat{\boldsymbol{z}}_n; \boldsymbol{w})|_+$, which is a tight, convex upper bound on the original loss in (7) (non-convex).

1: **Input:** $\boldsymbol{D} = \{(\boldsymbol{x}_n, \boldsymbol{z}_n^*)\}_{n=1}^N, \hat{\epsilon}$
2: Initialize $\boldsymbol{A} = \emptyset, \boldsymbol{b} = \emptyset, t = 1$, $\boldsymbol{w}$ (randomly)
3: **repeat**
4: **for all** $(\boldsymbol{x}, \boldsymbol{z}^*) \in \boldsymbol{D}$ **do**
5: Compute $\hat{\boldsymbol{z}}$ using Eq. 10
6: **end for**
7: Compute $\boldsymbol{a}_t$ and b_t as in Eq. 12 and Eq. 13
8: Set $\boldsymbol{A} = \boldsymbol{A} \cup \boldsymbol{a}_t$ and $\boldsymbol{b} = \boldsymbol{b} \cup b_t$
9: Update $\boldsymbol{w}$ with $\boldsymbol{A} = \{\boldsymbol{a}_1, \ldots\}$, $\boldsymbol{b} = \{b_1, \ldots\}$ (Eq. 14)
10: Compute approximation gap ϵ
11: Set $t = t + 1$
12: **until** $\epsilon \leq \hat{\epsilon}$
13: **Output:** $\boldsymbol{w}$

Fig. 4. Max-margin structured learning.

3) Optimization With Bundle Method: Since (9) involves an exponential number of constraints, the learning problem cannot be represented explicitly, let alone solved directly. We thus resort to the *bundle method* [35] which, in turn, is based on the *cutting-planes* approach [32]. The basic idea is as follows. Start with some parametrization $\boldsymbol{w}$ and no constraints. At iteration t, first find the optimum assignments for the current $\boldsymbol{w}$ by solving, for all n

$$\hat{\boldsymbol{z}}_n = \arg\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}_n} \left\{\mathcal{L}(\boldsymbol{x}_n, \boldsymbol{z}; \boldsymbol{w}) + \Delta(\boldsymbol{z}_n^*, \boldsymbol{z})\right\}. \quad (10)$$

Use all $\hat{\boldsymbol{z}}_n$ to identify the most violated constraint, which is a linear lower bound of the average of the slack variables [35]

$$\boldsymbol{a}_t' \boldsymbol{w} + b_t \leq \frac{1}{N} \sum_{n=1}^{N} \xi_n \quad (11)$$

where, given $\Psi(\boldsymbol{x}, \boldsymbol{z}^*, \hat{\boldsymbol{z}}) := \Phi(\boldsymbol{x}, \boldsymbol{z}^*) - \Phi(\boldsymbol{x}, \hat{\boldsymbol{z}})$, we have

$$\boldsymbol{a}_t = -\frac{1}{N} \left(\sum_{n}^{N} \Psi(\boldsymbol{x}_n, \boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n) \right) \quad (12)$$

$$b_t = -\frac{1}{N} \left[\sum_{n=1}^{N} \Delta(\boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n) + \boldsymbol{w}' \Psi(\boldsymbol{x}_n, \boldsymbol{z}_n^*, \hat{\boldsymbol{z}}_n) \right] - \boldsymbol{w}' \boldsymbol{a}_t. \quad (13)$$

We then update $\boldsymbol{w}$ by solving a variant of (9), which instead uses all those most-violated constraints (including those from previous iterations)

$$\begin{aligned} \underset{\boldsymbol{w}, \boldsymbol{\xi} \geq \boldsymbol{0}}{\arg\min} \quad & \frac{1}{N} \sum_{n=1}^{N} \xi_n + \lambda \Omega(\boldsymbol{w}) \\ \text{s. t.} \quad & \forall i \in \{1, \ldots, t\}, \boldsymbol{a}_i' w + b_i \leq \frac{1}{N} \sum_{n}^{N} \xi_n. \end{aligned} \quad (14)$$

We then move on to the next iteration: find new best assignments using the updated $\boldsymbol{w}$, etc. The procedure converges when those constraints form a tight lower bound of the original objective function [(7)], which is measured by so-called approximation gap ϵ (see [35] for more details). For a given parameter $\boldsymbol{w}$, the optimum assignments can be found by integer linear programming (ILP) [3], [18], [19]. Pseudocode is shown in Fig. 4.

1: **Input:** $\boldsymbol{D} = \{\boldsymbol{x}_n\}_{n=1}^{N}, \hat{\eta}, T$
2: Initialize $\boldsymbol{D}_{\mathrm{L}} = \emptyset, \boldsymbol{D}_{\mathrm{U}} = \boldsymbol{D}, t = 1$, $\boldsymbol{w}$ (randomly)
3: **repeat**
4: Find $\tilde{\boldsymbol{x}} = \arg\max_{\boldsymbol{x} \in \boldsymbol{D}_{\mathrm{U}}} q(\boldsymbol{x}, \boldsymbol{w})$
5: Annotate $\tilde{\boldsymbol{z}}^*$
6: Set $\boldsymbol{D}_{\mathrm{U}} = \boldsymbol{D}_{\mathrm{U}} \setminus \tilde{\boldsymbol{x}}$
7: Set $\boldsymbol{D}_{\mathrm{L}} = \boldsymbol{D}_{\mathrm{L}} \cup \{(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{z}})\}$
8: **for all** $(\boldsymbol{x}, \boldsymbol{z}^*) \in \boldsymbol{D}_{\mathrm{L}}$ **do**
9: Compute $\hat{\boldsymbol{z}}$ using Eq. 10
10: Update $\boldsymbol{w} = \boldsymbol{w} + \Phi(\boldsymbol{x}, \boldsymbol{z}^*) - \Phi(\boldsymbol{x}, \hat{\boldsymbol{z}})$
11: **end for**
12: Compute average uncertainty $\bar{q}_t = \dfrac{1}{|\boldsymbol{D}_{\mathrm{U}}|} \sum_{\boldsymbol{x} \in \boldsymbol{D}_{\mathrm{U}}} q(\boldsymbol{x}, \boldsymbol{w})$
13: Compute convergence measure $\eta(\bar{q}_{t-T:t})$ (Eq. 16)
14: Set $t = t + 1$
15: **until** $\eta(\bar{q}_{t-T:t}) \leq \hat{\eta}$ or $\boldsymbol{D}_{\mathrm{U}} \equiv \emptyset$
16: **Output:** $\boldsymbol{w}$

Fig. 5. Active structured learning with perceptron.

TABLE III
LIST OF UNCERTAINTY SAMPLING STRATEGY

Name	$q(\boldsymbol{x}, \boldsymbol{w})$ Formulation and Description
Random	Random number between 0 and 1
Scoring	$\exp\left(-\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z})\right)$ where higher value of $\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z})$ indicates higher confidence on the predicted tracking using existing parameter $\boldsymbol{w}$.
Best Vs. Worst	$\exp\left(-\left(\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z}) - \min_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z})\right)\right)$ where larger margin between those two terms indicates higher confidence towards the best predicted tracking w.r.t the worst one.
Best Vs. 2nd	$\exp\left(-\left(\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z}) - \max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^\circ} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z})\right)\right)$ where $\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}^\circ} \boldsymbol{w}'\Phi(\boldsymbol{x}, \boldsymbol{z})$ means computing the second best scoring and larger margin between those two terms indicates higher confidence towards the best predicted tracking w.r.t the second best one.

C. Training Data Retrieval via Active Learning

Given a learning algorithm, training a highly predictive tracking model is still dependent on the amount and quality of annotated training samples. Unfortunately, the annotation effort is particularly high for our tracking problem—hundreds of cells have to be examined and hundreds of events have to be marked per training sample (a pair of images). Therefore, we introduce an active learning approach for efficient training data retrieval at a low annotation cost. Our approach has the following core components: patchification, uncertainty measure, model update, and stopping criteria. We elaborate on the details following the pseudocode shown in Fig. 5.

1) Patchification: Refers to dividing a pair of full images (large, hundreds of events) into pairs of local patches (small, normally less than 20 cells per frame with patch size 128 × 128). Then each pair of local patches is considered a training sample. This certainly breaks the original (large) structure of event dependency induced by (4) and (5). Yet, we will empirically show that training on patchified samples is just as effective as training on pairs of full images. We consider patchification a necessary and viable preprocessing step for active learning. Otherwise, annotating a single sample is already too tedious and time-consuming, and part of the efforts is wasted on similar and repeated event patterns.

Note that patchification induces lots of artificial appearances and disappearances at the patch border. We eliminate them by discarding objects that touch the patch border unless it is the true image border.

2) Uncertainty Measure: Is the very core component of active learning [36]. We propose four different uncertainty measures described in Table III. They are direct extensions of uncertainty measures for flat data [37], [38] to structured data as in this paper. As lines 4–6 of Fig. 5 shows, at each iteration, we find the most uncertain sample (viz. pair of patches) from all unlabeled samples $\boldsymbol{D}_{\mathrm{U}}$ and demand annotation from the annotator. We will compare the learning curves of those uncertainty measures in results.

As technical details, to compute the worst predicted tracking with given $\boldsymbol{w}$ (see *Best vs. Worst*, Table III), we can simply flip the signs of $\boldsymbol{w}$ and call (6). Computing the second best tracking (see *Best vs. 2nd*, Table III) is a bit more complicated. Briefly, we first compute the best tracking $\hat{\boldsymbol{z}}$ using (6). Then the second best tracking $\hat{\hat{\boldsymbol{z}}}$ can be computed by (15) which is a variant of (6) with one extra constraint. The rational is as follows: $2\hat{\boldsymbol{z}} - 1$ transforms all the 0 s in the binary vector $\hat{\boldsymbol{z}}$ to -1 s, so the maximum possible value of $\langle 2\hat{\boldsymbol{z}} - 1, \boldsymbol{z}\rangle$ is $\|\hat{\boldsymbol{z}}\|_1$ and is achieved only when $\boldsymbol{z} = \hat{\boldsymbol{z}}$; by forcing $\langle 2\hat{\boldsymbol{z}} - 1, \boldsymbol{z}\rangle \leq \|\hat{\boldsymbol{z}}\|_1 - 1$, the optimizer can never pick $\hat{\boldsymbol{z}}$ as the optimal solution

$$\begin{aligned} \hat{\hat{\boldsymbol{z}}} = \arg\max_{\boldsymbol{z} \in \boldsymbol{\mathcal{Z}}} \quad & \mathcal{L}(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{w}) \\ \text{s. t.} \quad & \langle 2\hat{\boldsymbol{z}} - 1, \boldsymbol{z}\rangle \leq \|\hat{\boldsymbol{z}}\|_1 - 1. \end{aligned} \tag{15}$$

3) Model Update: Refers to updating the model parameter $\boldsymbol{w}$ after receiving a new annotated sample. Given labeled training set ($\boldsymbol{D}_{\mathrm{L}}$, Fig. 5), a naïve way is to invoke max-margin structured learning from the previous section [(9)]. However, this turns out inefficient in practice: max-margin structured learning is known very expensive (see [32] and our runtime result), which means that the annotator has to wait a few minutes before proceeding to the next sample. Therefore, we resort to structured perceptron [39] for model update (lines 8–11, Fig. 5). Briefly, it makes a one-pass run through all labeled samples and updates the parameter by incrementally (and locally) adding the gradient, viz. $\boldsymbol{w} = \boldsymbol{w} + \partial_{\boldsymbol{w}}(\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}^*; \boldsymbol{w}) - \mathcal{L}(\boldsymbol{x}, \hat{\boldsymbol{z}}; \boldsymbol{w}))$ (equivalent to line 10, Fig. 5).

Theorem 1: Assuming a pool of N unannotated samples, the complexity of the proposed algorithm in Fig. 5 is $\mathcal{O}(N^2)$. While the complexity of using max-margin structured learning for model update is at least $\mathcal{O}(N^3)$.

Proof: In Fig. 5, at each iteration t $(1 \leq t \leq N)$ we need N predictions, among which $N - t$ predictions are for uncertainty estimation on unlabeled samples (line 4) and the rest t for model update using labeled sampled (lines 8–11). This gives TN predictions after T iterations. Since T is a fraction of N, the overall complexity is $\mathcal{O}(N^2)$.

In the case of max-margin structured learning for model update (viz. replacing lines 8–11 with the algorithm in Fig. 4), [40] shows that, in SVM like max-margin formulation (including structured learning), the number of support vectors scales at least linearly with the number of training samples.

Thus, the complexity of max-margin structured learning is at least quadratic because we need compute the inner-product of each support vector and each sample. The overall complexity is then at least $\sum_t[(N-t)+t^2]$, which amounts to $\mathcal{O}(N^3)$. ■

We will discuss other pros and cons in Section II-D.

4) Stopping Criteria: Are another crucial component [36]. We chose a very popular measure proposed in [41]—the average uncertainty over all remaining unlabeled samples (see uncertainty measure in Table III). This does not require any holdout validation dataset. At iteration t, given a sequence of computed average uncertainty $\bar{q}_{t-T:t}$ (including previous ones), we compute the convergence measure η using (16) [42] (lines 12–13, Fig. 5). This convergence measure drops to a low value when the improvement on average uncertainty remains minor for several iterations. We stop the active learning when the convergence measure is below a given threshold or all samples are labeled (line 15, Fig. 5)

$$\eta(\bar{q}_{1:T+1}) = |\widehat{\text{mean}}(\bar{q}_{2:T+1}) - \widehat{\text{mean}}(\bar{q}_{1:T})|\,. \qquad (16)$$

Here, $\widehat{\text{mean}}(\cdot)$ is the robust mean (viz. mean of the elements within the 10% and 90% quantile).

D. A Combined Learning Strategy

Though gaining speed, using structured perceptron for model update has two drawbacks: lack of regularization and local (thus noisy) gradient update (line 10, Fig. 5). This makes the learned model prone to overfitting and also unstable in convergence (see Section IV). Therefore, in practice we use a combined approach: we use active structured perceptron *only* for training data retrieval and then use max-margin structured learning to obtain a regularized and globally optimized model.

III. Framework and Usability

This section covers other important considerations for a cell tracking framework, including tracking features, visual editing tool, and implementation/software.

A. Framework: Design of Tracking Features

The above structured learning for cell tracking allows the use of a much extended set of features to capture complicated events from different aspects, and thus boost the tracking performance. Many interesting tracking features have been proposed in previous work. We borrow some state-of-the-art [18], [19], and also contribute several new (e.g., Fig. 6). Table IV shows a categorization of our tracking features. For a detailed list, please refer to the **supporting document**. Note that these features can vary significantly in scale, which makes manual parameter tuning difficult. Normalization helps, yet can not address the problem at its root. In the following, we briefly describe the features designed for different events.

B. Framework: Implementation and Software Contributions

Our framework has been implemented using MATLAB and C++. The annotation GUI and active learning are written in MATLAB for better interactivity, while the computation-intensive components such as feature extraction, model inference and max-margin structured learning are provided in C++. In practice, to reduce the search space and eliminate hypotheses with no prospect of being realized, we constrain the hypotheses to a k-nearest neighborhood (usually $k = 4$) with distance thresholding. We use IBM CPLEX[1] as the underlying optimization platform for the ILP and quadratic programming as needed for solving (9) [35]. In addition to those tracking features proposed in this paper, the core library also provides a generic interface for adding additional features using the factory design pattern. The library also allows for configurable settings for different applications (e.g., worm tracking, no division event). This enables users to easily integrate their preferred features. Our software library is available online[2] .

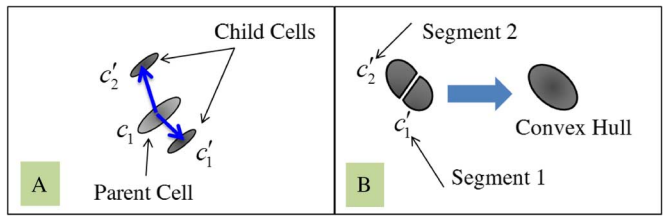

Fig. 6. Illustration of features. A: Angle pattern. B: Shape compactness. Angle pattern measures whether the vectors pointing from the parent cell to the child cells form an angle close to 180°. Shape compactness is the ratio of the total area of two proximate segments (c_1' and c_2') divided by the area of the convex hull of their union.

TABLE IV
Categorization of Features

Type	Description
Position	difference in position, distance to border, overlap with border
Intensity	difference in intensity histogram/sum/mean/deviation, intensity of father cell
Shape	difference in shape, difference in size, shape compactness, shape evenness
Others	angle pattern, mass evenness, eccentricity of father cell

C. Usability: Active Learning and Glyph Visualization

The improved usability first attributes to the use of active learning, which obtains the most informative sample to annotate. We further facilitate annotation by exploiting a visualization technique called "glyph visualization" [43] from the information visualization community [44]. Briefly, when two frames are placed side-by-side, instead of drawing arrows (e.g., Fig. 7A, cluttered view) or colored text (e.g., cell ID, more perception time), we draw two identical markers (viz. glyphs in the visualization language) at their respective cell centers to represent an assignment. We make the markers for proximate cells differ in several primitive attributes such as color and shape, which forms a local pattern. As a consequence, users no longer need to match cells by arrows or cell IDs but can use **local patterns** instead (viz. group by group in Fig. 7B, clean). This is efficient thanks to our powerful vision system: we instantly recognize and differentiate local patterns. Furthermore,

[1]Available online : http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[2]Available online: http://hci.iwr.uni-heidelberg.de/Staff/xlou/research/tracking.html

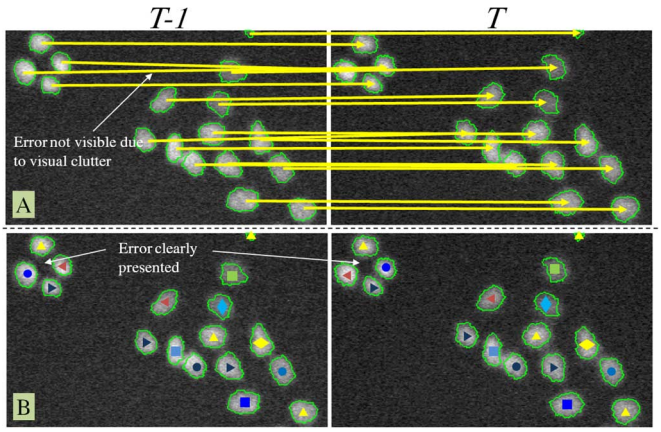

Fig. 7. Comparison of line visualization and our glyph visualization as assignment indicator. A: Line visualization, cluttered and errors not obvious. B: Glyph visualization, tends to be cleaner and afford an easier overview.

incorrect tracking is easily discernible because of the consistency constraints [see (4) and (5)], namely a single error usually provokes a domino effect, leading to a strong perturbation of the glyph pattern. Important events such as divisions are still highlighted using arrows; however, being rare, they barely give rise to any visual clutter.

IV. Results

We evaluated the proposed approach on five publicly available image sequences, one from DCellIQ[3] [19] and four from Mitocheck[4] [45] (referred to as MC1, MC2, etc.). The datasets differ in illumination, cell density and image compression artifacts (Fig. 8 and Table V). The imaging conditions for MC1 to MC4 are generally consistent, which differ from DCellIQ slightly. Note that retraining is necessary when applying the proposed approach to datasets with drastically different imaging conditions.

The GFP stained cell nuclei were segmented using the method in [13], yielding an F-score over 99.3% by counting. Some statistics about these two datasets are shown in Table V. Note that our tracking approach is not restricted to the specific segmentation method used. Yet, high quality segmentation is expected since many tracking features are extracted using segmentation masks.

A. Performance of Max-Margin Structured Learning

This section aims at evaluating max-margin structured learning. Therefore, we assume that sufficient annotation efforts can be provided. All results in this section are from models trained on full samples.

1) Task 1: Efficient Tracking for a Given Sequence: We first evaluate our method on a task that is frequently encountered in practice: the user simply wishes to obtain a good tracking for a given sequence with the smallest possible effort. The original graph matching based method on this sequence by [19] yields a loss of 6.18% (Table VI, second row). The model in [18] using only two types of features (size and position) reduces this loss to 1.64% (Table VI, first row). A detailed analysis of the error counts for specific events shows that the method accounts well for moves, but has difficulty with disappearance and split events. This is mainly due to the limited descriptive power of the simple features used. To understand this limit, we applied our max-margin structured learning to optimize the model in [18] and obtained a reduction of the total loss from 1.64% to 0.65% (Table VI, fourth row). This can be considered as the limit of this model. Note that the learned parametrization actually deteriorates the detection of divisions because the learning aims at minimizing the overall loss across all events. Local learning [1] also has a reasonable result (Table VI, third row).

Our model contains 38 features and thus manual tweaking becomes difficult (Table VI, fifth row). However, the proposed structured learning allows our model to fully profit from this richer description and achieve a total loss of only **0.30%** (Table VI, sixth row). More precisely, our model only missed **34** events which is much lower than the rest. Some example assignments are shown in Fig. 9.

The learned parameters are summarized in Fig. 10. They afford the following observations. Firstly, features on cell size and shape are generally of high importance, which is in line with the assumption in [18]. Secondly, the correlations of the features with the final assignment score are automatically learned. For example, shape compactness is positively correlated with split but negatively with division. This is in line with the intuition that an oversegmentation conserves compact shape, while a true division pushes the child cells far away from each other (in the present kind of data, where only DNA is labeled). Finally, many features are associated with large weights, which is key to the improved expressive power.

2) Task 2: Tracking for High-Throughput Experiments: The experiment described in the foregoing draws both training and test samples from the same time lapse experiment. However, in high-throughput experiments such as in the Mitocheck project [45], it is more desirable to train on one or a few sequences, and make predictions on many others. To emulate this situation, we have used the parameters $\boldsymbol{w}$ trained in the foregoing on DCellIQ

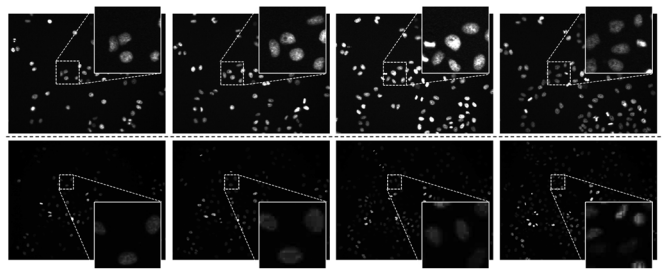

Fig. 8. Selected raw images from DCellIQ (top) and MC4 (bottom). MC4 exhibits higher cell density, larger intensity variability and "blockness" artifacts due to image compression.

TABLE V
Statistics of Datasets Used in Our Study

Data	Size ($X \times Y \times T$)	No. Cells	Seg. Acc.	Compressed
DCellIQ	$512 \times 672 \times 100$	10664	99.5%	No
MC1	$1024 \times 1344 \times 93$	37539	99.6%	No
MC2	$1024 \times 1344 \times 95$	19206	99.7%	No
MC3	$1024 \times 1344 \times 95$	12872	99.3%	No
MC4	$1024 \times 1344 \times 94$	24102	99.3%	Yes

[3]Available online: http://www.cbi-tmhs.org/Dcelliq/files/051606_HeLaMCF10A_DMSO_1.rar.

[4]Available online: http://www.mitocheck.org.

TABLE VI
PERFORMANCE COMPARISON ON THE DCELLIQ DATASET. HEADER ROW SHOWS THE NUMBER OF EVENTS OCCURRING FOR MOVES, DIVISIONS, APPEARANCE, DISAPPEARANCE, SPLITS, AND MERGERS. REMAINING ENTRIES GIVE THE ERROR COUNTS FOR EACH EVENT, SUMMED OVER THE ENTIRE SEQUENCE

DCellIQ - Event(Count)	Moves(10156)	Divisions(104)	Appear.(78)	Disapp.(76)	Splits(54)	Mergers(55)	Sum(10523)	Loss
[18] w/ suggested param.	71	18	16	26	30	12	173	1.64%
Original method [19]	-	-	-	-	-	-	-	6.18%[a]
Local learning by RF [1]	18	14	2	**0**	12	13	59	0.55%
[18] w/ our learning	21	25	5	5	6	10	72	0.65%
Ours w/ manual tweaking	56	24	16	19	2	**5**	122	1.12%
Ours w/ learning	**15**	**6**	**4**	1	**2**	6	**34**	**0.30%**

[a] Here we use the best reported error matching rate in [19] (similar to our loss).

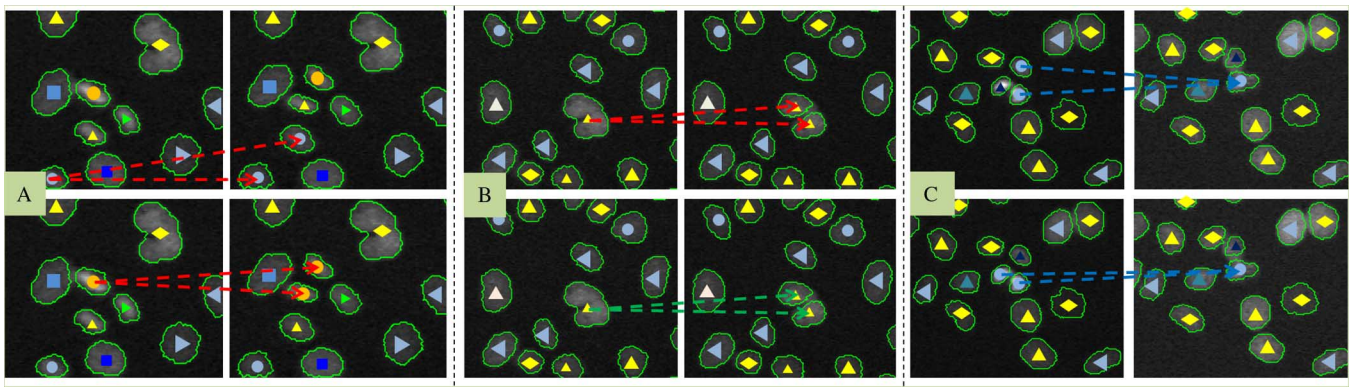

Fig. 9. Some diverging assignments by [18] (top) and our approach (bottom). A: Missing a false division event. B: Split mistaken for division. C: False merger and move event. Color code: move—glyph visualization ; red—division; green—split; blue—merger (best viewed in color).

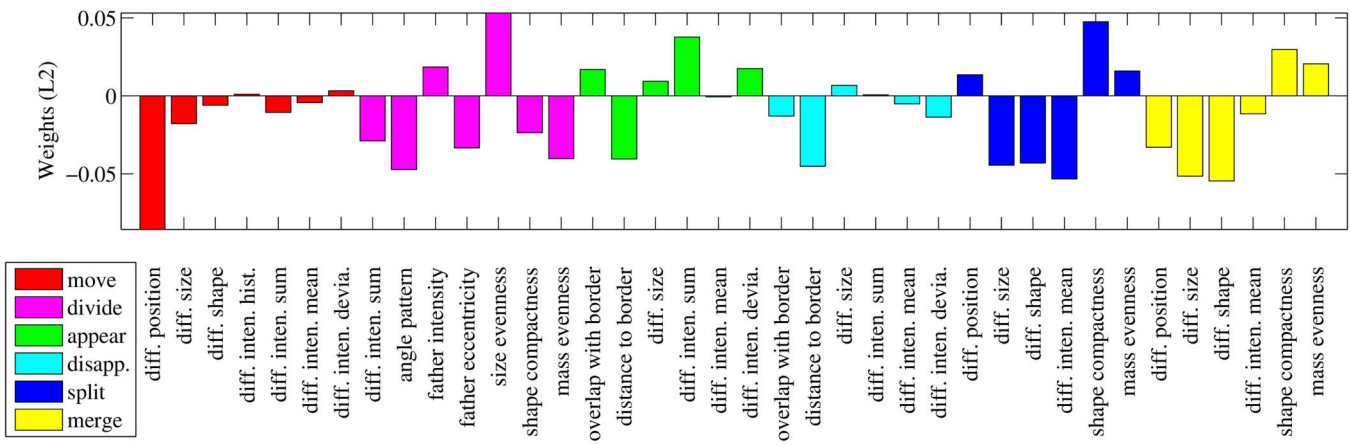

Fig. 10. Parameters $\boldsymbol{w}$ learned from the training data. Parameters weighing the features for different events are colored differently. The parameter values indicate the correlations of the features with the respective event, i.e., the importance of each feature for the according event is measured by the absolute value of its weight parameter. Note that the correlation can be positive or negative.

[19] and used these to track the Mitocheck dataset, without further training or refinement of the parameters. The main focus of the Mitocheck project is on accurate detection of mitosis (cell division). Despite the difference between those datasets, our method shows a high generalization capability and obtains an average total loss of 0.59% ± 0.33% (see Table VII). The relatively degraded performance on MC4 stems from the fact the video sequence is compressed which eventually yields noisy tracking features (e.g., blockness, intensity loss). Our method consistently outperforms [18] even if their parameters are optimized using our learning algorithm.

Fig. 11 shows the trajectories of lineages extracted from MC4. The background shows the maximum intensity projection through the image sequence. Each lineage is associated with a unique color—the root is indicated with a circle and all its descendants are colored the same. We see heterogeneous migrations and expansions, and cells are generally growing towards empty space in the field-of-view.

3) On Mitosis Event Detection: Specific Versus Conjunctive: Cell division, or mitosis, usually draws particularly high attention from life science researchers. Methods have been developed to specifically detect this event [45], [46]. We show that mitosis detection can be improved if it is considered in conjunction with other events, as is done in our approach. With appropriately learned features, we can significantly reduce ambiguity and eliminate false positives. For the DCellIQ sequence which is partially used for training, we successfully detected 98/104 events (see Table VI). For the more challenging MC4 dataset (for testing only), we correctly found 94.3% of 384 mitotic events.

TABLE VII
PERFORMANCE COMPARISON ON THE MITOCHECK DATASETS. METHOD WAS TRAINED ON THE DCELLIQ DATASET. HEADER ROW SHOWS THE NUMBER OF EVENTS OCCURRING FOR MOVES, DIVISIONS, APPEARANCE, DISAPPEARANCE, SPLITS, AND MERGERS. REMAINING ENTRIES GIVE THE ERROR COUNTS FOR EACH EVENT, SUMMED OVER THE ENTIRE SEQUENCE

MC1 - Event(Count)	Moves(35845)	Divisions(465)	Appear.(363)	Disappear.(335)	Splits(174)	Mergers(217)	Sum(37399)	Loss
[18] w/ our learning	50	66	30	36	109	63	354	1.04%
Ours w/ learning	30	23	30	30	56	30	199	0.56%
MC2 - Event(Count)	Moves(18957)	Divisions(195)	Appear.(59)	Disappear.(99)	Splits(90)	Mergers(78)	Sum(19478)	Loss
[18] w/ our learning	17	69	7	3	43	10	149	0.82%
Ours w/ learning	10	10	7	3	17	10	57	0.24%
MC3 - Event(Count)	Moves(12493)	Divisions(118)	Appear.(105)	Disappear.(78)	Splits(109)	Mergers(102)	Sum(13005)	Loss
[18] w/ our learning	13	40	7	13	69	20	162	1.13%
Ours w/ learning	7	20	7	10	23	13	80	0.56%
MC4 - Event(Count)	Moves(22520)	Divisions(384)	Appear.(310)	Disappear.(304)	Splits(127)	Mergers(132)	Sum(23777)	Loss
[18] w/ our learning	102	92	49	35	79	21	378	1.44%
Ours w/ learning	109	22	57	33	48	9	278	1.03%

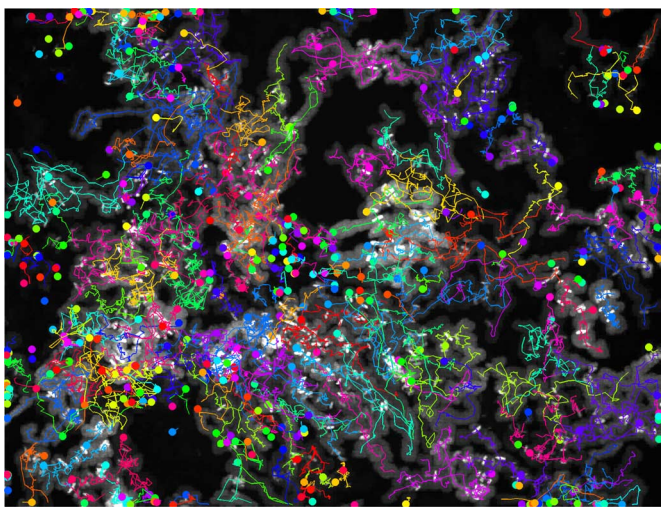

Fig. 11. Trajectory of lineages (best viewed in color).

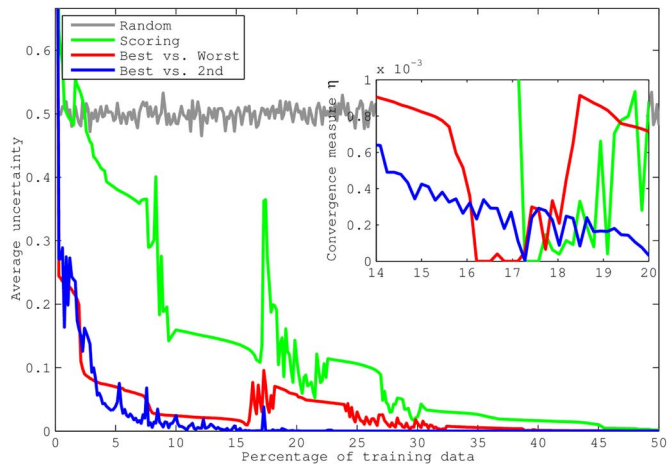

Fig. 12. Comparison of uncertainty measures: average uncertainty versus percentage of training data. The embedded figure shows the uncertainty convergence η versus the percentage of training data.

B. Performance of Active Structured Learning

Knowing that we can train a highly predictive cell tracking model given sufficient training data, we now evaluate our active learning for annotation cost reduction.

1) Justification of Patchification: As a strong prerequisite for the success of active learning, we first need to justify patchification. We made a direct comparison by applying regularized max-margin training on full samples (pairs of full images) and on their respective patchified samples. Training samples are sampled from DCellIQ and the test data is MC4. After 10 repeated experiments, the test error for training with patchified samples is 1.10 ± 0.01 (unit: %), which is comparable to that of full samples: 1.08 ± 0.01 (unit: %). Note that patchification is only applied to training data and the test data is **not** patchified.

2) Uncertainty Measures and Stopping Criteria: Using 660 patchified training samples from the DCellIQ dataset, in Fig. 12 we compare the learning curves (viz. average uncertainty) of the four uncertainty measures upto 50% of the total training samples. *Best vs. Worst* is stably converging at the beginning but has a second wave of significant changes after 16% of total training samples. The same applies to *Scoring* but the changes of average uncertainty are more drastic. *Best vs. 2nd* appears to be the best performing one: it converges to a stable state after 17% of total training samples. Regarding stopping criteria, *Random* is excluded because it is not suitable for the uncertainty convergence measure η [(16)]. To compute η, we chose $T = 80$ and used 10^{-4} as the stopping threshold. As the embedded figure in Fig. 12 shows, they all stop at around 17% of training data (*Best vs. Worst* is a bit earlier).

To further understand the learning curve in a practical setting, we tested all intermediate learned parameter $\boldsymbol{w}$ of the active learning process on MC4, respectively for all uncertainty measures. The result in Fig. 13 further supports our choice of *Best 2nd* not only because of its superiority in stability but also for its lower test error. Note that this observation is in line with the principle of max-margin.

The second wave in the learning curve of *Scoring* and *Best vs. Worst* suggests that there are two principle cohorts in the training datasets. Both *Scoring* and *Best vs. Worst* get stuck in choosing samples mostly from one cohort and consequently make the trained model overfit to this cohort. This is recognized only when the overfitting is too severe and the second cohort becomes dominant in the remaining unlabeled samples, namely the beginning of the second wave. *Best vs. 2nd* is more robust against this issue.

3) Runtime: In practice, using structured perceptron for model update yields pleasant runtime. Across iterations it requires (stably) less than 9 s to perform model update and

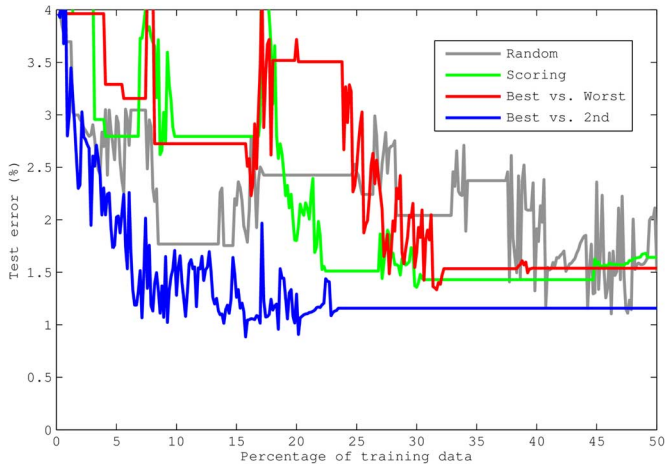

Fig. 13. Comparison of uncertainty measure: test error versus percentage of training data.

TABLE VIII
EVALUATION OF COMBINED LEARNING—UNIT %

	17%		30%		40%	
Measure	AL	CL	AL	CL	AL	CL
Random	1.77	1.66	2.14	1.53	2.43	1.31
Scoring	3.72	1.78	2.79	1.73	1.80	1.11
Best vs. Worst	2.73	2.23	2.73	3.06	3.72	1.36
Best vs. 2nd	1.33	**1.08**	1.26	1.06	1.29	1.09
Baseline	1.07					

uncertainty computation. We consider this a tolerable delay. The final max-margin structured learning run takes between 150 and 200 s. Note that this runtime is dependent on the hardware specification of the computer because the underlying solver CPLEX can run the branch-and-bound ILP algorithm in parallel. We used a 2.4-GHz Intel Xeon machine with 12 cores.

C. Performance of Combined Learning

We discussed that the purpose of active learning is to retrieve informative training samples while the truly useful model has to be learned via regularized max-margin learning. Table VIII shows the result of this combined learning strategy (CL) using 17% (viz. the stopping point by the convergence measure), 30% and 40% of training samples, compared against the active learning (AL) output. This affords the following observations. Firstly, using the same amount of training samples, regularized max-margin learning generally improves the output of active learning. Secondly, *Best vs. 2nd* performs better than the other uncertainty measures. Finally (and most importantly), using *Best vs. 2nd* as uncertainty measure and using only 17% of the training samples, we can train a tracking model as competent as the baseline model learned from all samples (last line, Table VIII).

D. Hyperparameter Optimization

Here, we discuss the selection of key hyperparameters used in the regularized max-margin structured learning.

First, we optimize the regularization parameter λ via cross validation (five-fold). It turned out that the regularization paths [47] for training with full samples and training with patchified

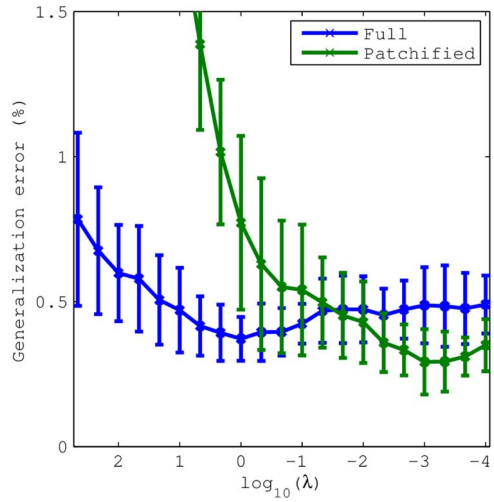

Fig. 14. Generalization error versus regularization parameter λ.

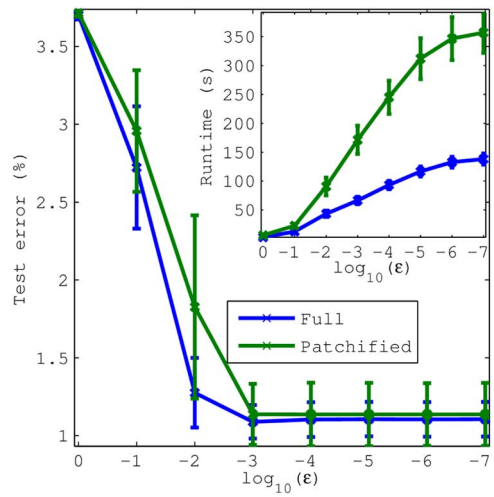

Fig. 15. Test error versus approximation gap ϵ.

ones are quite distinct (Fig. 14). In practice, we chose $\lambda = 1$ for the former and $\lambda = 10^{-3}$ for the later.

Second, the approximation gap ϵ controls the precision of approximating the learning objective function with piece-wise linear lower bounds (see Fig. 4 and [35]). A lower value means higher precision yet also more computation time (see the embedded figure in Fig. 15 about runtime). We optimize this parameter via empirical tests on a holdout validation dataset (MC4 in this case). As Fig. 15 shows, $\epsilon = 10^{-3}$ is sufficient for both training settings and any higher precision does not increase the test performance.

E. Additional Results

For other results such as sensitivity to training data size and sparse feature selection via $\boldsymbol{L_1}$ norm, please refer to the **supporting document** or [1].

V. DISCUSSIONS

As all detection/segmentation based tracking, our approach also relies on the quality of segmentation, and we consider its major limitations as follows. Firstly, it cannot recover missing

detections. This can be addressed by, for example, extending the technique from [48] which is a global model covering the *entire* sequence. Our learning approach can be adopted to learn this global tracking model, yet this can be prohibitively expensive. A more viable strategy is to train on pairwise models as in this paper, yet perform the tracking globally using the global model. Secondly, our model cannot handle overly segmented objects. This can be alleviated by improving the segmentation. However, a single segmentation cannot always suffice. For example, tuning the segmentation towards less over-segmentation may lead to more under-segmentation. A better solution is to use multiple segmentations that are complementary to each other (e.g., from multiple scales), and model the selection of segmentations jointly with the tracking.

VI. Conclusion and Future Work

We present a new cell tracking scheme that uses many expressive features and comes with a structured learning framework to train the larger number of parameters involved. We further propose an active learning approach for efficient training data retrieval that, empirically, reduces the annotation cost to only 17%. Comparison to related methods shows that this learning scheme brings significant improvements in performance and usability.

Acknowledgment

The authors would like to thank J.-K. Heriche from EMBL (Heidelberg) for providing Mitocheck datasets and C. Klein (University of Heidelberg) for ground truth annotation. The authors would like to thank all reviewers for their insightful and constructive comments.

References

[1] X. Lou and F. A. Hamprecht, "Structured learning for cell tracking," in *Neural Inf. Process. Syst.*, 2011, pp. 1296–1304.

[2] E. Meijering, O. Dzyubachyk, I. Smal, and W. A. van Cappellen, "Tracking in cell and developmental biology," *Seminars Cell Develop. Biol.*, vol. 20, no. 8, pp. 894–902, 2009.

[3] T. Kanade, Z. Yin, R. Bise, S. Huh, S. E. Eom, M. Sandbothe, and M. Chen, "Cell image analysis: Algorithms, system and applications," in *IEEE Workshop Appl. Comput. Vis.*, 2011.

[4] V. Abraham, D. Taylor, and J. Haskins, "High content screening applied to large-scale cell biology," *Trends Biotechnol.*, vol. 22, no. 1, pp. 15–22, 2004.

[5] A. K. Hadjantonakis, M. E. Dickinson, S. E. Fraser, and V. E. Papaioannou, "Technicolour transgenics: Imaging tools for functional genomics in the mouse," *Nature Rev. Genetics*, vol. 4, no. 8, pp. 613–625, 2003.

[6] Z. Bao, J. I. Murray, T. Boyle, S. L. Ooi, M. J. Sandel, and R. H. Waterston, "Automated cell lineage tracing in Caenorhabditis elegans," *Proc. Nat. Acad. Sci.*, vol. 103, no. 8, pp. 2707–2712, 2006.

[7] P. J. Keller, A. D. Schmidt, J. Wittbrodt, and E. H. K. Stelzer, "Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy," *Science*, vol. 322, no. 5904, p. 1065, 2008.

[8] T. Schroeder, "Long-term single-cell imaging of mammalian stem cells," *Nature Methods*, vol. 8, no. 4s, pp. S30–S35, 2011.

[9] J. W. Young, J. C. W. Locke, A. Altinok, N. Rosenfeld, T. Bacarian, P. S. Swain, E. Mjolsness, and M. B. Elowitz, "Measuring single cell gene expression dynamics in bacteria using fluorescence time-lapse microscopy," *Nature Protocols*, vol. 7, no. 1, pp. 80–88, 2011.

[10] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, 2006.

[11] G. Li, T. Liu, A. Tarokh, J. Nie, L. Guo, A. Mara, S. Holley, and S. T. C. Wong, "3-D cell nuclei segmentation based on gradient flow tracking," *BMC Cell Biol.*, vol. 8, no. 1, 2007.

[12] Y. Al-Kofahi, W. Lassoued, W. Lee, and B. Roysam, "Improved automatic detection and segmentation of cell nuclei in histopathology images," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 4, pp. 841–852, Apr. 2010.

[13] X. Lou, U. Köthe, J. Wittbrodt, and F. A. Hamprecht, "Learning to segment dense cell nuclei with shape prior," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 1012–1018.

[14] A. Dufour, V. Shinin, S. Tajbakhsh, N. Guillen-Aghion, J. C. Olivo-Marin, and C. Zimmer, "Segmenting and tracking fluorescent cells in dynamic 3-D microscopy with coupled active surfaces," *IEEE Trans. Image Process.*, vol. 14, no. 9, pp. 1396–1410, Sep. 2005.

[15] 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," *Med. Image Anal.*, vol. 12, no. 5, pp. 546–566, 2008.

[16] O. Dzyubachyk, W. A. van Cappellen, J. Essers, W. J. Niessen, and E. Meijering, "Advanced level-set-based cell tracking in time-lapse fluorescence microscopy," *IEEE Trans. Med. Imag.*, vol. 29, no. 3, pp. 852–867, Mar. 2010.

[17] I. Smal, K. Draegestein, N. Galjart, W. Niessen, and E. Meijering, "Particle filtering for multiple object tracking in dynamic fluorescence microscopy images: Application to microtubule growth analysis," *IEEE Trans. Med. Imag.*, vol. 27, no. 6, pp. 789–804, Jun. 2008.

[18] D. Padfield, J. Rittscher, and B. Roysam, "Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis," *Med. Image Anal.*, vol. 15, no. 4, pp. 650–668, 2011.

[19] F. Li, X. Zhou, J. Ma, and S. Wong, "Multiple nuclei tracking using integer programming for quantitative cancer cell cycle analysis," *IEEE Trans. Med. Imag.*, vol. 29, no. 1, pp. 96–105, Jan. 2010.

[20] X. Lou, F. O. Kaster, M. S. Lindner, B. X. Kausler, U. Köthe, H. Jänicke, B. Höckendorf, J. Wittbrodt, and F. A. Hamprecht, "DELTR: Digital embryo lineage tree reconstructor," in *Proc. IEEE Int. Symp. Biomed. Imag.: From Nano to Macro*, 2011, pp. 1557–1560.

[21] S. Avidan, "Ensemble tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2005, pp. 494–501.

[22] H. Grabner and H. Bischof, "On-line boosting and vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 260–267.

[23] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Robust tracking-by-detection using a detector confidence particle filter," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2009, pp. 1515–1522.

[24] C.-H. Kuo, C. Huang, and R. Nevatia, "Multi-target tracking by on-line learned discriminative appearance models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 685–692.

[25] B. Zhong, H. Yao, S. Chen, R. Ji, X. Yuan, S. Liu, and W. Gao, "Visual tracking via weakly supervised learning from multiple imperfect oracles," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2010, pp. 1323–1330.

[26] X. Wang, G. Hua, and T. X. Han, "Discriminative tracking by metric learning," in *Proc. Eur. Conf. Comput. Vis.*, 2010, pp. 200–214.

[27] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, 2003.

[28] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybrid-Boosted multi-target tracker for crowded scene," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 2953–2960.

[29] B. Yang, C. Huang, and R. Nevatia, "Learning affinities and dependencies for multi-target tracking using a CRF model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1233–1240.

[30] B. Yang and R. Nevatia, "An online learned CRF model for multitarget tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2034–2041.

[31] R. Yan, J. Yang, and A. Hauptmann, "Automatically labeling video data using multi-class active learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. 516–523.

[32] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, no. 2, pp. 1453–1484, 2006.

[33] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1048–1058, Jun. 2009.

[34] G. Bakir, T. Hofmann, B. Schoelkopf, A. J. Smola, B. Taskar, and S. Vishwanathan, *Predicting Structured Data*. Cambridge, MA: MIT Press, 2006.

[35] C. H. Teo, S. V. N. Vishwanthan, A. J. Smola, and Q. V. Le, "Bundle methods for regularized risk minimization," *J. Mach. Learn. Res.*, vol. 11, pp. 311–365, 2010.

[36] B. Settles, "Active learning literature survey," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.
[37] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," *J. Mach. Learn. Res.*, vol. 2, pp. 45–66, 2002.
[38] G. Schohn and D. Cohn, "Less is more: Active learning with support vector machines," in *Proc. Int. Conf. Mach. Learn.*, 2000, pp. 839–846.
[39] M. Collins, "Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms," in *Proc. Meet. Assoc. Computat. Linguist.*, 2002, pp. 1–8.
[40] L. Bottou, *Large-Scale Kernel Machines*. Cambridge, U.K.: MIT Press, 2007.
[41] A. Vlachos, "A stopping criterion for active learning," *Comput. Speech Language*, vol. 22, no. 3, pp. 295–312, 2008.
[42] F. Laws and H. Schätze, "Stopping criteria for active learning of named entity recognition," in *Proc. 22nd Int. Conf. Computat. Linguist.*, 2008, vol. 1, pp. 465–472.
[43] C. Ware, *Information Visualization: Perception for Design*. San Mateo, CA: Morgan Kaufmann, 2004.
[44] S. K. Card, J. D. Mackinlay, and B. Shneiderman, *Readings in Information Visualization: Using Vision to Think*. San Mateo, CA: Morgan Kaufmann, 1999.
[45] M. Held, M. H. A. Schmitz, B. Fischer, T. Walter, B. Neumann, M. H. Olma, M. Peter, J. Ellenberg, and D. W. Gerlich, "Cell cognition: Time resolved phenotype annotation in high-throughput live cell imaging," *Nature Methods*, vol. 7, no. 9, pp. 747–754, 2010.
[46] S. Huh, D. Ker, R. Bise, M. Chen, and T. Kanade, "Automated mitosis detection of stem cell populations in phase-contrast microscopy images," *IEEE Trans. Med. Imag.*, vol. 30, no. 3, pp. 586–596, Mar. 2011.
[47] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "The entire regularization path for the support vector machine," *J. Mach. Learn. Res.*, vol. 5, pp. 1391–1415, 2004.
[48] B. X. Kausler, M. Schiegg, B. Andres, M. Lindner, U. Köthe, H. Leitte, J. Wittbrodt, L. Hufnagel, and F. A. Hamprecht, "A discrete chain graph model for 3-D+T cell tracking with high misdetection robustness," in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 144–157.