DISSERTATION
submitted
to the
Combined Faculty for the Natural Sciences and
Mathematics
of
Heidelberg University, Germany
for the degree of
Doctor of Natural Sciences

Put forward by

Diplom-Wirtschaftsmathematiker Martin Josef Schiegg
Born in: Augsburg
Oral examination: _____

Multi-Target Tracking with Probabilistic Graphical Models

Advisor: Prof. Dr. Fred A. Hamprecht

Acknowledgements

First of all, I want to express my sincere gratitude to Prof. Fred Hamprecht who gave me the opportunity to pursue this enthralling research project. His vision to apply and advance cutting-edge machine learning algorithms in order to tackle fundamental biological questions, and his continued enthusiasm about his research was infective. His thorough supervision, and critical questions at the right times, the latitude he gave me to delve into subjects I got excited about during my research, the pleasant, helpful, and gifted members of his group which I joined in 2012, all contributed to the fact that I could not imagine a better environment for my doctoral studies.

Moreover, it is thanks to Lars Hufnagel and his group at the EMBL Heidelberg, particularly Stefan Günther, that I was fortunate enough to gain access to these terrific datasets from developing fruit flies. They always instantly shared novel datasets from their latest improvements on the light sheet microscope. These regular and valuable discussions left an imprint on the research directions I took. Many thanks for this great collaboration.

Furthermore, I am very grateful to Prof. Jochen Wittbrodt and his group, specifically Burkhard Höckendorf, for close collaborations and for giving me the opportunity to participate in their retreats and laboratory meetings. These regular interactions yielded fruitful inputs to my research, particularly Burkhard's frequent and patient explanations about cell biology.

I also spent great days with Jan Funke (ETH Zurich), Florian Jug, Tobias Pietzsch, and Dagmar Kainmüller (MPI Dresden) at our structured learning workshops (or "hackathons"). Cordial thanks for this friendly collaboration!

Moreover, I have benefited from many rewarding discussions with Bernhard Kausler, my precursor in the cell tracking project. It was thanks to his comprehensive knowledge and his helpful support that I was able to rapidly familiarize myself with the cell tracking project when I first joined Fred's research group. He coined the phrase that "everything is a graphical model", which I found myself

recalling ever so often when I studied many a model.

I would also like to extend my appreciation to our cell tracking experts Philipp Hanslovsky, Carsten Haubold, Ben Heuer, and Steffen Wolf. Philipp contributed to the method in Section 2.2.2 and during endless but productive nights at the institute, we jointly developed the method proposed in Chapter 3. With both him and Carsten, I had plenty of productive discussions about cell tracking and both always knew answers to any programming question. Carsten contributed to the runtime analysis in Appendix B, Ben implemented parts of the algorithm presented in Chapter 4, and Steffen made it possible to apply the structured learning algorithms in the same chapter. The daily discussions with these skilled researchers yielded many interesting ideas.

I would also like to give many thanks to my colleagues and friends Ferran Diego, Melih Kandemir, and Anna Kreshuk who always lent a friendly ear for discussing our works. In particular, Ferran's supervision for the work presented in Chapter 5 was very enriching.

Moreover, I would like to sincerely thank the current and former members of Fred's research group with whom it was a pleasure to work closely together, namely Janez Ales, Thorsten Beier, Stuart Berg, Christoph Decker, Kemal Eren, Burcin Eröcal, Luca Fiaschi, Jens Kleesiek, Christoph Klein, Ullrich Köthe, Thorben Kröger, Niko Krasowski, Xinghua Lou, David Stöckel, Christoph Strähle, Robert Walecki, Chong Zhang, Buote Xu, and especially Barbara Werner for managing all administrative matters.

I would like to give special thanks to my parents for making my studies possible and for supporting my decision for the doctoral studies. And not least, my deepest gratitude is expressed to Judith Werner for her loving support, her constant encouragement, and selfless understanding for the endless time spent on this work.

Abstract

Thanks to revolutionary developments in microscopy techniques such as robotic high-throughput setups or light sheet microscopy, vast amounts of data can be acquired at unprecedented temporal and spatial resolution. The mass of data naturally prohibits *manual* analysis, though, and life scientists thus have to rely more and more on *automated* cell tracking methods. However, automated cell tracking involves intricacies that are not commonly found in traditional tracking applications. For instance, cells may undergo mitosis, which results in variable numbers of tracking targets across successive frames. These difficulties have been addressed by tracking-by-assignment models in the past, which dissect the task into two stages, detection and tracking. However, as with every two-stage framework, the approach hinges on the quality of the first stage, and errors propagate partially irrevocably from the detection to the tracking phase.

The research in this thesis thus focuses on methods to advance tracking-by-assignment models in order to avoid these errors by exploiting synergy effects between the two (previously) separate stages. We propose two approaches, both in terms of probabilistic graphical models, which allow for information exchange between the detection and the tracking step to different degrees. The first algorithm, termed *Conservation tracking*, models both possible over- and undersegmentation errors and implements global consistency constraints in order to reidentify target identities even across occlusion or erroneous detections. Wrong detections from the first step can hence be corrected in the second stage. The second method goes one step further and optimizes the two stages completely *jointly* in one holistic model. In this way, the detection and tracking step can maximally benefit from each other and reach the overall most likely interpretation of the data. Both algorithms yield notable results which are state-of-the-art.

In spite of the distinguished results achieved with these methods, automated cell tracking methods are still error-prone and manual proof-reading is often unavoidable for life scientists. To avoid the time-consuming manual identification of errors on very large datasets, most ambiguous predictions ought to be detected automatically

so that these can be corrected by a human expert with minimal effort. In response, we propose two easy-to-use methods to sample multiple solutions from a tracking-by-assignment graphical model and derive uncertainty measures from the variations across the samples. We showcase the usefulness for guided proof-reading on the cell tracking model proposed in this work.

Finally, the successful application of structured output learning algorithms to cell tracking in previous work inspired us to advance the state-of-the-art by an algorithm called *Coulomb Structured Support Vector Machine* (CSSVM). The CSSVM improves the expected generalization error for unseen test data by the training of multiple *concurrent* graphical models. Through the novel diversity encouraging term, motivated from experimental design, the ensemble of graphical models is learned to yield *diverse* predictions for test data. The best prediction amongst these models may then be selected by an oracle or with respect to a more complex loss. Experimental evaluation shows significantly better results than using only one model and achieves state-of-the-art performance on challenging computer vision tasks.

Zusammenfassung

Dank revolutionärer Entwicklungen in der Mikroskopiertechnik, wie zum Beispiel *High-Throughput*-Roboter oder die Lichtscheibenmikroskopie, können gewaltige Datenmengen mit beispielloser zeitlicher und räumlicher Auflösung aufgenommen werden. Allerdings macht die riesige Masse an Daten eine *manuelle* Analyse unmöglich, weshalb Biowissenschaftler immer mehr auf *automatische* Zell-Tracking-Methoden angewiesen sind. Automatisches Zell-Tracking bringt jedoch Schwierigkeiten mit sich, welche in traditionellen Tracking-Anwendungen kaum eine Rolle spielen. Beispielsweise durchlaufen Zellen Mitose, wodurch sich die Anzahl der Tracking-Objekte in aufeinanderfolgenden Zeitschritten stark verändern kann. Diese Komplexitäten wurden in der Vergangenheit durch so genannte *Tracking-by-assignment*-Methoden angegangen, welche die Aufgabe in zwei Abschnitte aufteilen: Erkennung und Tracking. Wie jedes Zwei-Phasen-Modell steht und fällt jedoch auch dieses mit der Qualität des ersten Schrittes, und Fehler pflanzen sich teilweise unwiderruflich von der Erkennungs- zur Tracking-Phase fort.

Diese Forschungsarbeit beschäftigt sich deshalb mit Methoden, welche *Tracking-by-assignment*-Modelle durch das Ausnutzen von Synergieeffekten zwischen den beiden (bisher) getrennten Schritten verbessern sollen, um solche Fehler zu vermeiden. Wir schlagen zwei Ansätze vor, beide in Form von probabilistischen graphischen Modellen, die einen Informationsaustausch zwischen der Erkennungs- und der Tracking-Phase zu unterschiedlichen Graden ermöglichen. Der erste Algorithmus, den wir *Conservation Tracking* nennen, modelliert sowohl mögliche Über- als auch Untersegmentierungsfehler und führt globale Konsistenzbedingungen ein um Objektidentitäten sogar nach Verdeckung oder fehlerhafter Erkennung wiederzufinden. Falsch erkannte Objekte aus dem ersten Schritt können somit in der zweiten Phase korrigiert werden. Die zweite Methode geht noch einen Schritt weiter und optimiert die beiden Schritte komplett *gemeinsam* in einem ganzheitlichen Modell. Auf diese Weise können Erkennung und Tracking maximal voneinander profitieren und zusammen die insgesamt wahrscheinlichste Dateninterpretation erreichen. Beide Algorithmen erzielen hervorragende Ergebnisse, die dem neuesten Stand der Technik entsprechen.

Trotz der überzeugenden Ergebnisse, die mit diesen Methoden erzielt werden, ist das automatische Zell-Tracking jedoch immer noch fehleranfällig, was eine manuelle Fehlersuche für Biowissenschaftler unumgänglich macht. Um eine zeitraubende manuelle Identifikation solcher Fehler auf sehr großen Datensätzen zu vermeiden, sollten deshalb die unsichersten Vorhersagen automatisch gefunden werden, so dass diese von einem menschlichen Experten mit möglichst wenig Aufwand korrigiert werden können. Aus diesem Grund schlagen wir zwei einfach anwendbare Methoden vor um mehrere Lösungen von einem *Tracking-by-assignment*-Modell zu sampeln und leiten Unsicherheitsmaße von der Variabilität dieser Lösungen ab. Wir demonstrieren die Nützlichkeit für eine gesteuerte Fehlersuche auf dem in dieser Arbeit vorgestellten Zell-Tracking-Modell.

In früheren Arbeiten wurden strukturierte Lernalgorithmen erfolgreich auf Zell-Tracking-Modelle angewendet, was uns letztlich inspiriert hat, den Stand der Technik in dieser Richtung durch eine Methode namens *Coulomb Structured Support Vector Machine* (CSSVM) weiterzuentwickeln. Die CSSVM verbessert den erwarteten Generalisierungsfehler für zukünftige Testdaten durch das Training von mehreren *konkurrierenden* graphischen Modellen. Der neu eingeführte Term belohnt Diversität unter den Modellen und wird aus dem Bereich der Versuchsanordnung motiviert. Durch ihn wird ein Ensemble von graphischen Modellen so trainiert, dass es *diverse* Vorhersagen auf Testdaten liefert. Anschließend können die besten Vorhersagen von diesen Modellen durch ein Orakel oder durch eine komplexere Funktion ausgewählt werden. In der experimentellen Evaluation zeigen sich für die CSSVM deutlich bessere Ergebnisse gegenüber den Vorhersagen nur eines Modells und es werden Ergebnisse auf herausfordernden Problemen aus dem *Computer-Vision*-Bereich erzielt, die dem neuesten Stand der Technik entsprechen.

Contents

1

Introduction

Recent innovative developments in light sheet microscopy enable researchers in life sciences to examine research questions which have long been limited by imaging techniques. These novel microscopes (Keller et al. 2008; Keller et al. 2010; Tomer et al. 2012; Krzic et al. 2012) allow to acquire multidimensional images *in vivo* at outstanding high spatial and temporal resolutions, up to sub-cellular level. Thanks to selective plane illumination, and as a consequence, highly reduced effects of phototoxity and bleaching (Keller et al. 2008), (live) organisms may be recorded over *multiple* days without being harmed. As a result, one grand challenge of developmental biology, namely to understand embryogenesis (Keller et al. 2008; Keller et al. 2010; Höckendorf et al. 2012; Amat and Keller 2013), has been revived in recent years. Developmental biologists, hence, long for full reconstructions of so-called *digital embryos* (Keller et al. 2008), *i.e.* a fully documented lineage tree for each embryo from the egg cell to the tens of thousands of cells developed until the larval stage. From such building plans (Amat and Keller 2013), the scientists desire to analyze correlations at different scales (Höckendorf et al. 2012; Krzic et al. 2012; Amat and Keller 2013), between individual cells, between lineage trees of distinct embryos, or even across diverse species. From these insights, conclusions may be drawn about important biological research questions, such as:

- How are shapes generated during embryonic development and how can they be reproducible? Or more generally:
- Which mechanical forces govern embryogenesis? (Keller et al. 2008; Amat and Keller 2013)
- Which lineage relationships within and across tissues/organs can be found? For instance:

- Where and how are specialized cell types established? When do their lineages branch off the main tree? How related are the different cell types of an adult organ in the lineage tree? (Keller et al. 2008; Höckendorf et al. 2012)

Correlations in the lineage across tissues, individuals, or species would greatly contribute to the understanding of building patterns in developmental biology. Ultimately, these insights promise to be applicable in a wide range of contexts, including disease models, drug design, or analysis of mutant phenotypes (Amat and Keller 2013).

For simple organisms such as the nematode *Caenorhabditis elegans* (*C. elegans*), such a digital embryo has been successfully acquired already decades ago by Sulston and Horvitz (1977). Sulston and Horvitz (1977) performed their image analysis preponderantly manually, profiting from the invariant lineages, the transparency of the organism, and the fact that a male larva only consists of 671 cells. Such manual analysis, however, becomes prohibitive for more complex organisms such as zebrafish (*Danio rerio*) or fruit fly (*Drosophila*), as studied in this work. For such animals, which develop to tens of thousands of cells (Amat and Keller 2013), investigations for correlations between different embryos, and even within one single embryo, are only feasible if computing systems are able to *automatically* observe and reconstruct such lineage trees. Image analysis algorithms need to automatically follow each and every cell[1] over time to identify each cell at every point in time, and – at the same time – find all cell divisions.

Cell tracking, certainly, is not restricted to the *in vivo* experiments from developmental biology described above. Insights in other areas in life sciences are also highly dependent on robust and accurate *automated* cell tracking (Meijering et al. 2009; Kanade et al. 2011; Meijering et al. 2012; González et al. 2013; Maška et al. 2014), which is not only due to enormous amounts of data acquired by means of robotic high-throughput setups. In particular, many interesting research questions may only be approached by observing proliferating cells (*in vitro*) under various culture conditions, or by analyzing cell behavior in live tissues (*in situ*), both only feasible with the help of automated cell tracking methods.

However, since automated algorithms are not reliable enough yet due to the challenges outlined next, cell tracking is often still performed manually in these kind of experiments (Coutu and Schroeder 2013).

[1] Typically the cell *nuclei* are stained by fluorescence markers. Thus, technically speaking, the cell *nuclei* are to be tracked. Note that for brevity, we refer in this thesis to the tracking of cell *nuclei* as the tracking of *cells* or *cell tracking*.

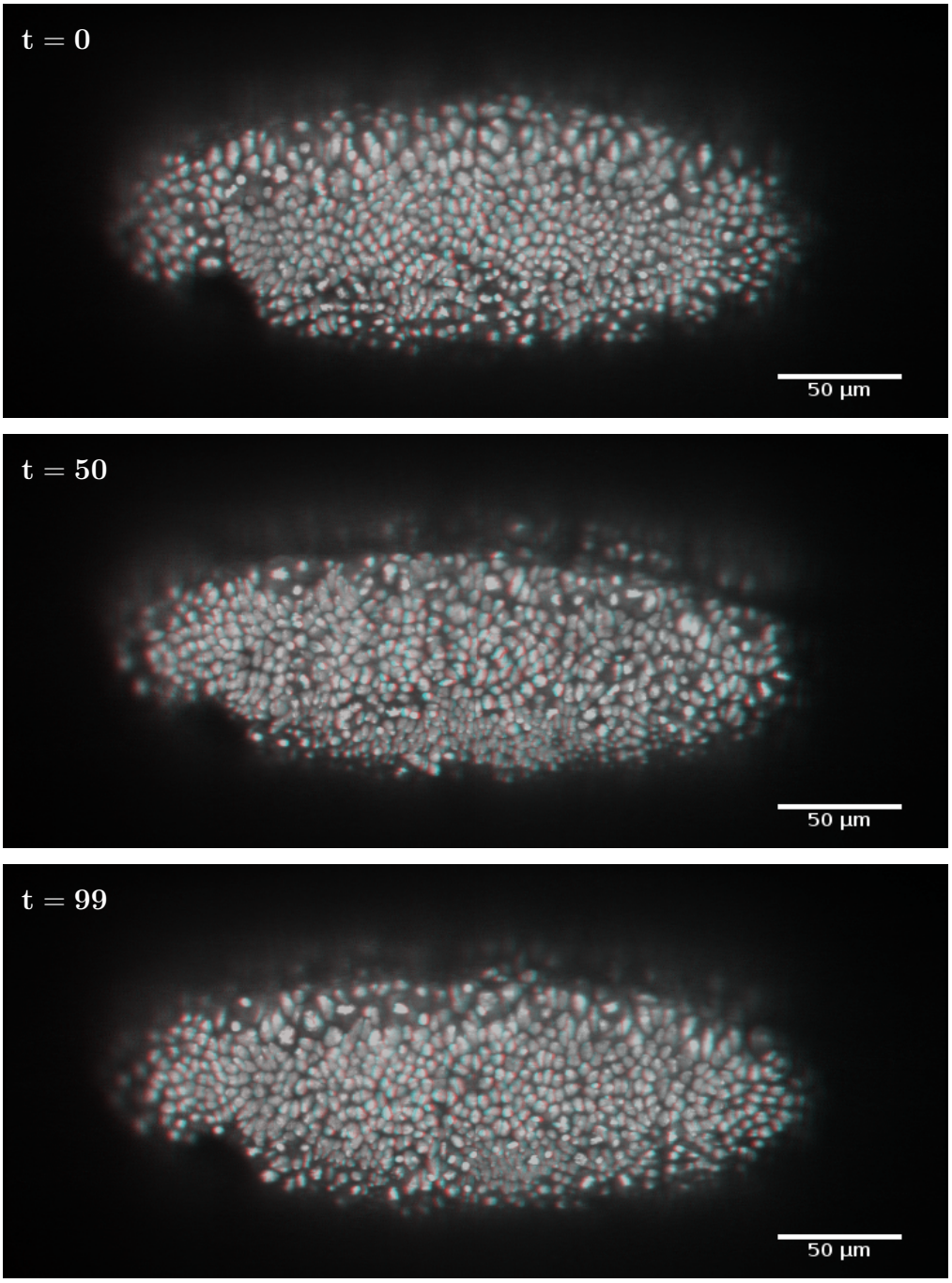


Figure 1.1: Developing *Drosophila* embryo during gastrulation. Depicted are three frames of the dataset described in Appendix A, which we will use for evaluation in the next chapters. The goal is to automatically find correspondences between all cells in the embryo. The image sequence was acquired with a recent light sheet microscope (Krzic et al. 2012). Best viewed in 3D with red/cyan glasses, red side left.

1.1 Challenges for Computer Vision

With all the improvements made in image acquisition in recent years, it is now essential to develop automated frameworks for cell tracking which yield high-quality cell lineage reconstructions. However, automated cell tracking is a highly challenging computer vision problem due to the properties outlined next. Moreover, the problem contrasts strongly from well established tracking applications (Yilmaz et al. 2006) such as those from military target tracking (Luo et al. 2002), or vehicle or pedestrian tracking (Enzweiler and Gavrila 2009), to name but a few.

First of all, **extremely high tracking accuracy is required** to allow for biological insights from the in-depth analysis of the results: Each false assignment of cells between a pair of frames results in the respective entire subtree of the cell lineage to be associated with the wrong cell, *i.e.* each single error propagates to all succeeding time points. The correctness of each and every assignment (cell-to-cell assignment or cell division) is hence of enormous importance.

Furthermore, when tracking **multiple targets**[2], the quality of the tracking result is directly dependent on the degree to which the following assumptions are fulfilled:

(i) Tracking targets must be distinctive in terms of appearance, *e.g.* unique color, characteristic shape, etc.:
However, as the examples from the embryogenesis development of a *Drosophila* embryo in Figure 1.1 show, cells are almost **indistinguishable** from each other in terms of appearance. In particular, **similar shapes which vary over time**, and almost identical intensity distributions, make all cells highly resemblant to each other. Although an ingenious technique termed *Brainbow* (Livet et al. 2007) has been developed in recent years for neuroscience, in which individual cells are marked with specific fluorescent proteins in order to express random colors (and thus generate a distinctive appearance criterion), this technique would evoke a deterioration in the temporal resolution since multiple channels need to be recorded and thus, *Brainbow* has not been applied in the context of entire digital embryos yet (Höckendorf et al. 2012).

(ii) Targets follow a known or predictable motion model, *e.g.* linear motion, or group motion (*i.e.* the motion of all targets in a neighborhood window is highly correlated):
Generally speaking, we cannot assume that this property holds in the movies we analyze. Whereas often a Brownian motion model is assumed (Meijering et al. 2012), it is yet **unknown which motion patterns** the cells follow during embryogenesis. In fact, in Chapter 2, we observe that during embryogenesis, cells within a proximal neighborhood temporally seem to follow similar motion

[2] In cell tracking, the cells (or cell nuclei) represent the *objects* or *targets* to be tracked.

patterns and we model this by a pre-processing step in Section 2.2.3. However, no such motion patterns apply *in general* and assuming a motion model a priori is risky in these cases.

(iii) The environment or background is not cluttered or noisy, or tracking targets are sparsely distributed:
However, the development of **tens of thousands of cells** during embryogenesis of complex organisms such as fruit fly or zebrafish, result in **dense cell populations** in late developmental stages. As a natural consequence of limited spatial resolution of microscopes, this results in each cell occupying only few pixels[3], with very few background pixels between distinct cells (*cf.* Figure 1.1). Due to autofluorescence, heterogeneous staining, or **low signal-to-noise ratio** (SNR), cells may falsely be clustered together if background between cells cannot be detected clearly. We refer to two or more cells detected in such clusters as falsely **merged cells** (or ***mergers***) in this thesis.

(iv) The sets of pixels which represent the same object in two successive frames must **spatially overlap**, *i.e.* objects migrate slowly relative to the temporal resolution of the movie:
Thanks to the most recent developments in light sheet microscopy (Tomer et al. 2012; Krzic et al. 2012), frame rates of one 3D volume per 10-30 seconds are made possible, and hence, cells between successive frames do spatially overlap.

To summarize, in the cell tracking datasets from developmental biology which we study throughout this work, properties (i)-(iii) are all violated, which makes successful frame-to-frame cell assignments highly dependent on the temporal resolution of the movies (iv). Thus, the most important assumption for successful tracking is that the cells are migrating slowly relative to the temporal resolution of the movie.

In fact, even if *all* properties (i)-(iv) were fulfilled, cell tracking remains an extremely complicated computer vision task, due to the fact that cells undergo mitosis and thus **divide**: Except for early developmental stages in embryogenesis, cell division is highly heterogeneous across individual cells, *i.e.* cells may divide at any point in time. As a result, there exists a **highly variable number of cells** across successive frames. Indeed, even if an expert user provided the exact number of cells in a subset of frames, the detection of cell divisions is still crucial for accurate reconstructions of cell lineages. Furthermore, since every cell must undergo a lengthy cell cycle before its duplication into two daughter cells, cell division is a **rare event** to be observed in time-lapse movies. Thus, only few observations of divisions are available even when long movies of proliferating cells are acquired, which challenges the training of automated cell division detection methods.

[3]In the context of 3D volumes, *pixels* are often termed *voxels*. For brevity, we sometimes use the term *pixel* for both 2D and 3D images.

It is mainly the division of the tracking targets which makes it impossible to apply well established tracking methods from computer vision as those listed above. In particular, multi target tracking is typically formulated as a *network flow* problem (Ahuja et al. 1993; Zhang et al. 2008) which can be solved in polynomial time. However, when it comes to the modeling of target *division*, the constraint matrix of the underlying optimization problem leaves the group of totally unimodular matrices, which lifts the problem into the class of integer linear programming problems, known to be **NP-hard** in the general case, *cf.* Section 2.1. The problem of tracking *dividing* targets, where the targets are *not* cells, has – to the best of our knowledge – to date only been studied in the car industry, namely for tracking of headlights (Rubio et al. 2012), and neuroscience, there known as object tracing (Funke et al. 2012).

Apart from these methodological challenges, the methods to be developed, moreover, need to be **easily approachable by non-professional computer users**. In particular, the high-dimensional data needs to be represented in an accessible way and the methods should only rely on very few auto-descriptive parameters or – even better – all parameters should be automatically learned from user annotations. We tackle the latter in this work by relying on methods from machine learning. However, there is **no dense ground truth** of tracked cells publicly available for late stages in embryogenesis of complex organisms such as fruit fly or zebrafish, from which model parameters could be automatically estimated or on which developed frameworks could be evaluated.

1.2 Tracking-by-assignment Methods for Cell Tracking

While other types of cell tracking approaches are discussed in the subsequent chapters, let us now introduce so-called *tracking-by-assignment* methods which are at the core of this thesis.

Tracking-by-assignment frameworks, sometimes referred to as *data association* (Kachouie et al. 2006; Bise et al. 2011) or *tracking-by-detection* (Magnusson et al. 2014) methods, typically comprise two stages:

(i) *detection* or *segmentation* of object candidates (*targets*) in every frame, and

(ii) a *tracking* or *association* step, in which the objects detected in step (i) are linked over time in order to form consistent tracks.

Here, the definition of *consistent* is application dependent: When dealing with non-

dividing objects, typically only one-to-one object matchings are feasible[4], whereas in cell tracking, where cells may divide into two daughter cells, each target may have up to two successors.

These assignment models are very flexible to explicitly model object properties such as target division. The methods may be formulated as integer linear programs (Bise et al. 2011) or probabilistic graphical models (Kausler et al. 2012b), which allow to exploit methods from discrete optimization or probabilistic modeling.

One major drawback of traditional tracking-by-assignment methods, however, is that they highly rely on the quality of the first stage. Errors introduced in the detection / segmentation step propagate to the second stage, the tracking. Due to low contrast or a low signal-to-noise ratio, errors such as undetected objects or false positive detections, false mergers (a cluster of multiple objects found as only one detection), or falsely split objects (oversegmentation), may occur. We will provide detailed examples for possible errors in Chapter 2, Figure 2.2. Of those only false positive detections can be corrected in the tracking step (Bise et al. 2011; Kausler et al. 2012b) so far, by handling cell candidates as random variables and taking a longer temporal context into account through global optimization over all time steps and variables.

1.3 Goals and Structure of the Thesis

The goal of this thesis is to advance the state-of-the-art of tracking-by-assignment models for cell tracking in the following directions:

- **Segmentation errors should be identified and corrected in the tracking step**. Moreover, for accurate lineage trees, it is important that results are **consistent** over time in the number of cells contained in each frame and detection, while at the same time, **cell division is detected**. To this end, we present an approach termed *Conservation tracking* in Chapter 2.
- To allow for maximal interaction (and **synergy effects**) between the two stages in tracking-by-assignment models, it is beneficial to handle both steps jointly in one model. In response, we present in Chapter 3 a novel approach for cell tracking which ***jointly* optimizes segmentation and tracking** globally over all time steps and space.
- Remaining errors should be spotted *automatically* in order to point the user to assignments which are likely to be erroneous. In Chapter 4, we propose

[4] Appearance or disappearance of an object may be modeled by introducing synthetic appearance or disappearance objects, respectively.

two methods to successfully **estimate ambiguities** in results of tracking-by-assignment models, the first of this kind for cell tracking. In this way, the user can be efficiently **guided for proof-reading**.

- The learning of model parameters from user annotations while taking the structure of the output space into account, is essential to achieve high quality cell tracking results. We propose a novel method in Chapter 5, which **trains *multiple diverse* models** from only one set of annotations in order to present multiple highly likely solutions to the user.

Before delving into these findings, we briefly review the concept of *factor graphs* as graphical representations of probability distributions for later reference.

1.4 Factor Graphs & Log-linear Models

Factor graphs (Kschischang et al. 2001) allow for visualizations of (in)dependence relationships in a probability distribution (Barber 2012, Chapter 4) and are representations for probabilistic graphical models (Koller and Friedman 2010, Chapter 4). In general, an (undirected) factor graph is defined by the tuple $\mathcal{G} = (\mathcal{Y}, \mathcal{F}, \mathcal{E})$ and graphically represents the factorization of a function

$$\tilde{p}(\mathcal{Y}) = \prod_{i=1}^{M} \psi_i(\mathcal{Y}_{C_i}). \tag{1.1}$$

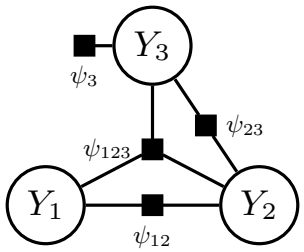
Here, $C_i \subset \{1, ..., N\}$ is an index set, $i \in \{1, ..., M\}$, $\mathcal{Y} = \{Y_1, ..., Y_N\}$ are variables, and $\mathcal{F} = \{\psi_1, ..., \psi_M\}$ are factors, both are nodes in the bipartite graph denoted by circles and squares, respectively. An undirected edge $E_{\nu j} \in \mathcal{E}$ is drawn between variable node $Y_\nu \in \mathcal{Y}$, $\nu \in \{1, ..., N\}$ and factor node $\psi_j \in \mathcal{F}$ if $\nu \in C_j$. Often, factor graphs represent a probability distribution

$$p(\mathcal{Y}) = \frac{1}{Z}\tilde{p}(\mathcal{Y}), \tag{1.2}$$

where the normalization constant Z is assumed implicitly. We refer the reader to (Koller and Friedman 2010; Wainwright and Jordan 2008; Barber 2012; Bishop 2006) for thorough introductions to probabilistic graphical models.

In this work, we study factor graphs for cell tracking and assume *log-linear* models (Koller and Friedman 2010, Chapter 4), *i.e.* factor graphs for which each factor $\psi_i(\mathcal{Y}_{C_i})$ (notation as in Equation (1.1)) can be written in the form

$$\psi_i(\mathcal{Y}_{C_i}) = \exp\left(-E_i\left(\mathcal{Y}_{C_i}\right)\right), \quad \textit{i.e. } E_i\left(\mathcal{Y}_{C_i}\right) = -\log\left(\psi_i(\mathcal{Y}_{C_i})\right), \tag{1.3}$$



Factor $\psi_{12}(Y_1, Y_2)$:

States of		**Indicator function** $\rho(Y_1, Y_2) =$				**Energy/Cost**
Y_1	Y_2	$[\rho^{0,0}_{\{1,2\}}$	$\rho^{0,1}_{\{1,2\}}$	$\rho^{1,0}_{\{1,2\}}$	$\rho^{1,1}_{\{1,2\}}]^\top$	$\boldsymbol{\theta}_{12} = \boldsymbol{\theta}_{12}(\mathbf{w}) =$
0	0	1	0	0	0	$\theta^{0,0}_{12} = \mathbf{w}^\top \phi_i(0,0)$
0	1	0	1	0	0	$\theta^{0,1}_{12} = \mathbf{w}^\top \phi_i(0,1)$
1	0	0	0	1	0	$\theta^{1,0}_{12} = \mathbf{w}^\top \phi_i(1,0)$
1	1	0	0	0	1	$\theta^{1,1}_{12} = \mathbf{w}^\top \phi_i(1,1)$

Figure 1.2: **(Left)** Factor graph for the function $\tilde{p}(Y_1, Y_2, Y_3) = \psi_3(Y_3) \cdot \psi_{12}(Y_1, Y_2) \cdot \psi_{23}(Y_2, Y_3) \cdot \psi_{123}(Y_1, Y_2, Y_3)$. Variables Y_1, Y_2, Y_3 are indicated by circles, factors ψ_3, ψ_{12}, ψ_{23}, ψ_{123} are depicted by black squares. A factor graph visualizes the decomposition of a function, and hence, (in)dependence relationships of random variables in probability distributions. Factors of order one (*e.g.* ψ_3) are called *unary factors*, those of order two (*e.g.* ψ_{12}, ψ_{23}) are termed *pairwise factors*, and those of order ≥ 2 are generally referred to as *higher-order factors*. **(Right)** As an example, we illustrate our notation through the pairwise factor $\psi_{12}(Y_1, Y_2)$. Indicator variable $\rho^{k,l}_{i,j} = 1$ if, and only if, $Y_i = k$ and $Y_j = l$, zero otherwise. Costs θ^{y_1,y_2}_{12} are linear in the joint features $\phi_i(y_1, y_2)$ where y_1, y_2 are the realizations of Y_1, Y_2.

with linear energy

$$E_i\left(\mathcal{Y}_{C_i}\right) = E_i\left(\mathcal{Y}_{C_i}; \mathbf{w}\right) = \boldsymbol{\theta}_i^\top(\mathbf{w}) \cdot \rho(\mathcal{Y}_{C_i}), \tag{1.4}$$

where $\rho(\cdot)$ are indicator functions yielding one binary indicator variable per possible configuration of $\mathcal{Y}_{C_i}$, and $\boldsymbol{\theta}_i(\mathbf{w})$ are the costs associated with factor ψ_i. Note that $\boldsymbol{\theta}_i(\cdot)$ themselves may be linearly parameterized through the *joint feature matrix* $\phi_i(\mathcal{Y}_{C_i})$ and the coefficients $\mathbf{w}$ and thus,

$$E_i\left(\mathcal{Y}_{C_i}\right) = \boldsymbol{\theta}_i^\top(\mathbf{w}) \cdot \rho(\mathcal{Y}_{C_i}) = \mathbf{w}^\top \phi_i(\mathcal{Y}_{C_i}) \rho(\mathcal{Y}_{C_i}). \tag{1.5}$$

The coefficients $\mathbf{w}$ are called the *parameters of the graphical model*.

Consequently, log-linear models may be rewritten as

$$\tilde{p}(\mathcal{Y}) = \prod_{i=1}^{M} \psi_i(\mathcal{Y}_{C_i}) = \prod_{i=1}^{M} \exp\left(-E_i\left(\mathcal{Y}_{C_i}\right)\right) = \exp\left(\sum_{i=1}^{M} \mathbf{w}^\top \phi_i(\mathcal{Y}_{C_i}) \rho(\mathcal{Y}_{C_i})\right). \tag{1.6}$$

A toy example for a factor graph with this type of parameterization is shown in Figure 1.2.

Strategies to tune the parameters $\mathbf{w}$ include manual tweaking, grid search over a pre-defined parameter range, unstructured learning (*i.e.* ignoring factors of order ≥ 2), *maximum likelihood estimation* (Koller and Friedman 2010, Chapter 20), or *maximum-margin learning* for structured models (Taskar et al. 2004), also known as

structured output learning (Tsochantaridis et al. 2005). While the first two methods are only applicable for small sets of parameters due to the highly time-consuming user or machine interaction needed, respectively, unstructured learning ignores the structure of the graphical model in the training of (flat) classifiers. Maximum likelihood based methods, in turn, train the model to *fit* the data. In contrast, in the structured learning framework, a discriminant function is learned to minimize the expected generalization error on unseen data while considering the entire structure described by the graphical model.

We propose cell tracking models in Chapters 2 and 3 in the form of factor graphs. They model a-priori knowledge in terms of linear constraints which yield a *structured output space*, *i.e.* only solutions which fulfill these linear constraints are feasible. *Structured* learning is hence a natural choice for the learning of the model parameters in cell tracking. While we use the time-consuming grid search in the experimental evaluations in Chapters 2 and 3 (the chapters focus on modeling rather than optimization/learning), we introduce *structured learning* strategies for our models in Chapters 4 and 5 in order to find parameters which maximally discriminate *correct* cell assignments in successive time steps from *any other* (false) solution. Further details on structured learning algorithms are discussed in Chapter 5.

2

Conservation Tracking

As pointed out in Section 1.2, multi-object tracking in general may be implemented as a two-step pipeline consisting of a detection/segmentation step and a data association or assignment/tracking step. Such approaches, however, are obviously susceptible to errors in the detection step which are propagated to the tracking model and typically cannot be corrected downstream. Therefore, the ultimate goal of data association tracking is to address detection and data association *jointly* such that both steps can maximally benefit from each other and information can be propagated from more to less obvious parts of the data. There are first approaches addressing joint detection and tracking (Wang et al. 2009; Wu et al. 2012), but none of them has been extended to deal with *dividing* objects. Given that the tracking of multiple dividing objects already is an NP-hard problem (Sahni 1974) in itself, joint detection and assignment is harder still.

As a first step into this direction, in this chapter, we propose a model that handles detection errors *explicitly* in the tracking step and can even correct most of them. Typical segmentation errors are depicted in the lower rows in Figure 2.1 and can be categorized into over- and undersegmentation errors occurring due to low contrast or noise in the images. Furthermore, a real data example is given in Figure 2.2. Oversegmentation may result in false detections whereas undersegmentation could lead to the appearance and vanishing of tracks or to accidental track merging. In this context, the divisibility of the objects is particularly challenging since demerging due to previous merging must be distinguished from object division. Note that we will differentiate between object *division* and object *demerging* throughout this chapter.

Contributions We present in this chapter[5] the first method which explicitly models *all* of the potential *segmentation errors* outlined above in one probabilistic graphical

[5]This chapter is an extended version of (Schiegg et al. 2013).

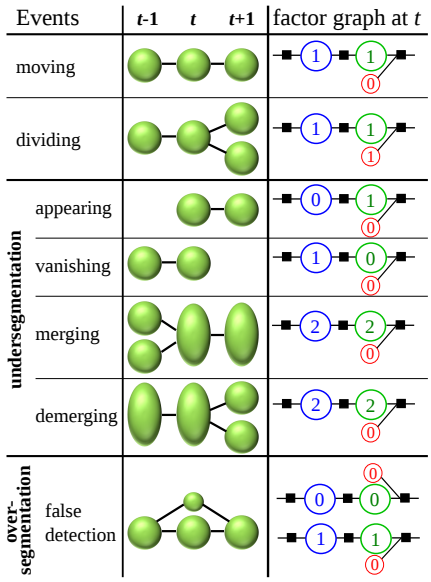


Figure 2.1: The proposed tracking-by-assignment model accounts for all of these events. Left column: Objects (represented as balls) are associated (edges) with each other over three time steps. Right: Excerpt of the proposed factor graph showing the three detection variables for the connected component at time t: Red variables are indicators for a division event. The other variables, taken together, represent the number of targets covered by a detection but they can also represent the other depicted scenarios such as disappearance or "demerging". See Figure 2.4 for more details.

model. The proposed factor graph models conservation laws for the number of objects contained in each detection to ensure *global consistency* of the solution. In other words, the model not only assures consistency between pairs of frames but can also resolve segmentation errors which only become evident from considering a complete time series at once (*cf.* Figure 2.2). In this way, temporarily merged targets can be *resolved under identity preservation* even for objects which are merged during longer sequences. For this purpose, a spatial Gaussian mixture model with the appropriate number of components is fitted to the undersegmented regions.
Object properties such as velocity are easy to represent in state space models, but notoriously difficult to model in a tracking-by-assignment approaches. To avoid bias towards slow motions and false assignments during group movement, we *estimate group motion* in a preprocessing step, using patch-wise cross correlation.

Structure The remainder of this chapter is structured as follows. We commence with the review of prior art and propose the tracking framework – and particularly the construction of the factor graph – in Section 2.2. To showcase the usefulness

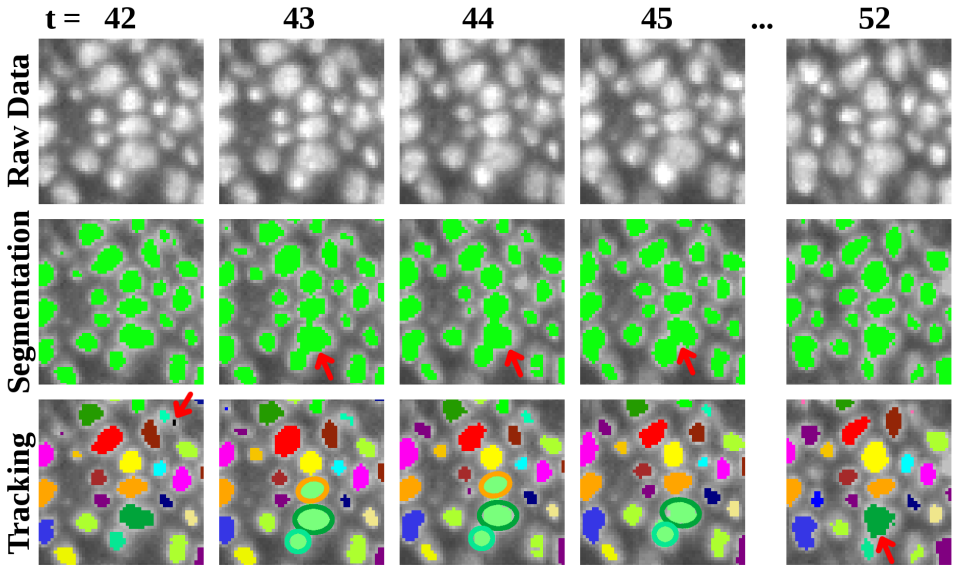


Figure 2.2: Tiny excerpt of dataset B with its almost indistinguishable objects. A short sequence of the raw data is depicted as 2D slices (top row) from 3D+time data and displays cells in a developing *Drosophila* embryo. Due to low contrast, multiple cells are segmented as only one connected component (undersegmentation) as pointed out in the middle row. Our tracking model (bottom row) can handle such errors and preserves the target identities as indicated by colors (see the three previously merged cells in $t = 52$) by fitting the correct number of Gaussians (ellipses) to detections containing multiple objects. Furthermore, the proposed factor graph can handle false detections (oversegmentation) as indicated by the black detection in frame 42 (bottom row).

and efficiency of this novel cell tracking approach, we finally present and discuss experiments on three challenging 2D+t and 3D+t datasets in Section 2.3.

2.1 Related Work

Existing tracking approaches can broadly be categorized into three:

(i) space-time segmentation,

(ii) state space models, and

(iii) tracking-by-assignment.

The first is only applicable at frame rates that make for small, ideally sub-pixel, displacements of objects between subsequent images.

State space models or Bayesian sequential filtering are not easily applicable to an unknown or variable number of objects, calling for costly strategies such as reversible jump MCMC or Gibbs sampling on Dirichlet process mixture models (Fox et al. 2006). In such a setup, dealing with *divisible* objects is harder still.
Tracking-by-assignment gracefully handles multiple, and even dividing objects; on the downside, object properties such as object velocity need to be implemented using factors that are higher-order in time.

We draw inspiration from, and build on, a series of excellent papers. The tracking of undersegmented objects was first described in (Nillius et al. 2006) and soon extended to deal with fragmentation (false positive detections) (Bose et al. 2007). Ben Shitrit et al. (2011) additionally address object identity preserving for possible occlusions of objects by exploiting global appearance constraints. Furthermore, Lou and Hamprecht (2011) account for both dividing objects and undersegmentation, and exploit local evidence in pairs of frames to find undersegmented objects. Their model, however, does not guarantee consistency over all time steps and detections.

The structure of our graphical model also builds on the network flow formulation in (Zhang et al. 2008). Note, however, that allowing for object division no longer permits to do inference via an ordinary network flow computation as in (Zhang et al. 2008). Instead, admitting divisions necessarily turns the problem into an integer flow problem with homologous arcs (*i.e.* flow along separate edges is required to be identical) and a constraint matrix which is not totally unimodular. Hence, there is no guarantee to obtain integer solutions in a network flow, and rounding (Padfield et al. 2009) gives only approximate solutions to a problem that is in general NP-hard (Sahni 1974).

Moreover, the only model which handles the tracking of dividing objects in a global probabilistic framework is the graphical model presented in (Kausler et al. 2012b). While oversegmentation is addressed in terms of false detections, it cannot deal with undersegmentation such as merged objects.

The method described in (Magnusson et al. 2014) – published shortly after the original publication of this chapter (Schiegg et al. 2013) – is most similar to our approach. They furthermore formulate their model as a dynamic programming problem which allows them to solve the underlying optimization problem approximately using a greedy strategy using heuristic rules.

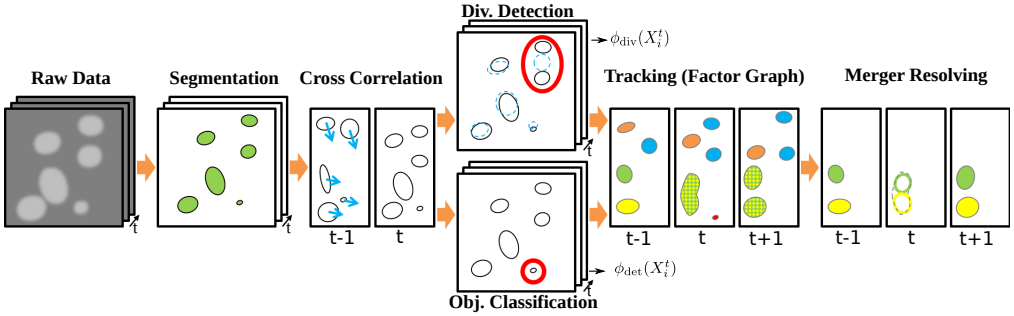
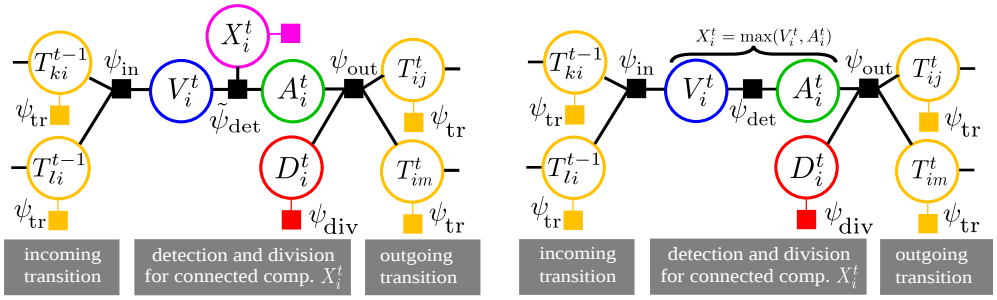


Figure 2.3: Objects are first detected from raw data by segmentation. Subsequently, on pairs of frames, patch-wise cross correlation on the binary images yields rough estimates about the displacement of groups of objects. Following this, probabilistic classifiers determine the unary potentials of each detection, *i.e.* they estimate the division probability and a probability mass function of the number of objects contained in each detection. These potentials are then used in the proposed factor graph (*cf.* Figure 2.4) to find a *globally consistent* tracking solution (here, tracks are indicated by colors). In the last step, detections which were found to contain more than one object (yellow/green in this example) are partitioned by fitting a spatial Gaussian mixture model with the respective number of kernels, and the demerged objects are being tracked again in order to find their original identities.

2.2 Tracking Divisible Objects in spite of Over- and Undersegmentation

The purpose of this work is to track dividing objects based on an error-prone segmentation. We therefore model data association in a probabilistic graphical model (Koller and Friedman 2010) where we explicitly handle over- and undersegmentation errors (*cf.* Figure 2.1). Here, the key idea is that all detections over all time steps are handled simultaneously in a holistic graphical model on which inference is performed globally. In this way, each segmented region is assigned the number of objects it contains while conservation laws across subsequent detections guarantee *global* consistency. Finally, each detection is partitioned into its inferred number of objects by fitting a Gaussian mixture model such that post-hoc linking yields identity preservation for temporarily merged targets. It should be noted that we distinguish between the terms *object* and *detection* which denote one target and one connected component, respectively, where a detection may comprise multiple objects. In the following, we describe our tracking workflow in detail for which a schematic overview is depicted in Figure 2.3.



(a) Full representation for one detection **(b)** Simplified representation

Figure 2.4: Factor graph for one detection X_i^t with two incoming and two outgoing transition candidates: (a) One detection is represented by three multi-state variables, X_i^t, V_i^t, and A_i^t, where X_i^t keeps the number of objects present in the corresponding detection, and the latter variables indicate whether objects are vanishing or appearing, respectively. Note that, since X_i^t is given by a deterministic function of A_i^t, V_i^t, it can be omitted in the simplified representation in (b). Furthermore, the binary variable D_i^t indicates whether object X_i^t is about to divide. See Figure 2.1 for different configurations of these variables. Moreover, transition variables $T \in \{0, ..., m\}$ indicate how many objects are associated between two respective detections. Here, the black squares implement conservation laws, *i.e.* the sum of the left-hand side must equal the sum of the right-hand side, whereas colored squares represent unary factors of the variables.

2.2.1 Graphical Model Implementing Global Conservation Laws

We design a factor graph (Kschischang et al. 2001), illustrated in Figure 2.4, which contains three categories of variables: *Detection* variables (comprising appearance and vanishing variables) for each connected component X_i^t from the segmented image, *division* variables which indicate whether an object is about to divide, and *transition* variables that associate detections in two adjacent time frames with each other.

In particular, each detection X_i^t is represented by an appearance variable $A_i^t \in \{0, ..., m\}$ and a vanishing variable $V_i^t \in \{0, ..., m\}$, where m is the maximal number of objects contained in one detection. The number of objects comprised by detection X_i^t is given by their maximum $X_i^t = \max(V_i^t, A_i^t)$.[6] The appearance and

[6]Note that X_i^t is given by a deterministic function of A_i^t, V_i^t and is hence omitted from the model in Figure 2.4(b).

vanishing variables of one detection are connected by factor ψ_{det} with energy

$$E_{\text{det}}(A_i^t, V_i^t, f_i^t) = \begin{cases} -\ln\left(\hat{P}(X_i^t = k \mid f_i^t)\right), & V_i^t = A_i^t = k \\ -\ln\left(\hat{P}(X_i^t = k \mid f_i^t)\right) + k w_{\text{app}}, & V_i^t = 0, A_i^t = k > 0 \\ -\ln\left(\hat{P}(X_i^t = k \mid f_i^t)\right) + k w_{\text{van}}, & V_i^t = k > 0, A_i^t = 0 \\ \infty, & \text{otherwise} \end{cases}, \tag{2.1}$$

where $k \in \{0, ..., m\}$. In other words, only three different kinds of configurations are allowed (*cf.* Figure 2.1): $V_i^t = A_i^t = k$ indicates that X_i^t comprises k objects (and X_i^t is a false detection if $k = 0$); $V_i^t = 0,\ A_i^t > 0$ means that the object(s) in X_i^t is/are appearing in this time step (*i.e.* starting a new track); whereas $V_i^t > 0,\ A_i^t = 0$ stands for their disappearance at time t.

Here, the model parameters w_{app} and w_{van} penalize spontaneous appearance and vanishing. $P(X_i^t = k \mid f_i^t)$ is determined by a probabilistic discriminative classifier where f_i^t stands for local evidence. In our experiments, we train a random forest (Breiman 2001) on local features such as the size of the connected component, its mean and variance of intensity, or the standard deviation along the principal components of the detected segment.

Now, we introduce a binary variable D_i^t which indicates whether an object in detection X_i^t is about to divide or not. Again, its unary potential is determined by a probabilistic classifier based on local evidence. In our experiments, we deal with cell tracking and therefore utilize domain specific features for cell division. These include the angle that the two nearest neighbors X_j^{t+1}, X_l^{t+1} at $t+1$ enclose with X_i^t, the mean and variance intensity of X_i^t, and the ratios of the squared distances to X_i^t, the mean intensities, and the sizes of X_j^{t+1} and X_l^{t+1}. For all features where appropriate, the region centers corrected by their cross correlation offset (*cf.* Section 2.2.3) are appended in addition. Division nodes are only added if the respective detection has at least two potential successors in the next time frame and the score from the division detection classifier is above some small threshold.

The third category of random variables in the proposed graphical model, the transition variables $T_{ij}^t \in \{0, ..., m\}$, are added for pairs of detections X_i^t, X_j^{t+1} in two subsequent frames. Their value indicates the number of objects of X_i^t which are assigned to X_j^{t+1} (this can be some or all). Local evidence for pairs of detections X_i^t, X_j^{t+1} is injected by

$$\hat{P}(T_{ij}^t = k \mid d_{ij}^t) = \begin{cases} 1 - \exp\left(-\frac{d_{ij}^t}{\alpha}\right), & k = 0 \\ \exp\left(-\frac{d_{ij}^t}{\alpha}\right), & k \neq 0 \end{cases}, \tag{2.2}$$

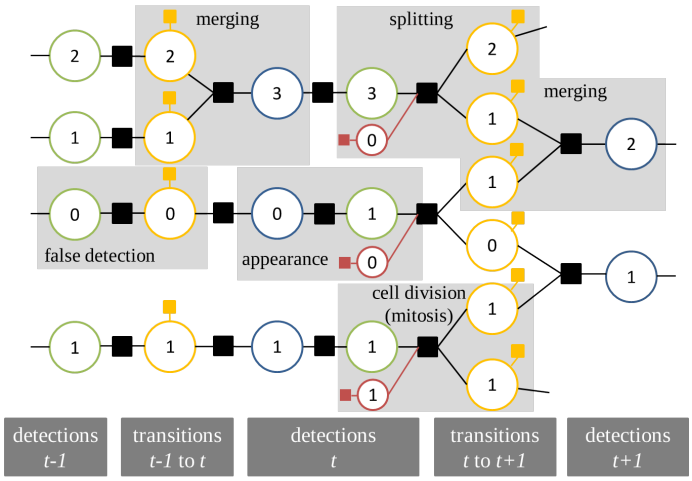


Figure 2.5: Concrete example for the representation of a small but complex toy sequence in terms of the proposed factor graph. The numbers indicate the states that were inferred for each random variable. See Figure 2.4 for color codes.

where $d_{ij}^t = \left(\mathbf{c}_j^{t+1} - (\mathbf{c}_i^t + \mathbf{u}_i^t)\right)^2$ is the squared distance of the region centers $\mathbf{c}$ corrected by the estimated cross correlation offset $\mathbf{u}$ (*cf.* Section 2.2.3), and α is a design parameter. It should be noted that by taking into account the *estimated* rather than the *detected* region centers, acceleration of an object is penalized instead of its velocity.

So far, only probabilistic *local* evidence has been implemented into the model. To ensure *global* consistency of the inferred solution, we augment the factor graph with conservation laws: In particular, the number of objects in detection X_i^t in t must equal to the sum of objects associated with X_i^t in time $t-1$ and $t+1$ (while taking object divisions into consideration). The conservation laws which guarantee these equivalences are implemented in the black squares in Figure 2.4, where, broadly speaking, the sum of objects on the left-hand side must equal the sum of the right-hand side. For instance, the conservation law for the outgoing transitions of X_i^t is expressed through factor ψ_{out} with energy

$$E_{\text{out}}(A_i^t, T_{ij_0}^t, ..., T_{ij_{n'}}^t) = \begin{cases} \infty, & \sum_{l \in \{j_0,...,j_{n'}\}} T_{il}^t \neq A_i^t + D_i^t \\ \infty, & \exists l \in \{j_0, ..., j_{n'}\} : T_{il}^t > A_i^t \\ \infty, & \sum_{l \in \{j_0,...,j_{n'}\}} T_{il}^t \neq 2 \text{ if } D_i^t = 1 \\ \infty, & A_i^t \neq 1 \text{ if } D_i^t = 1 \\ 0, & \text{otherwise} \end{cases} . \tag{2.3}$$

A toy example of our model over three time steps is depicted in Figure 2.5.

In addition, since sparse objects may lead to isolated (sub-)paths in the graphical model, *i.e.* paths where only one transition between two detections is possible, we subsume variables in such paths in *tracklets* and set their unary potential to the sum of the single detections' unaries plus their transition potentials for each possible configuration. In this way, whole tracklets may be treated just like single detections, which leads to major speed-ups in optimization.

Finally, the approximate maximum a-posteriori (MAP) solution of the proposed factor graph can be found using standard message passing algorithms. Alternatively, by minimizing the energy

$$\begin{aligned} &\operatorname*{argmin}_{\mathcal{A},\mathcal{V},\mathcal{D},\mathcal{T}} E(\mathcal{A},\mathcal{V},\mathcal{D},\mathcal{T}) = \\ &\operatorname*{argmin}_{\mathcal{A},\mathcal{V},\mathcal{D},\mathcal{T}} \sum_t \sum_i \left(E_{\text{det}}(A_i^t, V_i^t) + E_{\text{div}}(D_i^t) + \sum_j E_{\text{tr}}(T_{ij}^t) \right), \end{aligned} \tag{2.4}$$

subject to integrality constraints and the linear constraints implicitly given in the potentials ψ_{det}, ψ_{out}, and ψ_{in}, a solution can be obtained using integer linear programming (ILP) solvers. We opt for the latter since the problem can be solved exactly to global optimality for reasonably sized problems.

The energy terms in Equation (2.4) are obtained by reformulating probabilities in the energy domain utilizing the well-known Gibbs distribution $P(X) = \frac{1}{Z} e^{-E(X)}$, where Z is a normalizing factor.

2.2.2 Resolving Merged Objects

The inferred result of the described factor graph yields the number of objects covered by one detection X_i^t and the number of objects T_{ij}^t transferred between two detections X_i^t, X_j^{t+1} in adjacent time steps. Identities of individual objects are amalgamated into a cluster whenever undersegmentation leads to seeming mergers. To recover individual identities, we introduce the following model based on the inferred configuration of the factor graph.

Given the number of objects k contained in detection X_i^t, we fit a Gaussian mixture model with k normal distributions $\mathcal{N}(\mu_l, \Sigma_l)$ of unknown weight π_l to the connected component with pixels/voxels $\{\mathbf{p}_1, ..., \mathbf{p}_n\}$, *i.e.* we maximize

$$P(\mathbf{p}_1, ..., \mathbf{p}_n) = \prod_{j=1}^{N} \sum_{l=1}^{k} \pi_l P(\mathbf{p}_i \mid \mu_l, \Sigma_l). \tag{2.5}$$

The resulting clusters with means μ_l, $l \in \{1, ..., k\}$ are then treated as separate

detections with centers μ_l. Next, another factor graph as described in Section 2.2.1 is constructed comprising all newly resolved and all attaching original detections. We modify this *merger resolving factor graph* by setting all $A_i^t = V_i^t = 1$, *i.e.* in this post-processing step, we disallow appearance, vanishing, and false detections. This graphical model is again solved to global optimality and its solution hence preserves identities of objects, even for long sequences of merged objects.

2.2.3 Cross Correlation for Region Center Correction

Most tracking-by-assignment approaches penalize displacements of objects in terms of squared distance between objects of adjacent time frames. However, if a group of objects is moving rapidly in the same direction, this approach may lead to false assignments due to temporal aliasing. On this account, we adapt our model to penalize acceleration rather than velocity.

Before constructing the factor graph described in Section 2.2.1, we perform patch-wise cross correlation on the binary images on pairs of adjacent frames. In other words, for each patch $g^t(\mathbf{c})$ at time t of a user-specified size, we search for the best match in its neighborhood $h^{t+1}(\mathbf{c})$ at $t+1$ by maximizing

$$\gamma(\mathbf{c}) = \sum_{\mathbf{c}} g^t(\mathbf{c}) h^{t+1}(\mathbf{c} - \mathbf{u}^t) \tag{2.6}$$

to estimate an offset $\mathbf{u}^t$ for each pixel/voxel at t. The transition prior ψ_{tr} can then be computed based on the detection centers corrected by those offsets to find the displacement *relative* to the motion of the object's neighborhood.

2.2.4 Implementation

We have implemented the proposed algorithm in C++ and inference is performed using CPLEX. Optimization time is between one and 30 minutes on a current workstation, even for the experiments with large problem sizes presented in the following section. The source code together with a GUI for the complete workflow is freely available on `https://github.com/martinsch/pgmlink`.

2.3 Experiments & Results

Cell tracking is a natural application for the tracking of dividing objects, particularly challenging due to their almost texture-less appearances, which makes them nearly

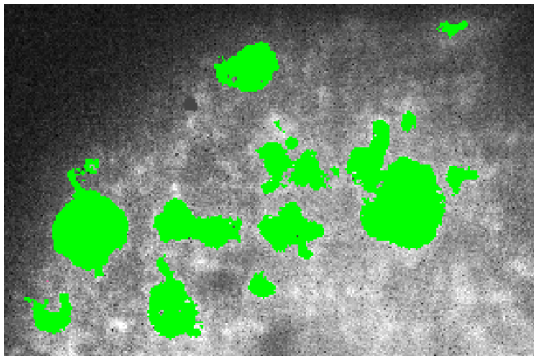


Figure 2.6: An excerpt of one time step of dataset A. Green color indicates detections including many false positives.

indistinguishable from each other. Furthermore, microscope images often suffer from low contrast which typically makes segmentation error-prone. Especially in dense cell populations, undersegmentation is a common cause for errors.

In order to show the efficiency and accuracy of the proposed algorithm, we perform experiments on three challenging datasets, datasets A and B are 3D+t, dataset C is 2D+t. Note that, although both datasets A and B show developing embryos of fruit flies, they are of drastic difference in appearance in terms of contrast. Furthermore, the density of cell populations due to their diverging stages in the developmental process of the embryo are highly differential.

In all experiments, we use random forests (Breiman 2001) each comprising 100 trees grown to purity as classifiers for the cell number in factor ψ_{det} and cell mitosis in factor ψ_{div}. Small training sets (≤ 30 samples for positive classes) are taken from the data. For a fair comparison, we used the same cell number classifier in our method and the competitive model.

First, we evaluate our model on the publicly available dataset from (Kausler et al. 2012b) which shows a *Drosophila* embryo in syncytial blastoderm (**dataset A**). Its segmentation (*cf.* Figure 2.6) consists of ≈ 300 connected components on average over 40 time steps of $2{,}362 \times 994 \times 47$ volumes and shows many false detections. We take the published segmentation of this dataset and its gold standard to compare with the cell tracking model from Kausler et al. (2012b). Their segmentation contains no merged objects and thus, we set in our model the maximal number of objects per detection to one, *i.e.* $m = 1$. In this experiment, we use the cell detection classifier of (Kausler et al. 2012b) and set our parameters to $\alpha = 25$, $w_{\text{app}} = 50$, $w_{\text{van}} = 50$, $w_{\text{tr}} = 13$, $w_{\text{div}} = 28$, where the latter two parameters weight the transition and division priors versus the detection prior. The results of this experiment are given in Table 2.1. Our model yields an f-measure of 0.94 taking all pairwise events, *i.e.*

	Overall: 12,289			**Divisions:** 380		
	Prec.	Rec.	F-meas.	Prec.	Rec.	F-meas.
(Kausler et al. 2012b)	0.96	0.96	0.96	0.92	0.91	0.92
Classifiers only*	*N/A*	*N/A*	*N/A*	0.79	0.69	0.74
Ours ($m = 1$)	0.94	0.95	0.94	0.92	0.88	0.90

Table 2.1: Cell tracking results on dataset A: precision ($= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$), recall ($= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$), and f-measure ($= 2 \cdot \frac{\mathrm{prec.} \cdot \mathrm{rec.}}{\mathrm{prec.} + \mathrm{rec.}}$) for the overall pairwise events (move, appearance, disappearance, divisions) and divisions in particular. For a description of *Classifiers only*, refer to Table 2.2.

moves, appearances, disappearances, and divisions, together, comparable to (Kausler et al. 2012b) (0.96). The f-measure for divisions in (Kausler et al. 2012b) is slightly better than ours, namely 0.92 compared to 0.90, which is due to their model making assumptions about minimal durations between division events, *cf.* (Kausler et al. 2012b, Figure 5).

The second dataset (**dataset B**) again shows a *Drosophila* fruit fly, but this time during gastrulation. Due to the embryonic development, the cell population is now much denser than in dataset B, resulting in a high number of undersegmented objects (*cf.* Figure 2.2). Furthermore, different from dataset A, cells enter mitosis highly heterogeneously in time. In this dataset, on average ≈ 800 cells are tracked over 100 time steps of $730 \times 320 \times 30$ volumes. We refer the reader to Appendix A for a detailed description of this dataset. A gold standard for this dataset has been manually acquired. This dataset is segmented using the segmentation toolkit *ilastik* (Sommer et al. 2011), a pixel-wise random forest classifier for segmentation. The design parameters in our factor graph – for the case of allowing maximally 4 cells in one detection (*i.e.* $m = 4$) – are set to $\alpha = 5$, $w_{\mathrm{app}} = 100$, $w_{\mathrm{van}} = 100$, $w_{\mathrm{tr}} = 24$, $w_{\mathrm{div}} = 36$. A 3D rendering of the resulting trajectories over all time steps is depicted in Figure 2.8(a). The results (*cf.* Table 2.2) show that our method outperforms the cell tracking model in (Kausler et al. 2012b). As indicated by the division f-measure of 0.71 compared to 0.06 of the competitive model, the explicit modeling and distinction of demerging and dividing – together with the probabilistic division prior ψ_{div} – brings a boost in the detection of mitotic events. Besides, due to the consideration of all detections of all frames in one holistic model and due to the conservation laws posed, our factor graph can accurately (precision of 0.78) detect the true number of targets contained in each detection. For this evaluation measure, only detections have been considered which contain more than one cell. Finally, with an f-measure of 0.68, our framework can resolve the original identities of such merged objects. In particular, the associations between the distinct objects after demerging are evaluated as true positives only if they link to the true respective objects before merging – possibly over long sequences of being merged.

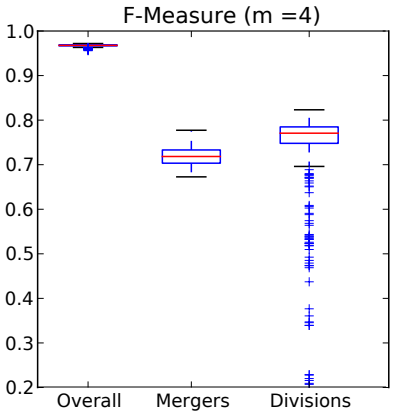


Figure 2.7: Parameter sensitivity: Box-plots for f-measures for dataset C for a search over 720 parameter configurations.

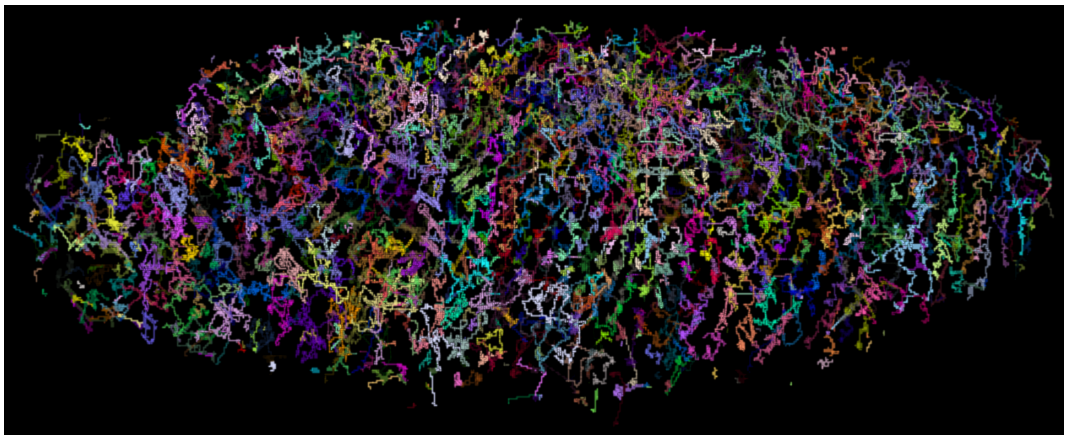
Dataset C is a publicly available 2D+t dataset taken from the Mitocheck project[7] (92 time steps, 1,344 × 1,024 pixels), segmented using the method in (Lou et al. 2012) and a gold standard is acquired manually. In our model, we again treat each connected component as one detection and set the parameters (for $m = 4$) to $\alpha = 5$, $w_{\text{app}} = 100$, $w_{\text{van}} = 100$, $w_{\text{tr}} = 10$, $w_{\text{div}} = 16$. Due to the global mass conservation, our model (f-measure of 0.80 and 0.76 for divisions and mergers) improves significantly over the results of the rather weak local division and merger classifiers (0.70 and 0.49, respectively), *cf.* Table 2.2.

In Figure 2.7 the robustness of the model parameters is addressed for this experiment in the case of $m = 4$ for a search over a reasonable parameter range (720 evaluations). The influence of the parameter setting on the overall result is marginal due to the domination of move events, which are robust to parameters. The results of mergers and divisions seem to depend more on the parameter setting, however, the standard deviation is only 0.02 and 0.08, respectively.

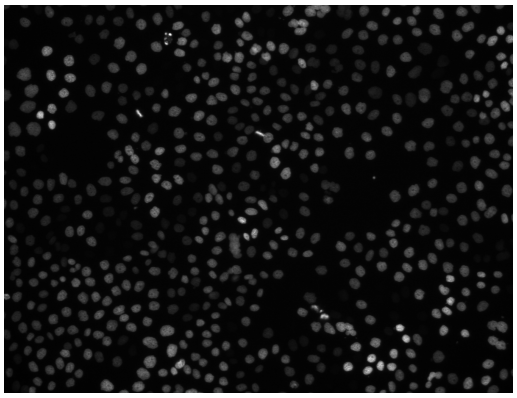
The results of our model can be further improved by designing even more features for object classification and division detection. This additional local evidence can then be put into global context within the factor graph. It should be noted that the object classification and division detection modules can be fully adopted to the particular application domain.

In the **next chapter**, we show that combining both segmentation and tracking *completely* into one holistic model such that the separate steps maximally benefit from each other, yields even superior results.

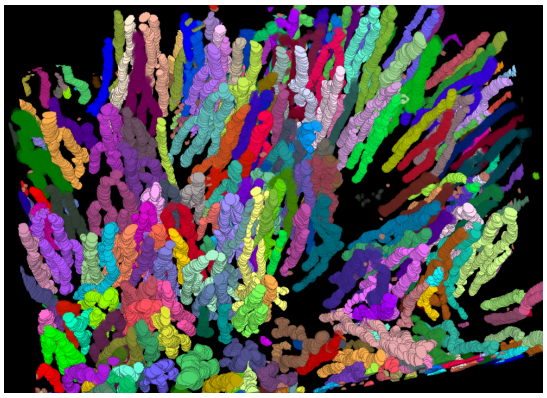
[7] http://www.mitocheck.org/archive/cgi-bin/mtc?action=show_movie;query=87214, last accessed on 09/02/2015.



(a) 2D projection of the 3D trajectories of dataset B (*Drosophila* 3D+t)



(b) One frame of dataset C (Mitocheck 2D+t)



(c) Space-time rendering of the trajectories of dataset C

Figure 2.8: (a) 2D projection of the 3D trajectories of dataset B (*Drosophila*, 3D+t) over all 100 time steps. (b) One frame of dataset C (2D+t) and (c) the 3D space-time rendering of its trajectories. Note that daughter cells inherit the color of their mother cell.

		Moves			Divisions			Mergers			Resolved Mergers		
		Prec.	Rec.	F-measure	Prec.	Rec.	F-measure	Prec.	Rec.	F-measure	Prec.	Rec.	F-measure
Dataset B	(Kausler et al. 2012b)	0.92	0.92	0.92	0.05	0.12	0.06	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
	Classifiers only*	*N/A*	*N/A*	*N/A*	0.83	0.64	0.72	0.63	0.31	0.41	*N/A*	*N/A*	*N/A*
	Ours ($m = 1$)	0.97	0.95	0.96	0.62	0.63	0.63	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
	Ours ($m = 2$)	0.97	0.97	0.97	0.53	0.79	0.64	0.71	0.54	0.61	0.72	0.61	0.66
	Ours ($m = 3$)	0.97	0.97	0.97	0.70	0.76	0.73	0.73	0.58	0.64	0.73	0.63	0.67
	Ours ($m = 4$)	0.97	0.97	0.97	0.65	0.77	0.71	0.78	0.59	0.67	0.74	0.63	0.68
Dataset C	(Kausler et al. 2012b)	0.99	0.97	0.98	0.65	0.68	0.66	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
	Classifiers only*	*N/A*	*N/A*	*N/A*	0.92	0.56	0.70	0.41	0.62	0.49	*N/A*	*N/A*	*N/A*
	Ours ($m = 1$)	0.99	0.97	0.98	0.68	0.71	0.70	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*
	Ours ($m = 2$)	1.00	0.99	0.99	0.85	0.76	0.80	0.73	0.60	0.66	0.79	0.70	0.74
	Ours ($m = 3$)	1.00	0.99	0.99	0.85	0.77	0.80	0.84	0.69	0.76	0.85	0.75	0.79
	Ours ($m = 4$)	1.00	0.99	0.99	0.85	0.76	0.80	0.84	0.69	0.76	0.85	0.75	0.80

Table 2.2: Cell tracking results on datasets B and C: Our model with a different maximal number of objects in one detection ($m = 1$ to $m = 4$) can best handle the under-/oversegmentation errors occurring in these datasets. Here, merged objects are only counted as true positives if the true number (≥ 2) of objects in the connected component is found. Finally, *resolved mergers* indicates, how many of the merged objects have been resolved to their original identities after demerging. (*) Note that in *Classifiers only*, it is only evaluated whether the particular cell is dividing whereas in the tracking models, we go beyond that and additionally require the correct links to the daughter cells. The ground truth of dataset B (dataset C) contains 56,029 (34,985) moves, 216 (440) divisions, 1,878 (1,189) mergers, and 1,466 (533) resolved mergers events.

3

Joint Cell Segmentation and Tracking

As discussed in the previous chapter, the modeling power in tracking-by-assignment approaches comes at the cost of propagating errors from the first stage (detection/segmentation[8]) to the second (tracking), and the overall achievable quality is limited by the lack of interaction between detection and assignment decisions. We addressed this challenge in Chapter 2 by combining a graphical model with a post-processing step to resolve many types of segmentation errors as shown in Figure 2.1.

Our work in this chapter aims at formulating a completely *joint* model for segmentation and tracking for the single stages to maximally interact between each other. Instead of a single fixed segmentation as used in Chapter 2 and other tracking-by-assignment models (Bise et al. 2011; Kausler et al. 2012b), the detection phase generates superpixels/-voxels from which regions (possible cell segmentations) are extracted as sets of the original superpixels. In particular, these regions can be understood as a selection of possible segmentation hypotheses. Global temporal and spatial information guides the selection of those hypotheses that best fit the overall tracking. During inference, each superpixel is assigned either a cell track identifier or the identifier of the background (*cf.* Figure 3.1). Put another way, our algorithm simultaneously produces both, a valid cell segmentation and an assignment of each cell to its cell lineage.

Contributions Our main contribution in this chapter[9] is the formulation of a probabilistic graphical model for *joint* segmentation and tracking for divisible and almost

[8] For brevity, we mostly refer to the combination of detection and segmentation as *detection* only.
[9] This chapter is an extended version of (Schiegg* et al. 2014).

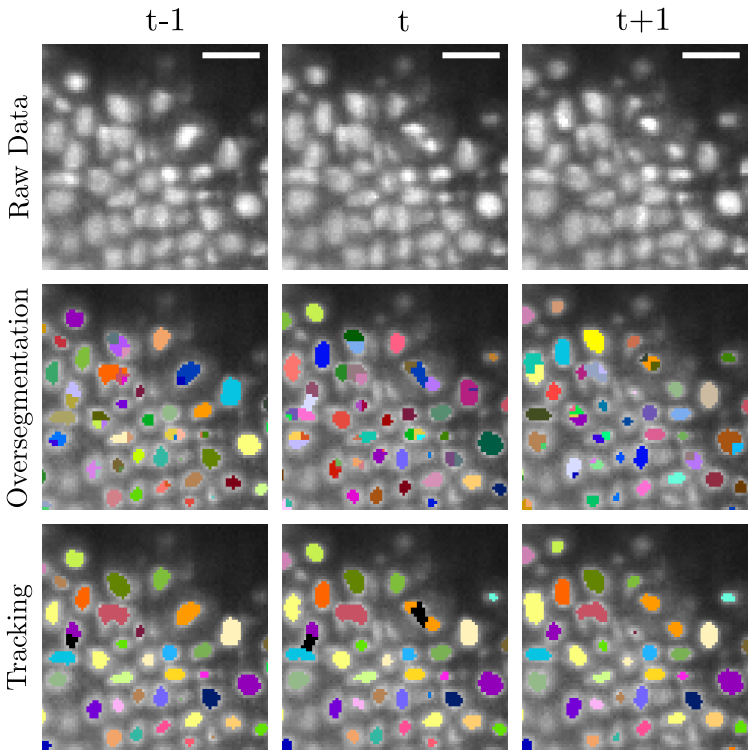


Figure 3.1: An excerpt of three consecutive time steps of the *Drosophila* dataset (2D slices out of 3D volumes). The raw data (top row) is oversegmented into superpixels (middle row). Our graphical model then tracks the cells over time and assigns each segment to a track (indicated by the same random color) or background (black). Offspring cells are assigned the color of their parent cell after mitosis (here: orange). Note that one cell may be represented by multiple superpixels. Scale bars are $10\mu m$.

indistinguishable cells. This undirected graphical model incorporates prior beliefs from multiple local classifiers and guarantees consistency in time and space. We also present a method to generate an oversegmentation which respects the borders between cells and generates an overcomplete set of superpixels even for cells in dense populations. Furthermore, the 3D+t *Drosophila* dataset we use for evaluation, as well as our manually acquired dense annotations are provided to the public. This is the first dataset of this size and kind for which manual annotations are freely available.

Structure We start off with the review of related work in Section 3.1, followed by a detailed description of the proposed pipeline for joint segmentation and tracking in Section 3.2: We develop a method to effectively oversegment a volume of cells, and design the joint graphical model for cell segmentation and tracking. The chapter is concluded in Section 3.3 with a discussion of experiments on a challenging 3D+t

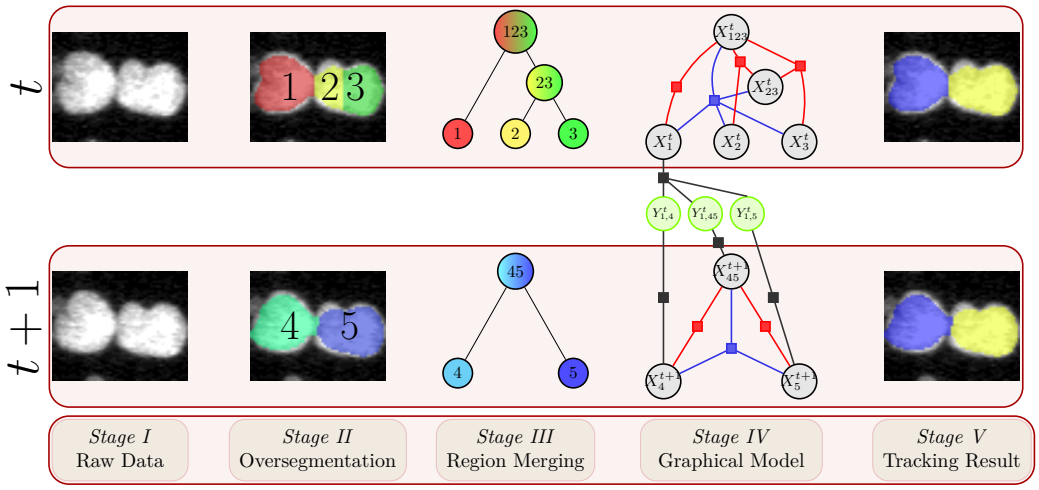


Figure 3.2: First, the raw data is oversegmented in all timesteps separately (stage II). Then, in stage III, segmentation hypotheses are generated by merging adjacent segments into bigger segments (e.g. 2, 3 may be merged into 23). From this structure, a graphical model is constructed (stage IV): Overlapping segmentation hypotheses are connected by intra-frame factors (*red*: conflicting segmentation hypotheses; *blue*: local evidence for the number of cells in one connected component) and inter-timestep transition hypotheses are modeled by binary random variables (green nodes) indicating whether the corresponding cell in t has moved to, divided to, or is not associated with the corresponding cell in $t+1$. Note that, for simplicity, only *one* connected component in only *two* timesteps is visualized. The proposed factor graph in stage IV, in fact, models *all* detections and *all* timesteps in one holistic model at once. Also for simplicity, only a small subset of transition variables is shown. After performing inference on this factor graph, the most probable selection of active regions (actual cells) and their transitions between timesteps are found as visualized by the two cells marked in yellow and blue in stage IV.

cell tracking dataset from *Drosophila* embryogenesis as well as on a 2D+t dataset of proliferating cells in a dense population with frequent overlaps. We show that the proposed method achieves state-of-the-art results.

3.1 Related Work

Joint object detection and tracking is handled naturally in tracking algorithms based on active contours (Xiong et al. 2006), space-time segmentation (Lezama et al. 2011), or video segmentation of multiple objects (Vazquez-Reina et al. 2010; Budvytis et al. 2011). However, these methods either cannot deal naturally with divisible objects and heuristics must be used, or they cannot cope with dense object populations where ob-

jects may overlap. In a very recent study, Amat et al. (2014) present a fast pipeline to simultaneously segment and track cells by propagating Gaussian mixture models through time, but again heuristic rules remain to detect cell divisions. Furthermore, optical flow has been extended to jointly deal with segmentation and tracking (Amat et al. 2013). These authors propose to augment an optical flow algorithm by a regularization term based on similarities of neighboring superpixels modeled in a Markov random field.

In tracking-by-assignment models, however, *joint* optimization of segmentation and tracking is only rarely tackled. Instead, to reduce errors in the final results, errors are minimized in each step of the two-stage tracking-by-assignment separately, the segmentation step and the tracking step: For the former, specialized segmentation approaches for the detection of overlapping objects have been developed (Park et al. 2013; Arteta et al. 2013; Lou et al. 2012). These approaches aim to find most accurate segmentations, however, they do not incorporate any time information. To reduce errors in the *tracking step*, probabilistic tracking-by-assignment methods for dividing objects have been proposed (Bise et al. 2011; Kausler et al. 2012b), which associate a random variable with each detected object to make allowance for false positive detections. This idea has been extended in Chapter 2 to further correct for *under*segmentation errors by introducing conservation constraints between time steps to guarantee a consistent number of objects contained in each detected region. In a postprocessing step, we corrected the original segmentations. Our idea in this chapter goes one step further and aims to avoid segmentation errors already in the first place by *jointly* optimizing segmentation (*i.e.* selection of foreground-superpixels) and tracking.

Most similar to our proposed method are the models in (Funke et al. 2012; Hofmann et al. 2013; Jug et al. 2014). Funke et al. (2012) propose an algorithm which segments an anisotropic 3D volume of branching neurons by generating segmentation hypotheses in 2D slices separately and posing constraints between overlapping segmentation hypotheses. In contrast to our model, the authors do not need to model background for their specific use-case whereas in our domain it is important to infer both whether a segment should be activated as foreground and to which segments in the consecutive timesteps it should be linked. Moreover, in contrast to the model we propose, they do not model detection variables directly; instead they introduce additional transition variables which model appearance, disappearance, and divisions. The explicit modeling of detection variables allows us to pose a prior on the count of cells in connected components. The authors in (Hofmann et al. 2013) propose a similar idea for joint tracking and object reconstruction from multiple cameras. Both methods have in common that they solve an integer linear program with a large set of hard constraints between superpixels within one (time/z-slice) instance and across instances. In independent work, Jug et al. (2014) jointly segment and track bacteria in 1D+t.

The original idea to refine a segmentation by modeling the conflicts between multiple overlapping segmentation hypotheses was introduced by Brendel and Todorovic (2010) and Ion et al. (2011). Whereas Brendel and Todorovic (2010) propose algorithms to efficiently find the best independent sets in a conflict graph, Ion et al. (2011) present a complementary approach to search for maximum cliques in the graph of possible hypotheses (where contradicting tiles are not connected). Their ideas were extended to the temporal domain in (Brendel et al. 2011), but they cannot deal with dividing objects. Extending this idea to *dividing* cells is a much harder problem and the main contribution of this chapter.

3.2 Pipeline for Joint Segmentation and Tracking

The purpose of this work is to segment and track multiple *dividing* cells in a tracking-by-assignment framework. To avoid error-propagation from the segmentation to the tracking stage, we propose to jointly segment and track the targets based on an oversegmentation. This process is illustrated in Figure 3.2: We first run an oversegmentation algorithm on the volumes with overlapping cells to generate multiple segmentation hypotheses. This is followed by the construction of a graphical model for the joint segmentation and tracking. It models competing (intra-frame) relations between the potential cell segmentations which overlap in space, as well as possible inter-frame hypotheses between regions of adjacent timesteps. In this section, we specify each step of this pipeline consecutively, starting with the oversegmentation step.

3.2.1 Competing Segmentation Hypotheses

To make joint segmentation and tracking computationally feasible in tracking-by-assignment approaches, the time-series of 2D/3D images/volumes must be coarse-grained into superpixels/-voxels to reduce the problem space (stage (II) and (III) in Figure 3.2). Note that the resulting superpixels also afford the extraction of more expressive features at the object rather than the pixel level. To this end, first superpixels are obtained which are as large as possible but at the same time small enough to respect all cell boundaries. Next, neighboring superpixels are grouped to generate different segmentation hypotheses. Here, we choose to merge the superpixels in a hierarchical fashion. However, the proposed model does not rely on or exploit the resulting tree structure, so any other means of generating complementary but conflicting segmentations could be used.

Oversegmentation

In stage (II), the purpose is to obtain an oversegmentation on every image which is sufficiently fine but as coarse as possible. That is, we prefer single segments (superpixels) for (isolated) objects without ambiguities, whereas multiple (smaller) segments are desired in cases where objects overlap in space. To this end, we propose the following oversegmentation algorithm:

1. Obtain a coarse segmentation which only distinguishes potential foreground from definite background (high sensitivity, low specificity).

2. Automatically select seeds fulfilling the requirements outlined above.

3. Compute the seeded watershed on the foreground mask.

4. Merge resulting segments hierarchically to potential regions.

Here, the first step may be performed by any segmentation algorithm which can be adjusted in a way that only those pixels are predicted as background where we are sufficiently certain. This step's output is either a hard segmentation or a probability map of the foreground (soft segmentation). Note that typically, it is not desirable to track the resulting connected components directly, since large clusters of cells may be contained in each connected component. Hence, we continue by splitting these connected components into multiple segments. To this end, the watershed algorithm is applied on the probability map of the potential foreground (the *foreground mask* is obtained by truncating probabilities below a chosen threshold; we choose 0.5). The seeds for the watershed algorithm are then the local maxima of the distance transform on the foreground mask. This gives rise to regularly shaped compact segments.

Region Merging

Finally, superpixels are grouped into regions which form possibly competing cell segmentations (stage (III) in Figure 3.2). These segmentation candidates can be generated in very different ways. For simplicity, we choose a hierarchical region merging in a region adjacency graph using L tree levels. Its edge weights between neighboring segments/regions may be arbitrarily complex and the regions may be merged in an order determined by these edge weights.

Since the segmentation hypotheses are composed from the same superpixels, natural conflicts between these regions exist and are resolved by our graphical model (stage (IV) in Figure 3.2) as discussed in the next section.

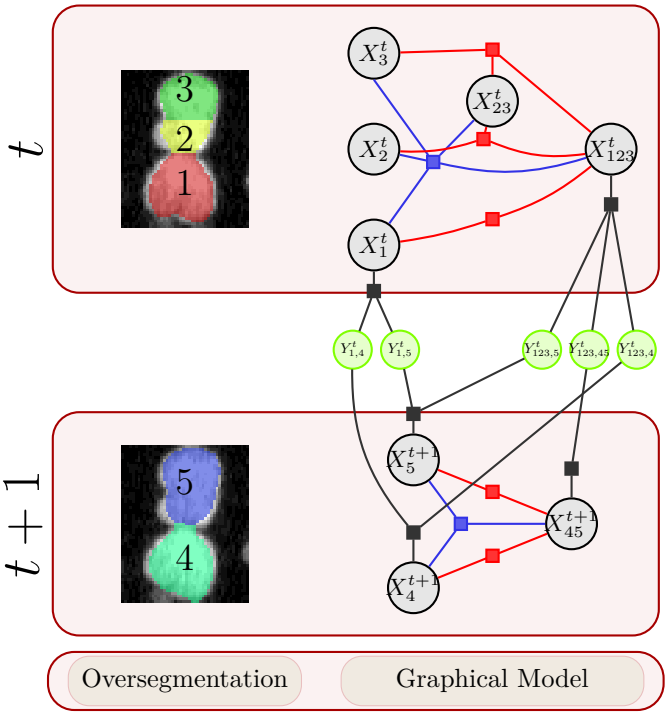


Figure 3.3: Close-up on stage IV from Figure 3.2. In the factor graph, *detection* variables for possible cell segmentations are shown in black while their allowed inter-timestep transitions are modeled by random variables depicted in green (most of them are omitted for clarity). Blue factors give a prior probability to each connected component for how many cells it may contain. By introducing intra-timestep conflict hard constraints (red factors), it is guaranteed that at most only one variable in each conflict set, *e.g.* $\mathcal{C} = \{\{123\}, \{23\}, \{3\}\}$, may be active at a time. Outgoing and incoming factors (black squares) connect inter-frame transition with detection variables and ensure a unique lineage of cells.

3.2.2 Graphical Model for Joint Segmentation and Tracking

Based on the oversegmentation described in Section 3.2.1, a graphical model (here: a factor graph (Kschischang et al. 2001)) is constructed whose factors collect evidence from local classifiers and, at the same time, guarantee consistency due to linear constraints. That is, impossible configurations are disallowed, *e.g.* a cell dividing into more than two children. Building the graphical model corresponds to stage (IV) in Figure 3.2. The construction of the factor graph and the meaning of contained factors and random variables are described in detail in this section. We will refer to the toy example depicted in Figure 3.3 as a running example.

Random Variables

To build the factor graph for joint segmentation and tracking, we first introduce two types of binary random variables, *detection* variables and *transition* variables. In particular, each possible cell segmentation (region) gets assigned a *detection* variable $X_\alpha^t \in \{0,1\}$ with region identifier α and timestep t. Secondly, variables $Y_{\alpha,\beta}^t \in \{0,1\}$ for each possible inter-frame transition between two regions in adjacent timesteps are added. In our illustrative example in Figure 3.2, one detection variable is X_4^{t+1}, referring to region 4 at time $t+1$. $Y_{123,4}^t$ is an exemplary inter-frame transition variable, where the indices mean that region 123 at time t may be associated with region 4 at time $t+1$.

Factors

We continue the construction of our graphical model by adding factors. Factors may disallow specific configurations (see the *constraints* paragraph) and score possible configurations of their associated variables based on estimated probabilities $\hat{P}$ that are here determined by probabilistic classifiers using local evidence f_α^t. In the following, intra-frame factors (detection and count factors) and inter-frame factors (outgoing and incoming factors) are described.

Obviously, all regions in each path from a leaf node to the root node in the region merging graph (see stage (III) of Figure 3.2) form competing segmentation hypotheses and are represented by a conflict set $\mathcal{C}_k^t$ each of which contains indices of such conflicting regions. For each such conflict set $\mathcal{C}_k^t$, a higher order *detection factor* ψ_{det} is added to the graphical model with energy[10]

$$E_{\text{det}}(\mathcal{X}_k^t, \mathcal{F}_k^t) = \begin{cases} -w_{\text{det}} \log\left(\hat{P}_{f_\alpha^t}\left(X_\alpha^t = 1\right)\right), & X_\alpha^t = 1 \\ -w_{\text{det}} \max\limits_{X_\alpha^t \in \mathcal{X}_k^t} \log\left(\hat{P}_{f_\alpha^t}\left(X_\alpha^t = 0\right)\right) + c_{\text{bias}}, & X_\alpha^t = 0 \ \forall X_\alpha^t \in \mathcal{X}_k^t \end{cases}, \tag{3.1}$$

where $\mathcal{X}_k^t = \{X_\kappa^t\}_{\kappa \in \mathcal{C}_k^t}$, $\mathcal{F}_k^t = \{f_\kappa^t\}_{\kappa \in \mathcal{C}_k^t}$ are the detection variables (and their corresponding features) of regions contained in conflict set $\mathcal{C}_k^t$ and w_{det} weighs the detection factor against other factors. Equation (3.1) translates to the following: A prior probability $P_{f_\alpha^t}\left(X_\alpha^t = 1\right)$ obtained from a pre-trained local classifier (see Section 3.2.3 for details) with features f_α^t is transformed into an energy for the configuration where exactly one X_α^t is found to be a true cell. In the second case, none of the regions in the conflict set is a true cell, a penalty has to be paid based on the classifier's belief of each of the regions being false positive detections. The model parameter c_{bias} can put

[10] A factor $\psi(X)$ can be obtained from the given energy $E(X)$ by the following transformation: $\psi(X) = \exp\left(-E(X)\right)$. For the sake of brevity, we will only describe the energies in the remainder of this chapter.

a bias on regions to be activated rather than deactivated in case of doubt. Note that impossible configurations, such as the selection of more than one competing region, are forbidden by constraint $\mathfrak{C}_1$, see Table 3.1. In Figure 3.3, the potential ψ_{det} ideally obtains a high energy (*i.e.* low probability) for the single region 2 being one cell, while region 23 has a low energy since it better represents an entire cell.

Moreover, local evidence is further leveraged by a higher-order *count factor*

$$E_{\text{count}}(\{X_\alpha^t\}_{\alpha\in\mathcal{K}_i^t}) = -w_{\text{count}} \log\left(\hat{P}_{\text{count}}\Big(\sum_{X\in\{X_\alpha^t\}_{\alpha\in\mathcal{K}_i^t}} X = k\Big)\right), \tag{3.2}$$

where $\{X_\alpha^t\}_{\alpha\in\mathcal{K}_i^t}$ denotes the detection variables for all regions belonging to connected component $\mathcal{K}_i^t$ at time t. It injects prior beliefs for each connected component i to contain k actual cells. To this end, a probabilistic count classifier (see Section 3.2.3) is trained using features such as total intensity or size, and applied on connected components. For instance, two active regions are favored for connected component $\mathcal{K}_{123}^t = \{\{1\},\{2\},\{3\},\{23\},\{123\}\}$.

The factors above are both associated with variables from single timesteps only. To achieve temporal associations of cells across timesteps, the model has to be extended by *inter-frame factors* which connect detection with transition variables. Firstly, *outgoing factors* with energy

$$E_{\text{out}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) = E_{\text{dis}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) + E_{\text{move}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) + E_{\text{div}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) \tag{3.3}$$

associate each variable X_α^t with all possible transitions $\mathcal{Y}_{\alpha\rightarrow}^t$ to variables in the successive timestep. This factor decomposes into three energy terms:

1. Disappearance: The termination of a track is penalized by the *disappearance* energy

$$E_{\text{dis}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) = \begin{cases} w_{\text{dis}}, & X_\alpha^t = 1 \wedge \sum_{Y\in\mathcal{Y}_{\alpha\rightarrow}^t} Y = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{3.4}$$

   In other words, cost w_{dis} is charged when a detection variable is active, but all outgoing transition variables are inactive.

2. Cell division: The division energy is given by

$$E_{\text{div}}(\mathcal{Y}_{\alpha\rightarrow}^t) = w_{\text{div}} \cdot \frac{|X_\alpha^t|}{|\{X_\alpha^t\}_{\alpha\in\mathcal{K}_i^t}|} \times \tag{3.5}$$
$$\times \begin{cases} -\log\left(\hat{P}_{\text{div}}\left(Y_{\alpha,\mu_1}^t + Y_{\alpha,\mu_2}^t = 2\right)\right), \text{if } \sum_{Y\in\mathcal{Y}_{\alpha\rightarrow}^t} Y = 2 \ \wedge \ Y_{\alpha,\mu_1}^t, Y_{\alpha,\mu_2}^t = 1 \\ 0, \qquad\qquad\qquad\qquad \text{otherwise} \end{cases},$$

where $\hat{P}_{\text{div}}$ is estimated by the division classifier (*cf.* the *classifier* paragraph). Here, the first case applies if region α divides into regions μ_1 and μ_2, and the second if not. The prefactor $\frac{|X_\alpha^t|}{|\{X_\alpha^t\}_{\alpha \in \mathcal{K}_i^t}|}$ weighs the energy by the ratio of the number of superpixels contained in region α to the total number of those contained in its connected component $\mathcal{K}_i^t$. This weight avoids bias towards large objects since the cost of activating multiple small regions is balanced with the cost of activating one large region.

3. Cell migration: The *transition energy*

$$E_{\text{move}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) = w_{\text{move}} \cdot \frac{|X_\alpha^t|}{|\{X_\alpha^t\}_{\alpha \in \mathcal{K}_i^t}|} \times \tag{3.6}$$
$$\times \begin{cases} -\log\left(\hat{P}_{\text{move}}\left(Y_{\alpha,\mu}^t = 1\right)\right), \text{if } \sum_{Y \in \mathcal{Y}_{\alpha\rightarrow}^t} Y = 1 \ \wedge \ Y_{\alpha,\mu}^t = 1 \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

assigns the cost estimated by the move classifier if, and only if, region α is associated with region μ. The prefactor is justified as above.

The second inter-frame factor, the *incoming factor*, assigns a cost in case a cell appears, *i.e.* X_β^{t+1} is one, but all of the transition variables in $\mathcal{Y}_{\rightarrow\beta}^{t+1}$ are zero. The incoming factor is defined by the energy

$$E_{\text{in}}(X_\beta^{t+1}, \mathcal{Y}_{\rightarrow\beta}^{t+1}) = \begin{cases} w_{\text{app}}, & \text{if } X_\beta^{t+1} = 1 \ \wedge \sum_{Y \in \mathcal{Y}_{\rightarrow\beta}^{t+1}} Y = 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

and connects the detection variable at $t+1$ with all appropriate transition variables coming from the previous timestep t.

Omitted in these factors so far are impossible configurations, such as more than one ancestor or more than two descendants for one cell. These configurations are prohibited by adding the following constraints.

Constraints

We add linear constraints to guarantee that only feasible configurations are part of a solution. Constraints within individual timesteps are referred to as *intra-frame* constraints while *inter-frame* constraints regularize the interaction of detection with transition variables. The constraints are summarized in Table 3.1 and explained in the following.

Since overlapping – and hence conflicting – regions are contained in the segmen-

	Constraint Name	Description	Linear Formulation	ID
	Intra-Frame Segmentation Conflicts	Conflicting (*i.e.* overlapping) regions may not be active at the same time.	$\sum_{\kappa\in\mathcal{C}}^{t} X_{\kappa}^{t} \leq 1$ $\forall \mathcal{C} \in \{\mathcal{C}_k^t\}_{k,t}$	$\mathfrak{C}_1$
Inter-Frame	Couple-Detection-Outgoing	Inter-frame hypotheses may not be active when the corresponding detection variable is inactive.	$Y_{\alpha,\beta}^{t} \leq X_{\alpha}^{t}\ \forall\beta$	$\mathfrak{C}_2$
	Descendants-Outgoing	A region may not have more than two descendants.	$\sum_{\beta} Y_{\alpha,\beta}^{t} \leq 2\ \forall\alpha$	$\mathfrak{C}_3$
	Couple-Detection-Incoming	Inter-frame hypotheses may not be active when the corresponding intra-frame hypotheses are inactive.	$Y_{\alpha,\beta}^{t} \leq X_{\beta}^{t+1}\ \forall\alpha$	$\mathfrak{C}_4$
	Ancestors-Incoming	A region may not have more than one ancestor.	$\sum_{\alpha} Y_{\alpha,\beta}^{t} \leq 1\ \forall\beta$	$\mathfrak{C}_5$

Table 3.1: Linear constraints for random variables

tation hypotheses, constraints need to restrict the space of feasible solutions to non-contradicting solutions. For this purpose, conflicting hypotheses are subsumed into *conflict sets* $\mathcal{C}_k^t$ (Red factors and their associated detection variables in Figure 3.3.). Constraint $\mathfrak{C}_1$ in Table 3.1 ensures that at most one detection variable is active in each conflict set. Taking conflict set $\mathcal{C} = \{\{123\}, \{23\}, \{3\}\}$ in Figure 3.3 as an example, the constraint states: $X_3^t + X_{23}^t + X_{123}^t \leq 1$.

Those intra-frame constraints added, *outgoing* and *incoming* constraints model inter-frame interactions and couple detection variables with transition variables. These constraints ($\mathfrak{C}_2$ and $\mathfrak{C}_4$ in Table 3.1) ensure compatibility of detection and assignment variables: No transition variable may be active if the corresponding detection variable has state zero. In terms of the factor graph in Figure 3.3, this means that, *e.g.* $Y_{123,45}^t \leq X_{123}^t$.

In a similar fashion, constraints $\mathfrak{C}_3$ and $\mathfrak{C}_5$ in Table 3.1 enforce compliance with the tracking requirement that a cell can have at most two descendants and one ancestor, respectively. A feasible tracking solution must fulfill all constraints $\mathfrak{C}_1$–$\mathfrak{C}_5$. It should be noted that only $\mathfrak{C}_3$ needs to be adjusted appropriately if non-divisible objects are to be tracked.

Inference

In our global graphical model, the total energy

$$E(\mathcal{X},\mathcal{Y}) = \sum_t \Bigg(\sum_i \Big(\sum_{k\in\mathcal{C}_i} E_{\text{det}}(\mathcal{X}_k^t) + E_{\text{count}}(\{X_\alpha^t\}_{\alpha\in\mathcal{K}_i^t}) \Big) + \sum_\alpha \Big(E_{\text{out}}(X_\alpha^t, \mathcal{Y}_{\alpha\rightarrow}^t) + E_{\text{in}}(X_\alpha^t, \mathcal{Y}_{\rightarrow\alpha}^{t-1}) \Big) \Bigg) \tag{3.8}$$

subject to all constraints in Table 3.1,

is the sum of all factors over all possible variable configurations of detection variables $\mathcal{X}$ and transition variables $\mathcal{Y}$. Note that $\mathcal{X}$ and $\mathcal{Y}$ contain *all* random variables of *all* time steps taking all information available into account in one holistic graphical model. The probability for a configuration $\mathcal{X}$, $\mathcal{Y}$ is then given by the Gibbs distribution $P(\mathcal{X},\mathcal{Y}) \propto e^{-E(\mathcal{X},\mathcal{Y})}$ and the optimal tracking corresponds to its MAP solution. We solve the energy minimization problem to global optimality by solving the corresponding integer linear program.

After inference, the optimal configuration of the factor graph can be interpreted as a segmentation *and* tracking result as illustrated in stage (IV) in Figure 3.2. The graphical model assigns a track identifier to each foreground superpixel and sets segment values to zero which are inferred to be background.

3.2.3 Local Classifiers

The factors of the probabilistic graphical model introduced in Section 3.2.2 are based on the predictions of local classifiers for

1. the number of cells in a connected component: the *count classifier* is trained based on the appearance (*e.g.* the size, intensity, radius) of a connected component and predicts the number of cells that are contained within. The predictions are then injected into the count factors in Equation (3.2) as prior belief for the number of cells contained in a connected component.

2. true detections: the *detection classifier* estimates how strongly a region resembles a cell (*cf.* Equation (3.1)).

3. cell divisions: the *division classifier* rates the probability of triples of regions, ancestor and two children from consecutive frames, to represent a division.

4. cell migration (moves): the *move classifier* rates every pair of regions associated with a transition variable.

In our implementation, we train random forest classifiers, but any classifier which provides (pseudo-)probabilistic predictions can be used. These classifiers are trained on user annotated training examples.

3.2.4 Implementation Details

In this cell tracking application, we use the following methods and parameters for the oversegmentation algorithm sketched in Section 3.2.1. To obtain a coarse foreground mask, we use the segmentation toolkit ilastik (Sommer et al. 2011) which can segment both the phase-contrast images from the *Rat stem cells* dataset as well as the stained cell nuclei from the *Drosophila* dataset: Here, prediction maps for each timestep are computed independently using a pixel-wise random forest trained on few training examples from the respective dataset. We use 100 trees in every experiment and select the following features at different scales: Gaussian smoothing, Gaussian Gradient Magnitude, Difference of Gaussians, Structure Tensor Eigenvalues, and Hessian of Gaussian Eigenvalues. Then, the seeds are determined by the local maxima of the distance transform on the slightly smoothed foreground mask (Gaussian smoothing with $\sigma = 0.3$ and $\sigma = 1.0$ in the case of *Drosophila* and *Rat stem cells*, respectively) and nearby seeds are pruned by dilating with a disc/ball of radius 2 pixels. Resulting segments are merged hierarchically with edge weights determined by the ratio of the length of their common border and the perimeter of the smaller region. While much more expressive weights could be used here, we find that these simple features already perform sufficiently well. Then, at every level $l \in \{0, ..., L\}$ of the hierarchical segmentation hypotheses (we choose the tree depth $L = 4$ in the 2D+t and $L = 5$ in the 3D+t dataset), edge weights are ordered and the p% best neighbors (in our case, highest weights[11]) are merged iteratively. Here, we set $p = 20$ for $l \in \{0, ..., L-1\}$ and $p = 100$ for $l = L$, which yields the connected components of the foreground mask as the root node of the segmentation hypotheses trees. Our model and implementation is not limited to hierarchical segmentation hypotheses. In fact, any algorithm which generates competing segmentation hypotheses could be used.

The graphical model described in Section 3.2.2 is implemented in C++ using the open-source library OpenGM (Andres et al. 2012). For tractability, the number of inter-frame hypotheses is pruned to a reasonable number of candidates in the spatial proximity of each region: In particular, inter-frame hypotheses between frames t and $t + 1$ are generated by finding the 2 nearest neighbors in $t + 1$ for each region in frame t as well as the 2 nearest neighbors in t for each region in frame $t + 1$. This procedure yields many inter-frame hypotheses ($\gg 2$) in dense cell populations and only few hypotheses in the parts of the image where cells are sparse. To create training

[11] In this way, segments completely contained within other segments are merged first, whereas regions which only touch in few pixels are merged last.

examples for the classifiers, a small subset of the raw data is selected and sparsely annotated to train a random forest (Breiman 2001) for each classifier suggested in Section 3.2.3. We choose 100 trees for each and train the random forests to purity. The parameters of the factor graph are then tuned to best fit a small, fully annotated subset of the data. These parameters are used for the final predictions on the entire dataset to report the performance measures. To do inference on our graphical model, we use the (integer) linear programming solver CPLEX. The globally optimal solution for the entire time sequence is found within $\approx 10 - 70$ minutes. We refer the reader to Appendix B more detailed runtime discussion.

3.3 Results & Discussion

We perform comparative experiments on two datasets – a cell culture (2D+t), and a developing *Drosophila* embryo (3D+t). The former is challenging due to severe mutual overlap while the latter is difficult owing to its ambiguity in the segmentation hypotheses due to high cell density under low contrast.

The first dataset is publicly available from (Rapoport et al. 2011) (their dataset A) and consists of a time-series of 209 images ($1\,376 \times 1\,038$ pixels) of about 240 000 pancreatic stem cells of a *rattus norwegicus* ("Rat stem cells"). This dataset is particularly challenging due to the cells changing their appearance (shape, size, intensity) over time from long elongated to round cells. Moreover, the proliferating stem cells quickly grow to a dense population causing frequent overlaps between cells. Due to the dataset's high temporal resolution, it is difficult to pinpoint a cell division to a specific point in time. Instead, mitosis occurs over multiple timesteps. For this reason, we subsample the sequence in time, processing every second image only (leaving us with 104 time steps) and relax the evaluation criterion for divisions (see Section 3.3.1). We further resample the ground truth provided by (Rapoport et al. 2011) to guarantee that no cell division is lost in the subsampling.

The second dataset is dataset B from Chapter 2, a developing *Drosophila* embryo, described in more detail in Appendix A. On average, about 800 cells are tracked over 100 time steps ($730 \times 320 \times 30$ voxels, voxel resolution $0.5\mu m$). In Chapter 2, we evaluated the conservation tracking method on this dataset conditioned on a given segmentation. To evaluate the performance of the joint approach of segmentation *and* tracking as proposed in this chapter, we extend the manual annotations such that they also cover previously missing cells, and that voxels of falsely merged cells are assigned to individual cell identities.[12] In this way, we can further report segmentation/detection measures in addition to tracking measures *unconditioned* on the

[12]Both the dataset and our manual annotations are freely available online on http://hci.iwr.uni-heidelberg.de/MIP/Research/tracking/.

segmentation result.

3.3.1 Evaluation Measures

In contrast to the typical evaluation of tracking-by-assignment methods, for which an evaluation conditioned on the segmentation is sufficient to determine the efficiency of the tracking algorithm, here, both segmentation and tracking must be compared against a ground truth. To evaluate the segmentation quality, we use the Jaccard index as a similarity measure between a region r_{res} of the result and ground truth region r_{gt}, *i.e.*

$$\zeta(r_{\text{res}}, r_{\text{gt}}) = \frac{|r_{\text{res}} \cap r_{\text{gt}}|}{|r_{\text{res}} \cup r_{\text{gt}}|}. \tag{3.9}$$

The best-matching region

$$r^*_{\text{res}}(r_{\text{gt}}) = \underset{r_{\text{res}}}{\operatorname{argmax}}\, \zeta(r_{\text{gt}}, r_{\text{res}}) \tag{3.10}$$

for some ground truth region r_{gt} counts as a true positive segmentation for that region if its Jaccard index is greater than some threshold τ (we set $\tau = 0.5$)[13]. Unmatched ground truth/tracking result regions are considered false negative/false positive detections.

We then compare the frame-to-frame tracking events (*moves* and *divisions*) from the ground truth to those of the tracking result. We report an *unconditioned* tracking result as well as *conditioned* performance measures. The former evaluates the tracking on the raw data directly, the latter is conditioned on the true segmentation hypotheses. Note that it is often not clear from the raw data, in which exact timestep a cell division is occurring. We hence allow cell divisions to be off from the ground truth by one timestep, *i.e.* a division is still counted as a true positive if it occurs one timestep earlier or later within the same track. Finally, based on the number of true/false positives and false negatives, *precision*, *recall* and *f-measure* are computed for detections, moves, and divisions.

3.3.2 Results for Joint Segmentation and Tracking

To evaluate the performance of our model for joint cell segmentation and tracking (JST), we perform experiments on the two datasets described above. We compare with two state-of-the-art cell tracking algorithms:

[13] For the competitive method (Amat et al. 2014), we dilate the resulting centroids of Gaussians with a disc/ball of radius 5. For evaluation, we then choose $\tau = 0.0$ which accepts matching segmentations already at only 1 pixel overlap.

Dataset **Method**	**Segmentation** Precision	Recall	F-Measure
Rat stem cells (2D+t) (Rapoport et al. 2011)			
(Rapoport et al. 2011)		0.95	
Conservation tracking (CT)	0.75	0.99	0.85
Conservation tracking on oversegmentation	0.79	0.99	0.88
(Amat et al. 2014) on raw data	0.94	0.95	0.94
(Amat et al. 2014) on our prediction maps	0.92	0.95	0.93
Joint segmentation and tracking (JST)	0.99	0.96	**0.97**
Drosophila embryo (3D+t)			
Conservation tracking (CT)	0.82	0.93	0.87
Conservation tracking on oversegmentation	0.77	0.95	0.85
(Amat et al. 2014) on raw data	0.97	0.93	**0.95**
(Amat et al. 2014) on our prediction maps	0.96	0.89	0.93
Joint segmentation and tracking (JST)	0.99	0.88	0.93

Table 3.2: Segmentation quality after tracking (higher is better). Note that in the *joint segmentation and tracking* method proposed in this chapter, segmentation and tracking are optimized concurrently. The *rat stem cells* dataset contains a ground truth of 121 632 cells across all frames, whereas the *Drosophila embryo* data consists of 65 821 true cells.

1. the *conservation tracking* method (CT) proposed in Chapter 2 based on a given segmentation, which can correct for falsely merged cells in a post-processing step. In order to show that the method proposed in this chapter operates on a reasonably fine oversegmentation and that it is not enough to merely track the superpixels in this oversegmentation, we also perform experiments using the conservation tracking method but use an oversegmentation as input. To this end, we set the parameter of maximally allowed cells (parameter m in Chapter 2) in a single detection to $m = 1$. In all three methods, we use the same *count* and *division* classifier, to which in our JST method *move* and *detection* classifiers are added.

2. a cell tracking pipeline designed to track entire embryos (Amat et al. 2014). We evaluate their algorithm on both the raw data directly and our prediction maps as input. Note that their code was primarily designed for 3D+t datasets, and all computations on the 2D *Rat stem cells* movie are internally performed for 3D using one z-slice only; we report these results. Note that stacking duplicates of the 2D images to artificially construct 3D volumes yielded inferior results.

In the 2D+t dataset, we furthermore compare with the results of (Rapoport et al. 2011) for the quantitative results reported there.

Segmentation Quality

We first investigate the quality of cell segmentations, see Table 3.2 for results. Note that in both methods CT and JST, cell candidates may be set inactive by the graphical model. In both datasets, the JST method outperforms the segmentation quality of the CT model from Chapter 2 with an f-measure of 0.97 and 0.93 compared to 0.88 and 0.87. Since our JST model groups superpixels into cells or deactivates them, it is not crucial in this approach whether cell candidates (or superpixels) are touching in the segmented image. In the CT method, in contrast, the complexity of the model is determined by the worst case cluster size, *i.e.* the number of potentially merged cells. Hence, in the CT approach, the need for correctly segmented individual cells leads to parameter settings that in turn make for many false negatives in the segmentation. We consider it a strong advantage of our JST method to deal with competing segmentation hypotheses rather than repairing a fixed segmentation. Moreover, Rapoport et al. (2011) achieve on the *Rat stem cells* data a recall of 0.95 (they do not report precision), whereas our JST method obtains a recall of 0.96 under very high precision (0.99). Note that (Rapoport et al. 2011) use $\tau = 0.3$ (*cf.* Section 3.3.1) whereas we set $\tau = 0.5$ as a stronger criterion. Amat et al. (2014) achieve similar or slightly better detection accuracies on the 3D+t dataset. Their detection accuracy on the 2D+t dataset only increases in the course of the movie, seemingly due to the following reasons: The cells adopt a Gaussian shape only after a number of frames, but their model is tailored towards Gaussian shaped objects. Moreover, due to non-homogeneous illumination, initialization with the correct number of cells seems to be imperfect. Of course, these detection errors in this dataset are also mirrored when inspecting their tracking quality.

Tracking Quality

The detection/segmentation errors usually propagate to the next stage, the tracking stage. Our JST model aims at avoiding such error-propagation, the performance measures for the tracking quality are reported in Table 3.3. On both datasets, the proposed JST method is on par with CT (Chapter 2) and (Amat et al. 2014) in terms of (frame-to-frame) move events. For the division events, we show through the f-measures of 0.70 (unconditioned) and 0.84 (conditioned) that our JST method can deal with mitosis in the challenging 2D+t dataset slightly better than (Rapoport et al. 2011) (f-measure of 0.67), and improves significantly upon CT (f-measures of 0.32 and 0.56, respectively), although using the same classifier. On the other hand, the CT method yields a slightly better detection rate of division events on the 3D+t dataset. We believe that this fluctuation is due to a lack of training data for the graphical model (only 16 divisions occur in our training set) which is more critical in our JST approach since it has more degrees of freedom. In particular, when dealing with oversegmented

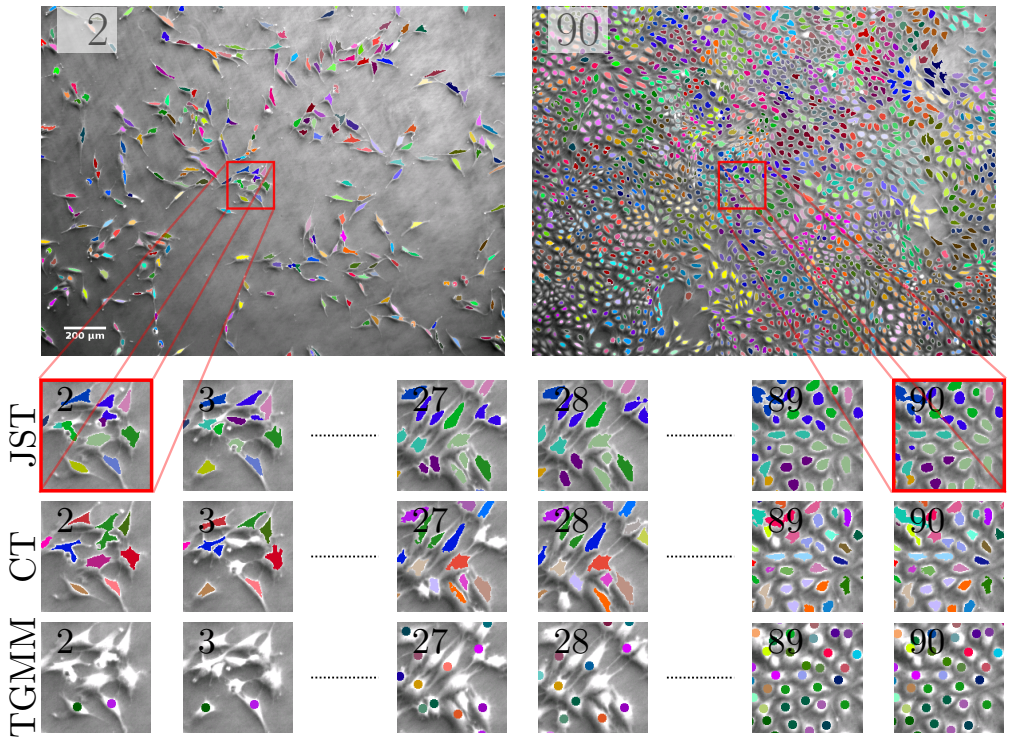


Figure 3.4: Qualitative results for the *Rat stem cells* dataset. Cells are assigned a random color identity in the first frame, which is inherited to their children in later timesteps. The magnified views illustrate that cells can be tracked reliably by our JST method in spite of frequent overlap. CT is short for *Conservation Tracking* (Chapter 2), TGMM stands for *Tracking with Gaussian Mixture Models*, (Amat et al. 2014), and Joint segmentation and tracking (JST) is the model proposed in this chapter.

objects, a strong division classifier is crucial since the introduced ambiguity may lead to increased confusion in division events. If higher division accuracies are desired, the training set needs to be expanded at the cost of more user annotations.

Furthermore, the division detection accuracy our proposed JST model achieves is significantly better than that of (Amat et al. 2014). We believe this is due to the reason that divisions are handled naturally in tracking-by-assignment approaches (compared to heuristic rules) and further evidence can be injected through local classifiers trained on this specific event.

Qualitative results for our experiments on the *Rat stem cells* dataset and the *Drosophila* dataset are depicted in Figures 3.4 and 3.5, respectively.

Although both quantitative and qualitative results prove high accuracy of the algorithm, the tracking results are still far from being 100% accurate on these challenging datasets. We tackle this issue in the **next chapter**, where we present methods to

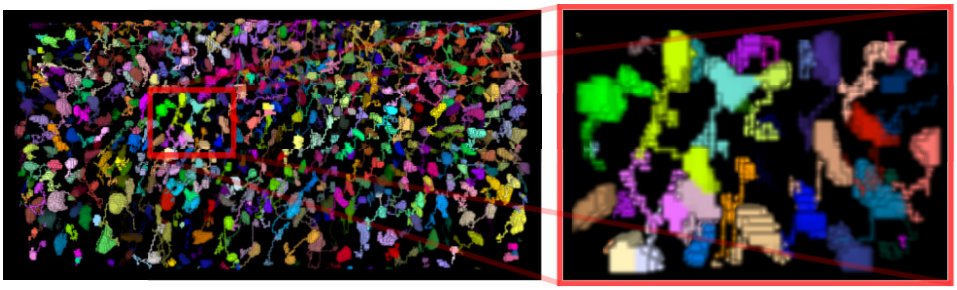


Figure 3.5: 3D rendering for the *Drosophila embryo* dataset. Depicted are the cell segmentations in timestep 50 with their trajectories as one-voxel-traces over the previous 50 timesteps (the remaining 50 timesteps are omitted for clarity). In the close-up view (right), the two yellow cells are the result of a cell division many timesteps ago and the lower one is touching with cells indicated in gray and pink. Thanks to the joint optimization of segmentation and tracking, the identity of the yellow cell is preserved in spite of this heavy overlap.

automatically guide the human expert to assignments which are most probably erroneous in order to manually correct these if necessary. With the user in the loop, we can significantly improve the reliability of our algorithms with only little human interaction time needed.

	unconditioned						*conditioned on segmentation*					
Dataset	**Moves**			**Divisions**			**Moves**			**Divisions**		
Method	Prec.	Rec.	F-Meas.	Prec.	Rec.	F-Meas.	Prec.	Rec.	F-Meas.	Prec.	Rec.	F-Meas.
Rat stem cells (2D+t) (Rapoport et al. 2011)												
(Rapoport et al. 2011)				0.55	0.87	0.67						
Conservation tracking (CT)	0.96	0.89	0.92	0.68	0.26	0.32	0.98	0.90	0.94	0.72	0.26	0.38
Conservation tracking on oversegmentation	0.89	0.90	0.90	0.22	0.44	0.29	0.99	0.91	0.95	0.77	0.45	0.56
(Amat et al. 2014) on raw data	0.92	0.63	0.75	0.62	0.17	0.26	0.96	0.68	0.80	0.64	0.24	0.35
(Amat et al. 2014) on our pred. maps	0.90	0.88	0.89	0.74	0.31	0.44	0.97	0.94	0.95	0.8	0.41	0.54
Joint segmentation and tracking (JST)	0.97	0.93	**0.95**	0.74	0.67	**0.70**	0.98	0.97	0.98	0.90	0.78	0.84
Drosophila embryo (3D+t)												
Conservation tracking (CT)	0.95	0.85	0.90	0.65	0.74	**0.69**	0.97	0.92	0.94	0.80	0.77	0.78
Conservation tracking on oversegmentation	0.73	0.77	0.75	0.04	0.78	0.08	0.97	0.82	0.89	0.28	0.82	0.42
(Amat et al. 2014) on raw data	0.93	0.91	**0.92**	0.25	0.75	0.38	0.97	0.98	0.97	0.35	0.78	0.48
(Amat et al. 2014) on our pred. maps	0.91	0.86	0.89	0.18	0.70	0.29	0.96	0.97	0.96	0.25	0.85	0.38
Joint segmentation and tracking (JST)	0.96	0.86	0.91	0.54	0.75	0.63	0.98	0.99	0.98	0.60	0.89	0.72

Table 3.3: Quantitative results for cell tracking. Reported are precision, recall, and f-measure for (frame-to-frame) events *move* (*i.e.* transition assignments) and *cell divisions* (*i.e.* mitosis). *Rat stem cells* comprises 119 266 and 1 998 such events, respectively, whereas *Drosophila embryo* includes 63 548 moves and 226 divisions. Results are shown for the tracking being *conditioned* on its segmentation result and directly compared to ground truth (*unconditioned*).

4

Proof-reading Guidance by Sampling

Biomedical applications often require a *highly accurate* tracking of proliferating cells (Meijering et al. 2009). Existing cell tracking methods (Al-Kofahi et al. 2006; Padfield et al. 2009; Bise et al. 2011; Amat et al. 2014) perform well on sparse cell populations with high signal-to-noise ratio (SNR). However, they are far from achieving gold-standard-accuracy on high-dimensional data with low SNR. To analyze their data with highest reliability, biomedical specialists have to spend their precious time to proof-read the predicted cell lineages and correct them where necessary. When working with high-throughput setups, unguided proof-reading quickly becomes prohibitive, though. In response, what is needed is automatic guidance which presents the most ambiguous frame-to-frame assignments to the user, omitting trivial assignments.

Tracking-by-assignment methods as proposed in Chapters 2 and 3, or in (Padfield et al. 2009; Bise et al. 2011) typically consist of a data term (local features) and an interaction term (e.g. lineage consistency constraints, penalties for large displacements, etc.). While the latter term is based on model assumptions, the first term may be corrupted by missing features, noisy signal, or poor image quality. When estimating the maximum a-posteriori (MAP) solution to obtain the most likely cell lineages, these cell tracking methods ignore this signal uncertainty.

Contributions In this chapter[14], we take into account the uncertainty in the measurements to derive ambiguity quantification measures in order to guide the user to those predicted assignments which are most uncertain. Both types of predicted events, cell migration (*moves*) and mitosis (*divisions*), are taken into consideration

[14] This chapter is an extended version of (Schiegg et al. 2015b).

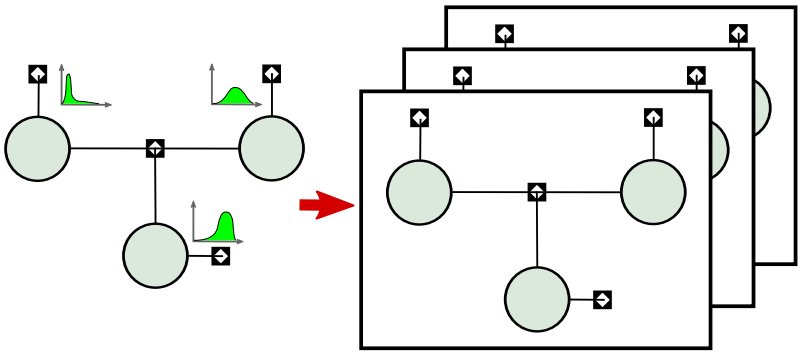


Figure 4.1: To derive proof-reading priorities, we propose to perturb the unary potentials of a tracking-by-assignment cell tracking model according to *(i)* a Gumbel distribution (Papandreou and Yuille 2011), or *(ii)* a Gaussian distribution predicted from a Gaussian process. Instances of the graphical model can then be generated by sampling locally from the distributions of the unary potentials. Estimating the MAP solution of each individual graphical model instance allows to compute robustness measures for each individual predicted event and helps to guide the biomedical specialist to the most ambiguous assignments.

independently. In order to sample multiple proposal solutions from the tracking-by-assignment method, we present two methods, the application of Perturb-and-MAP (Papandreou and Yuille 2011) and a novel approach based on Gaussian process predictions for the unary terms. Both alternatives allow sampling multiple instances of the underlying graphical model on which, in turn, individual MAP solutions are estimated, *cf.* Figure 4.1. We find that the variations in each assignment among these proposal solutions is an efficient guidance measure for proof-reading.

Structure This chapter is structured as follows. We first elaborate on prior work related to uncertainty estimation in (cell) tracking in Section 4.1. In Section 4.2, we present two approaches to effectively sample from tracking-by-assignment models, followed by the proposition of a useful ambiguity measure in Section 4.3. With an empirical evaluation of the proposed approaches in Section 4.4, we conclude this chapter.

4.1 Related Work

Tracking-by-assignment models aim at solving the cell tracking task with a structured model spanning pairs of frames (Padfield et al. 2009) or a larger, possibly the entire, time sequence (Bise et al. 2011; Jug et al. 2014) as also proposed in Chapters 2 and 3. These methods can be formulated as probabilistic graphical models which

describe a Gibbs distribution defined by local potentials. To quantify uncertainty in such graphical models, sampling from this distribution by Markov chain Monte Carlo (MCMC) techniques would be the natural first choice. However, due to signal-sensitive local potentials, the Gibbs distribution may be "ragged" (Maji et al. 2014) and sampling will be expensive. In contrast, recent developments in machine learning turn to generating multiple instances of graphical models and perform a MAP estimation on each of those instances to obtain proposal solutions (Papandreou and Yuille 2011), similar to k-best data association hypotheses (Kragel et al. 2012). We apply Perturb-and-MAP (Papandreou and Yuille 2011) to cell tracking, *i.e.* all unary potentials are perturbed with the same distribution, and we propose an alternative where the perturbations are based on local uncertainties in the features.

In cell tracking, assignment ambiguities have only been explored rarely. Rapoport et al. (2011) find potential error types and remove all lineages which are likely to contain such errors to only show the most "trustworthy" lineages, *i.e.* aiming for a high precision at the cost of low recall. Lou et al. (2014b) exploit uncertainty in an active learning framework to request the most uncertain assignment from the user in order to improve the weights of the structured cell tracking method.

4.2 Sampling from Tracking-by-Assignment Models

First, we briefly recapitulate tracking-by-assignment models to introduce a simplified notation, then review Perturb-and-MAP as "one shot approximate random sampling" approaches (Papandreou and Yuille 2011) and finally propose a novel related sampling method using Gaussian processes.

4.2.1 Tracking-by-Assignment Models

As discussed in previous chapters, tracking-by-assignment models typically decouple into two subsequent stages, a detection/segmentation stage followed by a tracking/assignment stage. For the ease of notation, let us now rewrite the energy minimization problem for tracking-by-assignment as

$$\begin{aligned} \mathbf{y}^* = &\operatorname*{argmin}_{\mathbf{y}\in\mathcal{Y}} E(\mathbf{y}) = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{Y}} \boldsymbol{\theta}^\top \rho(\mathbf{y}) \\ &s.t. \quad \text{consistency constraints,} \end{aligned} \tag{4.1}$$

which minimizes real-valued costs (energy) E associated with specific configurations $\mathbf{y} \in \mathcal{Y}$ of the detections and cell-to-cell transitions to find the optimal assignments/detections $\mathbf{y}^*$. In general, the energy function can be expressed in terms of indicator functions $\rho(\mathbf{y})$ and energies $\boldsymbol{\theta}$ (which express costs in unary and higher-order potentials). We refer the reader to Section 1.4 for more details on this notation. Linear constraints forbid inconsistent solutions such as more than one cell ancestor or more than two cell successors; these constraints can equivalently be added to the objective function. Thus we drop the constraints in our following notations.

In Chapter 2, we propose to choose multi-state random variables, $\mathcal{Y} \in \{0, ..., m\}^D$, where m is the maximal number of cells expected to appear in one detection (falsely merged cells)[15] and D is the number of random variables in the graphical model.

Energy minimization problems as in Equation (4.1) can be cast as maximum a-posteriori (MAP) estimation problems through the maximization of the underlying Gibbs distribution,

$$P(\mathbf{y}) = \frac{1}{Z} \exp\left(-E(\mathbf{y})\right) = \frac{1}{Z} \exp\left(-\boldsymbol{\theta}^\top \rho(\mathbf{y})\right), \tag{4.2}$$

where Z is the normalizing partition function.

4.2.2 Sampling through Perturb-and-MAP Random Fields

The key idea of Perturb-and-MAP (Papandreou and Yuille 2011) is to inject random noise $\boldsymbol{\varepsilon}$ into the Gibbs distribution in Equation (4.2), *i.e.*

$$P_{\boldsymbol{\varepsilon}}(\mathbf{y}) = \frac{1}{Z} \exp\left(-(\boldsymbol{\theta} + \boldsymbol{\varepsilon})^\top \rho(\mathbf{y})\right), \tag{4.3}$$

and then repeatedly find the MAP estimates of the perturbed objective. It has been shown (Papandreou and Yuille 2011) that if the perturbation variables $\boldsymbol{\varepsilon}$ are distributed according to a Gumbel distribution, the distribution defined in Equation (4.3) approximates the Gibbs distribution in Equation (4.2).

In practice, each cost of the (unary) potentials is perturbed independently and identically according to a Gumbel distribution. The scale parameter for this single distribution needs to be set manually. In the following, we propose to model each data term with a distinct Gaussian distribution that reflects the ambiguity of the local observations.

[15] For the ease of notation, we are here only referring to one variable type $\mathcal{Y}$, whereas in the original model in Chapter 2, a finer distinction is made.

4.2.3 Sampling through Gaussian Processes

The costs $\boldsymbol{\theta}$ introduced in Section 4.2.1 relate to the energies defined by unary or higher-order potentials in the graphical model. In many graphical models, the unary potentials are based on predictions of classifiers fed with local features. This is also true for the conservation tracking method studied in this work: The unary costs are results of a cell division classifier, a cell detection classifier, and a transition classifier. We propose to utilize Gaussian process classifiers[16] for the generation of unary potentials since they provide full predictive distributions for each query point. In particular, let us denote the energy associated with some variable $Y_i \in \{0, ..., m\}$ by $\theta_i^k,\ k \in \{0, ..., m\}$. Note that θ_i^k are the costs which were perturbed in Section 4.2.2. Typically, an unstructured classifier with parameters $\boldsymbol{\eta}$ is trained on a training set of input/output pairs, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1,...,N} =: (\mathbf{X}, \mathbf{y})$, and the (mean) prediction $p(Y_i = k|\mathbf{x}_i, \mathbf{X}, \mathbf{y}, \boldsymbol{\eta})$ from local features $\mathbf{x}_i$ determines these costs θ_i^k for variable Y_i as

$$\theta_i^k = -w \log \left(p(Y_i = k|\mathbf{x}_i, \mathbf{X}, \mathbf{y}, \boldsymbol{\eta}) \right), \tag{4.4}$$

where w is a parameter of the graphical model; see, *e.g.* Equation (2.1). For a Gaussian process classifier, reviewed in the next paragraph, θ_i^k may either come from the mean of the predictive distribution in Equation (4.6) specific for variable Y_i, or drawing M samples in Equation (4.6) yields multiple $\theta_{i,l}^k,\ l \in \{1, ..., M\}$. Drawing samples from the predictive distributions of each variable in the graphical model yields a multitude of perturbations of the graphical model similar to Equation (4.3). Their non-parametric nature avoids choosing a value for the perturbation variance, and the unary potentials in the tracking-by-assignment model are perturbed according to the predicted second-order uncertainty in the measured data, in other words the data term is perturbed locally *w.r.t.* the predictive uncertainty from the Gaussian process classifier. An example for the uncertainty of the cell detection classifier on our *Drosophila* dataset is shown in Figure 4.2. Similarly to Section 4.2.2, these perturbed graphical models can finally be solved independently and distributedly to approximate samples from the Gibbs distribution.

Gaussian Process Classification

Gaussian processes (GPs), denoted by $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, are stochastic processes each realization of which defines a multi-variate joint Gaussian distribution (Rasmussen and Williams 2006). The mean function $m(\mathbf{x})$ is typically assumed to be constant zero whereas the covariance function $k(\mathbf{x}, \mathbf{x}')$ may be any Mercer kernel. GP classification aims at finding a mapping from an input space to a categorical output space given unstructured training data $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ and it can be shown (Ras-

[16] Gaussian process regression may also be used directly for the generation of unary potentials. We refer the reader to (Rasmussen and Williams 2006) for a thorough introduction to Gaussian processes.

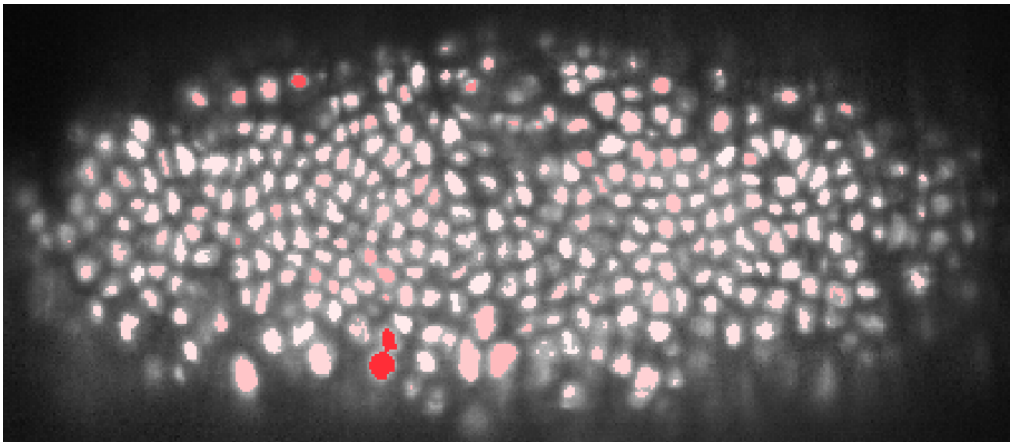


Figure 4.2: One slice of the 3D+t *Drosophila* sequence: The more opaque the red color the higher the *classifier uncertainty* that the connected component contains exactly one cell.

mussen and Williams 2006) that the latent predictive distribution for query point $\mathbf{x}_*$ is given as

$$f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{k}_*^\top K^{-1}\mathbf{y}, k_{**} - \mathbf{k}_*^\top K^{-1}\mathbf{k}_*), \tag{4.5}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes the Gaussian distribution, $[K]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $[\mathbf{k}_*]_i = k(\mathbf{x}_*, \mathbf{x}_i)$ with $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$, $k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$, and the parameters of $k(\cdot,\cdot)$ are the hyperparameters $\boldsymbol{\eta}$ of the GP. For *binary* classification with class label $y \in \{0, 1\}$, the prediction for the latent f_* in Equation (4.5) is "squashed" through a sigmoid function:

$$\begin{aligned} p(Y_* = +1|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\eta}) &= \mathbb{E}\left(\text{sigmoid}\left(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\eta}\right)\right) \\ &\approx \text{sigmoid}\left(\mathbb{E}(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}, \boldsymbol{\eta})\right). \end{aligned} \tag{4.6}$$

Applying the sigmoid function on multiple samples of f_* rather than its expected value yields samples from the non-Gaussian class-conditional distribution. In our experiments, we choose the error function as the sigmoid function. To train a GP classifier, the marginal likelihood of the model is optimized, and approximations need to be made due to the non-Gaussian likelihood function utilized in GP classification. We refer to (Rasmussen and Williams 2006) for more details.

To extend the binary GP classifier to multiple classes, uncorrelated latent functions are assumed, one per class. The hyperparameters of these latent processes may either be learned jointly using the softmax (Williams and Barber 1998) or independently as a set of binary classifiers in a one-vs.-all scheme. We use the latter in our experiments.

4.2.4 Graphical Model Point of View

We summarize in the probabilistic graphical model depicted in Figure 4.3 the assumptions made in this chapter. The simplistic graphical model is a conditional random field (Koller and Friedman 2010, Chapter 4) and shows two input variables X_i, X_j, and their observations $\mathbf{x}_i$, $\mathbf{x}_j$ are called the *features*. The corresponding outputs Y_i, Y_j are correlated, which is a simple example for a structured output space. Throughout this work, we assume that the energies of the potentials are linearly parameterized as in Equation (1.6), the weights of which are unknown and estimated from training data, for instance using structured learning as in Section 5.2.

In our cell tracking application, we further augment the locally observed features by the predictions from unstructured classifiers such as random forests (Breiman 2001) or Gaussian process classification as in this chapter. The parameters $\boldsymbol{\eta}$ of these classifiers are learned from unstructured input/output pairs only, without taking the correlations of the outputs into consideration, as depicted in Figure 4.4(a). For the structured prediction, in turn, these parameters $\boldsymbol{\eta}$ are fixed, the observed inputs $\mathbf{x}_i$, $\mathbf{x}_j$ are fed into the classifiers, which finally determines the classification estimates f_i, f_j, respectively. The local feature vector is then augmented by these estimates. In fact, note that in Equation (4.4), we only use these classifier estimates as features in our cell tracking model (*i.e.* we set all other model weights to zero).

In this chapter, we propose two strategies to analyze the robustness of the model predictions. The first, Perturb-and-MAP in Section 4.2.2, uses a point estimate of the unstructured classifier as input and assumes that the resultant energies are perturbed by random noise $\boldsymbol{\varepsilon}$, *cf.* Figure 4.4(b). Sampling different values for $\boldsymbol{\varepsilon}$ from a Gumbel distribution yields multiple proposal predictions for the structured outputs. In contrast, in Section 4.2.3, we propose to sample different predictions from the posterior distribution of the Gaussian process classifier which naturally considers local uncertainty in the observed data. This process is illustrated in Figure 4.4(c).

4.3 Uncertainty in Cell Tracking

The goal in cell tracking is to find full cell lineages which, in turn, decompose over frame-to-frame *events*, assignments either between two or three cell candidates, denoted as *moves* or cell *divisions*, respectively. For manual proof-reading, we want to guide the user to the most uncertain events, while assuming that ground truth is not available for the test dataset. We propose to estimate the event uncertainty (also referred to as event *ambiguity*) by analyzing the proposal solutions drawn from the tracking-by-assignment model, as outlined in the previous sections. Both events, *moves* and *divisions* are modeled as random variables Y_i^{move} and Y_i^{div} in the graphical

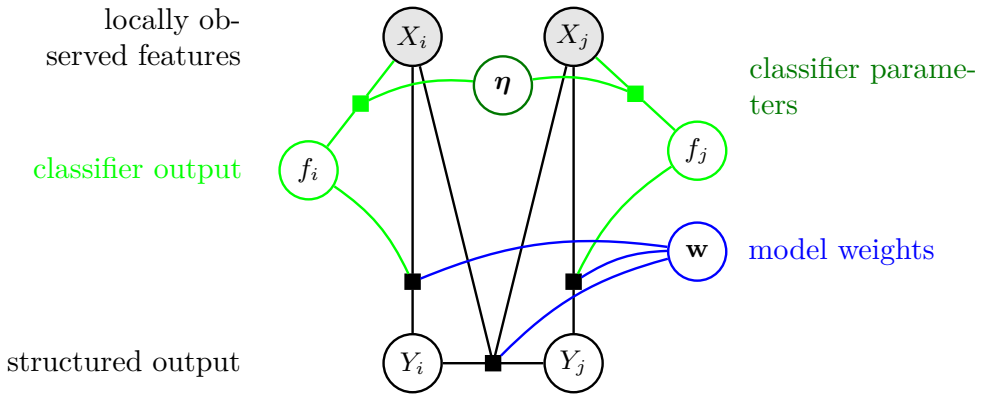
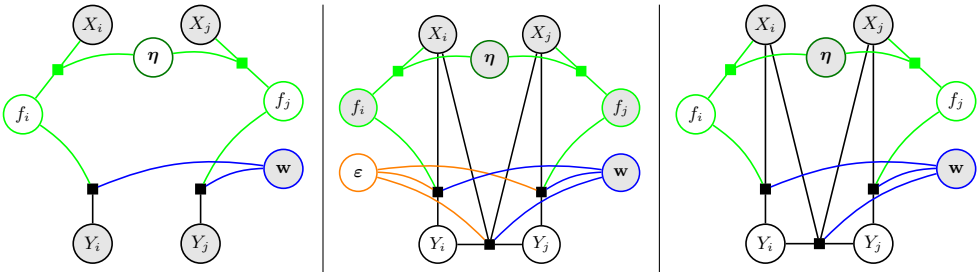


Figure 4.3: Underlying graphical model The probabilistic graphical model consists of two output variables Y_i, Y_j, which are interdependent through a pairwise factor (a simple structured output space). In a conditional random field, the pairwise interaction term as well as the unary potentials (all black) depend on the observations $\mathbf{x}_i$, $\mathbf{x}_j$ from X_i, X_j. Note that additionally to the directly observed features, other features might be derived such as $g(f_i)$, $g(f_j)$, where f_i, f_j are predictions of a classifier and $g(\cdot)$ is an arbitrary, possibly non-linear, function. This classifier has parameters $\boldsymbol{\eta}$ estimated from an unstructured training set, as shown in Figure 4.4(a). Shaded nodes are observed.



a: Classifier learning **b:** Perturb-and-MAP **c:** GP-sampling

Figure 4.4: (a) Classifier learning The graphical model from Figure 4.3 is modified for the estimation of the parameters $\boldsymbol{\eta}$ of the unstructured classifier. During this phase, direct interactions between input and output variables are omitted. **(b) Perturb-and-MAP** Variable $\mathbf{w}$ is kept fixed after structured learning (see Section 5.2) and f_k is deterministically computed from $\boldsymbol{\eta}$ and $\mathbf{x}_k$ ($k = \{i, j\}$). For perturbations, a noise injecting variable $\boldsymbol{\varepsilon}$ is introduced, which is assumed to be Gumbel distributed and perturbs the costs. **(c) GP sampling** Again, $\mathbf{w}$ is fixed after structured learning, and different f_i are sampled from a Gaussian process with parameters $\boldsymbol{\eta}$ learned in (a), *cf.* Equation (4.5). Shaded nodes are (in)directly observed. See Figure 4.3 for notations.

model proposed in Chapter 2, whose realizations in the nth proposal are denoted as $y_{i,n}^{\text{move}}$ and $y_{i,n}^{\text{div}}$; for brevity we omit the event type.

To reason about the reliability of a selected labeling $\mathbf{y}_*$, we introduce the *labeling*

uncertainty measure as

$$U(Y_i = y_{i,*}) = 1 - \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}[y_{i,n} = y_{i,*}], \tag{4.7}$$

which normalizes the number of votes amongst the solutions of the perturbed models *against* labeling $\mathbf{y}_*$. In other words, the labeling uncertainty finds the most uncertain decisions, *i.e.* the predictions in the selected proposal which are likely to be wrong, compared to the labelings of the other proposals.

Note that in order to guarantee a consistent solution, it is not possible to *round* the approximated marginal distributions for an "averaged" distribution. Instead, one proposal solution (*e.g.* the MAP solution of the unperturbed model) needs to be chosen to guarantee that no consistency constraint in Equation (4.1) is violated. In the GP based sampling approach, this unperturbed proposal solution corresponds to the mean predictions.

4.4 Experiments & Results

We evaluate the usefulness of the proposed uncertainty measures on a challenging 3D+t sequence from the developing *Drosophila* embryo with annotations as in Chapter 2. For this purpose, we adapt the tracking-by-assignment model from Chapter 2 (we set $m = 2$) and use Gaussian process classifiers for the detection, transition, and division priors using the GP implementations in the GPy package (GPy authors 2014). The parameters of the graphical model are trained using the structured max-margin learning implementation from (Funke 2014). We used the first 20 frames as training set and the remaining 80 frames as test set. Structured learning will be studied in the next Chapter in Section 5.2. Using this technique, significantly less iterations involving inference on the graphical model are necessary compared to an exhaustive grid search over a reasonable parameter range as done in the previous chapters. In this way, learning is performed orders of magnitudes faster with better generalization properties on test data.

As performance measure, we choose the

$$\text{f-measure} = \frac{2 \cdot \text{True Positives (TP)}}{2 \cdot \text{TP} + \text{False Negatives} + \text{False Positives}}, \tag{4.8}$$

since it balances the number of true events found (recall) and the rate of true predictions (precision). Note that although tracking-by-assignment is a two-stage model, we here only evaluate the second stage, namely the tracking model, since the ambiguity measures proposed only apply to that; we hence only compare to the ground truth

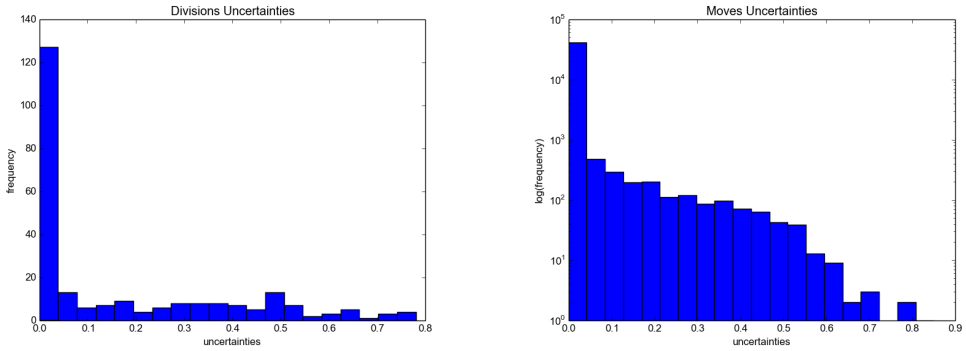
Events presented	0	10	50	100	150	250	500
GP sampling (5x)	68.3	69.3	71.8	74.9	78.5	80.5	-
GP sampling (20x)	68.3	70.0	75.3	77.9	81.9	84.2	87.2
GP sampling (50x)	68.3	69.8	75.0	79.8	82.4	84.1	86.6
GP sampling (100x)	68.3	69.8	75.1	80.0	82.5	84.1	86.6
Perturb-and-MAP (5x)	68.3	69.3	72.7	-	-	-	-
Perturb-and-MAP (20x)	68.3	69.1	73.4	76.4	80.8	-	-
Perturb-and-MAP (50x)	68.3	69.1	73.4	76.6	79.7	-	-
Perturb-and-MAP (100x)	68.3	69.3	73.3	76.6	79.1	-	-

Table 4.1: Division accuracy after N events presented to the user (and corrected if necessary); "-" indicates that the method did not generate enough events with positive uncertainty. The number "(tx)" is the number of samples drawn.

actually seen by the tracking model. The numbers may deviate from those reported in Chapter 2 since a separate test set and different classifiers are used. We want to show in this evaluation that presenting the most ambiguous events to the user and asking for corrections may quickly boost performance. The samples are drawn from the graphical model offline, so no expensive human interaction time is needed during the generation of the uncertainty values. In our experiments, drawing a sample took ≈ 30 seconds on a contemporary workstation.

Our experimental setup is as follows. We iteratively present the event with the highest labeling uncertainty according to Equation (4.7) to an oracle which corrects the event if necessary. For both divisions and moves, most of the highly uncertain events must be corrected by the oracle as shown in Figure 4.6(b) and Figure 4.7(b), and less corrections need to be made for less uncertain events. Our baseline is a random suggestion of events. Both presented sampling methods perform similarly well with a slight advantage for the GP sampling approach for division events. As expected, the more samples are drawn, the more meaningful are the uncertainty measures. As shown in Table 4.1 and Figure 4.6(a), manually correcting the divisions proposed by the GP sampling method can boost the performance from 68.3% to 80.0% already after 100 presented examples. The f-measure of *moves* (*cf.* Figure 4.7(a)) increases more slowly due to the large number of move events in the dataset. Examples of uncertain (and falsely predicted) division and move events are provided in Figure 4.8 and Figure 4.9, respectively. The frequencies of division and move uncertainties in our *Drosophila* dataset are shown in Figure 4.5.

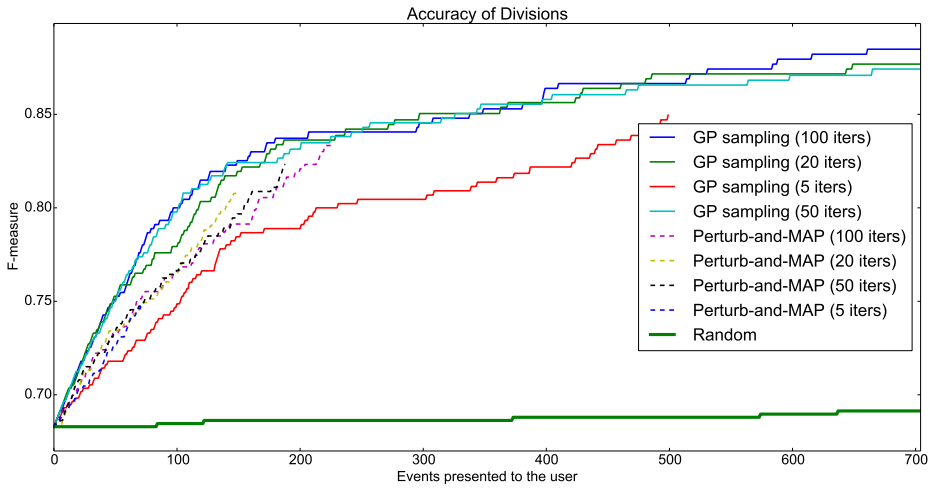
To further improve the uncertainty estimations, we propose in the **next chapter** to learn an ensemble of *diverse* structured models. By analyzing diverse but similarly likely solutions of the tracking problem, ambiguous assignments may be easily found by only examining very few proposal solutions. Furthermore, since diversity is already considered at the training time of the structured models, it is ensured that all models in the ensemble fit and represent the training data well while still yielding diverse



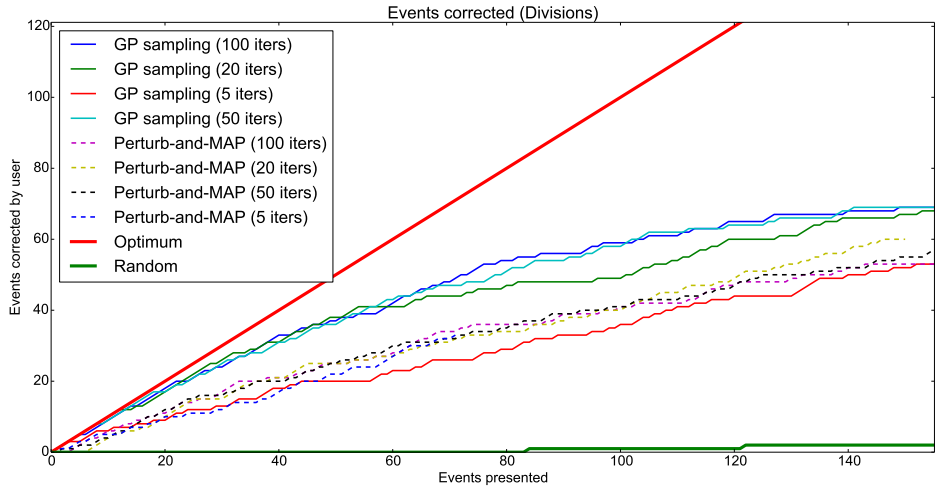
(a) Division uncertainties (b) Move uncertainties

Figure 4.5: Frequencies of division and move uncertainties

predictions. This is in contrast to the methods proposed in this chapter, where only one structured model is being learned and samples are generated with this unique set of model parameters.

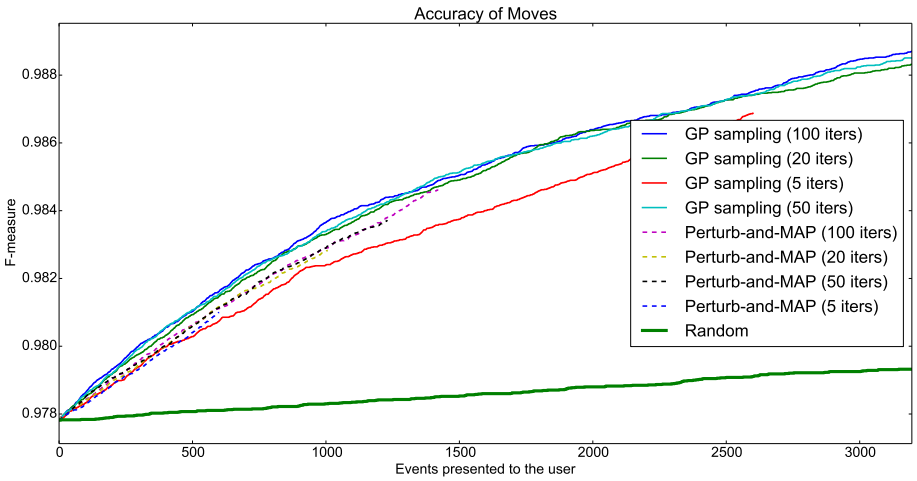


(a) Division: F-Measure

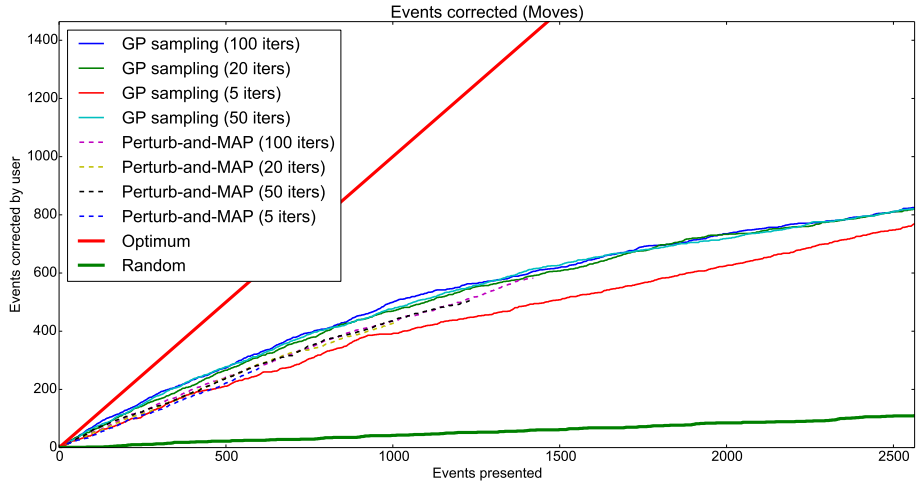


(b) Division: Corrected Events

Figure 4.6: Division Events: Comparison of sampling methods and the resulting labeling uncertainties. The curve terminates prematurely if all remaining uncertainty estimates are zero. "iters" stands for the number of samples generated from one model. See main text for details.



(a) Move: F-Measure



(b) Move: Corrected Events

Figure 4.7: Move Events: Comparison of sampling methods and the resulting labeling uncertainties. The curve terminates prematurely if all remaining uncertainty estimates are zero. "iters" stands for the number of samples generated from one model. See main text for details.

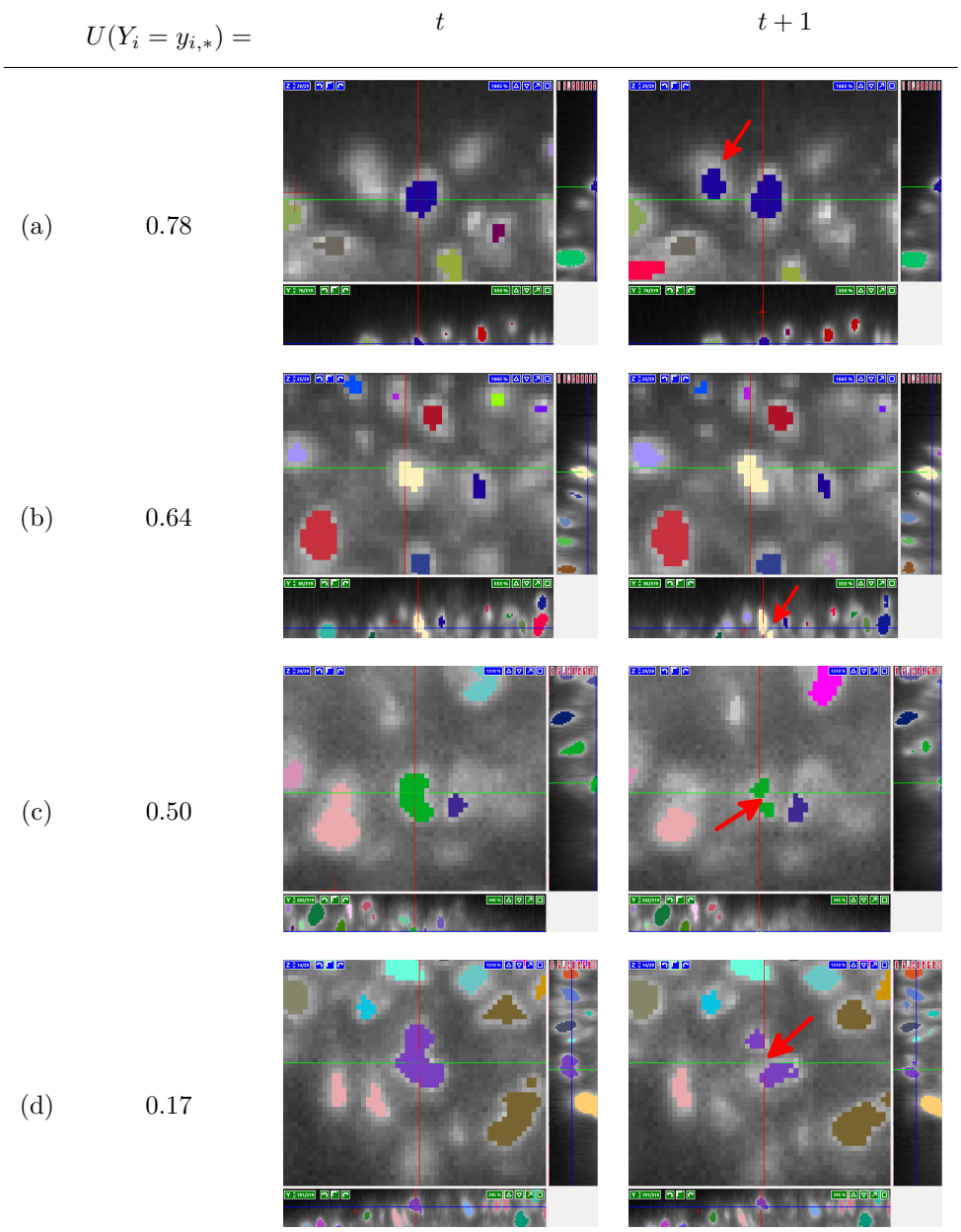


Figure 4.8: Uncertain **division events**: The cells of interest are centered in orthoviews (green, red, and blue lines indicate the cuts of each view) and incorrect predictions are indicated with a red arrow. Appearance at the image border (a), undetected demerging (b), and oversegmentation (c, d) lead to falsely predicted divisions. These false predictions are identified by our GP sampling method as the uncertainty quantities (first column) indicate.

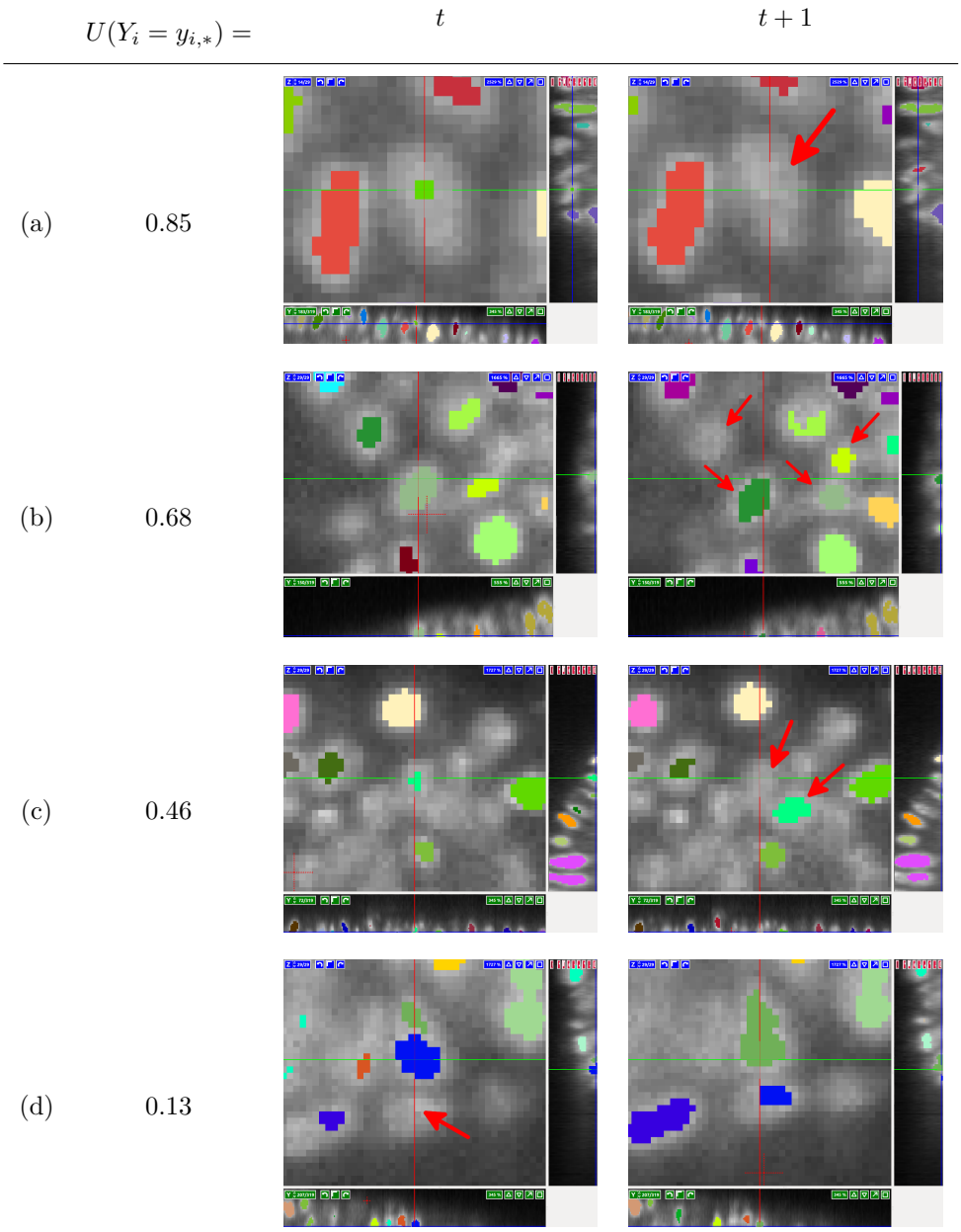


Figure 4.9: Uncertain **move events**: The cells of interest are centered in orthoviews (green, red, and blue lines indicate the cuts of each view) and incorrect predictions are indicated with a red arrow. Too high priors for being a false detection (a, c), or missing segmentations (b, d) result in wrong move predictions. In (a), the green cell falsely disappears, whereas in (b, c, d), the cell of interest is associated with the wrong descendant, in (b) even leading to a chain of wrong associations. The uncertainty quantities (first column) estimated by our GP sampling method lead the user to these wrong predictions.

5

Learning Diverse Models

Tracking-by-assignment methods, as presented in this thesis, map a structured input in terms of features to a structured output space of consistent tracking solutions. To learn the model parameters from annotations on the structured space, it is standard procedure to discriminatively train a single model that is then used to make a single prediction for each input, as in the structured support vector machine (SSVM) (Tsochantaridis et al. 2005).

The success of such large margin methods for structured output learning, is partly due to their good generalization performances achieved on test data, compared to, *e.g.* maximum likelihood learning on structured models (Nowozin and Lampert 2011). Despite such regularization strategies, however, it is not guaranteed that the model which optimizes the learning objective function really generalizes well to unseen data. Reasons include wrong model assumptions, noisy data, ambiguities in the data, missing features, insufficient training data, or a task loss which is too complex to model directly.

To further decrease the generalization error, it is beneficial to either *(i)* generate multiple likely solutions from the model (Yanover and Weiss 2003; Papandreou and Yuille 2011; Batra et al. 2012) or, *(ii)* learn *multiple* models which generate diverse predictions (Guzman-Rivera et al. 2012; Guzman-Rivera et al. 2014; Gane et al. 2014).

The different predictions for a given structured input may then be analyzed to compute robustness/uncertainty measures, or may be the input for a more complex model exploiting higher-order dependencies, as is done in re-ranking models, *e.g.* Yadollahpour et al. (2013) augment their features with global ones for automatic re-ranking. Other successful applications include prediction of diverse hypotheses for machine translation (Gimpel et al. 2013), on-demand feature computation (Roig et al. 2013), or active learning methods (Maji et al. 2014; Premachandran et al.

2014). Furthermore, an oracle may choose amongst all predictions that one which is closest to the ground truth. This becomes handy for proof-reading tasks in order to keep manual interactions at a minimum. It is particularly beneficial in structured output spaces to present to the user not only similarly likely, but also *diverse* proposal solutions. The set of diverse predictions may still contain a low-loss solution, even if the most likely prediction of the single model has a large loss. As a consequence, instead of minimizing the expected generalization error of a *single* model in structured learning, (cf. Figure 5.1(a)), it is favorable to minimize the expected generalization error *amongst multiple* models, see Figure 5.1(b,c).

Contributions Our main contribution in this chapter[17] is an algorithm termed the *Coulomb structured support vector machine* (CSSVM) which learns an ensemble of M models with different parameters, thanks to a corresponding diversity-encouraging prior. This is qualitatively different from previous work which requires that the *outputs* of the M models are diverse. In particular, we allow the M models in the ensemble to make identical predictions (and hence perfectly fit the data) at training time. Another benefit is that CSSVM can learn diverse models even if only a single structured training example is available. In Section 5.3.4, we generalize our algorithm to allow for structured clustering. Note that, in contrast to the approaches presented in Chapter 4, where proposal samples are generated from only one model through (local) perturbations, we present in this chapter a novel algorithm to learn *multiple* models which are coupled at training time to encourage *diverse* predictions on unseen data.

Structure The remainder of this chapter is organized as follows. We first discuss recent work on generating M (diverse) structured outputs in Section 5.1, followed by a review of structured learning in Section 5.2. In Section 5.3, we propose the *Coulomb structured support vector machine*, a novel algorithm to learn M structured models coupled by a diversity term. We finally empirically demonstrate the utility of the proposed ensemble method in Section 5.4.

5.1 Related Work

One major research avenue is to generate at prediction time multiple (possibly diverse) solutions from a single previously trained structured model (Yanover and Weiss 2003; Papandreou and Yuille 2011; Batra et al. 2012). In order to find M similarly likely solutions, Yanover and Weiss (2003) propose a message passing scheme to iteratively

[17] This chapter is an extended version of (Schiegg et al. 2015a), which is still under review at the time of writing.

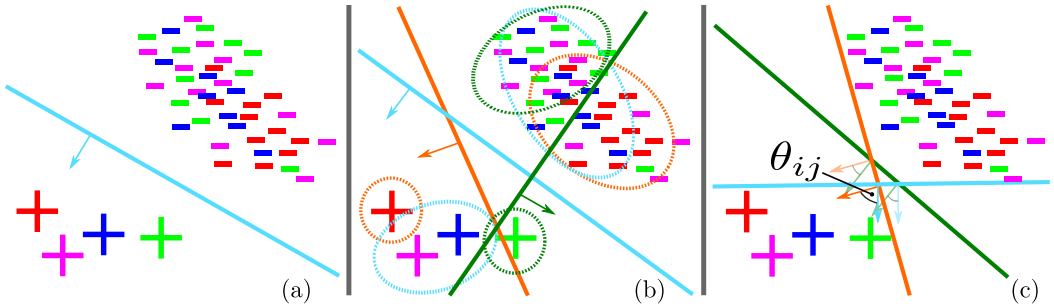


Figure 5.1: Structured SVM learning. "+" indicates a structured training example whereas "−" in the same color are the corresponding structured outputs with task loss $\Delta(+,-) > 0$. (a) A standard linear SSVM maximizes the margin between positive and all "negative" examples (decision boundary with its normal vector in cyan). (b) Multiple choice learning (Guzman-Rivera et al. 2012) learns M SSVMs (here: 3) which cluster the space (clusters for positive and negative examples are depicted in the same color) to generate M outputs. (c) We propose the Coulomb Structured SVM which learns an ensemble of M SSVMs through a diversity term which maximizes the pairwise angles θ_{ij} between their (linear) decision boundaries, while seeking to best fit all training data.

add constraints forbidding the previous solutions. Batra et al. (2012) build on the same idea but incorporate these constraints directly into the objective function. This yields a deterministic framework which tries to find *diverse* solutions by requiring a minimum distance to the previous solutions. Papandreou and Yuille (2011), instead, perturb model parameters repeatedly with noise from a Gumbel distribution, and subsequently solve for the maximum-a-posteriori (MAP) solution to sample M plausible solutions. Their idea of perturbing the data term is natural when data is assumed to be noisy. We applied this idea to cell tracking in Chapter 4.
Sampling M solutions could of course also be achieved using Gibbs sampling or other MCMC techniques, however with very slow mixing time on general graphs; more efficient sampling strategies have been proposed recently (Hazan et al. 2013). Most recent work aims at finding the M best modes of the probability distribution (local maxima) directly (Chen et al. 2013; Chen et al. 2014). While promising, their algorithms are yet not applicable to general graphs.

Rather than learning one model and then sampling successively (possibly diverse) solutions from the model, recent developments (Guzman-Rivera et al. 2012; Guzman-Rivera et al. 2014; Gane et al. 2014) allow to *train* multiple diverse models, *i.e.* diversity is already considered at training time. Typically, only one ground truth solution is provided per training sample rather than a diverse set, and thus diversity amongst the models can not be directly measured by means of training data. There are multiple works which tackle this challenge successfully: Gane et al. (2014) learn (multi-modal) distributions over the perturbations in Perturb-and-MAP models using

latent variable models which include inverse convex programs to determine relations between the model parameters and the MAP solution. Most similar to our work is (Guzman-Rivera et al. 2012; Guzman-Rivera et al. 2014), where a set of M SSVMs is optimized while trading diversity versus data fit. In the former, diversity is encouraged through clustering: Each structured training example is assigned to the learner which achieves the lowest task loss for this sample in the current iteration. Their idea builds on the assumption that there are M clusters present in the data, thus requiring at least M (implicitly) diverse training samples. This requirement may be a crucial problem on small training sets. Our approach, in contrast, can learn M diverse models even if only one training example is present, as is often the case in CRF learning. In their more recent work, Guzman-Rivera et al. (2014) extend their idea by augmenting the learning objective directly with a convex term which explicitly rewards diversity in the *outputs* of different learners. In our approach, in contrast, the diversity prior is posed on the *parameters* of the M models, and thus, all learners might achieve the same loss on the training samples while still providing diverse predictions on test data, *cf.* Figure 5.1(b,c).

5.2 Structured Support Vector Machine

Structured learning has shown to be a useful framework to learn the parameters of a graphical model from training data, while minimizing the expected generalization error on unseen test data (Nowozin and Lampert 2011). Lou and Hamprecht (2011) and Lou and Hamprecht (2012) successfully applied this technique to cell tracking. We briefly review in this section the structured support vector machine (Tsochantaridis et al. 2005), a popular technique for structured learning, and extend this algorithm to a diverse ensemble method in Section 5.3.

In Section 1.4, we defined a log-linear model through the energy $E(\mathbf{x}, \mathbf{y}; \mathbf{w})$ which is linearly parameterized through observed features $\phi(\mathbf{x}, \mathbf{y})$ on the input-output pair $(\mathbf{x}, \mathbf{y})$, *i.e.* $E(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \mathbf{w}^\top \phi(\mathbf{x}, \mathbf{y}) \rho(\mathbf{x}, \mathbf{y})$. Here, $\mathbf{w}$ are the model weights which parameterize the energy function $E(\cdot)$, and $\rho(\cdot)$ are indicator functions which we omit in the following without loss of generality[18].

The goal of structured (output) learning is to estimate the model weights $\mathbf{w}^*$, such that the true solution $\mathbf{y}$ for the observed input $\mathbf{x}$ obtains the lowest energy amongst all possible labelings $\mathbf{y}' \in \mathcal{Y}$ in the structured output space $\mathcal{Y}$, *i.e.* we want to find $\mathbf{w}^*$ such that for any future input-output pair $(\mathbf{x}, \mathbf{y})$ holds:

$$\mathbf{y} = \operatorname*{argmin}_{\mathbf{y}' \in \mathcal{Y}} E(\mathbf{x}, \mathbf{y}'; \mathbf{w}^*). \tag{5.1}$$

[18] These indicators may be incorporated into the feature function $\phi(\cdot)$.

To learn these parameters from a training set of N structured input-output pairs $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,\dots,N}$, Tsochantaridis et al. (2005) propose an algorithm called the *structured support vector machine* (SSVM) or *large margin learning* for structured models (Taskar et al. 2005). Here, the regularized empirical risk over the training set is minimized in order to find the optimal $\mathbf{w}^*$ among all $\mathbf{w} \in \mathcal{W}$ from a set of feasible parameters[19] $\mathcal{W}$, *i.e.*

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathcal{W}}{\operatorname{argmin}} \ \Omega(\mathbf{w}) + C \cdot R_1(\mathbf{w}, \mathcal{D}), \tag{5.2}$$

where $\Omega(\mathbf{w})$ is the convex regularization term, typically the $L2$ regularizer, $\Omega(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$, and

$$R_1(\mathbf{w}, \mathcal{D}) = \frac{1}{N} \cdot \sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) \tag{5.3}$$

is the *empirical risk* over all training examples in $\mathcal{D}$ with $L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})$ being the structured loss of the i-th training example. To guarantee convexity of the objective function, typically the structured hinge loss is chosen for $L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})$, which is defined by

$$L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) = \max_{\mathbf{y}\in\mathcal{Y}} \left(\mathbf{w}^\top \phi(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^\top \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) \right). \tag{5.4}$$

Here, $\Delta(\mathbf{y}_i, \mathbf{y})$ is the *task loss*, for instance the Hamming distance, and measures the similarity between labeling $\mathbf{y}$ and the ground truth solution $\mathbf{y}_i$, with $\Delta(\mathbf{y}', \mathbf{y}) \geq 0$ and $\Delta(\mathbf{y}, \mathbf{y}) = 0$.

The resultant (convex) quadratic objective function in Equation (5.2), however, at a first glance seems hard to optimize since it involves a maximum operation over exponentially many labelings $\mathbf{y} \in \mathcal{Y}$. Nevertheless, recent works successfully applied optimization strategies such as cutting planes (Tsochantaridis et al. 2005) or the subgradient method (Ratliff et al. 2007). For the first, slack variables are introduced to rewrite the maximum operation as (exponentially) many constraints of which the most violated ones are successively added to the optimizer. In this chapter, however, we will resort to the latter method.

With the choices of $\Omega(\cdot)$ and $L(\cdot)$ in Equation (5.2) as being made above, a subgradient[20] $\mathbf{g}_i$ for the convex loss function $L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})$ at $\mathbf{w}$ is defined by

$$\mathbf{g}_i^\top (\mathbf{w}' - \mathbf{w}) \geq (L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}') - L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w})) \ \ \forall \mathbf{w}' \in \mathcal{W}. \tag{5.5}$$

Note that a subgradient is not unique for non-differentiable $L(\cdot)$. A subgradient for

[19] For instance, in submodular energy minimization, this $\mathcal{W}$ may be a half-space.
[20] The subgradient is a generalization of the gradient for non-differentiable convex functions.

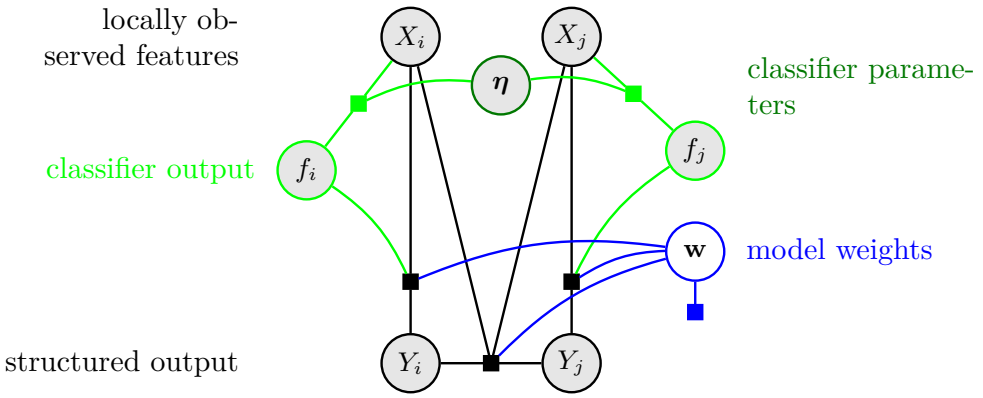


Figure 5.2: Structured Learning Figure 4.3 is modified to illustrate the state of the graphical model during structured learning. All variables except the model weights are (implicitly) observed: The observed states of X and Y serve as training data $\mathbf{x}$ and $\mathbf{y}$, and the classifier parameters $\boldsymbol{\eta}$ are already learned as depicted in Figure 4.4(a), which determines the classifier predictions f, which augment the features $\phi(\mathbf{x})$. The difference to the learning of the unstructured classifiers in Figure 4.4(a) should be noted, to wit the structure in the output space is preserved. Shaded nodes are (implicitly) observed.

$L(\cdot)$ from Equation (5.4) at $\mathbf{w}$ may be computed through

$$\begin{aligned} \hat{\mathbf{y}} &= \max_{\mathbf{y}\in\mathcal{Y}} \left(\mathbf{w}^\top \phi(\mathbf{x}_i, \mathbf{y}_i) - \mathbf{w}^\top \phi(\mathbf{x}_i, \mathbf{y}) + \Delta(\mathbf{y}_i, \mathbf{y}) \right), \\ \mathbf{g}_i &= \phi(\mathbf{x}_i, \mathbf{y}_i) - \phi(\mathbf{x}_i, \hat{\mathbf{y}}). \end{aligned} \tag{5.6}$$

Finally, using the subgradient method (Boyd et al. 2003), $\mathbf{w}$ is iteratively updated through

$$\mathbf{w}' \leftarrow \mathbf{w} - \zeta_t \left(\mathbf{w} + \frac{C}{N} \sum_{i=1}^{N} \mathbf{g}_i \right), \tag{5.7}$$

since $\mathbf{w}$ is the (sub)gradient of the $L2$ regularizer. Boyd et al. (2003) show that the subgradient method converges to the optimal objective value if the step size ζ_t fulfills certain requirements.

We adopt Figure 4.3 in Figure 5.2, and illustrate that all variables during structured learning are kept fixed except for $\mathbf{w}$. The optimal $\mathbf{w}^*$ is found through a structured learning method such as SSVM.

We refer the reader to (Nowozin and Lampert 2011) for a thorough tutorial on structured output learning.

5.3 Coulomb Structured Support Vector Machine

The goal of this chapter is to learn M mappings from one structured input to M possibly *diverse* structured outputs from a training set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1,...,N}$.

5.3.1 Problem Description and Diversity Prior

For this purpose, we propose to learn an ensemble of M concurrent structured SVMs, which amounts to the following optimization problem:

$$\underset{\mathbf{w}_1,...,\mathbf{w}_M}{\operatorname{argmin}} \underbrace{\alpha\Gamma(W)}_{\text{diversity}} + \underbrace{\Omega(W)}_{\text{generalization}} + \underbrace{C \cdot R_M(W, \mathcal{D})}_{\text{data term}}, \tag{5.8}$$

$$\text{where } R_M(W, \mathcal{D}) = \frac{1}{MN} \cdot \sum_{m=1}^{M} \left(\sum_{i=1}^{N} L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}_m) \right)$$

is the empirical risk with $L(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}_m)$ being the structured loss of the i-th training example evaluated by the m-th learner. $\Omega(W)$ is the regularization term on the parameters $W = [\mathbf{w}_1, ..., \mathbf{w}_M]$ (as in Section 5.2, typically an $L2$ regularizer on each single $\mathbf{w}_i$), and a bias term is omitted since it does not have an influence on the optimization problem (Lampert 2011). Diversity amongst the M learners is encouraged by the diversity prior $\Gamma(W)$ on the parameters W, where α regulates the degree of diversity. In this way, $\alpha = 0$ reveals the standard SSVM formulation in Equation (5.2), since all M weights converge to the same optimum.

For the ease of argument, let us now assume the training set is linearly separable[21] as in Figure 5.1. Our illustration of the structured learning problem in Figure 5.1 is analogous to representations of flat classification problems, where we regard the ground truth labeling of the structured training samples as the single positive examples and all other (exponentially many) labelings as corresponding negative examples. The objective in Figure 5.1(a) is to find a weight vector $\mathbf{w}$ which separates the positive from the negative examples and maximizes the margin (Tsochantaridis et al. 2005). We define the *version space* $V(\mathcal{D})$ analogously as in flat classification (Mitchell 1997; Herbrich et al. 2004), as

$$V(\mathcal{D}) = \{\mathbf{w} \in \mathcal{W} \mid R_1(\mathbf{w}, \mathcal{D}) = 0\}, \tag{5.9}$$

where R_1 is the empirical risk as in Equation (5.8) with $M = 1$, and $\mathcal{W}$ is the space

[21] Note that this is almost always true once we have a sufficient number of independent features, see the function counting theorem (Cover 1965).

of feasible weight vectors. In other words, the version space is the set of all feasible weight vectors which yield zero loss on the training set $\mathcal{D}$. For linear classifiers, the weight vectors $\mathbf{w} \in \mathcal{W}$ are linear combinations of the training points $\mathbf{x}_i$ (Herbrich et al. 2004), *i.e.* $\mathbf{w} = \sum_{i=1}^{N} c_i \mathbf{x}_i$ for coefficients c_i, and the version space may be restricted appropriately. Note that the error of a structured model induced from a weight vector in version space may still be large for randomly chosen query points (*i.e.* high generalization error), in spite of achieving zero loss on the training set.

Typically, version space is only summarized by a single point such as the center of the largest inscribed sphere (the hard-margin SVM) or the center of mass of the version space (the Bayes point machine (Herbrich et al. 2001)). To learn an ensemble of classifiers, our goal is to distribute M weight vectors $\mathbf{w}_m \in \mathcal{W}$, $m = 1, ..., M$, in version space such that the most diverse predictions on unseen points are obtained. To this end, it is sufficient for structured models with energy functions linear in $\mathbf{w}$ – similar to flat linear classification (Graepel and Herbrich 2000) – to only investigate weight vectors on the unit sphere (*i.e.* $\|\mathbf{w}\|_2 = 1$): At prediction time, labelings are scored by the energy function of the structured model as shown in Equation (5.1). Replacing $\mathbf{w}^*$ by $\lambda\mathbf{w}^*$, $\lambda > 0$, still yields the same ordering of the labelings.

We hence have to solve an experimental design problem on parts of the unit sphere to get an ensemble of *diverse* structured models, in other words – disregarding training data – we want to evenly distribute M points on the unit sphere. Following experimental design (Hardin and Sloane 1993), we introduce in the next section the repulsive diversity energy term $\Gamma(W)$ which makes Equation (5.8) a non-convex optimization problem.

5.3.2 Diversity through Coulomb Potential

Distributing M points evenly on the unit sphere is much studied in information theory and is known as a spherical code (Conway and Sloane 1987): Different variants include *sphere packing* (maximize the minimal angle between any two parameter vectors) and *covering problems* (minimize the distance between any point on the sphere and the closest parameter vector). In three dimensions, the problem is known as the *Thomson problem*[22]: The goal is to minimize the energy configuration of M charges on a unit sphere while the charges repel each other with forces determined by Coulomb's law. While yet unsolved exactly, approximate solutions have been proposed in the literature, including spiral approximations (Saff and Kuijlaars 1997), subdivisions of polyhedrons (Katanforoush and Shahshahani 2003), or gradient descent methods (Claxton and Benson 1966; Erber and Hockney 1991; Lakhbab et al. 2012) which correspond to electrostatic repulsion simulations exploiting Coulomb's law: Particles of equal charge repel each other with a force proportional to the square

[22]Note that we want to approximate this problem in a high dimensional space instead of only 3 dimensions.

of their pairwise distance, the Coulomb force. More generally, in the equilibrium state of the M particles $\mathbf{p}_1, ..., \mathbf{p}_M$ on the unit sphere, the Riesz energy,

$$E_s(\mathbf{p}_1, ..., \mathbf{p}_M) = \sum_{i=1}^{M} \sum_{\substack{j=1 \\ j \neq i}}^{M} \frac{1}{\|\mathbf{p}_i - \mathbf{p}_j\|_2^s} \text{ s.t. } \|\mathbf{p}_i\|_2^2 = 1 \ \forall i \tag{5.10}$$

is minimal. In the following, we set $s = 1$ which yields the Coulomb energy $E_C = E_1$. The Coulomb force which affects particle $\mathbf{p}_i$ amounts to the negative gradient vector of Equation (5.10) *w.r.t.* $\mathbf{p}_i$ (Claxton and Benson 1966; Neubauer et al. 1998; Lakhbab et al. 2012) and is given by

$$\bar{F}_i^C = -\frac{\partial E_C}{\partial \mathbf{p}_i}(\mathbf{p}_1, ..., \mathbf{p}_M) = -\sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_i - \mathbf{p}_j\|_2^3} = \sum_{\substack{j=1 \\ j \neq i}}^{N} \frac{\mathbf{e}_{ij}}{\|\mathbf{p}_i - \mathbf{p}_j\|_2^2}, \tag{5.11}$$

where $\mathbf{e}_{ij}$ is the unit vector from $\mathbf{p}_i$ to $\mathbf{p}_j$. Projecting the resultant of force $\bar{F}_i^C$ on $\mathbf{p}_i$ back to the unit sphere by the projection $\mathcal{P}(\mathbf{p}) = \frac{\mathbf{p}}{\|\mathbf{p}\|}$ yields the projected gradient descent update on $\mathbf{p}_i$,

$$\mathbf{p}_i' = \mathcal{P}(\mathbf{p}_i + \bar{F}_i^C). \tag{5.12}$$

5.3.3 Optimization by an Electrostatic Repulsion Model

In the following, we will specify the diversity term $\Gamma(W)$ in Equation (5.8) and minimize it by utilizing the electrostatic repulsion simulation from the previous section. As derived in Section 5.3.1, the magnitudes of vectors $\mathbf{w}_m$ do not contribute to the diversity term $\Gamma(W)$. Thus, we project the weight vectors to the unit sphere, *i.e.* $\bar{\mathbf{w}}_m = \frac{\mathbf{w}_m}{\|\mathbf{w}_m\|}$, and use the Coulomb energy E_C as the diversity term in Equation (5.8),

$$\Gamma(\mathbf{w}_1, ...\mathbf{w}_M) = E_C(\bar{\mathbf{w}}_1, ..., \bar{\mathbf{w}}_M). \tag{5.13}$$

Note that the weights in both the regularizer $\Omega(W)$ and the risk $R_M(W, \mathcal{D})$ are *not* constrained to the unit sphere.

In Equation (5.12), we derived the projected Coulomb forces which act on the point $\bar{\mathbf{w}}_m$ on the unit sphere. This update step can be projected to $\mathbf{w}_m$ utilizing the intercept theorem (*cf.* Figure 5.3),

$$F_m^C = \|\mathbf{w}_m\|_2^2 \cdot \mathcal{P}(\bar{\mathbf{w}}_m + \alpha \bar{F}_m^C). \tag{5.14}$$

Next, let us derive force F_m^{RR} which acts on particle $\mathbf{w}_m$ according to the regular-

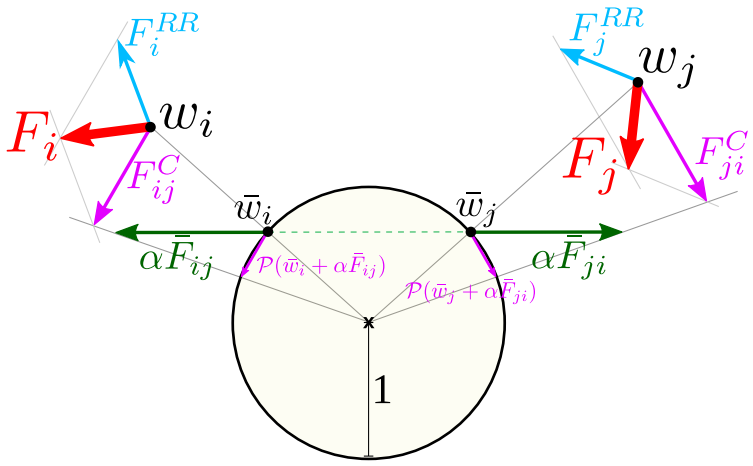


Figure 5.3: In each iteration of the subgradient algorithm, the current weights w of the competing M learners (here: 2) are projected to the unit sphere, $\bar{w}$, their Coulomb forces (green) are computed, and the resultant weight updates $\mathcal{P}(\bar{w} + \alpha\bar{F})$ are projected from the unit sphere to the original weight vectors w, yielding F^C (pink). Independently, the negative gradient of the regularized risk determines forces F^{RR} (blue). Added together, F^{RR} and F^C yield the update F of the weight vector (red).

ized risk $\Omega(W) + C \cdot R_M(W, \mathcal{D})$ in Equation (5.8). The regularized risk in a structured SVM can be minimized using subgradient methods as derived in Section 5.2 and the negative subgradient for the learner m amounts to the force F_m^{RR}, *i.e.* the direction to go in the next optimization step when only considering the regularized risk. When choosing the structured hinge loss as defined in Equation (5.4), then the subgradient $\mathbf{g}_k^m$ for training example k is given analogously to Section 5.2 by

$$\begin{aligned} \hat{\mathbf{y}}^m &= \max_{\mathbf{y} \in \mathcal{Y}} \left(\mathbf{w}_m^\top \phi(\mathbf{x}_k, \mathbf{y}_k) - \mathbf{w}_m^\top \phi(\mathbf{x}_k, \mathbf{y}) + \Delta(\mathbf{y}_k, \mathbf{y}) \right), \\ \mathbf{g}_k^m &= \phi(\mathbf{x}_k, \mathbf{y}_k) - \phi(\mathbf{x}_k, \hat{\mathbf{y}}^m), \end{aligned} \tag{5.15}$$

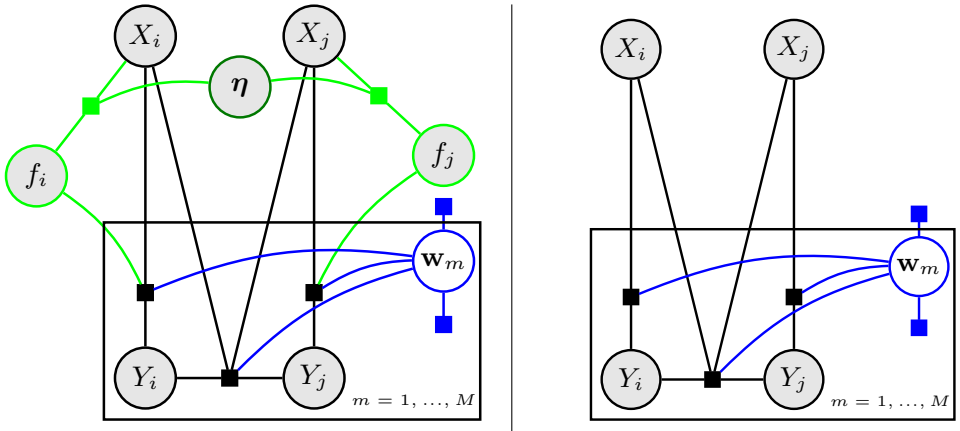
i.e. the regularized risk force on particle $\mathbf{w}_m$ is $F_m^{RR} = -\frac{1}{N} \sum_{k=1}^N \mathbf{g}_k^m$.

Finally, all forces acting on $\mathbf{w}_m$ can be summed to the total force F_m which determines the next update of $\mathbf{w}_m$: $F_m = F_m^{RR} + F_m^C$; in other words, the update of $\mathbf{w}_m$ is given by

$$\mathbf{w}'_m \leftarrow \mathbf{w}_m - \zeta_t \left(\mathbf{w}_m + \frac{C}{N} \sum_{k=1}^N \mathbf{g}_k^m - F_m^C \right), \tag{5.16}$$

or, in the *stochastic* subgradient algorithm with a random $l \in \{1, ..., N\}$,

$$\mathbf{w}'_m \leftarrow \mathbf{w}_m - \zeta_t \left(\mathbf{w}_m + C\mathbf{g}_l^m - F_m^C \right), \tag{5.17}$$



a: CSSVM **b:** CSSVM (simplified)

Figure 5.4: Coulomb Structured Support Vector Machine Training Figure 5.2 is extended to learn an ensemble of multiple model weights $W = [\mathbf{w}_1, ..., \mathbf{w}_m]$ all connected by a repulsive diversity encouraging term, as in the Coulomb Structured Support Vector Machine (CSSVM). Rectangles denote plates (Buntine 1994) and replicate the surrounded objects according to the key in their bottom right corner. In this chapter, we omit the classifier outputs (features in X may be extended appropriately), which makes for the simplified version of the CSSVM graphical model in (b).

where ζ_t is a step size at iteration t and $\mathbf{g}_l^m$ as in Equation (5.15). Note that element $[\mathbf{w}'_m]_i$ may be projected to zero to guarantee submodular energies during training as proved in (Prasad et al. 2014).

For initialization of the CSSVM, we train one SSVM to get the optimum $\mathbf{w}_*$. Then M random perturbations of $\mathbf{w}_*$ give starting points for $\mathbf{w}_1, ...\mathbf{w}_M$.

To illustrate the proposed CSSVM approach, the graphical model from Figure 4.3 is augmented in Figure 5.4 accordingly. Note that here, all model parameters $\mathbf{w}_m$, $m \in \{1, ..., M\}$, are coupled by a common factor, the diversity term $\Gamma(W)$.

5.3.4 Extension: Structured Clustering

Our model suggests a straightforward extension to structured clustering: In the stochastic subgradient update given in Equation (5.17), a random training sample is chosen for each learner to update the weight vector. Instead of random selection, a steered selection of training samples for each individual learner would increase diversity. Similarly to the structured K-means block-coordinate descent algorithm proposed in (Guzman-Rivera et al. 2012), we assign training examples to individual learners: After each subgradient iteration in Section 5.3.3, the task losses

$[\sigma(\boldsymbol{\pi}_i)]_m =$	**Description**	**Abbrev.**
1	Assign the sample i to every learner $m \in \{1, ..., M\}$, *i.e.* Equation (5.16).	**all**
$\mathbb{1}\left[m = \underset{m'}{\operatorname{argmin}}\{\pi_i^{m'}\}\right]$	Assign the sample i to the learner m which achieves the best task loss.	**best**
$\mathbb{1}\left[m = \hat{m}(\pi_i^1, ..., \pi_i^M)\right]$	Sample a learner index $\hat{m}$ from the distribution defined by $q_i^1, ...q_i^M$ and assign the sample i to learner $\hat{m}$; here, $q_i^m = \frac{1-\pi_i^m}{\sum_j (1-\pi_j^m)}$, $\sum_m q_i^m = 1$.	**sampled**
$\mathbb{1}\left[i = \hat{j}_m\right]$	Sample one training example index $\hat{j}_m \in \{1, ..., N\}$ for each learner $m \in \{1, ..., M\}$, *i.e.* Equation (5.17).	**stochastic**

Table 5.1: Possible mappings for the assignment of training samples to individual learners.

$\Delta(\mathbf{y}^m, \mathbf{y}_i; \mathbf{w}_m)$ between prediction $\mathbf{y}^m$ and ground truth $\mathbf{y}_i$ are computed for each learner m, $m \in \{1, ..., M\}$, and normalized over all learners, *i.e.*

$$\pi_i^m = \frac{\Delta(y^m, y_i; \mathbf{w}_m)}{\sum_{k=1}^M \Delta(y^k, y_i; \mathbf{w}_k)}, \quad \sum_m \pi_i^m = 1. \tag{5.18}$$

Training example i is then assigned to any of the M learners according to some indicator vector $\sigma(\boldsymbol{\pi}_i)$, where $[\sigma(\boldsymbol{\pi}_i)]_m = 1$ if training sample i is assigned to learner m, 0 otherwise. In Table 5.1, we propose different alternatives for the mapping $\sigma(\cdot)$. The subgradient update step in Equation (5.16) is then modified accordingly:

$$\mathbf{w}'_m \leftarrow \mathbf{w}_m - \zeta_t \left(\mathbf{w}_m + \frac{C}{\sum_{k=1}^N [\sigma(\boldsymbol{\pi}_k)]_m} \sum_{k=1}^N [\sigma(\boldsymbol{\pi}_k)]_m \cdot \mathbf{g}_k^m - F_m^C \right). \tag{5.19}$$

5.4 Experiments & Results

To evaluate the performance of our approach, we run experiments on three challenging tasks from computer vision: *(i)* co-segmentation, *(ii)* foreground/background segmentation, and *(iii)* semantic segmentation. We use the iCoseg (Batra et al. 2010) database for the first two and PASCAL VOC 2010 (Everingham et al. 2010) for the latter task.

We implemented our algorithm in Python using the PyStruct (Müller and Behnke

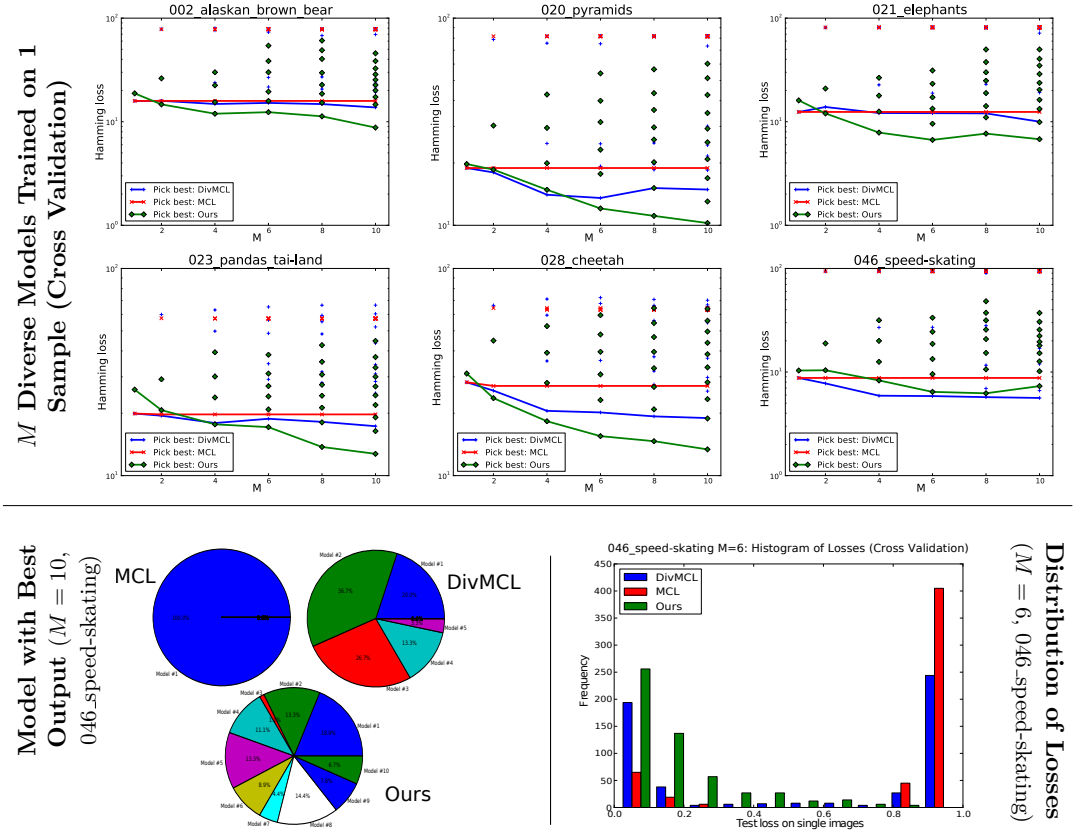


Figure 5.5: **Top:** Hamming losses on the respective datasets of the iCoseg database averaged after cross-validation (lower is better): Each fold consists of exactly one image. We train our model, MCL (Guzman-Rivera et al. 2012), and DivMCL (Guzman-Rivera et al. 2014) on one fold, validate on three other folds, and take the remaining $N_c - 4$ folds as test folds, the errors of which we report. For each test example, we compute the M task losses of the predictions to the ground truth, report the minimum as the pick best error (line), and mark the averages of the second, third, etc. best errors in the graphs. In other words, the line represents the losses which an oracle achieves when selecting always the best out of the M predictions. Note that the average error when always selecting the prediction with highest task error (*i.e.* the worst prediction), is constantly lower in our model than in the competing MCL and DivMCL. **Bottom left:** Frequency of how often model $\#i$, $i \in \{1, ..., M\}$, generates the best test prediction; here $M = 10$, speed-skating dataset. Note that in our algorithm, there is no dominant model and each of the M models achieves the pick-best error on a reasonable number of test samples, whereas in MCL and DivMCL the pick-best losses are attributed to only one or few models, respectively. **Bottom right:** Frequencies of task losses achieved among all test folds and models. All models in our CSSVM ensemble yield predominantly low losses whereas in Div-/MCL many predictions are useless.

2014) framework. The code will be available on the authors' website[23]. On all three tasks, we are comparing our results with the state-of-the-art diversity inducing methods Multiple Choice Learning (Guzman-Rivera et al. 2012) (MCL) and Diverse Multiple Choice Learning (Guzman-Rivera et al. 2014) (DivMCL), the Matlab implementations of which as well as their features/splitting criterions for the iCoseg dataset in task *(ii)* were kindly provided by the authors. The energies for tasks *(i)* and *(ii)* are submodular, and we thus use graph-cut as inference method; for the multi-label problem in *(iii)*, we utilize TRWS (Kolmogorov 2006).

Generating M diverse outputs is particularly useful in early stages of cascaded approaches, where at a later stage, *e.g.* a human or a second complex model may choose the best of M predictions according to a higher-order loss function. The goal of our approach is, hence, to generate M diverse predictions some of which ought to achieve better task loss than the prediction of the single max-margin model. We therefore stick to the evaluation criterion as applied in prior works, where an oracle chooses the best out of M predictions. In this way, we can evaluate the usefulness of such an approach for cascade models. We relate to this loss as *pick best* error, *i.e.* the lowest task loss among the M predictions.

5.4.1 Co-Segmentation

The design of the proposed CSSVM allows to learn an ensemble of diverse models on very small training sets, in fact, even on training sets which consist of one structured training example only. To demonstrate the usefulness of our approach on such tasks, we run experiments on a co-segmentation dataset. The goal in co-segmentation, in general, is the simultaneous segmentation of two images each containing similar objects (Rother et al. 2006). In our experiments, we assume that a model can be learned on the annotations of one image to predict the segmentation of similar images. We choose six categories from the iCoseg database and use the superpixels and features from (Guzman-Rivera et al. 2014), their 12-dimensional color features for the nodes and a contrast-sensitive and -insensitive Potts term for the edges.

The results for MCL, DivMCL, and our model are depicted in Figure 5.5. For each category, we vary the number of models in the ensemble M from 1 to 10, where $M = 1$ may be viewed as the baseline and corresponds to the training of a standard SSVM. We perform a full N_c-fold crossvalidation on each category, where N_c is the number of images in category c, and report the test losses of all M models. We choose the regularization and diversity trade-off parameters of each method on a hold-out validation set consisting of three images per category. Note that these losses are computed on superpixel level rather than pixel level which makes for a fair comparison since all three models are using the same superpixels and features. In these datasets,

[23] http://github.com/martinsch/

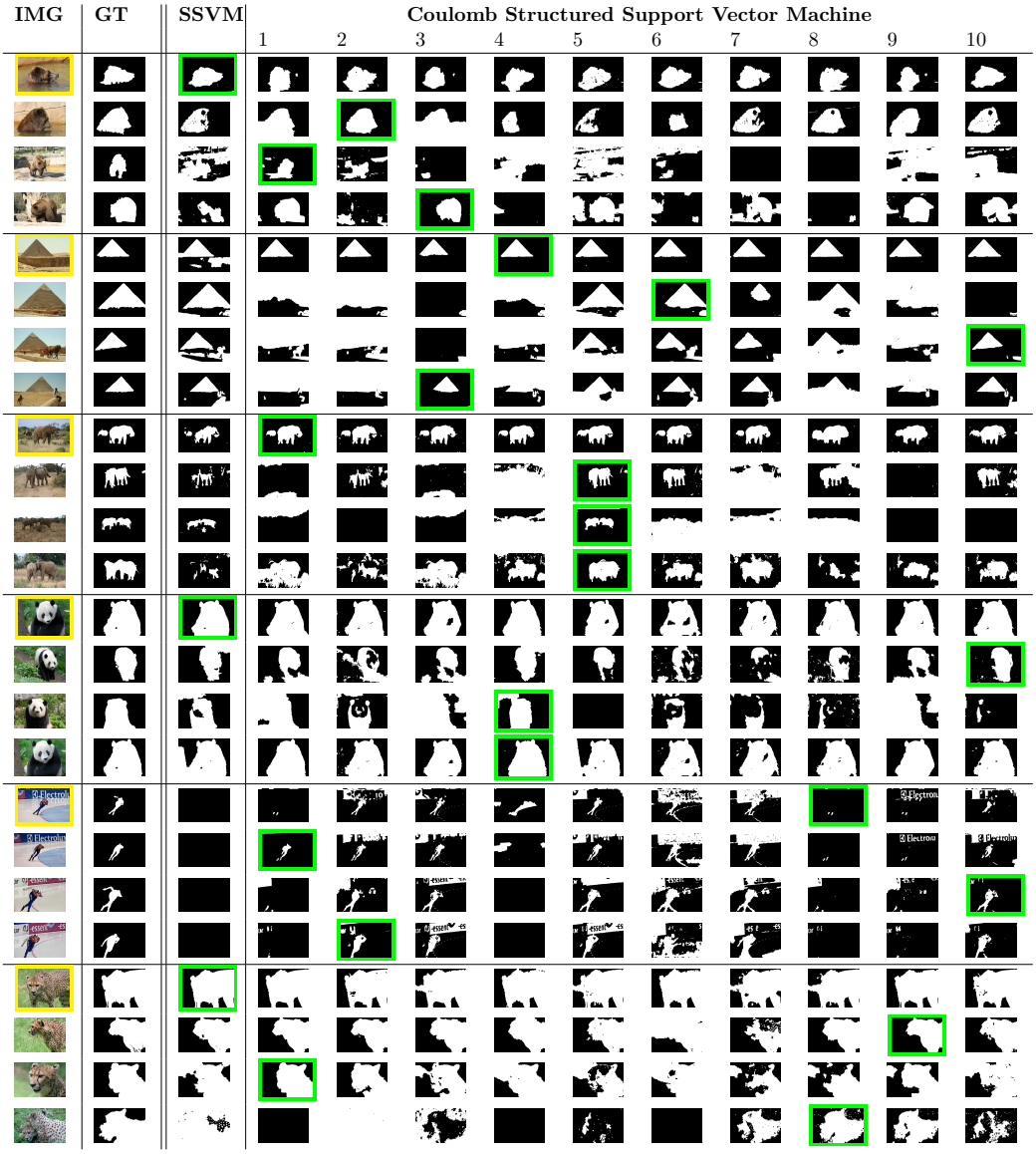


Figure 5.6: Foreground/background Co-segmentation (white/black, respectively). The single training image in each dataset is marked in yellow, the best prediction is framed in green. Note that all $M = 10$ models of CSSVM fit the training images similarly well, whereas high diversity amongst the M models is present in the predictions of the test set. GT stands for ground truth.

N_c's are in the range of 10 to 33, dependent on the dataset. Obviously, we use strategy "all" from Table 5.1 for these experiments.

It should be noted that, if we took the same implementations, exactly the same losses for all three competing models for $M = 1$ would be obtained (since all three models are direct generalizations of SSVM). The deviations here are probably due

to different optimization strategies, *e.g.* different minima on a plateau or not enough iterations for the subgradient method (Div-/MCL use cutting-plane optimization instead).

On all six datasets, our method clearly improves over the baseline of only one SSVM ($M = 1$) and achieves better pick-best errors for large M than MCL and DivMCL do, with the exception of the *speed-skating* category. We show for this category exemplarily, however, that our algorithm learns M models which are all performing similarly well while in DivMCL only few models are strong, and in MCL, there exists only one strong model since diversity is only encouraged by assigning the training samples (here: 1) to specific models (shown for $M = 10$ in bottom left of Figure 5.5). The phenomenon that our method yields significantly better average errors across all predictors in the ensemble is also reflected in the histogram of all losses from the full cross validation, as provided in Figure 5.5 bottom right. The fact that most of the predictions achieve low loss in the proposed CSSVM is a strong advantage when the model is used in a cascade model since *all* predictions are good candidates to be selected as the best solution.

Example images for $M = 10$ are presented in Figure 5.6. Note that for CSSVM, all models in the ensemble achieve similar training performances while yielding high diversity on the test images. By design, diversity on the training samples is *not* rewarded but models are distributed diversely in version space as argued in Section 5.3.1 in order to achieve a low generalization error on unseen data when the predictions of all M models are considered jointly. This is in contrast to the competing methods, where diversity among the models is also enforced on the training set.

5.4.2 Foreground/background Segmentation

In this experiment, we use all these categories together (166 images in total) and use the same split criterion for the 5-fold cross validation as in (Guzman-Rivera et al. 2014). We train the models on one fold, select regularization and diversity trade off parameters on two validation folds and report the test error on the remaining two folds. Figure 5.7 presents the results for MCL, DivMCL, and our model with different sample assignment strategies as in Table 5.1. Since this dataset consists of different categories, it seems natural that the models which cluster the training data by assigning training instances to distinct models (as in Div-/MCL, Ours-sampled, and Ours-best) perform better than the models which try to fit all M models to the *entire* dataset (Ours-all, Ours-stochastic). Our model achieves similar accuracies as the state-of-the-art method DivMCL in this experiment.

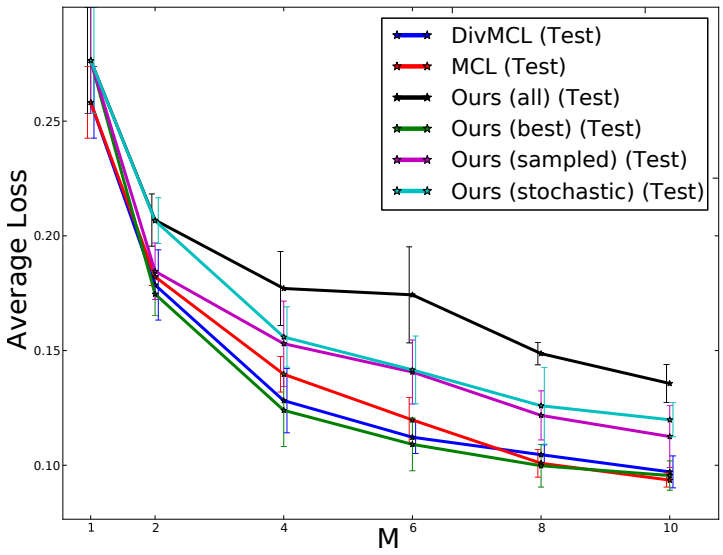


Figure 5.7: Foreground/background Segmentation (iCoseg). Average pick-best error (Hamming distance, lower is better) on the set of all categories. Shown are the test errors with one standard deviation (error bars are slightly perturbed on the x-axis for illustration purposes). Our training sample assignment strategies are denoted as in Table 5.1.

5.4.3 Semantic Segmentation

We also evaluate our algorithm on the PASCAL VOC 2010 benchmark dataset for object class segmentation (challenge 5). The dataset consists of an official training set and validation set comprising 964 images each, which contain 21 object classes. We use the SLIC superpixels and Textonboost potentials (Krähenbühl and Koltun 2011) publicly available from (Müller and Behnke 2014). Due to the lack of a publicly available test set, we are selecting the parameters of all three models on the official validation set and report these validation errors in Table 5.2 using the PASCAL VOC evaluation criterion, the Jaccard index. For structured learning, all models use a loss weighted by the inverse class frequency present in the training data.

The baselines for this experiment are given by an argmax operation on our features ("unaries only"), a linear SVM on the unary features, and a structured SVM ($M = 1$). With the publicly available features, these baselines achieve average accuracies of 21.6%, 27.4%, and 29.1% which is much lower than the current best results reported on this challenge. In this experiment, however, we want to focus on how much a baseline algorithm can be improved thanks to a diverse ensemble, and not indulge in feature and pipeline tuning.

By training $M = 6$ diverse models and selecting the best predictions amongst them

according to the ground truth, all three competing methods yield significantly higher pick-best accuracies than a single SSVM. We can even improve the accuracy from 29.1% to 37.6% with the assignment strategy "best" (*cf.* Table 5.1). This massive relative improvement underlines the usefulness of a diverse ensemble approach. MCL (35.0%) and DivMCL (34.5%) yield inferior performance.

Method	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	diningtable	dog	horse	motorbike	person	plant	sheep	sofa	train	tvmonitor	**Average Accuracy**
Unaries only	80.2	25.0	0.1	10.6	14.3	13.8	32.1	44.0	30.0	4.9	9.5	4.4	11.9	15.4	27.5	35.5	10.5	19.8	12.0	28.6	22.3	21.6
Linear SVM	80.0	36.6	2.8	17.3	23.0	25.6	40.4	48.7	27.6	8.3	19.5	10.5	13.3	21.9	34.4	36.4	16.9	22.8	17.0	37.0	34.3	27.4
SSVM (M=1)	79.9	39.9	2.1	18.5	27.5	28.4	43.2	49.2	28.7	8.4	21.6	12.3	14.1	23.7	35.2	37.2	22.0	23.6	18.3	38.9	39.4	29.1
MCL (M=6)	82.0	49.1	1.0	31.5	21.2	31.4	55.3	57.7	37.0	12.0	33.0	27.9	28.0	28.8	40.9	39.4	15.1	32.1	23.2	42.4	46.5	35.0
DivMCL (M=6)	82.2	30.3	0.5	25.7	26.4	30.4	51.1	56.3	42.7	7.9	33.5	22.9	45.4	27.3	45.6	43.3	21.3	39.5	17.4	42.9	32.2	34.5
Ours (M=6, all)	83.4	44.4	1.7	37.4	34.1	34.2	47.7	54.9	42.8	8.9	34.4	22.8	40.4	24.7	33.2	44.5	25.7	29.1	20.3	40.1	41.3	35.5
Ours (M=6, stochastic)	83.5	40.1	2.4	25.1	23.0	28.5	57.4	51.8	35.3	8.4	33.7	18.9	31.3	24.9	37.9	42.0	22.7	37.8	24.3	44.4	51.3	34.5
Ours (M=6, best)	83.2	48.8	3.2	38.3	28.4	33.3	58.1	60.3	51.1	7.7	34.5	21.6	34.6	32.0	39.3	43.4	17.7	27.7	26.6	48.3	51.3	**37.6**
Ours (M=6, sampled)	83.9	42.4	1.7	27.6	27.5	33.1	55.9	53.0	46.6	7.6	34.1	25.6	34.6	26.1	41.6	45.6	27.4	32.6	25.8	46.5	48.4	36.6

Table 5.2: Pascal VOC 2010 Validation Accuracy (higher is better). We tune a popular conditional random field (Müller and Behnke 2014) as baseline structured models (top rows). We here focus on the relative improvement that different diversity strategies can achieve (bottom rows), rather than tweaking the baseline model itself.

6

Discussion

Tracking-by-assignment methods, formulated as graphical models, are well suited for cell tracking. We have shown that their modeling flexibility renders it possible to cope with cell division and segmentation errors in one holistic tracking model. Furthermore, the machinery naturally coming with probabilistic graphical model representations may be exploited for parameter learning or uncertainty estimation.

Structure In this final chapter, we summarize our contributions in the subsequent section. We conclude in Section 6.2 with a discussion of the limitations of the proposed algorithms and relevant future research directions.

6.1 Conclusions & Contributions

This work presented two approaches for cell tracking which reduce and correct errors which originally raised from the clear separation into a detection and a tracking phase in previous works. The synergy effects which appear in our novel formulations make it possible to improve over state-of-the-art. Moreover, we discussed how uncertainty measures may be derived from tracking-by-assignment approaches and how these could guide the user for more efficient proof-reading. We finally presented a method to learn multiple diverse graphical models to decrease the expected generalization error for predictions on test data. We now summarize our contributions in detail.

6.1.1 Higher Accuracy through Synergy Effects

The tracking of dividing cells involves phenomena that are not commonly found in other typical tracking applications. In particular, one wants to track multiple targets

that are almost indistinguishable, that may divide (undergo mitosis), that may be undersegmented due to mutual occlusion or low contrast (yielding cells falsely clustered together in only one detection), and that may be recorded in the presence of noise, provoking false positive detections.

Most existing tracking-by-assignment approaches break down the problem by solving the segmentation / detection and the assignment / association problems separately. To prevent the propagation of errors resulting from wrong but fixed decisions, it is desirable to solve the detection and the assignment problems jointly.

(a) Identify and Correct Segmentation Errors

We presented in Chapter 2 the first principled treatment that addresses all of the above complications simultaneously. Our formulation is in terms of an undirected graphical model which – due to the explicit modeling of global conservation laws – can robustly correct errors from the previous detection step. We have shown that the proposed factor graph can outperform a recently published cell tracking method on sequences of proliferating cells in dense populations thanks to

- the joint consideration of over- and undersegmentation errors in one probabilistic model: Guaranteeing consistent solutions through conservation laws makes it possible to identify and correct errors from the preceding segmentation step such as false detections or falsely clustered cells. It can thus tolerate detection errors from a foregoing detection step.
- the coupling of all detections over all time steps in one model: This ensures maximal temporal context and decisions are not only based on local influences.
- the exact optimization: We solve this model to global optimality using integer linear programming. We thus avoid approximation errors.

In addition, our model can partition and track previously merged objects *while preserving their original identities*.

We empirically demonstrated the effectiveness of this novel tracking method on three different and challenging 2D+t and 3D+t datasets from developmental biology and present state-of-the-art results.

(b) Avoid Segmentation Errors through Joint Optimization of Segmentation and Tracking

The work in Chapter 3 was motivated by the desire to further increase the synergy effects between the two (previously independent) steps of detection and tracking by

joint optimization of the two steps together in one model. In response, we proposed an undirected graphical model which simultaneously selects a subset of competing segmentation hypotheses and links them across time steps. All of these decisions are made to interact so as to reach the overall most likely interpretation of the data. Although the model is one of significant complexity, it remains solvable to global optimality in practicable runtimes of less than an hour on the large datasets used.

The benefits of this approach were borne out by experimental results which are a significant improvement over the state-of-the-art, in particular in terms of the division detection accuracy reached.

6.1.2 Uncertainty Measures for Guided Proof-Reading

The formulation of cell tracking in terms of probabilistic graphical models, allows us to sample multiple likely solutions from these models. In Chapter 4, we first proposed to apply the recently developed technique of Perturb-and-MAP (Papandreou and Yuille 2011) random fields to sample multiple instances of graphical models through the synthetic injection of global noise. Moreover, we presented a novel procedure based on Gaussian processes in order to estimate *local* uncertainty in the observed data. The latter method proved particularly useful for the robust detection of cell division errors. Since training sets for cell division detection are typically small, an explicit modeling of the ambiguity of such events is crucial for accurate cell tracking results.

The uncertainty measures derived from the variations between these proposal solutions guide the user to the most ambiguous predictions. We demonstrated on a challenging *Drosophila* dataset shown in Figure 1.1 that these measures guide the user reliably to false predictions for proof-reading. Human interaction time is hence greatly reduced compared to full manual tracking (or random guidance) while achieving a much higher tracking accuracy than with purely automatic methods.

6.1.3 Parameter Learning for Diverse Models to Improve Generalization on Test Data

Previous work in cell tracking has shown that the discriminative learning of parameters of tracking-by-assignment models may lead to significantly better results than the hand-tuning of the model weights (Lou et al. 2014b). We applied a structured learning strategy to our tracking model in Chapter 4 which sped up training time from multiple days as for the grid search used in Chapters 2 and 3 to a few hours for large datasets.

We furthermore enhanced structured learning techniques in Chapter 5 by the

proposition of a novel algorithm termed the *Coulomb Structured Support Vector Machine* (CSSVM). The CSSVM learns an ensemble of multiple models which yield *diverse* predictions on test data. This method models a diversity prior on the set of model weights, trading off diversity in the test data against data fit. We demonstrated on numerous real world datasets that selecting the best prediction from the set of diverse outputs yields to significantly improved results compared to only one model trained. Furthermore, we outperform on challenging computer vision tasks state-of-the-art methods for diverse output predictions. The applicability of the proposed method to cell tracking is straightforward and is left for future work.

6.1.4 Dense Ground-Truth for Dataset from Developmental Biology

One major challenge to develop cell tracking algorithms is that there are only very few public datasets with ground truth available. In response, we manually annotated 100 time steps of late developmental stages of a *Drosophila* embryo, and released this dataset. The data was utilized in Chapters 2–4 for evaluation purposes and is described in Appendix A. Three example frames are shown in Figure 1.1.

The dataset together with our manual annotations may be used free of charge by the cell tracking community to develop and evaluate their algorithms.

6.1.5 Open Source Development

In addition, we made the code developed for the cell tracking models proposed in Chapters 2 and 3 available to the public. Moreover, we plan to release the code of the methods proposed in Chapters 4 and 5 in the future. This open source development will facilitate other researchers to use our algorithms for their applications, advance these methods, compare performances of competitive methods, or reproduce the results achieved in this work.

6.1.6 Applicability for Non-expert Users

As mentioned in Section 1.1, to allow for impacts in life sciences, our algorithms must be approachable for non-expert computer users. It is hence not sufficient to only release the research code, but graphical user interfaces (GUIs) need to be provided to ease the use of the proposed methods. We therefore implemented the *Conservation tracking* algorithm from Chapter 2 into *ilastik*[24], a tool for automatic analysis of

[24] www.ilastik.org

high-dimensional image data with an easy-to-use GUI. Its availability for all major platforms allows users worldwide to use our methods on their own data and active development guarantees support in the future.

6.2 Limitations & Outlook

There are several immediately relevant avenues for future work arising from the research done in this thesis.

6.2.1 Gap Closing: Find Undetected Cells

In Chapters 2 and 3, we proposed graphical models which refine previously segmented cells by taking into account temporal context. Both methods are initiated from a segmentation or an oversegmentation, and can automatically correct errors from this preceding stage by the coupling of the two stages in tracking-by-assignment approaches.

There is, however, one type of segmentation error which has not been explicitly addressed so far, viz *undetected* objects. We proposed in Chapter 3 to circumvent this phenomena by a more conservative segmentation method with high recall and low precision, *i.e.* detecting considerably more objects than present in the data. The joint segmentation and tracking model can then assign false positive detections to background. Yet this strategy comes at the cost of a significant increase of random variables and thus shifts the problem towards the already hard optimization problem. Besides, due to low contrast or noise in the data, it is not always possible to pick up each cell in each frame without temporal context.

As a result, tracking-by-assignment models should be refined to close gaps between falsely disappearing and later reappearing cells. Introducing higher-order edges over multiple time frames is, unfortunately, out of question due to the high complexity it would introduce to the optimization problem. Instead, Jaqaman et al. (2008) propose a linear program formulation to link tracks that terminate early with those that start in later frames, and Dicle et al. (2013) present a regression method which relies on locally constant motion patterns. These approaches could be extended to cope with divisions of objects. It is also conceivable to integrate the raw signals recorded in the movies into these approaches by, for instance, locally reducing the threshold of the segmentation when gaps are detected.

6.2.2 Approximate or Decoupled Inference

The exact optimization over a maximally possible time range in our cell tracking algorithms has proven to yield highly accurate results. However, this comes at the

cost of solving an NP-hard optimization problem. While runtimes less than an hour are achieved on the datasets used throughout this work with ≤ 1000 cells per frame, it is obvious that exact optimization will not scale to tens of thousands of cells.

It is thus necessary to focus research on the acceleration of the inference process. Different directions include:

- **Approximate Inference**: If approximate results are sufficient for the application, the optimization problem may be simplified. For instance, integer linear programming allows to perform *approximate* inference by early stopping and comes with upper bounds for the gap to the real optimum. Furthermore, greedy optimization strategies as in (Magnusson et al. 2014) may be followed, however, at the cost of high approximation errors. In this case, crafty heuristics are necessary to revoke and refine previous decisions.

- **Domain Decomposition**: One straightforward strategy could be a so-called *sliding window* approach, where the time sequence is dissected into multiple chunks with at least one frame overlap. These sequences can then be solved subsequently, fixing the result of the previous chunk on the respective overlap to guarantee consistency in the overlaps. Another possibility which even guarantees global optimality, is to decompose the graphical model temporally (and/or spatially) into smaller portions with overlaps, for each of which inference is fast. Dual decomposition techniques (Komodakis et al. 2007) then guarantee globally consistent solutions by iteratively enforcing a consensus on the overlaps. To scale up, these subproblems may also be solved approximately. Funke (2014, Chapter 5) experiments with dual decomposition on similar problem structures and concludes that, unfortunately, the runtime even increases with these methods. It is hence necessary to turn to more informed decomposition strategies.

Furthermore, coupling the fast method of Amat et al. (2014), which achieves outstanding performance on one-to-one cell associations (*moves*), with our graphical model formulations to precisely detect cell divisions, may be a good compromise between exact inference and approximate optimization for the easier decisions. In particular, a set of tracklets[25] may be generated with the Gaussian mixture model based method of Amat et al. (2014) and subsequently, may be stitched together through a graphical model formulation similar to those proposed in this work, while, at the same time, finding cell divisions. This might yield both significant speed-ups and performance improvements due to less ambiguities.

[25] As a reminder from Chapter 2, a *tracklet* is a fixed linking of one object over multiple frames.

6.2.3 Higher-order Relations through Tracklets

Such tracklets not only speed-up inference for easy decisions, they come with another great advantage. When stitching them together, expressive higher-order features may be computed on the tracklets (Pirsiavash et al. 2011; Arora and Globerson 2013). Higher-order features such as acceleration, constant velocity or direction, or other higher-order motion dynamics, avoidance of outliers within one track in terms of appearance features, etc., are prohibitive to model in a framework consisting of single-frame objects only, since they span over multiple frames and hence involve exponentially many objects and higher-order factors. In contrast, they are cheap to determine on a small set of tracklet proposals.

The bottom line is that although tracklets involve irrevocable hard decisions, if only used for highly likely assignments, they may decrease the problem size drastically and at the same time allow for more expressive modeling power due to higher-order features.

Note that although we introduced tracklets in Chapter 2, we did not exploit the use of more expressive features but only reduced the number of variables in the integer linear program. Including these features in this simplified representation obviously increases the number of model parameters to be learned.

6.2.4 Non-linear Energy Parameterization

To make structured learning feasible in our models, we are restricted to use linear parameterizations of the energies. Kernel based extensions of the structured support vector machine (SSVM) have been presented (Tsochantaridis et al. 2005), however, they involve a prohibitive number of kernel evaluations during training. To avoid expensive training, our work in Chapters 4 and 5 is hence based on *linear* SSVMs, using features from unstructured non-linear classifiers. Approximations for *non-linear* SSVM may be investigated. One attempt has been taken in (Lucchi et al. 2012) where a set of features is constructed from kernel evaluations with support vectors in a previously trained (kernel-based) support vector machine. Their results seem promising and their method should be directly applicable to our applications.

6.2.5 Re-ranking of Proposal Solutions

The method we proposed in Chapter 5 yields diverse predictions for structured output problems and in the experimental evaluation, an oracle selects the best among these predictions. The method may be extended by a second-stage model which automatically selects the best of these predictions based on a previously trained model which,

in turn, takes into account higher-order features. Note that due to the same reasons as outlined in Section 6.2.3, it is not possible to model these features already in the first-stage model.

Applied to cell tracking, such an automatic re-ranking (Yadollahpour et al. 2013) may yield a significant decrease in proof-reading time.

6.2.6 Fusion of Proposal Solutions

Ultimately, however, one would like to fuse the (diverse) proposal solutions (and the corrections made by the user) to one consistent solution. One possible strategy could be to rely on move-making paradigms as done in (Beier et al. 2014).

Furthermore, if it is possible to fuse multiple solutions *consistently*, concurrent solutions resulting from early stopping in domain decomposition based approaches as discussed in Section 6.2.2 could be corrected.

To conclude, since major insights in life sciences rely heavily on a highly accurate tracking of cells, human verifications will always be necessary. The focus of research on cell tracking algorithms will hence slowly shift from the design of reliable cell tracking algorithms to the question of how to ***efficiently*** *integrate a human expert into the cell tracking process, trading off the human interaction time with accuracy of results. The cell tracking models, parameter learning, and proof-reading guidance presented in this thesis, are an important step towards this goal.*

A

Drosophila Dataset Description

The fruit fly (*Drosophila*) dataset on which we evaluate the algorithms presented in this thesis, has been recorded by the Hufnagel group, EMBL, Heidelberg. The acquisition process is described in detail in (Krzic et al. 2012).

For evaluation purposes, we cropped the volumes to the dimensions described in Table A.1 and depicted in Figure 1.1, segmented the cells automatically with *ilastik* (Sommer et al. 2011), and refined the result with a seeded watershed. We then tracked *all* segmented cells manually[26] over 100 time steps. For this purpose, we developed the *Manual Tracking workflow* in *ilastik*, which is made publicly available on `http://www.ilastik.org`.

[26] Philipp Hanslovsky and Christoph Klein (both University of Heidelberg) assisted with the manual annotations.

	Original Dataset	**Cropped Dataset for Manual Annotations**
Organism	*Drosophila melanogaster* embryo	
Dataset label	20120803_170524 Fusion Histone	
Stage in embryonic development	Gastrulation	
Fluorescent protein	H2Av-mCherry	
Microscope	Multiview selective-plane illumination microscope (MuVi-SPIM) (Krzic et al. 2012)	
Volume dimensions	$1050 \times 435 \times 361$ voxels	$730 \times 320 \times 30$ voxels
Voxel size	$0.52\mu m \times 0.52\mu m \times 0.52\mu m$ (3D isotropic)	
Temporal resolution	1 frame per 30 seconds	
Time sequence	650 time steps	frames 300–399

Table A.1: The dataset was recorded by the Hufnagel group, EMBL Heidelberg, Germany. We extracted from the original dataset the volumes (t, x, y, z) from $(300, 160, 60, 0)$ to $(400, 890, 380, 30)$ for dense manual tracking annotations. These were acquired by Martin Schiegg, Philipp Hanslovsky, and Christoph Klein (University of Heidelberg, Germany).

B

Runtime Comparisons

For an analysis of runtimes of the algorithms proposed in this work, we compare the *Conservation tracking* method introduced in Chapter 2 and the joint model for segmentation and tracking proposed in Chapter 3 with the method from (Amat et al. 2014). For this purpose, we run each algorithm on a contemporary workstation (2x Intel(R) Xeon(R) CPU E5606, 4x 2.13 GHz each, GeForce GTX 680). A detailed comparison of implementation language, parallel implementations, and runtimes (wall clock time) in seconds are provided in Table B.1.

In total, tracking using TGMM (Amat et al. 2014) runs significantly faster than using the graphical models of Conservation Tracking (CT) from Chapter 2 or the joint segmentation and tracking model (JST) proposed in Chapter 3. This is not only due to the GPU implementation vs. the CPU implementations but mainly comes from the nature of the underlying model: Whereas in TGMM the tracking is based on a forward propagation of the parameters of the Gaussian mixture models based on a short history only, the graphical models (CT and JST) aim at solving the exact optimization problem taking all temporal information in one holistic model into account. The optimization problems in these graphical models are formulated as energy minimizations and the integer linear programs are solved exactly. Using approximate solvers may significantly speed up inference, but inconsistent solutions may occur. We leave research on approximations for future work as discussed in Section 6.2.

Moreover, it should be noted that all of the expensive preprocessing computations to construct the graphical models can be run without user interaction and intermediate results may be stored to enable fast parameter training. Computing features on the (competing) segmented cell candidates allow for expressive classifiers to be used as priors and are an important ingredient to achieve high accuracies in cell division detection.

Furthermore, the biggest portion in runtime of the joint tracking and segmentation approach is due to an expensive operation in our implementation for the export of the MAP solution of the graphical model: To ease evaluation of results, we export the sequence of images/volumes with a unique cell lineage identifier on each and every pixel/voxel. In contrast, the export of results in CT or TGMM is much faster since, for our evaluation, we only export a mapping from an object identifier to a lineage identifier or the coordinates of the centroid (and a lineage identifier), respectively.

Method **Workflow Component**	**Implementation**	**Sequential/Parallel**	**Dataset (Runtimes in sec.)** Drosophila	 Rat stem cells
Conservation tracking (Chapter 2)				
Segmentation with ilastik (Sommer et al. 2011)	Python/C++	parallel	842	277
Object feature extraction and classification	Python/C++	partly parallel	745	259
Tracking w/o inference	C++	sequential	111	105
Optimization (with CPLEX)	commercial	parallel	270	2181
TGMM (Amat et al. 2014) on raw data				
Hierarchical Segmentation	C++	parallel	64	57
Tracking using Gaussian mixture models	GPU	parallel	118	126
Joint segmentation and tracking (Chapter 3)				
Prediction maps with ilastik (Sommer et al. 2011)	Python/C++	parallel	787	347
Generation of competing segmentation hypotheses	Python/C++	sequential	4811	1779
Object feature extraction and classification	Python/C++	parallel	2240	933
Graphical model construction	C++	sequential	620	1291
Optimization (with CPLEX)	commercial	parallel	498	4032
(Pixelwise) Export of results	C++	sequential	9276	3555

Table B.1: Runtime comparison in seconds (wall clock time).

List of Figures

List of Tables

Bibliography

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc. ISBN: 0-13-617549-X (cited on p. 6).

Amat, F. and Keller, P. J. (2013). "Towards comprehensive cell lineage reconstructions in complex organisms using light-sheet microscopy". In: *Development, Growth & Differentiation* 55.4, pp. 563–578 (cited on pp. 1, 2).

Amat, F., Lemon, W., Mossing, D. P., McDole, K., Wan, Y., Branson, K., Myers, E. W., and Keller, P. J. (2014). "Fast, accurate reconstruction of cell lineages from large-scale fluorescence microscopy data". In: *Nature Methods* July. ISSN: 1548-7091. DOI: 10.1038/nmeth.3036 (cited on pp. 30, 41–44, 46, 47, 88, 93, 95).

Amat, F., Myers, E. W., and Keller, P. J. (2013). "Fast and robust optical flow for time-lapse microscopy using super-voxels". In: *Bioinformatics* 29.3, pp. 373–380 (cited on p. 30).

Andres, B., Beier, T., and Kappes, J. H. (2012). "OpenGM: A C++ Library for Discrete Graphical Models". In: *CoRR* (cited on p. 39).

Arora, C. and Globerson, A. (2013). "Higher Order Matching for Consistent Multiple Target Tracking". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 177–184. DOI: 10.1109/ICCV.2013.29 (cited on p. 89).

Arteta, C., Lempitsky, V., Noble, J. A., and Zisserman, A. (2013). "Learning to Detect Partially Overlapping Instances". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3230–3237. DOI: 10.1109/CVPR.2013.415 (cited on p. 30).

Barber, D. (2012). *Bayesian Reasoning and Machine Learning*. Cambridge University Press. ISBN: 0521518148 (cited on p. 8).

Batra, D., Kowdle, A., Parikh, D., Luo, J., and Chen, T. (2010). "iCoseg: Interactive co-segmentation with intelligent scribble guidance". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3169–3176. DOI: 10.1109/CVPR.2010.5540080 (cited on p. 74).

Batra, D., Yadollahpour, P., Guzman-Rivera, A., and Shakhnarovich, G. (2012). "Diverse M-best solutions in Markov random fields". In: *European Conference on Computer Vision (ECCV)*. DOI: 10.1007/978-3-642-33715-4_1 (cited on pp. 63–65).

Beier, T., Kroeger, T., Kappes, J. H., Koethe, U., and Hamprecht, F. A. (2014). "Cut, Glue & Cut: A Fast, Approximate Solver for Multicut Partitioning". In: *IEEE Conference on*

Computer Vision and Pattern Recognition (CVPR), pp. 73–80. DOI: 10.1109/CVPR.2014.17 (cited on p. 90).

Ben Shitrit, H., Berclaz, J., Fleuret, F., and Fua, P. (2011). "Tracking multiple people under global appearance constraints". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 137–144. DOI: 10.1109/ICCV.2011.6126235 (cited on p. 14).

Bise, R., Yin, Z., and Kanade, T. (2011). "Reliable Cell Tracking by Global Data Association". In: *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, pp. 1004–1010. ISBN: 9781424441280. DOI: 10.1109/ISBI.2011.5872571 (cited on pp. 6, 7, 27, 30, 47, 48).

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc. ISBN: 0387310738 (cited on p. 8).

Bose, B., Wang, X., and Grimson, E. (2007). "Multi-class object tracking algorithm that handles fragmentation and grouping". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2007.383175 (cited on p. 14).

Boyd, S., Xiao, L., and Mutapcic, A. (2003). "Subgradient methods". In: *Lecture Notes of EE392o, Stanford University, Autumn Quarter* (cited on p. 68).

Breiman, L. (2001). "Random forests". In: *Machine Learning* 45.1, pp. 5–32. DOI: 10.1023/A:1010933404324 (cited on pp. 17, 21, 40, 53).

Brendel, W., Amer, M., and Todorovic, S. (2011). "Multiobject tracking as maximum weight independent set". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1273–1280. ISBN: 978-1-4577-0394-2. DOI: 10.1109/CVPR.2011.5995395 (cited on p. 31).

Brendel, W. and Todorovic, S. (2010). "Segmentation as Maximum-Weight Independent Set". In: *Neural Information Processing Systems (NIPS)*, pp. 307–315 (cited on p. 31).

Budvytis, I., Badrinarayanan, V., and Cipolla, R. (2011). "Semi-supervised video segmentation using tree structured graphical models". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2257–2264. DOI: 10.1109/CVPR.2011.5995600 (cited on p. 29).

Buntine, W. L. (1994). "Operations for learning with graphical models". In: *Journal of Artificial Intelligence Research* 2, pp. 159–225 (cited on p. 73).

Chen, C., Kolmogorov, V., Zhu, Y., Metaxas, D. N., and Lampert, C. H. (2013). "Computing the M Most Probable Modes of a Graphical Model". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 161–169 (cited on p. 65).

Chen, C., Liu, H., Metaxas, D. N., and Zhao, T. (2014). "Mode Estimation for High Dimensional Discrete Tree Graphical Models". In: *Neural Information Processing Systems (NIPS)*, pp. 1323–1331 (cited on p. 65).

Claxton, T. and Benson, G. (1966). "Stereochemistry and Seven Coordination". In: *Canadian Journal of Chemistry* 44.2, pp. 157–163 (cited on pp. 70, 71).

Conway, J. H. and Sloane, N. J. A. (1987). *Sphere-packings, Lattices, and Groups*. Springer-Verlag New York, Inc. (cited on p. 70).

Coutu, D. L. and Schroeder, T. (2013). "Probing cellular processes by long-term live imaging–historic problems and current solutions". In: *Journal of Cell Science* 126.17, pp. 3805–3815 (cited on p. 2).

Cover, T. M. (1965). "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14.3, pp. 326–334. ISSN: 0367-7508. DOI: 10.1109/PGEC.1965.264137 (cited on p. 69).

Dicle, C., Camps, O. I., and Sznaier, M. (2013). "The Way They Move: Tracking Targets with Similar Appearance". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2304–2311. DOI: 10.1109/ICCV.2013.286 (cited on p. 87).

Enzweiler, M. and Gavrila, D. M. (2009). "Monocular pedestrian detection: Survey and experiments". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.12, pp. 2179–2195. DOI: 10.1109/TPAMI.2008.260 (cited on p. 4).

Erber, T. and Hockney, G. M. (1991). "Equilibrium configurations of N equal charges on a sphere". In: *Journal of Physics A: Mathematical and General* 24.23 (cited on p. 70).

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). *The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results*. [Online; last accessed 09/02/2015]. URL: http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html (cited on p. 74).

Fox, E. B., Choi, D. S., and Willsky, A. S. (2006). "Nonparametric Bayesian methods for large scale multi-target tracking". In: *Asilomar Conference on Signals, Systems, and Computers*, pp. 2009–2013. DOI: 10.1109/ACSSC.2006.355118 (cited on p. 14).

Funke, J., Andres, B., Hamprecht, F. A., Cardona, A., and Cook, M. (2012). "Efficient automatic 3D-reconstruction of branching neurons from EM data". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1004–1011. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6247777 (cited on pp. 6, 30).

Funke, J. (2014). "Automatic Neuron Reconstruction from Anisotropic Electron Microscopy Volumes". PhD thesis. Institute of Neuroinformatics, ETH Zurich (cited on pp. 55, 88).

Gane, A., Hazan, T., and Jaakkola, T. (2014). "Learning with Maximum A-Posteriori Perturbation Models". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 247–256 (cited on pp. 63, 65).

Gimpel, K., Batra, D., Dyer, C., Shakhnarovich, G., and Tech, V. (2013). "A Systematic Exploration of Diversity in Machine Translation". In: *Conference on Empirical Methods on Natural Language Processing* (cited on p. 63).

González, G., Fusco, L., Benmansour, F., Fua, P., Pertz, O., and Smith, K. (2013). "Automated quantification of morphodynamics for high-throughput live cell time-lapse datasets". In: *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, pp. 664–667. DOI: 10.1109/ISBI.2013.6556562 (cited on p. 2).

GPy authors, the (2014). *GPy: A Gaussian process framework in Python*. https://github.com/SheffieldML/GPy (cited on p. 55).

Graepel, T. and Herbrich, R. (2000). "The Kernel Gibbs Sampler". In: *Neural Information Processing Systems (NIPS)*, pp. 514–520 (cited on p. 70).

Guzman-Rivera, A., Batra, D., and Kohli, P. (2012). "Multiple Choice Learning: Learning to Produce Multiple Structured Outputs". In: *Neural Information Processing Systems (NIPS)*, pp. 1808–1816 (cited on pp. 63, 65, 66, 73, 75, 76).

Guzman-Rivera, A., Kohli, P., Batra, D., and Rutenbar, R. A. (2014). "Efficiently Enforcing Diversity in Multi-Output Structured Prediction". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 284–292 (cited on pp. 63, 65, 66, 75, 76, 78).

Hardin, R. H. and Sloane, N. J. A. (1993). "A new approach to the construction of optimal designs". In: *Journal of Statistical Planning and Inference* 37.3, pp. 339–369 (cited on p. 70).

Hazan, T., Maji, S., and Jaakkola, T. (2013). "On sampling from the Gibbs distribution with random maximum a-posteriori perturbations". In: *Neural Information Processing Systems (NIPS)*, pp. 1268–1276 (cited on p. 65).

Herbrich, R., Graepel, T., and Campbell, C. (2001). "Bayes Point Machines". In: *Journal of Machine Learning Research* 1, pp. 245–279. ISSN: 1532-4435 (cited on p. 70).

Herbrich, R., Graepel, T., and Williamson, R. C. (2004). *The Structure of Version Space*. Tech. rep. MSR-TR-2004-63. Microsoft Research, p. 17 (cited on pp. 69, 70).

Höckendorf, B., Thumberger, T., and Wittbrodt, J. (2012). "Quantitative analysis of embryogenesis: A perspective for light sheet microscopy". In: *Developmental Cell* 23.6, pp. 1111–1120 (cited on pp. 1, 2, 4).

Hofmann, M., Wolf, D., and Rigoll, G. (2013). "Hypergraphs for Joint Multi-View Reconstruction and Multi-Object Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3650–3657. DOI: 10.1109/CVPR.2013.468 (cited on p. 30).

Ion, A., Carreira, J., and Sminchisescu, C. (2011). "Image Segmentation by Figure-Ground Composition into Maximal Cliques". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2110–2117. DOI: 10.1109/ICCV.2011.6126486 (cited on p. 31).

Jaqaman, K., Loerke, D., Mettlen, M., Kuwata, H., Grinstein, S., Schmid, S. L., and Danuser, G. (2008). "Robust single-particle tracking in live-cell time-lapse sequences". In: *Nature Methods* 5.8, pp. 695–702 (cited on p. 87).

Jug, F., Pietzsch, T., Kainmüller, D., Funke, J., Kaiser, M., Nimwegen, E. van, Rother, C., and Myers, G. (2014). "Optimal Joint Segmentation and Tracking of Escherichia Coli in the Mother Machine". In: *Bayesian and Graphical Models for Biomedical Imaging (BAMBI-MICCAI)*, pp. 25–36. DOI: 10.1007/978-3-319-12289-2_3 (cited on pp. 30, 48).

Kachouie, N. N., Fieguth, P. W., Ramunas, J., and Jervis, E. (2006). "Probabilistic model-based cell tracking". In: *International Journal of Biomedical Imaging*. DOI: 10.1155/IJBI/2006/12186 (cited on p. 6).

Kanade, T., Yin, Z., Bise, R., Huh, S., Eom, S., Sandbothe, M. F., and Chen, M. (2011). "Cell image analysis: Algorithms, system and applications". In: *IEEE Workshop on Applications of Computer Vision (WACV)*, pp. 374–381. DOI: 10.1109/WACV.2011.5711528 (cited on p. 2).

Katanforoush, A. and Shahshahani, M. (2003). "Distributing points on the sphere". In: *Experimental Mathematics* 12.2, pp. 199–209. DOI: 10.1080/10586458.2003.10504492 (cited on p. 70).

Kausler, B. X., Schiegg, M., Andres, B., Lindner, M. S., Koethe, U., Leitte, H., Wittbrodt, J., Hufnagel, L., and Hamprecht, F. A. (2012b). "A Discrete Chain Graph Model for 3d + t Cell Tracking with High Misdetection Robustness". In: *European Conference on Computer Vision (ECCV)*, pp. 144–157. DOI: 10.1007/978-3-642-33712-3_11 (cited on pp. 7, 14, 21, 22, 25, 27, 30).

Keller, P. J., Schmidt, A. D., Santella, A., Khairy, K., Bao, Z., Wittbrodt, J., and Stelzer, E. H. (2010). "Fast, high-contrast imaging of animal development with scanned light sheet-based structured-illumination microscopy". In: *Nature Methods* 7.8, pp. 637–642 (cited on p. 1).

Keller, P. J., Schmidt, A. D., Wittbrodt, J., and Stelzer, E. H. (2008). "Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy". In: *Science* 322.5904, pp. 1065–1069 (cited on pp. 1, 2).

Al-Kofahi, O., Radke, R. J., Goderie, S. K., Shen, Q., Temple, S., and Roysam, B. (2006). "Automated Cell Lineage Construction: A Rapid Method to Analyze Clonal Development Established with Murine Neural Progenitor Cells". In: *Cell Cycle* 5.3, pp. 327–335 (cited on p. 47).

Koller, D. and Friedman, N. (2010). *Probabilistic Graphical Models* (cited on pp. 8, 9, 15, 53).

Kolmogorov, V. (2006). "Convergent tree-reweighted message passing for energy minimization". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.10, 1568–1583. DOI: 10.1109/TPAMI.2006.200 (cited on p. 76).

Komodakis, N., Paragios, N., and Tziritas, G. (2007). "MRF optimization via dual decomposition: Message-passing revisited". In: *IEEE International Conference on Computer Vision (ICCV)*. DOI: 10.1109/ICCV.2007.4408890 (cited on p. 88).

Kragel, B., Herman, S., and Roseveare, N. (2012). "A Comparison of Methods for Estimating Track-to-Track Assignment Probabilities". In: *IEEE Transactions on Aerospace and Electronic Systems* 48.3, pp. 1870–1888. ISSN: 0018-9251. DOI: 10.1109/TAES.2012.6237567 (cited on p. 49).

Krähenbühl, P. and Koltun, V. (2011). "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials". In: *Neural Information Processing Systems (NIPS)*, pp. 109–117 (cited on p. 79).

Krzic, U., Gunther, S., Saunders, T. E., Streichan, S. J., and Hufnagel, L. (2012). "Multiview light-sheet microscope for rapid in toto imaging". In: *Nature Methods* 9.7, pp. 730–733. ISSN: 1548-7105. DOI: 10.1038/nmeth.2064 (cited on pp. 1, 3, 5, 91, 92).

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. (2001). "Factor graphs and the sum-product algorithm". In: *IEEE Transactions on Information Theory* 47.2, pp. 498–519. ISSN: 0018-9448. DOI: 10.1109/18.910572 (cited on pp. 8, 16, 33).

Lakhbab, H., Bernoussi, S. E., and Harif, A. E. (2012). "Energy Minimization of Point Charges on a Sphere with a Spectral Projected Gradient Method". In: *International Journal of Scientific & Engineering Research* 3.5 (cited on pp. 70, 71).

Lampert, C. H. (2011). "Maximum margin multi-label structured prediction". In: *Neural Information Processing Systems (NIPS)*, pp. 289–297 (cited on p. 69).

Lezama, J., Alahari, K., Sivic, J., and Laptev, I. (2011). "Track to the future: Spatio-temporal video segmentation with long-range motion cues". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3369–3376. DOI: 10.1109/CVPR.2011.6044588 (cited on p. 29).

Livet, J., Weissman, T. A., Kang, H., Draft, R. W., Lu, J., Bennis, R. A., Sanes, J. R., and Lichtman, J. W. (2007). "Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system". In: *Nature* 450.7166, pp. 56–62. ISSN: 1476-4687. DOI: 10.1038/nature06293 (cited on p. 4).

Lou, X. and Hamprecht, F. A. (2011). "Structured Learning for Cell Tracking". In: *Neural Information Processing Systems (NIPS)*, pp. 1296–1304 (cited on pp. 14, 66).

Lou, X. and Hamprecht, F. A. (2012). "Structured Learning from Partial Annotations". In: *International Conference on Machine Learning (ICML)* (cited on p. 66).

Lou, X., Koethe, U., Wittbrodt, J., and Hamprecht, F. A. (2012). "Learning to Segment Dense Cell Nuclei with Shape Prior". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1012–1018. DOI: 10.1109/CVPR.2012.6247778 (cited on pp. 23, 30).

Lou, X., Schiegg, M., and Hamprecht, F. A. (2014b). "Active Structured Learning for Cell Tracking: Algorithm, Framework and Usability". In: *IEEE Transactions on Medical Imaging* 33.4, pp. 849–860. ISSN: 0278-0062. DOI: 10.1109/TMI.2013.2296937 (cited on pp. 49, 85).

Lucchi, A., Li, Y., Smith, K., and Fua, P. (2012). "Structured image segmentation using kernelized features". In: *European Conference on Computer Vision (ECCV)*. Springer, pp. 400–413. DOI: 10.1007/978-3-642-33709-3_29 (cited on p. 89).

Luo, R. C., Yih, C.-C., and Su, K. L. (2002). "Multisensor fusion and integration: approaches, applications, and future research directions". In: *Sensors Journal* 2.2, pp. 107–119. ISSN: 1530-437X. DOI: 10.1109/JSEN.2002.1000251 (cited on p. 4).

Magnusson, K., Jalden, J., Gilbert, P., and Blau, H. (2014). "Global linking of cell tracks using the Viterbi algorithm". In: *IEEE Transactions on Medical Imaging* PP.99. ISSN: 0278-0062. DOI: 10.1109/TMI.2014.2370951 (cited on pp. 6, 14, 88).

Maji, S., Hazan, T., and Jaakkola, T. (2014). "Efficient Boundary Annotation using Random Maximum A-Posteriori Perturbations". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)* (cited on pp. 49, 63).

Maška, M., Ulman, V., Svoboda, D., Matula, P., Matula, P., Ederra, C., Urbiola, A., España, T., Venkatesan, S., Balak, D. M. W., Karas, P., Bolcková, T., Štreitová, M., Carthel, C., Coraluppi, S., Harder, N., Rohr, K., Magnusson, K. E. G., Jaldén, J., Blau, H. M., Dzyubachyk, O., Křížek, P., Hagen, G. M., Pastor-Escuredo, D., Jimenez-Carretero, D., Ledesma-Carbayo, M. J., Muñoz-Barrutia, A., Meijering, E., Kozubek, M., and Ortiz-de-Solorzano, C. (2014). "A Benchmark for Comparison of Cell Tracking Algorithms". In: *Bioinformatics* 30.11, pp. 1609–1617. ISSN: 1460-2059. DOI: 10.1093/bioinformatics/btu080 (cited on p. 2).

Meijering, E., Dzyubachyk, O., and Smal, I. (2012). "Methods for cell and particle tracking". In: *Methods in Enzymology: Live Cell Imaging* 504.9, pp. 183–200 (cited on pp. 2, 4).

Meijering, E., Dzyubachyk, O., Smal, I., and Cappellen, W. A. van (2009). "Tracking in cell and developmental biology". In: *Seminars in Cell & Developmental Biology*. Vol. 20. 8. Elsevier, pp. 894–902 (cited on pp. 2, 47).

Mitchell, T. M. (1997). *Machine Learning*. 1st ed. McGraw-Hill, Inc. ISBN: 0070428077 (cited on p. 69).

Müller, A. C. and Behnke, S. (2014). "PyStruct - Learning Structured Prediction in Python". In: *Journal of Machine Learning Research* 15.1, pp. 2055–2060 (cited on pp. 74, 79, 81).

Neubauer, Schilling, Watkins, and Zeitlin (1998). "An algorithm for finding potential minimizing configurations of points on a sphere". In: [Online; last accessed 09/02/2015]. URL: http://www.csun.edu/~hcmth007/algorithm.html (cited on p. 71).

Nillius, P., Sullivan, J., and Carlsson, S. (2006). "Multi-Target Tracking – Linking Identities using Bayesian Network Inference". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2, pp. 2187–2194. DOI: 10.1109/CVPR.2006.198 (cited on p. 14).

Nowozin, S. and Lampert, C. H. (2011). "Structured learning and prediction in computer vision". In: *Foundations and Trends in Computer Graphics and Vision* 6.3–4, pp. 185–365. DOI: 10.1561/0600000033 (cited on pp. 63, 66, 68).

Padfield, D. R., Rittscher, J., and Roysam, B. (2009). "Coupled minimum-cost flow cell tracking". In: *Information Processing in Medical Imaging*. Ed. by J. Prince, D. Pham, and K. Myers. Vol. 5636. Lecture Notes in Computer Science. Springer, pp. 374–85. ISBN: 978-3-642-02497-9. DOI: 10.1007/978-3-642-02498-6_31 (cited on pp. 14, 47, 48).

Papandreou, G. and Yuille, A. L. (2011). "Perturb-and-MAP random fields: Using discrete optimization to learn and sample from energy models". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 193–200. DOI: 10.1109/ICCV.2011.6126242 (cited on pp. 48–50, 63–65, 85).

Park, C., Huang, J. Z., Ji, J. X., and Ding, Y. (2013). "Segmentation, Inference and Classification of Partially Overlapping Nanoparticles". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.3. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.163 (cited on p. 30).

Pirsiavash, H., Ramanan, D., and Fowlkes, C. C. (2011). "Globally-Optimal Greedy Algorithms for Tracking a Variable Number of Objects". In: *IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR), pp. 1201–1208. DOI: 10.1109/CVPR.2011.5995604 (cited on p. 89).

Prasad, A., Jegelka, S., and Batra, D. (2014). "Submodular meets Structured: Finding Diverse Subsets in Exponentially-Large Structured Item Sets". In: *Neural Information Processing Systems (NIPS)*, pp. 2645–2653 (cited on p. 73).

Premachandran, V., Tarlow, D., and Batra, D. (2014). "Empirical Minimum Bayes Risk Prediction: How to extract an extra few% performance from vision models with just three more parameters". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1043–1050. DOI: 10.1109/CVPR.2014.137 (cited on p. 63).

Rapoport, D. H., Becker, T., Madany Mamlouk, A., Schicktanz, S., and Kruse, C. (2011). "A novel validation algorithm allows for automated cell tracking and the extraction of biologically meaningful parameters". In: *PLOS ONE* 6.11, e27315. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0027315 (cited on pp. 40, 42, 43, 46, 49).

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian processes for machine learning*. Vol. 14. 2. ISBN: 026218253X (cited on pp. 51, 52).

Ratliff, N. D., Bagnell, J. A., and Zinkevich, M. A. (2007). "(Online) Subgradient Methods for Structured Prediction". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 380–387 (cited on p. 67).

Roig, G., Boix, X., Nijs, R. de, Ramos, S., Kühnlenz, K., and Van Gool, L. J. (2013). "Active MAP Inference in CRFs for Efficient Semantic Segmentation". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2312–2319. DOI: 10.1109/ICCV.2013.287 (cited on p. 63).

Rother, C., Minka, T. P., Blake, A., and Kolmogorov, V. (2006). "Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 993–1000. DOI: 10.1109/CVPR.2006.91 (cited on p. 76).

Rubio, J. C., Serrat, J., Lopez, A. M., and Ponsa, D. (2012). "Multiple-Target Tracking for Intelligent Headlights Control". In: *IEEE Transactions on Intelligent Transportation Systems* 13.2, pp. 594–605. ISSN: 1524-9050. DOI: 10.1109/TITS.2011.2175219 (cited on p. 6).

Saff, E. B. and Kuijlaars, A. B. (1997). "Distributing many points on a sphere". In: *The Mathematical Intelligencer* 19.1, pp. 5–11 (cited on p. 70).

Sahni, S. (1974). "Computationally related problems". In: *SIAM Journal on Computing* 3.4, pp. 262–279. DOI: 10.1137/0203021 (cited on pp. 11, 14).

Schiegg, M., Diego, F., and Hamprecht, F. A. (2015a). *Learning Diverse Models: The Coulomb Structured Support Vector Machine*. (under review) (cited on p. 64).

Schiegg*, M., Hanslovsky*, P., Haubold, C., Koethe, U., Hufnagel, L., and Hamprecht, F. A. (2014). "Graphical Model for Joint Segmentation and Tracking of Multiple Dividing Cells". In: *Bioinformatics*. [* contributed equally], in press. DOI: 10.1093/bioinformatics/btu764 (cited on p. 27).

Schiegg, M., Hanslovsky, P., Kausler, B. X., Hufnagel, L., and Hamprecht, F. A. (2013). "Conservation Tracking". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2928–2935. DOI: 10.1109/ICCV.2013.364 (cited on pp. 11, 14).

Schiegg, M., Heuer, B., Haubold, C., Wolf, S., Koethe, U., and Hamprecht, F. A. (2015b). "Proof-reading guidance in cell tracking by sampling from tracking-by-assignment models". In: *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*. (in press) (cited on p. 47).

Sommer, C., Straehle, C., Koethe, U., and Hamprecht, F. A. (2011). "ilastik: Interactive learning and segmentation toolkit". In: *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*, pp. 230–233. ISBN: 978-1-4244-4127-3. DOI: 10.1109/ISBI.2011.5872394 (cited on pp. 22, 39, 91, 95).

Sulston, J. E. and Horvitz, H. R. (1977). "Post-embryonic cell lineages of the nematode, Caenorhabditis elegans". In: *Developmental Biology* 56.1, pp. 110–156 (cited on p. 2).

Taskar, B., Chatalbashev, V., Koller, D., and Guestrin, C. (2005). "Learning structured prediction models: A large margin approach". In: *International Conference on Machine Learning (ICML)*. ACM, pp. 896–903 (cited on p. 67).

Taskar, B., Guestrin, C., and Koller, D. (2004). "Max-Margin Markov Networks". In: *Neural Information Processing Systems (NIPS)*, pp. 25–32 (cited on p. 9).

Tomer, R., Khairy, K., Amat, F., and Keller, P. J. (2012). "Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy". In: *Nature Methods* 9.7, pp. 755–763. ISSN: 1548-7105. DOI: 10.1038/nmeth.2062 (cited on pp. 1, 5).

Tsochantaridis, I., Joachims, T., Hofmann, T., and Altun, Y. (2005). "Large Margin Methods for Structured and Interdependent Output Variables". In: *Journal of Machine Learning Research* 6, pp. 1453–1484. ISSN: 1532-4435 (cited on pp. 10, 63, 66, 67, 69, 89).

Vazquez-Reina, A., Avidan, S., Pfister, H., and Miller, E. L. (2010). "Multiple Hypothesis Video Segmentation from Superpixel Flows". In: *European Conference on Computer Vision (ECCV)*, pp. 268–281. DOI: 10.1007/978-3-642-15555-0_20 (cited on p. 29).

Wainwright, M. J. and Jordan, M. I. (2008). "Graphical models, exponential families, and variational inference". In: *Foundations and Trends in Machine Learning* 1.1-2, pp. 1–305 (cited on p. 8).

Wang, C., La Gorce, M. de, and Paragios, N. (2009). "Segmentation, ordering and multi-object tracking using graphical models". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 747–754. DOI: 10.1109/ICCV.2009.5459247 (cited on p. 11).

Williams, C. K. I. and Barber, D. (1998). "Bayesian Classification With Gaussian Processes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.12, pp. 1342–1351. ISSN: 0162-8828. DOI: 10.1109/34.735807 (cited on p. 52).

Wu, Z., Thangali, A., Sclaroff, S., and Betke, M. (2012). "Coupling Detection and Data Association for Multiple Object Tracking". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1948–1955. DOI: 10.1109/CVPR.2012.6247896 (cited on p. 11).

Xiong, G., Feng, C., and Ji, L. (2006). "Dynamical Gaussian mixture model for tracking elliptical living objects". In: *Pattern Recognition Letters* 27.7, pp. 838–842. DOI: 10.1016/j.patrec.2005.11.015 (cited on p. 29).

Yadollahpour, P., Batra, D., and Shakhnarovich, G. (2013). "Discriminative Re-ranking of Diverse Segmentations". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1923–1930. DOI: 10.1109/CVPR.2013.251 (cited on pp. 63, 90).

Yanover, C. and Weiss, Y. (2003). "Finding the M most probable configurations using loopy belief propagation". In: *Neural Information Processing Systems (NIPS)* (cited on pp. 63, 64).

Yilmaz, A., Javed, O., and Shah, M. (2006). "Object tracking: A Survey". In: *ACM Computing Surveys* 38.4. ISSN: 03600300. DOI: 10.1145/1177352.1177355 (cited on p. 4).

Zhang, L., Li, Y., and Nevatia, R. (2008). "Global data association for multi-object tracking using network flows". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2008.4587584 (cited on pp. 6, 14).

List of Publications

Peer-reviewed Journal Articles

Schiegg*, M., Hanslovsky*, P., Haubold, C., Koethe, U., Hufnagel, L., and Hamprecht, F. A. (2014). "Graphical Model for Joint Segmentation and Tracking of Multiple Dividing Cells". In: *Bioinformatics*. [* contributed equally], in press. DOI: 10.1093/bioinformatics/btu764.

Peer-reviewed Conference Articles

Schiegg, M., Diego, F., and Hamprecht, F. A. (2015a). *Learning Diverse Models: The Coulomb Structured Support Vector Machine*. (under review).

Schiegg, M., Hanslovsky, P., Kausler, B. X., Hufnagel, L., and Hamprecht, F. A. (2013). "Conservation Tracking". In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 2928–2935. DOI: 10.1109/ICCV.2013.364.

Schiegg, M., Heuer, B., Haubold, C., Wolf, S., Koethe, U., and Hamprecht, F. A. (2015b). "Proof-reading guidance in cell tracking by sampling from tracking-by-assignment models". In: *IEEE International Symposium on Biomedical Imaging: From Nano to Macro (ISBI)*. (in press).

Contributions beyond the Scope of this Thesis

Fiaschi, L., Diego, F., Gregor, K., **Schiegg, M.**, Koethe, U., Zlatic, M., and Hamprecht, F. A. (2014). "Tracking Indistinguishable Translucent Objects over Time using Weakly Supervised Structured Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2736–2743. DOI: 10.1109/CVPR.2014.356.

Kausler, B. X., **Schiegg, M.**, Andres, B., Lindner, M. S., Koethe, U., Leitte, H., Wittbrodt, J., Hufnagel, L., and Hamprecht, F. A. (2012a). "A Discrete Chain Graph Model for 3d + t Cell Tracking with High Misdetection Robustness". In: *European Conference on Computer Vision (ECCV)*, pp. 144–157. DOI: 10.1007/978-3-642-33712-3_11.

Lou, X., **Schiegg, M.**, and Hamprecht, F. A. (2014a). "Active Structured Learning for Cell Tracking: Algorithm, Framework and Usability". In: *IEEE Transactions on Medical Imaging* 33.4, pp. 849–860. ISSN: 0278-0062. DOI: 10.1109/TMI.2013.2296937.