Dissertation

submitted to the
Combined Faculties for the Natural Sciences and for Mathematics
of the Ruperto-Carola University of Heidelberg, Germany
for the degree of
Doctor of Natural Sciences

Put forward by
Dipl.-Phys. Bernhard Kausler
born in Hirschau

Oral examination: 2013-07-19

Tracking-by-Assignment as a Probabilistic Graphical Model with Applications in Developmental Biology

Referees: Prof. Dr. Fred A. Hamprecht
Prof. Dr. Tilmann Gneiting

Zusammenfassung

Tracking-durch-Zuordnung als ein probabilistisches graphisches Modell mit Anwendungen in der Entwicklungsbiologie

Diese Dissertation präsentiert einen neuartigen Ansatz um eine variable Anzahl von ähnlich aussehenden und sich teilenden Objekten in Gegenwart einer nicht zu vernachlässigenden Anzahl von falsch-positiv Detektionen (mehr als 10%) zu tracken. Er wird dazu verwendet, Zellstammbäume in sich entwickelnden Zebrabärbling- und Fruchtfliegen-Embryos aus 3d Zeitrafferaufnahmen zu rekonstruieren. Das zugrunde liegende Modell ist ein *chain graph* – ein gemischt gerichtetes und ungerichtetes probabilistisches graphisches Modell. Ein Tracking wird aus der Maximum-a-posteriori Konfiguration des Modells über alle Zeitschritte gleichzeitig ermittelt.

Das Tracking Modell dient als zweiter Schritt einer Pipeline zur Aufzeichnung von Digital Embryos, d.h. Karten aller Zellen in einem Embryo zusammen mit ihrer Entwicklungshistorie. Der erste Schritt der Pipeline besteht aus der Segmentierung von Zellkernen, die mit Fluoreszenzmarkern in Lichtscheiben-Mikroskopie Bildern sichtbar gemacht wurden.

Die Pipeline ist als Software mit einer intuitiven, grafischen Benutzeroberfläche implementiert. Es ist das erste frei verfügbare Programm seiner Art und macht die präsentierten Methoden einem breiten Anwenderkreis aus den Lebenswissenschaften zugänglich.

Summary

Tracking-by-Assignment as a Probabilistic Graphical Model with Applications in Developmental Biology

This thesis presents a novel approach for tracking a varying number of divisible objects with similar appearance in the presence of a non-negligible number of false positive detections (more than 10%). It is applied to the reconstruction of cell lineages in developing zebrafish and fruit fly embryos from 3d time-lapse recordings. The model takes the form of a *chain graph*—a mixed directed-undirected probabilistic graphical model—and a tracking is obtained simultaneously over all time slices from the maximum a-posteriori configuration.

The tracking model is used as the second step in a two-step pipeline to produce digital embryos—maps of cell nuclei in an embryo and their ancestral fate; the first step being the segmentation of the fluorescently-stained cell nuclei in light sheet microscopy images.

The pipeline is implemented as a software with an intuitive graphical user interface. It is the first freely available program of its kind and makes the presented methods accessible to a broad audience of users from the life sciences.

Contents

Sources of Figures

No third party sources were used for the figures presented in the thesis at hand. Some figures appeared previously in Kausler et al. (2012). Fig. 2.1 on page 9 is based on an illustration from the paper by Lou et al. (2011) which I coauthored.

Acknowledgements

Prof. Fred Hamprecht supervised this thesis and he once told me that he likes to see things grow. I want to thank him for planting me in his group and I hope he is satisfied with what he seeded.

I thank Prof. Tilmann Gneiting for being my second adviser and taking the time to delve into this project.

The endeavor to create digital embryos is a collaborative effort and I would like to thank all the great people who participated in the work at hand. In the beginning there were Xinghua Lou, Frederik Kaster, and Martin Lindner. Frederik Kaster formulated the "optimal joint assignment" tracking method (Sec. 3.3 on page 19) and Martin Lindner implemented the cell object features (Sec. 4.5.2 on page 37). Xinghua Lou orchestrated our first efforts to implement a digital embryo recording pipeline (Chap. 2 on page 9). Then Martin Schiegg joined the team. He implemented and evaluated the "minimal cell cycle length extension" (Sec. 4.4.2 on page 36) and manually assembled the lineage tree synopsis (Appendix B). Jörg Kappes ran experiments to compare inference methods on the chain graph model and others (Sec. 5.3.2 on page 50). Björn Andres helped with etching out the details of the chain graph model (Chap. 4 on page 21). The VOLUMINA library was developed in symbiosis with Thorben Kröger (Sec. 6.2 on page 62). Prof. Heike Leitte provided the code to plot lineage trees (Fig. 5.6 on page 55 and Appendix B). The Random Forest implementation (Sec. 4.5.2 on page 37) is part of Ullrich Köthe's excellent VIGRA library. I especially thank Ullrich Köthe for asking the critical questions before the reviewers could.

During the last three years I had the pleasure to collaborate with several outstanding scientists. I thank Prof. Heike Leitte and Jens Fangerau for providing advice all things visualization and granting us access to their software SCIFER which let us see the embryos for the first time. I thank Prof. Jochen Wittbrodt and Burkhard Höckendorf for providing zebrafish datasets, Alexis Maizel for sharing his Arabidopis datasets, and Lars Hufnagel and his students for their jaw-dropping Drosophila recordings.

All members in my group provided steady intellectual and emotional support. We had a great time together. Anna Kreshuk shared many offices with me and she was the first of my colleagues who became my friend. Along with her I want to thank them, namely: Ullrich Köthe, Thorben Kröger, Christoph Straehle, Xinghua Lou, Frederik Kaster, Bernhard Renard, Michael Hanselmann, Marc Kirchner, Martin Lindner, Björn Andres, Christoph Sommer, Martin Schiegg, Jan Funke, Luca Fiaschi, Ferran Diego, Chong Zhang, Melih Kandemir, Nathan Hüsken, Rahul Nair, Philipp Hanslovsky, Buote Xu, Robert Walecki, Thorsten Beier, Svenja Reith,

Markus Nullmeier, Kemal Eren, and Stuart Berg. For their support in all administrative affairs, I would like to thank Barbara Werner, Evelyn Wilhem, and Ole Hansen.

I gratefully acknowledge the financial support I received by the German Research Foundation (DFG) within the programme "Spatio-/Temporal Graphical Models and Applications in Image Analysis", grant GRK 1653. I thank Prof. Christoph Schnörr, Dr. Thordis Thorarinsdottir, and all the other PIs in the programme for providing great learning opportunities and insightful interdisciplinary discussions. I thank my fellows in the research training group for their comradeship, namely, Borislav Antic, Fabian Bachl, Eva-Maria Didden, Johannes Dück, Dominic Edelmann, Jochen Fiedler, Gabriell Mate, Annette Möller, Fabian Rathke, Bernhard Schmitzer, Markus Speth, Jörg Kappes, Bogdan Savchynskyy, and Alex Lenkoski.

Finally, I thank my wife Tiffy for her loving support.

1. Introduction

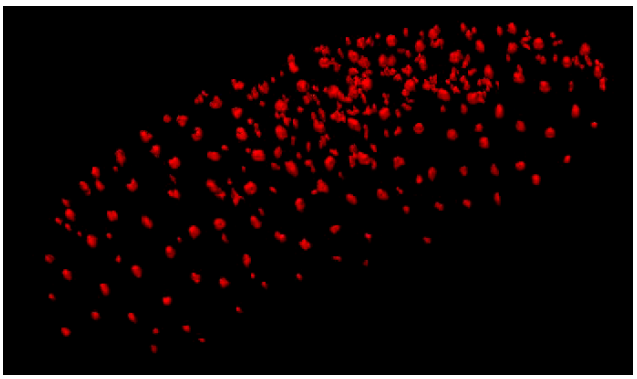
1.1. False Positive Detection-tolerant Tracking-by-Assignment

One grand challenge of developmental biology is to find the lineage tree of all cells in a growing embryo, i.e. the complete ancestry of each cell (Meijering et al., 2009). To date the complete cell lineage is only known for the nematode *Caenorhabditis Elegans* which was manually reconstructed in the early 80s from 2D light microscopy images. The task was greatly assisted by the fact that the nematode's lineage is invariant between specimens. This made it possible to extract a single lineage by processing many specimens (Sulston et al., 1983; Sulston, 2002). As a next step the same should be achieved for more complex organisms with lineages that are in general not invariant. Besides model organisms that are at least semi-transparent and the availability of suitable cell markers, a new kind of 3d microscopy had to be developed which allows the embryos to survive sufficiently long under observation and provides short enough image acquisition times to record typical cell activities like movements and divisions.

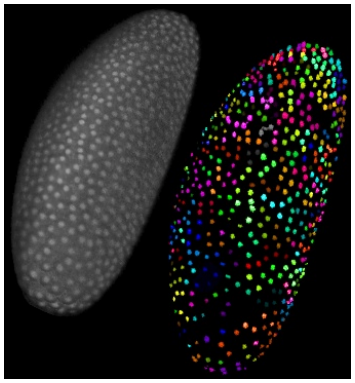
Recently, a new light sheet-based microscopy technology was introduced that exhibits sufficiently low phototoxicity (Jemielita et al., 2012) and can acquire fully isotropic 3d images at a rate of up to three 3d images per minute at a resolution suitable to identify cell nuclei and membranes (Santi, 2011; Weber and Huisken, 2011; Keller and Dodt, 2012). Sequences showing a significant portion of the embryonic development have already been successfully recorded for model organisms like zebrafish (Keller et al., 2008), *Drosophila* (Keller et al., 2010; Krzic et al., 2012), and the plant *Arabidopsis thaliana* (Maizel et al., 2011). The immediate goal of these experiments is the recording of *digital embryos*—the complete record of cell locations, shapes, tracks, and ancestry relations.

A reliable digital embryo recording pipeline is a necessary condition for large scale quantitative studies of embryonic development. Besides the contribution to fundamental science this could lead to the development of new drugs and a better understanding of hereditary diseases (Kari et al., 2007; Chakraborty et al., 2009). Currently, mainly 2d cell cultures are used in drug discovery. In contrast, cells in an embryo can be examined in their natural three-dimensional configuration, which could lead to new pharmaceutical discoveries (Kunz-Schughart et al., 2004).

Keller et al. (2008) introduced a pipeline consisting of a cell nuclei segmentation step via local adaptive thresholding followed by a tracking step employing a nearest neighbor search and applied the pipeline to early embryogenesis of zebrafish. In



(a) Segmentation of one volume in a time lapse sequence of a developing Drosophila embryo. More than 10% of the objects are noise instead of actual cell nuclei.



(b) Maximum intensity projection of a Drosophila embryo and tracking. Common ancestry is indicated by common color.

Figure 1.1.: Tracking of cell nuclei in a Drosophila embryo. Despite more than 10% noise objects in the segmentation (left panel) the proposed method is able to track the cell nuclei successfully (right panel).

Lou et al. (2011) we adopted the approach and improved both steps. Our proposed segmentation employed refined blob filter responses and a tracking based on the optimisation of a scoring function that can account for cell moves, divisions, and (dis-)appearances. Later, Lou and Hamprecht (2011) improved the segmentation with a regularized graph cut-based postprocessing. Kaster (2011, p. 152) compared the latter segmentation method with the voxel[1] classification-based approach of the freely available software ilastik (Sommer et al., 2011) and found the performance of both methods on par. However, due to its convenient graphical user interface ilastik is preferable to the regularized graph cut.

A ceiling analysis[2] reveals that the pipeline performance is mostly limited by false positive detections during the segmentation step caused by clutter objects originating from fluorescence markers outside the nuclei and phantom cells segmented out of background noise. The tracker cannot distinguish these misdetections from actual cells and the obtained cell lineages are distorted. In particular, tracking of divisions suffers because clutter objects near cells are frequently interpreted as the two descendant cells after a division in the previous time slice. This is especially unfortunate since a single wrong division invalidates the ancestry of all cells originating from that division.

[1] A voxel is a volumetric pixel.

[2] Ceiling analysis asserts that the overall performance of a data analysis pipeline is determined by the performance of each subcomponent. Evaluating each component with regard to its isolated improvement potential and impact on the overall pipeline performance, we can most effectively decide which component to improve next.

Therefore, to advance the state-of-the-art a method has to be developed that

- simultaneously tracks a *large* (several thousand objects), *unknown* and *variable* number of cells,
- allows for cell *division*,
- is highly accurate, because each tracking error affects a complete subtree of the lineage,
- is highly robust against false positive segmentations and clutter objects.

In response, the thesis at hand improves the state-of-the-art in tracking offline a variable and large number of divisible objects in presence of clutter in terms of performance, modeling flexibility, and accessibility to non-experts.

We will argue that instead of improving the quality of existing segmentation algorithms we improve the tracking method such that it can correct segmentation errors after the fact. In particular, our work has the following features:

- In recent years graphical models proved to be exceptionally successful in computer vision (Blake et al., 2011). We adopt the approach and present the first probabilistic graphical model for cell tracking that can track an unknown number of divisible objects that may appear or disappear and that can cope with false detections.
- The model achieves robustness against noise detections by solving the tracking problem for all time steps at once taking the expected time between divisions into account. The most likely cell lineage is obtained as the model configuration with maximum a-posteriori (MAP) probability using exact inference.
- While all literature with similar data focuses on single organisms, we present results on zebrafish and—for the first time—on Drosophila together with gold standard lineages, benchmarking measures and tracking results.
- The implementation of the method is the first freely available software for automatic 3d+t tracking of divisible objects with a graphical user interface.

Fig. 1.1 on the facing page illustrates the capabilities of the proposed method.

1.2. Related Work

1.2.1. Tracking

Tracking in science comprises many different application domains and approaches (Yilmaz et al., 2006). Our work is concerned with the tracking of a varying number of divisible objects with similar appearance in images with its main application in cell lineage reconstruction resp. cell tracking (Miura, 2005; Meijering et al., 2009). It has to be distinguished from tracking many similar particles that are not dividing (Smal et al., 2008).

When the temporal resolution of the raw data is high compared to the rate of change, derivative-based methods such as optical flow (Melani et al., 2007) or level sets (Padfield et al., 2009; Dzyubachyk et al., 2010) can be used to simultaneously segment and track the evolution of one or more targets in spacetime.

A lower temporal resolution—as it is the case in our application—makes the problem harder and in response, most algorithms separate the detection / segmentation from the tracking problem: objects are detected in each time slice and the detections are subsequently fed into a tracking routine. Two different kinds of models are typically used: state space models and assignment models. The Kalman filter, a linear Gaussian state space model, is the archetype of the former class, which interprets detections as caused by a hidden target (Kalman and Bucy, 1961; Yang et al., 2006). It has been generalized in a number of ways allowing for nonlinear motion models, discrete state spaces, non-Gaussian distributions (Arulampalam et al., 2002; Doucet and Johansen, 2011), multiple target hypotheses (Ong et al., 2010), or even an unknown (but fixed) number of targets (Fox et al., 2006). While state space models easily accommodate target properties such as velocity, size, and appearance and are robust against noisy detections by design, they are unfortunately hard to coax into dealing with a variable number of dividing objects.

Tracking-by-assignment, on the other hand, treats every detection as a potential target itself. It easily accommodates multiple or dividing objects; however, this increased flexibility must be reined in by enforcing the consistency of a tracking. For example, each target must have a unique ancestry and it may not divide into more than two parts, if prior knowledge so dictates. This difficulty has so far been addressed in three ways. Firstly, approximate methods can be applied that forego a consistency guarantee (Jiang et al., 2007). Secondly, tracks can be generated hierarchically from tracklets (Bise et al., 2011; Li et al., 2009; Brendel et al., 2011), akin to the use of superpixels in image processing. Thirdly, tracking is performed across pairs of frames only (Chen et al., 2006; Kachouie and Fieguth, 2007; Kanade et al., 2011; Padfield et al., 2011; Lou et al., 2011). Finally, other approaches combine several methods into sophisticated tracking pipelines to compensate the weaknesses of the individual methods (Li et al., 2008).

The approach by Bise et al. (2011) is most similar to ours since it is—to our best knowledge—the only other global model of detections and assignments over many

time slices. It is therefore a good candidate for a comparison with our method (see Sec. 5.3.3 on page 52).

1.2.2. Digital Embryo Recording

The field that tries to quantitatively reconstruct embryonic development at the cellular level using computational image analysis is fairly young. In fact, the term *digital embryo* was (most likely) coined as late as 2008 by Keller et al. (2008). The pioneers of the field come from the Waterston lab (Washington). They published the first digital cell lineages of *C. elegans* in 2006 (Bao et al., 2006). Their software tools StarryNite and AceTree (Murray et al., 2006) for the analysis of confocal microscopy images of nematode embryos are still in active use and were recently employed to study differences between normal and stressed *C. elegans* embryos quantitatively (Richards et al., 2013).

Huisken et al. (2004) introduced light-sheet microscopy to developmental biology and showed first *in vivo* recordings of *Drosophila* and Medaka embryos, even though the contrast was not yet high enough to clearly discern single cells. Keller et al. (2008) presented an improved version of the microscope employing a laser scanning technique together with the first digital zebrafish embryos at early development stages (see also Sec. 1.1 on page 3). Later, they upgraded the microscope with structured-illumination and published recordings of *Drosophila* embryos (Keller et al., 2010). Their expertise so far is summarized in Keller (2013).

In another line of work Olivier et al. (2010) applied label-free nonlinear microscopy to the reconstruction of early zebrafish embryogenesis. Besides a tracking of the cell nuclei they also reconstructed the spatial arrangement of the cell membranes in 3D. Their image processing pipeline to segment the cell membranes is described by Luengo-Oroz et al. (2012) and is based on a watershed segmentation. Furthermore, Mikula et al. (2011) published an advanced approach based on PDE methods to segment cell nuclei and membranes in similar zebrafish data. Later, the same microscopy technique was shown to work also on *C. elegans* embryos (Tserevelakis et al., 2011).

2. Digital Embryos Recording Pipeline

Light sheet microscopy paves the way to create *digital embryos* of organisms as complex as zebrafish or Drosophila. That is, a complete record of cell locations, shapes, tracks, and ancestry relations. Current microscopes can provide 3d images high in contrast at a steady temporal frequency of two to three images per minute. In the images one can observe cell nuclei and—in same cases—other structures like cell membranes. To make these artifacts visible they have to be stained with fluorescent makers allowing us to study the growth of embryos at the cellular level. Even fast movements that happen during cell migration and division can be observed, albeit not as a smooth motion and typically with no spatial overlap between the images of a nucleus in consecutive time slices.

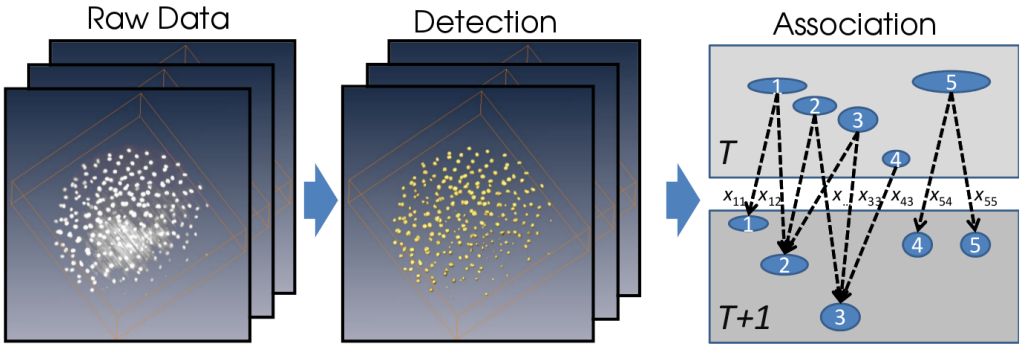


Figure 2.1.: Digital embryos recording pipeline. Data acquisition by 3d light sheet microscopy is followed by a detection-by-segmentation step. Finally, the cell lineage is established in a tracking step. [modified from Lou et al. (2011)]

In this chapter we present an approach to record digital embryos from microscopy images of stained cell nuclei. Given the above scenario a sensible design for a digital embryo recording pipeline is a two-step procedure where a segmentation of the cell nuclei for all time slices is followed by a tracking. Fig. 2.1 shows a schematic of the approach. For the segmentation step we employed a classifier to predict foreground vs. background for every voxel. Several options for the tracking step will be described in Chapters 3 and 4.

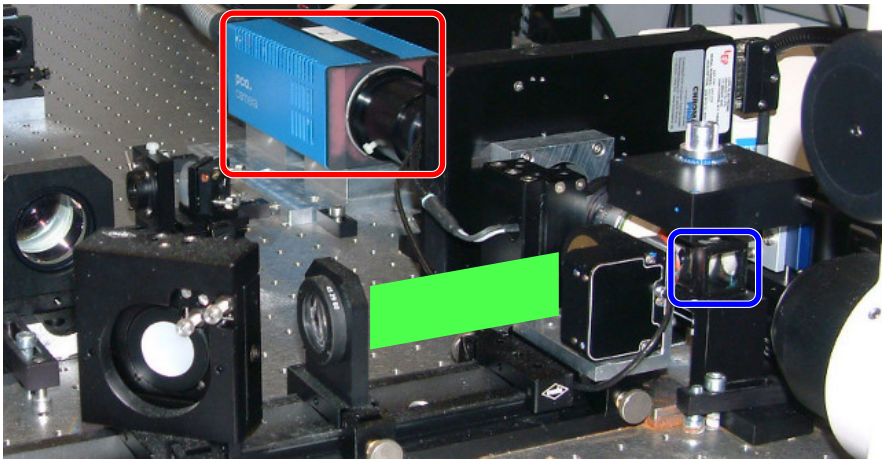


Figure 2.2.: Light sheet microscope (Wittbrodt lab, Univ. Heidelberg). The camera (*red*) points at the specimen container (*blue*). The light sheet enters the container perpendicular to the camera axis as indicated by the green plane.

2.1. Light Sheet Microscopy to Record Embryogenesis

The recently rediscovered light sheet microscope (Siedentopf and Zsigmondy, 1902; Voie et al., 1993; Huisken et al., 2004) is an emerging tool in current developmental biology (Tomer et al., 2011; Höckendorf et al., 2012). It allows the recording of three dimensional images at high resolution and speed by illuminating only a single plane in the specimen at any time. Compared to conventional widefield or confocal fluorescence microscopy, photobleaching of the fluorophores is reduced and specimens suffer less from phototoxicity.

2.1.1. Light Sheet Microscopes

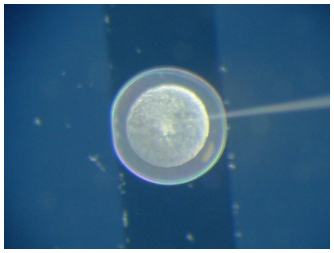
Fig. 2.2 shows the essential components of a light sheet microscope similar to the one described by Keller et al. (2008), employing a single camera and a single light sheet. In this model, the light sheet moves through the sample once typically every 30 to 90 seconds while the high-speed camera is recording around 400 images. One image has a size of four megapixel with a isotropic resolution of 0.3 µm per pixel. Resolution along the stack is roughly 1µm per slice. The light sheet in this model is generated by rapidly scanning the specimen with a thin laser beam horizontally and vertically. There are other approaches; all with the goal to produce a light sheet as thin and homogeneous as possible, with Bessel beam plane illumination being one of the most advanced (Planchon et al., 2011).

Images recorded with this type of microscope show a degradation of contrast along the light sheet penetration axis. This is caused by light absorption, scattering, and shadowing. Recently, Krzic et al. (2012) and Tomer et al. (2012) overcame this limitation by recording multiple views with multiple light sheets simultaneously.

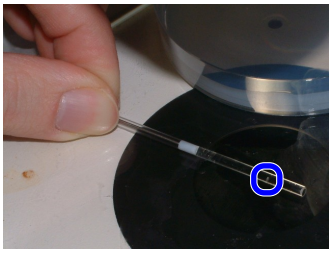
At the same time the first commercial microscope was released as a sealed box system making the technology available to a broader audience (Zeiss, 2012). The technology is developing rapidly and there is a good case to believe that fast, high-contrast light sheet microscopes will be a standard tool in many developmental biology labs in the near future.

2.1.2. Imaging Embryos

(a) Two zebrafishes in mating tank.



(b) Fluorescent staining of the zygote.



(c) Egg (*blue*) embedded in agarose-filled tube.

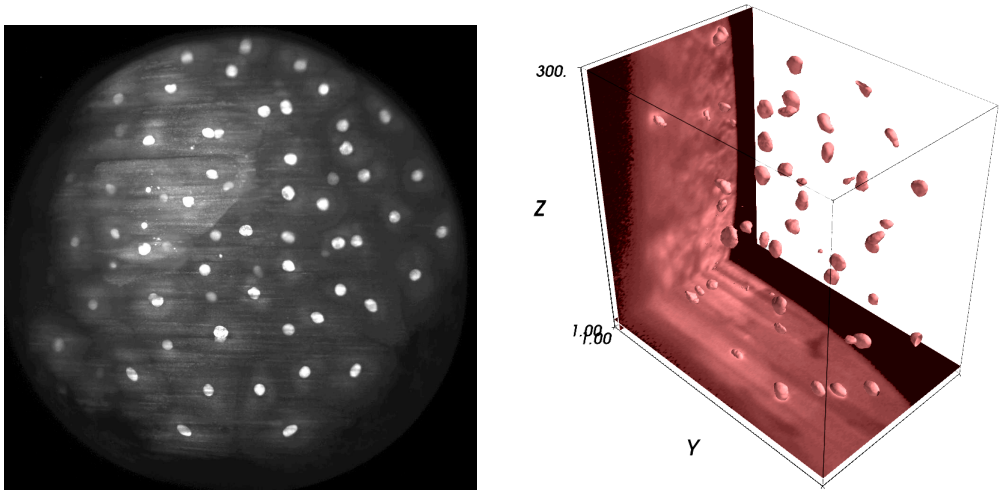
Figure 2.3.: Preparing zebrafish embryos for imaging.

To be mounted inside the light sheet microscope the developing embryos have to be embedded in an imaging cylinder filled with a semi-elastic gel. In general only embryos that stay constant in size for the recording time survive the embedding procedure. Furthermore, they should be at least semi-transparent and suitable to express fluorophores. For that reason fruit fly and fish embryos are popular organisms to be analyzed with light sheet microscopes (Keller et al., 2008; Truong et al., 2011; Tomer et al., 2012; Krzic et al., 2012). They are also the two model organisms for which we present tracking results in the thesis at hand. Other model organisms examined with light sheet microscopes in the context of developmental biology are the nematode *C. elegans* (Wu et al., 2011) and the plant *Arabidopsis thaliana* (Maizel et al., 2011).

The zebrafish (*Danio rerio*) is a popular aquarium fish and also one of the most studied model organisms in developmental biology (Detrich, III et al., 2009). Fig. 2.3 illustrates the steps necessary to prepare a zebrafish embryo for imaging in a light sheet microscope. First the zygote inside the fertilized eggs is stained with a fluorescence marker like green fluorescent protein (GFP) by injecting it with a thin needle (the gray line entering the egg from the right in Fig. 2.3b). Afterward the egg is embedded in a transparent cylinder filled with agarose—an organic gelatinous substance—to fix it in place. Subsequently, cylinder and egg are put in the microscope specimen container for imaging. The experimenter has to be fast in preparing the specimen otherwise the early stages of embryogenesis couldn't be recorded. A lot of manual skill and experience is necessary to create a dataset suitable for further analysis.

Fruit fly embryos are prepared in a similar manner. However, the manual staining is not necessary. Instead the experiments are typically conducted with transgenic lines, i.e. the cell nuclei are expressing fluorescent proteins on their own. On the one hand this simplifies the experiments, but on the other hand the fluorophores are also found in high concentrations outside the nuclei worsening the contrast of the microscope images.

2.2. Nuclei Segmentation



(a) Maximum intensity projection of zebrafish blastula at the 64 cell stadium.

(b) Ortho-surfaces illustrating the spatial distribution of the nuclei (cropped view).

Figure 2.4.: Zebrafish light sheet microscopy images. The two panels show images of a zebrafish blastula. Cell nuclei are visible as round spots together with other artifacts. Typically up to two 3d volumes are recorded per minute over a total time span of several hours.

After the successful imaging of a developing embryo the next step is a foreground–background segmentation of the fluorescent cell nuclei (Fig. 2.4). Kaster (2011) investigated two methods geared towards nuclei segmentation in light sheet microscopy data. The one method employs a shape-regularized graph cut scheme (Lou et al., 2012) and the other a voxel-wise classification using Random Forest—a non-linear state-of-the-art classification algorithm—on filter-based intensity, edge, and texture descriptors (Sommer et al., 2011). Both methods are on par regarding segmentation accuracy. The latter is implemented in terms of the open source "interactive learning and segmentation toolkit" (ilastik)[1] and is easily accessible for the end user thanks to a polished graphical user interface. Furthermore, the classifier can be trained in an interactive manner with a direct quality feedback at

[1] http://ilastik.org

any time. For that reasons we choose ilastik as our preferred method for nuclei segmentation (see also Sec. 5.3.1 on page 48 for a quantitative evaluation).

The software takes sparse foreground / background labels and a selection of voxel features as input. Kaster (2011) investigated several subsets of (rotationally invariant) features and published a table of the best performing ones (Kaster, 2011, pg. 144). In general, the best performance can be achieved with a as-large-as-possible set of informative features; the only limitation being an insufficient amount of computer main memory. Furthermore, Random Forests are quite robust against uninformative features when provided with enough training data (Breiman, 2001). As a consequence we use as many features as possible in the order given by the table, only limited by the available computing power.

2.3. Summary

In this chapter we described a two-step pipeline consisting of cell identification and tracking to reconstruct cell lineages (in general) and record digital embryos (in particular) from 3d+t microscopy images. We gave a quick overview of light sheet microscopes as sources of the spatio-temporal images and described the steps necessary to prepare embryos for imaging. Finally, we introduced ilastik pixel classification as a viable method to segment and identify cell nuclei in the obtained images; being the first step of the proposed pipeline. A description of the second step, tracking, is postponed to the following chapters.

3. The Tracking-by-Assignment Problem

In Chapter 2 we introduced a pipeline to record digital embryos or—more general—reconstruct cell tracks and lineages from time-lapse microscope images in two steps: a segmentation step for cell identification followed by a tracking step. There, we postponed a more detailed description of the tracking step to this chapter, which will serve as the foundation for the main contribution of the thesis: stating the tracking-by-assignment problem that we will later solve with a graphical model approach (see Chap. 4).

We face a scenario where we want to track a large and typically unknown number of objects (up to several thousands). Additionally these objects may divide and (dis-)appear in any time slice and some objects may be false positive identifications due to clutter or noise in the data. By assigning objects in consecutive time slices to each other, we can easily establish tracks. Objects that are false positives are labelled as such and not assign to any other object. Disappearance and appearance can be expressed by no outgoing or incoming assignments to the future or from the past. Divisions can be handled gracefully by assigning one ancestor object to two descendant objects. The task of finding these assignments given a set of object candidates in slices at regular intervals is the *tracking-by-assignment problem*. We will formulate the problem in terms of a graph labelling and present a baseline procedure to solve the problem in the context of cell nuclei tracking.

3.1. Hypotheses Graph

We want to state the tracking-by-assignment problem formally in terms of an abstract *hypotheses graph*. The solution of the problem is then expressed as a labelling of the graph. Procedures to obtain the labelling are called *reasoners*.

Formulating the tracking problem in terms of a graph and the clear separation of modelling and solving the problem has several advantages. The procedures concerned with constructing the hypothesis graph are independent from the reasoners. That is, we can offer several options for both parts, each with its own advantages and disadvantages and can tailor our design to a given application by reusing independent components. Furthermore, it allows a fair performance evaluation of different options for one part by keeping the rest fixed. Finally, there are several minor advantages like the ability to serialize the graph (when implemented in software) and to connect to the vast literature on graph algorithms. Note, that this idea is very similar to the separation of learning, inference, and modelling in probabilistic graphical models (one of the cornerstones of their success).

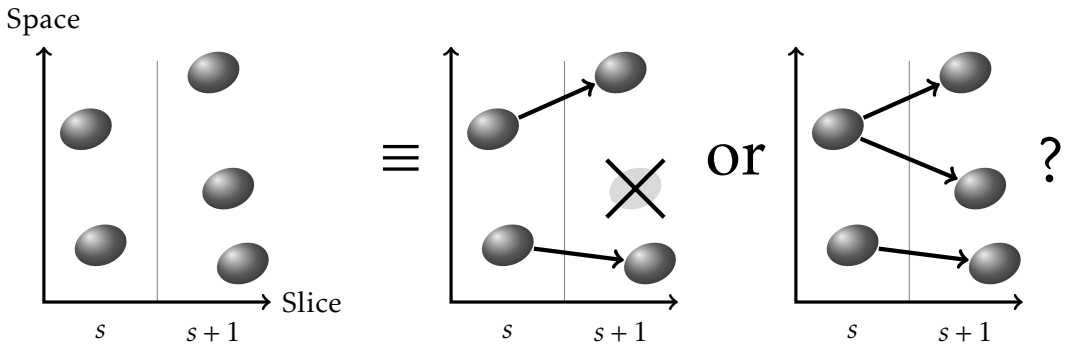


Figure 3.1.: Ambiguous object assignments. Given the object candidates shown in the left-most panel there are several hypotheses how the transition from slice s to $s+1$ happened. In the middle panel we explain the situation in terms of two one-to-one assignments and mark one object as clutter. The right panel in contrast offers an object split and a one-to-one assignment as an interpretation. There are many more possible explanations including trivial ones like everything marked as clutter or instant appearance and disappearance of all objects.

3.1.1. Definition

Before we give a formal definition of the hypotheses graph let us start with the intuition behind it. Fig. 3.1 shows some object candidates for two consecutive slices and two out of many possible hypotheses how the transition from one slice to the next happened. The term *object candidate* subsumes both the actual objects to track and undesired objects like clutter or false positive segmentations. For brevity we will use *object candidate* and *object* interchangeably where it is not ambiguous. The hypotheses are given as assignments between two or more objects in different slices (when representing events like moves or divisions) or as markings of the objects themselves (for instance, in case of disappearing objects or false positive object candidates).

The task of a reasoner is to select the correct or most likely hypotheses given information like the distance between objects or shape features. To avoid enumerating all possible hypotheses—there are exponentially many in the number of objects–we build-up a *hypotheses graph*. For every object we add a node to the graph and introduce edges between nodes when we believe that such an assignment could be possible. For example one could link objects in a certain spatio-temporal neighbourhood to each other or when they have a similar shape. Instead of selecting from a set of hypotheses a reasoner can now label nodes and edges as active or inactive.

A hypotheses graph covering seven object candidates at three slices is shown in Fig. 3.2 on the next page. Note that not every two nodes are connected in consecutive slices, stating that assignments between the two objects are considered

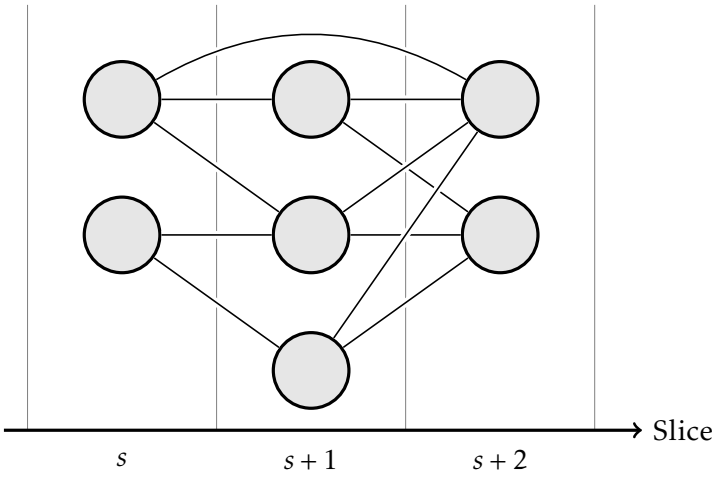


Figure 3.2.: A hypotheses graph spanning three slices.

impossible. Furthermore, nodes in slices further apart may also be connected allowing for missing detections. (Alternatively a new node representing a missing detection candidate can be added to the graph.)

We now want to give a formal definition of the hypotheses graph. A prerequisite is the notion of a total order:

Definition. A *total order* is a binary relation over a set P, i.e. for all elements a, b, and c in P the following statements are true:

Antisymmetry: $a \leq b \wedge b \leq a \Rightarrow a = b$

Transitivity: $a \leq b \wedge b \leq c \Rightarrow a \leq c$

Totality: $a \leq b \vee b \leq a$
(i.e., you can compare any two elements of the set P)

Based on that, a hypotheses graph is defined as follows.

Definition. A *hypotheses graph* is an ordered, undirected graph

$$G = (\mathcal{N}, \mathcal{E}, s\colon \mathcal{N} \to \mathbb{Z}, \lesssim)$$

where $\mathcal{N}$ is a set of nodes, $\mathcal{E}$ a set of edges. The function s a node labelling of the graph assigning a slice number to every node n. The graph is totally ordered under $\lesssim$ ("less or similar"). The order is induced by the slice labels:

$$n \lesssim n' \Leftrightarrow s(n) \leq s(n')$$

3.1.2. Tracking-by-Assignment as Hypotheses Graph Labelling

Tracking-by-assignment can be restated as labelling all the edges and nodes of a hypotheses graph as either "active" or "inactive" according to the following rules of consistency. Active nodes mark the actual objects to track. The inactive nodes are other objects like clutter or false positive segmentations. Object transitions like moves, divisions, or mergers are represented by one or more active edges coming from previous ("incoming edge") or following ("outgoing edge") slices. Appearing (disappearing) objects have no incoming (outgoing) active edges. There are no active edges attached to inactive nodes.

It is the task of the reasoner to ensure the consistency of the labelling. Furthermore some reasoners are less capable than others. For instance, a reasoner that cannot track dividing objects would instead initialize new tracks for the descending objects. It would never label more than one outgoing edge as active. This is not an issue as long as the labelling is consistent.

Of course, reasoners take more than a hypotheses graph as input. Typical additional information is the geometric configuration of the objects in space and features describing their shapes. A good reasoner produces the most likely of all consistent labellings given these additional inputs.

In software the graph labelling and any additional information can be represented as map data structures defined over the hypotheses graph data structure. This allows to attach or remove annotations without changing the underlying graph data structure and helps to reduce software defects caused by data mutability issues.

3.2. Events

Events are a more detailed way to label a hypotheses graph resolving some of the ambiguities that can arise from the simple "active" vs. "inactive" labelling described in Sec. 3.1.2. In cell tracking we typically encounter event types such as "move", "division", "appearance", "disappearance", "true positive detection", and "false positive detection". Formally they are described in terms of subsets of hypotheses graph nodes. For instance, division events are the three-set of nodes representing the ancestor and the two descendant objects. Events from different event classes can share the same set of nodes such as "appearance" and "true positive detection".

Some labellings of a hypotheses graph can be ambiguous. For example, both a (cell) division and a split due to oversegmentation is indicated by two active outgoing edges. A reasoner can give the event as an additional output to resolve the situation. Furthermore, events can be used to specify mutually exclusive assignments. A node marked as inactive corresponds to the "false positive detection" event. Since we typically don't want to allow assignments between inactive nodes this event is incompatible with all the assignment events like move, division etc.

Event	Notation	Cost
i *moves* to j	$i \to j$	$d^2_{i\to j}$
i *divides* into j and k	$i \to j+k$	$d^2_{i\to j} + d^2_{i\to k} + c_{\text{div}}$
i *disappears* in slice s	$i \to \emptyset$	c_{dis}
j *appears* in slice $s+1$	$\emptyset \to j$	c_{app}

Table 3.1.: Summary of the events considered by the reasoner. Here $d_{i\to j}$ is the Euclidean distance between the centers of mass of nucleus i (from slice s) and j (from slice $s+1$).

Such relations can be encoded with a hypotheses graph and a complementary set of events.

3.3. A Baseline Reasoner for Cell Tracking: Optimal Joint Assignment[1]

We present an example for a basic cell nuclei tracking reasoner as a baseline for solving the tracking-by-assignment problem. It cannot mark nodes in the hypotheses graph as inactive i.e. distinguish true cells from false positive objects but will mark all nodes as active by default and should therefore only be applied when the true positive rate of object candidates is very high. However, it is capable enough to track dividing objects. Furthermore, since it only considers two consecutive slices at once it cannot handle links that span several slices. Then again it can be easily parallelized by tracking many pairs of slices simultaneously.

It finds the optimal joint assignment between nuclei for every pair of subsequent time slices by minimizing the total sum of squared nuclei distances and fixed event costs. Formally, let i denote a nucleus from slice s and j,k denote nuclei from slice $s+1$, and let $\emptyset$ represent *no assignment*. We consider the following events: *move*, *division*, *disappearance* and *appearance*. While an (apparent) cell disappearance may be caused by cell death, it is typically caused by the cell leaving the field of view or a misdetection in the segmentation step. An (apparent) appearance happens when it is segmented again at a later time slice or reenters the field of view. Note that we initialize a new track for a reentering nucleus. All ancestry information is lost. In order to make the optimization problem tractable, we only consider at most k nearest neighbors of i within a given distance threshold above the maximal observed moving distance of the nuclei. That is there are at most k outgoing edges at any node in the hypotheses graph.[2]

As shown in Table 3.1 all these events have associated costs and the constants c_{div}, c_{dis} and c_{app} are chosen such that the cost of appearance and disappearance events are always higher than the costs of all allowed division and move events.

[1] Content of section is based on the publication by Lou et al. (2011).

[2] Setting $k = 6$ works well in practice.

Therefore cells that are assumed to (dis)appear cannot be accounted for by any other event. We choose the movement costs to be zero for no translation and quadratically increasing with the travelled distance. This assumed that objects stay more or less at the same location in consecutive time slices. This is true when the sampling frequency of the slices is high relative to the average movement speed of the objects.[3]

Let $\mathcal{M}$ be the set of all possible moves and $\mathcal{D}$ the set of all possible divisions. For each event in $\mathcal{M}$ and $\mathcal{D}$, we define a binary variable x indicating whether this event takes place or not. Finding the optimum joint association is then an integer linear programming (ILP) problem:

$$\min_{\boldsymbol{x}} \sum_{(i \to j) \in \mathcal{M}} x_{i \to j}(c_{i \to j} - c_{i \to \emptyset} - c_{\emptyset \to j}) + \sum_{(i \to j+k) \in \mathcal{D}} x_{i \to j+k}(c_{i \to j+k} - c_{i \to \emptyset} - c_{\emptyset \to j} - c_{\emptyset \to k}) \quad (3.1)$$

subject to

$$\begin{aligned} \sum_{j:(i \to j) \in \mathcal{M}} x_{i \to j} + \sum_{j,k:(i \to j+k) \in \mathcal{D}} x_{i \to j+k} \leq 1 \quad \forall\, i, \\ \sum_{i:(i \to j) \in \mathcal{M}} x_{i \to j} + \sum_{i,k:(i \to j+k) \in \mathcal{D}} x_{i \to j+k} \leq 1 \quad \forall\, j. \end{aligned} \quad (3.2)$$

Here, the two constraints guarantee that one nucleus can either divide into two cells or perform a move when not disappearing.

All cells not accounted for by either a division or a move are assumed to appear or disappear. Typically there are a few ten thousand variables (one for each division or move) and a few thousand constraints (proportional to the number of nuclei in each frame). We use a state-of-the-art ILP solver (ILOG CPLEX [4]) to solve this problem to global optimality within less than a minute per frame pair on a standard desktop machine.

3.4. Summary

In this chapter we formulated tracking-by-assignment as a graph labelling problem and presented a baseline reasoner to solve the problem in the context of cell tracking. Representing the problem in terms of a data structure produces a flexible modelling framework for tracking algorithms. The baseline reasoner calculates a optimal joint assignment between consecutive time slices by solving an integer linear program. It cannot distinguish between true positive and false positive objects and needs a very high object identification accuracy to be applied successfully. We will present a more advanced reasoner in form of a graphical model in Chapter 4.

[3] As a rule of thumb, most objects should have a spatial overlap with themselves in consecutive slices.

[4] http://www.ilog.com/products/cplex/

4. Tracking-by-Assignment as a Graphical Model[1]

In the introduction of the thesis (pg. 5) we listed the requirements for a cell tracking method to improve the state-of-the-art for automatic cell lineage reconstruction. In particular we assert that it is necessary to handle a high number of false positive objects—that is $\approx 10\%$ of all identified objects.

In chapter 3 on page 15 we formulated tracking-by-assignment as a hypotheses graph labelling problem and called the solving procedures "reasoners". In this chapter we present a reasoner formulated as a probabilistic graphical model that meets all the demanded requirements and introduces the possibility to mark object candidates as false positives (equivalent to labelling nodes as inactive in a hypotheses graph) to address a high false positive rate.

The model takes the form of a *chain graph* (Frydenberg, 1990)—a directed graphical model (Bayesian network) of "supernodes", each of which consists of a conditional random field over a set of "subnodes". A chain graph model turns out to be necessary because neither Bayesian networks nor Markov random fields alone are compatible with the independence assumptions imposed by tracking.

4.1. Background

In this section we will summarize the most important methods necessary to understand the chain graph tracking model. That is on the one hand the theory of graphical models including Bayesian networks, Markov random fields, and the chain graph model as a hybrid of the two. On the other hand we have the Random Forest discriminative classifier which is used to parametrize the detection factor in the chain graph tracking model.

We will only mark the most important concepts and kindly refer the reader to the cited literature for more details. For graphical models in particular we highly recommend the book by Daphne Koller and Nir Friedman (Koller and Friedman, 2009). It is both a gentle introduction and comprehensive guide to almost all topics concerning graphical models.

4.1.1. Discriminative Classifier: Random Forest

Random Forest was introduced to the machine learning community by Breiman (2001). As a discriminative classifier Random Forest is part of a broad family of

[1] Content of chapter is based on the publication by Kausler et al. (2012).

ensemble-based methods (Rokach, 2010), which aggregate the predictions of many, typically weak classifiers of the same type to achieve an overall better performance than can be achieved by each single classifier alone. In case of Random Forest, decision trees play the role of the weak learners and the overall class predictions are given in form of relative frequencies of the single tree predictions.

To improve prediction accuracy a *bagging* technique is employed (Breiman, 1996). That is, each tree is only trained on a bootstrapped subset of the training samples, drawn uniformly and randomly with replacement from all samples. Furthermore, each tree sees only a random subset of all features at each split node. The trees are typically trained to purity (each leaf associated with samples of only the same class) without pruning. As a splitting criterion we have chosen the popular Gini impurity measure (Breiman et al., 1984; Ceriani and Verme, 2012). Random Forest only has two tunable parameters: the number of drawn features and the number of trees in the forest. As empirical studies[2] have shown, its predictive performance is mostly independent of the exact parametrization and is on par with other state-of-the-art non-linear classifiers like SVM or neural networks. Furthermore, it is robust against noisy, mostly uninformative features.

We employ Random Forest to learn prior factors in the chain graph tracking model (see Sec. 4.5.2 on page 37). Random Forest is well suited for this task. The relative frequencies of the tree votes can be directly interpreted as normalized probabilities (other than—for instance—the binary output of the standard SVM) and it is fast in training and prediction without the need to worry about cross validation schemes (other than—for example—neural networks).

4.1.2. Graphical Models

A graphical model links two aspects of probability theory with the help of a graph data structure: conditional independence statements about a set of possibly correlated random variables and the representation of multivariate distribution over these variables as a product of factors. Given two random variables A and B they are said to be independent ($A \perp\!\!\!\perp B$) if

$$A \perp\!\!\!\perp B : \quad P(A,B) = P(A) \cdot P(B) \tag{4.1}$$

or (after introducing a third variable C) conditionally independent ($A \perp\!\!\!\perp B \mid C$) if

$$A \perp\!\!\!\perp B \mid C : \quad P(A,B|C) = P(A|C) \cdot P(B|C). \tag{4.2}$$

One possible factorization of a distribution $P(A,B,C)$ could be obtained using Bayes' theorem

$$P(A,B,C) = P(A|C) \cdot P(B|C) \cdot P(C) \tag{4.3}$$

[2] Guo et al. (2004); Caruana and Niculescu-Mizil (2006); Díaz-Uriarte and De Andres (2006); Menze et al. (2009)

or the same distribution could be represented as a product of nonnegative potentials ϕ_i with a normalizing partition function Z

$$P(A,B,C) = \frac{1}{Z}\phi_1(A,C)\cdot\phi_2(B,C)\cdot\phi_3(A)\cdot\phi_4(B). \tag{4.4}$$

Graphical models have many advantages compared to merely probabilistic, energy-based (LeCun et al., 2006), or combinatorial optimization-based (Papadimitriou and Steiglitz, 1998) approaches (and others). Graphical models can pose as any of the three approaches by only concentrating on certain aspects. For example, MAP inference can be seen as a combinatorial optimization problem and Markov random fields can be converted to energy-based models by translating them to log-space. In fact, one advantage of graphical models is their ability to disguise as many different approaches allowing us to use the best methods from many fields with the same underlying model (for example, sampling techniques borrowed from Bayesian statistics or inference algorithms based on convex and combinatorial optimization techniques). Furthermore, graphical models can be bootstrapped from first principles by identifying random variables to describe real world problems and clearly stating conditional independence assumptions. From that a factorization with the necessary parameters can be derived, alleviating the issue of over- or underparametrization.

Another strong point is the graph data structure associated with every graphical model. It allows to investigate independence relationships in a much more intuitive and effective manner compared to purely probabilistic models where independence has to be checked using involved arithmetic. The lower hurdle encourages the practitioner to actually investigate the independence relationships in his or her model and helps to find possibly hidden and unwanted model assumptions. The same structure also encodes factorization information without the need to spell it out in formulas. To put it in another way, graphical models help humans to structure a problem and to reason about it using intuitively accessible graphical elements instead of opaque formulas and allow to concentrate more on the problem domain itself instead of the modeling tools.

Representation

An instance of a graphical model is given as a graph $G = (N, E)$ comprising a set of nodes N and set of edges E. It can be directed, undirected, or both depending on the particular type of graphical model. Below we give descriptions of three types: *Bayesian networks* (directed), *Markov random fields* (undirected), and *chain graphs* (mixed). There is one and only one node for each random variable. Edges between nodes represent a direct influence between the two random variables. That is, if the two variables are probabilistic dependent, no conditioning on other variables in the model can make them independent. This is not necessarily true the other way around. Two variables connected by an edge can still be probabilistic independent as induced by a certain parametrization of the distribution.

(Conditional) independence relations between variables that are not directly connected can be read-out from the graph by examining connecting paths between

the variables (the rules depend on the type of graphical model). Furthermore, the factorization of probability distributions over the random variables has to be compatible with the connectivity of variables / nodes in the scope of each factor (the rules again depending on the type of graphical model).

In general there can be many factorizations of a distribution that are compatible with a single graph. Kschischang et al. (2001) and Frey (2003) introduced and refined *factor graphs* to resolve this ambiguity. Factor graphs are undirected bipartite graphs consisting of factor and variable nodes. Each factor node is connected to the variables in the scope of the factor. Fig. 4.1 shows an example factor graph with four factor nodes f_i and three random variables X_i. The factor f_2 is connected with X_1 and X_2 indicating the factor scope $f_2(X_1, X_2)$. The complete factorization encoded by this factor graph is then $\tilde{P}(X_1, X_2, X_3) = f_1(X_1) \cdot f_2(X_1, X_2) \cdot f_3(X_1, X_2) \cdot f_4(X_2, X_3)$. In general $\tilde{P}(X_1, X_2, X_3)$ needs to be normalized to obtain a proper distribution $P(X_1, X_2, X_3)$.

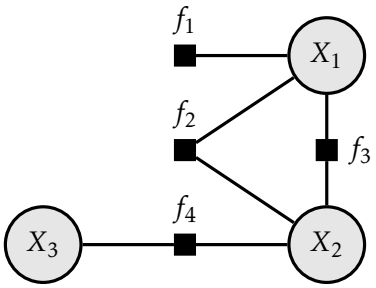


Figure 4.1.: Factor graph example.

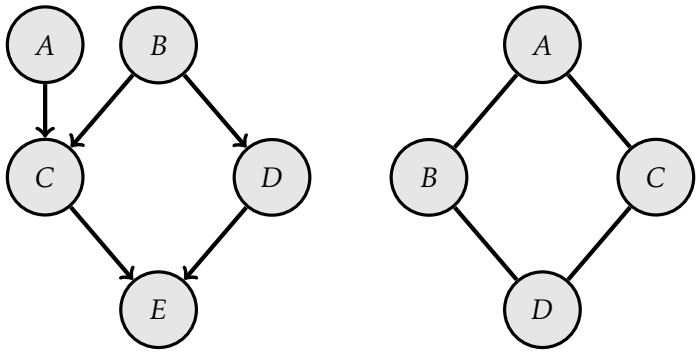
Inference

The primary application of multivariate probability distributions modeled as a graphical model are inference queries. They come in three flavors: calculating marginal distributions $P(\mathcal{X} \backslash \tilde{\mathcal{X}})$, conditional distributions $P(\mathcal{X} \backslash \tilde{\mathcal{X}} | \tilde{\mathcal{X}})$, and mode queries—typically the maximum a-posteriori (MAP) or most likely configuration $x' = \operatorname{argmax} P(\mathcal{X})$. All three can be solved in principle using the *variable elimination* algorithm, which combines basic arithmetic manipulations applying Bayes' theorem and dynamic programming. The algorithm was invented many times in many different fields dating back as early as 1880 (Thiele, 1880). In the context of graphical models (in particular, Bayesian networks) it was introduced by Zhang and Poole (1994). The gist of the method is already present in a simple calculation of a conditional. Given a distribution over two binary variables $P(A, B)$ we can obtain the conditional distribution of A as follows:

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(A,B)}{P(A_0,B) + P(A_1,B)}. \tag{4.5}$$

In case of more variables, operations like these are applied recursively on subsets of variables while reusing already calculated intermediate results. That way, a combinatorial blow-up is prevented as it happens when calculating the queries in a single step.

Unfortunately, the inference problem is still $\mathcal{NP}$-hard in general in the number of random variables (Bertele and Brioschi, 1972; Dechter, 1999). There are special cases with polynomial runtime, though. For tree-structured graphical models



(a) Bayesian network. (b) Markov Random Field.

Figure 4.2.: Examples of directed and undirected graphical models.

all three kinds of queries can be calculated efficiently (Shafer and Shenoy, 1990; Shenoy and Shafer, 2008). If the distribution is a member of the Gibbs family ($P(\mathcal{X})\frac{1}{Z}e^{-E(\mathcal{X})}$) and the energy function E is submodular, MAP inference is possible in polynomial time (Kolmogorov and Zabih, 2002). As we show in the work at hand, MAP inference can nevertheless be conducted in some cases in reasonable time on real world problem sizes for the general case (even though it will scale exponentially for even larger problems).

Furthermore, there are many approximate methods that can be roughly categorized into sampling approaches (Pearl, 1987; Gamerman and Lopes, 2006) and variational approaches (Yedidia et al., 2005; Wainwright and Jordan, 2007). Another important approach poses MAP inference in terms of an integer linear program allowing us to use exact solvers (Schrijver, 1998) or linear programming relaxations (Wainwright et al., 2005).

4.1.3. Bayesian Networks

A Bayesian network is represented by a directed graph. The factorization and independence rules over the graph are governed by Bayes' rule

$$P(A, B) = P(A|B) \cdot P(B) \tag{4.6}$$

(for two random variables A and B). In particular, there exist only two types of factors: prior factors (like $P(B)$) and conditional factors (like $P(A|B)$) depending on a single variable each. Both types are properly normalized and there is exactly one factor associated with each variable node in the graph. If the node has incoming edges the factor is conditioned on the sources nodes / variables of the edges, otherwise it is an unconditioned prior factor. Naturally, this construction induces a directed acyclic graph.

An example Bayesian network with five random variables is shown in Fig. 4.2a.

The corresponding factorization is

$$P(A,B,C,D,E) = P(A)P(B)P(C|A,B)P(D|B)P(E|C,D) \tag{4.7}$$

as can be easily read from the graph. Furthermore, conditional independence statements can be derived from the graph, too. It is—for instance—intuitively clear that $A \perp\!\!\!\perp \{B,D,E\} \mid C$. Formally, two variables are conditional independent in a Bayesian network if they are d-separated, that is, if they are not connected by any non-blocking trail.[3] A trail is blocked if one ore more of the following statements are true:

1. The trail contains a segment $B \leftarrow M \leftarrow E$, where M is observed.
2. The trail contains a segment $B \leftarrow M \rightarrow E$, where M is observed.
3. The trail contains a segment $B \rightarrow M \leftarrow E$ and neither M nor any of its descendants is observed.

In our example graph, C and D are thus not independent since they are connected via B. If we observe B, this trail would become blocked and the two variables were d-separated (and thus independent) since the only other trail via E is blocked, too. If we would subsequently observe E, the latter trail unblocks and C and D become dependent again.

4.1.4. Markov Random Fields

Other than Bayesian networks Markov Random Fields (MRFs) are represented by undirected graphs. A factorization of a distribution is compatible with a Markov Random Field if each factor is defined over a single clique in the graph. Before we elaborate these concepts we have to define some notions first. A clique is a fully connected subgraph and a maximal clique is a clique where no node could be added without breaking the clique property. The set of variables a factor takes as arguments is called the *scope* of the factor. A factor over a clique then is a factor whose scope is identical with the clique variables and—as already stated—no factor's scope is allowed to have variables from more than one clique (otherwise the factorization wouldn't be compatible with the MRF anymore). In particular, the largest scopes of any factorization are limited by the maximal cliques in the graph and the coarsest factorization that is still compatible with the MRF is

$$P(\mathcal{X}) = \frac{1}{Z} \prod_{C \in \{\text{max. cliques}\}} \phi_C(\mathcal{X}_C) \tag{4.8}$$

where $\mathcal{X}_C$ are the variables in clique C and the constant Z is called the *partition function*. The partition function normalizes the product of semipositive, unbounded factors. The factors ϕ are called the *potentials*. Note, that there are usually several

[3] A trail—other than a path—ignores the direction of edges.

factorizations that are compatible with a given MRF. In that case a factor graph (see above) can be used to exactly specify the factorization.

If a factorization of a distribution is compatible with a MRF we can read conditional independence relations from the graph that can be fulfilled by the factorization. Fig. 4.2b on page 25 shows an example MRF with the following maximal clique factorization:

$$P(A,B,C,D) = \frac{1}{Z}\phi_1(A,B)\,\phi_2(A,C)\,\phi_3(B,D)\,\phi_4(C,D) \tag{4.9}$$

Two variables in a MRF are conditionally independent if all connecting paths are blocked. A path is blocked if it contains an observed variable. The two variables are then said to be *separated*. For instance, $B \perp\!\!\!\perp C \mid A,D$ and $A \perp\!\!\!\perp D \mid B,C$ in the example graph since in both cases the only two paths connecting the variables are blocked. Note, that this rule is simpler than the three rules for d-separation in Bayes networks.

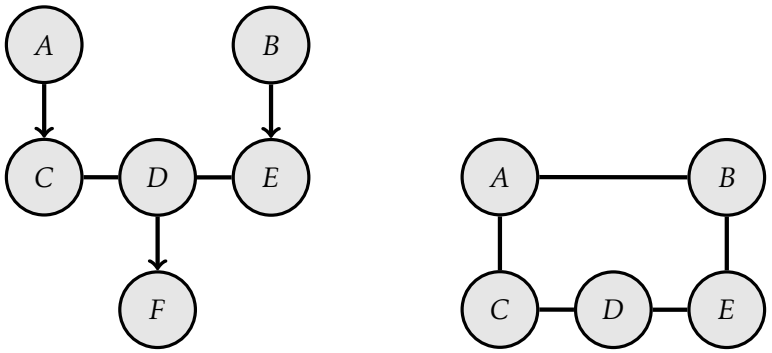
4.1.5. Chain Graph Models

Bayesian networks and Markov Random Fields can both express independence relations—sometimes called *Markov properties* of the graphical model—that the respective other cannot. In case of Bayesian networks the "V"-pattern constellation $B \rightarrow M \leftarrow E$ induces $B \perp\!\!\!\perp E$ when M is not observed and nullifies the independence when M is observed. Such a mechanism does not exist in MRFs since there one cannot remove independence relations by the act of observing variables. An analog example exists the other way around. A diamond-pattern MRF like the one shown in Fig. 4.2b on page 25 can expresses the conditional independence relations $B \perp\!\!\!\perp C \mid A,D$ and $A \perp\!\!\!\perp D \mid B,C$. There is no Bayesian network over four random variables that could express both of these relations (which can be proven by exhaustive construction of all possible four-node directed acyclic graphs).

A *chain graph model* is a mixed graphical model with both directed and undirected edges that allows to express independence relations exclusive to directed (Bayesian network) or undirected (MRF) graphs in the same model (Frydenberg, 1990; Studeny and Bouckaert, 1998). To understand chain graphical models we first have to introduce two other concepts: conditional random fields (CRFs) and partially directed acyclic graphs.

CRFs model a distribution over a set of target variables $\mathcal{Y}$ given another set of feature variables $\mathcal{X}$ where independence relations involving target variables are determined by MRF Markov properties (feature dependencies are not modeled since they are thought of being observed):

$$\begin{aligned} P(\mathcal{Y}|\mathcal{X}) &= \frac{1}{Z(\mathcal{X})}\tilde{P}(\mathcal{X},\mathcal{Y}) \\ \tilde{P}(\mathcal{X},\mathcal{Y}) &= \prod_i \phi_i(\mathcal{D}) \end{aligned} \tag{4.10}$$



(a) A chain graph $\mathcal{G}$. (b) The graph $\mathcal{M}[\mathcal{G}^+[C, D, E]]$.

Figure 4.3.: Example for a chain graph. The chain graph in the left panel consists of four components $\mathcal{K}_1 = \{A\}$, $\mathcal{K}_2 = \{B\}$, $\mathcal{K}_3 = \{C, D, E\}$, and $\mathcal{K}_4 = \{F\}$. The moralized subgraph of the chain graph in the right panel is used to reason about the dependence of variable C and D in the context of c-separation.

where $\mathcal{D} \subseteq \{\mathcal{X}, \mathcal{Y}\}$ and $\mathcal{D} \not\subseteq \mathcal{X}$, that is, there are no factors with scopes only spanning feature but no target variables. Note that the partition function is not a constant but is depending on the feature variables. A CRF is represented by a partially directed graph with an undirected subgraph over the target variables and parent–child relationships between feature and target variables indicated by directed edges. Sutton and McCallum (2012) give an comprehensive introduction to CRFs.

A partially directed acyclic graph (PDAG) is a directed acyclic graph (DAG) that can also contain undirected edges. Because there are no directed cycles such a graph consists of one or more distinct components. The nodes in a single component are connected with each other only through undirected edges and are connected to nodes from other components only with directed edges. This macro structure resembles a chain of beads. A PDAG is therefore also called a *chain graph* over *chain components* and a distribution that factorizes over such a graph is a chain graph model. Fig. 4.3a shows an example for a chain graph with four components and six variables.

The chain components in a chain graph model are laid out as a DAG and we can interpret this DAG as a Bayesian network whose nodes are the chain components. In contrast to a common Bayesian network the nodes themselves expose a finer structure in form of undirected subgraphs. In fact, a chain graph model generalizes both Bayesian networks and MRFs. It reduces to a common Bayesian network in the case of only one node per chain component and to a MRF in case of only one chain component with no parents. Let t be the number of chain components, $\mathcal{K}_i$ the set of all variables in component i of t, and $\mathrm{Pa}_{\mathcal{K}_i}$ the set of parent nodes connected with nodes in component i by directed edges which are pointing towards the component. Then—in general—the factorization of a distribution over a chain graph takes the

form of a product of CRFs:

$$P(\mathcal{X}) = \prod_{i=1}^{t} P_i(\mathcal{K}_i|\mathrm{Pa}_{\mathcal{K}_i}) \tag{4.11}$$

The Markov properties of chain graph models subsume the properties of MRFs and Bayesian networks and are consequently more complex than each of the latter two. They are defined in terms of *c-separation* and two variables in a chain graph are conditionally independent given a third set of variables if the two variables are c-separated (Madigan et al., 1995). To understand c-separation we need to introduce the concept of *moralization* first. Moralization converts a directed into an undirected graph in two steps. First, all parent nodes of the same child node, that are not already connected, are connected with an edge. (They are "married" to each other, removing their "immoral" relationship.) Second, all edges are converted to undirected edges. The moralized version of a graph $\mathcal{G}$ is denoted by $\mathcal{M}[\mathcal{G}]$. The graph $\mathcal{G}^+[\mathbf{U}]$ is the induced subgraph over the nodes $\mathbf{U}$ together with all ancestors of these nodes based on a chain graph model. (In an undirected graph all other nodes are ancestors). Given two variables A and B and a set of variables $\mathbf{Z}$ ($A, B \notin \mathbf{Z}$), A is said to be c-separated from B if they are separated (in the MRF sense) in the graph $\mathcal{M}[\mathcal{G}^+[A \cup B \cup \mathbf{Z}]]$. To summarize, the statement $A \perp\!\!\!\perp B \mid \mathbf{Z}$ is true if A is c-separated from B given $\mathbf{Z}$.

For example, Fig. 4.3b on the facing page shows the graph $\mathcal{M}[\mathcal{G}^+[C, D, E]]$ derived from our example chain graph and we can read out that the statement $C \perp\!\!\!\perp E \mid D$ is not true since there is a non-blocking path between C and E via A and B. We additionally need to observe either A or B or both to make the statement true.

4.1.6. Bootstrapping Graphical Models

In previous sections we introduced the craft of graphical models that is mainly concerned with reasoning in network structures and inference queries. In contrast this section will briefly discuss the art of graphical models, that is, design decisions involved when engineering a model. Laying out a graphical model involves three steps: choosing random variables, picking a network structure, and parametrizing the factors. The three steps cannot just be done one by one in linear order because decisions made at any step are influencing all the others. The main goal of the graphical model design process is to balance out all three steps to arrive at a solution with a as low as possible complexity while still being useful for practical applications.

The complexity of a graphical model can be estimated by the number of (free) parameters in the model. To keep inference tractable and fast we want to reduce model complexity without losing too much descriptive power. For instance, a model consisting of a single factor with two binary variables has four entries and thus three free parameters. In contrast, a model with two factors over one binary variable each has only two free parameters and would be preferable in

terms of model complexity. However, we also loose descriptive power because the latter model cannot consider interactions between the two variables (they are independent in the model). Inference in less complex models is in general faster because we can think of inference—on the very highest level of abstraction—as a search through the state space of the model whose size is roughly proportional to the number of parameters in the model.

There are several ways to influence the model complexity. The more (conditional) independence assumptions we make the more the model can be factorized and in return reduces the model complexity. Or in other words, we want to have as few edges as possible in the graph. Furthermore, it is desirable to have as few variables as possible, too. Frequently one can trade off the number of edges against the number of variables. Consider four variables all connected to each other with in total six edges. We can introduce a fifth (hidden) variable and mediate the dependency between the original variables via this helper variable. That way we have increased the number of variables by one but reduced the number of edges to four.

When designing a (partially) directed model a widely accepted rule of thumb is to choose a structure that reflects the causal relationships between the variables such that parent variables are considered the cause for the effect taking place in the children variables. Experience of practitioners in the field shows that this approach leads in general to sparser models. Another aspect to consider is the parametrization of Bayesian factors. The concept of a child variable depending on some parent variables maps naturally to the concept of discriminative classifiers like Random Forest which can output a probability estimate for classes given certain features. If we factorize the network neatly, some factors can be learned directly using a discriminative classifier instead of using maximum likelihood based learning approaches which are usually costlier than a classifier. In particular, one should design factors such that they model the probability of a target variable given some features than the other way around.

4.2. A Graphical Model for Tracking-by-Assignment

4.2.1. Random Variables and Layout

Our model contains two types of random variables: *detection* variables and *assignment* variables. A binary detection variable $X_i^{(t)}$ is associated with the ith object (cell candidate) in time slice t, and $\mathcal{X}^{(t)}$ denotes the set of all detection variables at time t. Values of these variables determine if the corresponding detections are accepted as a true object (and hence incorporated into the tracking interpretation), or rejected as a false positive. We argue to operate object identification in a high recall regime, that is, we try to identify all actual objects at the cost of introducing false positives (lowering precision). Compared to singling out false positives it is harder to recover objects that were lost in the first place. We therefore forgo

modeling of false negatives to rein in model complexity and assume an object identification step with high recall.

Binary assignment variables $Y_{ij}^{(t)}$ are associated with every pair of object candidates i, j at times $t \in [1, \ldots, T-1]$, $t+1$, and $\mathcal{Y}^{(t)}$ is the set of all assignment variables between steps t and $t+1$. A value of 1 expresses the belief that cell candidate j at time step $t+1$ is identical with, or a child of, cell candidate i at time step t, and a value of 0 says that i and j are unrelated.

In addition, a number of natural consistency constraints are imposed on these variables by our application in developmental biology: First, each cell must have at most one predecessor, i.e. $\sum_i Y_{ij}^{(t)} \leq 1$. Second, each cell must have a unique fate—if it does not disappear (by leaving the data frame or other reasons), it can either move to (be assigned to) a unique cell candidate in the next time slice, or it can divide and be associated with exactly two cell candidates in the next time slice. These children, in turn, may not have any other nonzero incoming assignment variables. Third, there is a biological upper bound on the distance traveled between time frames, excluding all assignments that are too far apart. This leads to a significant reduction in the number of required assignment variables.

These consistency constraints define which configurations are impossible, i.e. have zero probability. In the following subsections, we describe how the probabilities of feasible configurations are defined by connecting variables into a chain graph whose factors encode our knowledge about the plausibility of different trackings as a function of data-dependent properties such as distance traveled, probability of being a misdetection, etc.

Conditional random field over assignment variables

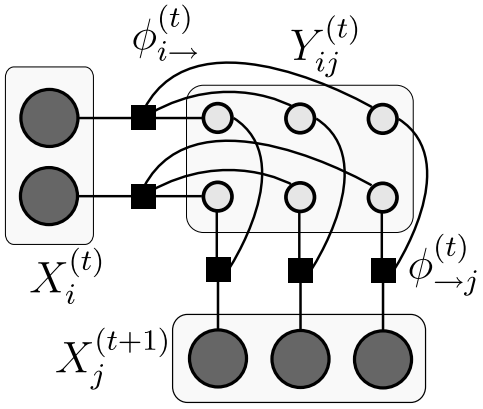
Let us pretend for the moment that the optimal values of the detection variables are already known. Then the probability of a configuration $\mathcal{Y}^{(t)}$ of assignment variables connecting slices t and $t+1$ *given* the detection variables $\mathcal{X}^{(t)}$ and $\mathcal{X}^{(t+1)}$ can be expressed with an undirected graphical model, in particular a conditional random field (CRF):

$$\mathrm{CRF}^{(t)}: \; P^{(t)}(\mathcal{Y}^{(t)} \mid \mathcal{X}^{(t)}, \mathcal{X}^{(t+1)}) \tag{4.12}$$

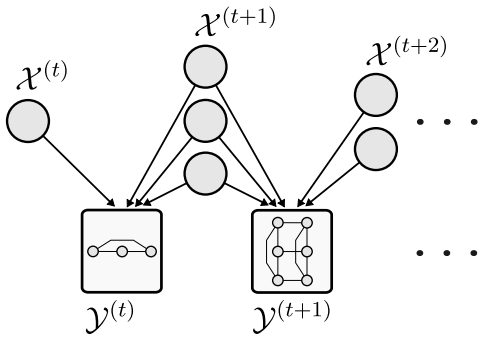
where

$$P^{(t)}(\mathcal{Y}^{(t)} | \mathcal{X}^{(t)}, \mathcal{X}^{(t+1)}) = \frac{1}{Z^{(t)}} \prod_{X_i^{(t)} \in \mathcal{X}^{(t)}} \phi_{i\rightarrow}^{(t)}(X_i^{(t)}, \mathcal{Y}_{i\rightarrow}^{(t)}) \prod_{X_j^{(t+1)} \in \mathcal{X}^{(t+1)}} \phi_{\rightarrow j}^{(t)}(\mathcal{Y}_{\rightarrow j}^{(t)}, X_j^{(t+1)}), \tag{4.13}$$

and $\phi_{i\rightarrow}^{(t)}$ and $\phi_{\rightarrow j}^{(t)}$ denote the factors for outgoing and incoming transitions respectively. Note that the CRF partition function $Z^{(t)}(\mathcal{X}^{(t)}, \mathcal{X}^{(t+1)})$ is only calculated over the assignment variables, since the detection variables are considered as given. A factor graph Kschischang et al. (2001) representation of eq. (4.13) is shown in



(a) **Undirected part of the chain graph**. A single assignment CRF is shown as a factor graph Kschischang et al. (2001). The dark nodes represent fixed detection variables $\mathcal{X}$ (at t and $t+1$) and the smaller, light nodes assignment variables $\mathcal{Y}$.



(b) **Directed part of the chain graph**. The boxes symbolize supernodes containing assignment CRFs according to Fig. 4.4a.

Figure 4.4.: Chain graphical model for global tracking by assignment.

Fig. 4.4a. Each of the maximal cliques of the underlying undirected graph corresponds to a factor in the CRF. These CRFs will form the supernodes of the chain graph, while the individual assignment variables are the subnodes.

Analyzing the separation properties of the graph in terms of connecting paths between variables, implicit independence assumptions in the CRF can be identified: an assignment $Y_{ij}^{(t)}$ is conditionally independent of all assignments $Y_{kl}^{(t)}$ involving different detection variables $k \neq i$, $l \neq j$, provided that all incoming assignments $\mathcal{Y}_{\rightarrow j}^{(t)}$ of detection j and all outgoing assignments $\mathcal{Y}_{i\rightarrow}^{(t)}$ of detection i are given:

$$Y_{ij}^{(t)} \perp\!\!\!\perp Y_{kl}^{(t)} \mid \mathcal{Y}_{\rightarrow j}^{(t)}, \mathcal{Y}_{i\rightarrow}^{(t)} \quad l \neq j, i \neq k\,. \tag{4.14}$$

In other words, the assignment decisions only influence a small neighborhood of other assignment variables directly. By fixing enough variables the CRF can even decouple in two or more completely independent tracking problems—promoting the tractability of the chain graph model.

Combining assignment CRFs in a chain graph

In order to get rid of the requirement that detection variables are already known, we connect all CRFs defined above by means of directed edges from detection variables to factors, in a manner analogous to a Bayesian network. The joint probability over

all time steps can now be factorized as

$$P(\mathcal{X},\mathcal{Y}) = \prod_{t=1}^{T} \prod_{X_i^{(t)} \in \mathcal{X}^{(t)}} P_{\text{det}}^{(t,i)}(X_i^{(t)}) \cdot \prod_{t=1}^{T-1} P^{(t)}(\mathcal{Y}^{(t)} | \mathcal{X}^{(t)}, \mathcal{X}^{(t+1)}), \tag{4.15}$$

where $P_{\text{det}}^{(t,i)}(X_i^{(t)})$ is the probability that a detection hypothesis should be accepted into the tracking. The corresponding chain graphical model is sketched in Fig. 4.4b.

To analyze the independence properties in such a model one has to use the quite involved rules of c-separation. By treating each CRF as a single random variable (supernode) with as many states as configurations of the assignment variables (subnodes) and assuming that all variables in the supernode are observed at once, the less complex d-separation analysis can be applied to the directed part of the model. The (unobserved) supernodes have only incoming edges and are blocking the path between detection variables, revealing the following conditional independence relations:

1. Detection variables are independent from each other, i.e. $X_i \perp\!\!\!\perp (\mathcal{X} \setminus X_i)\ , \forall X_i$.
2. Consecutive assignment variables are conditionally independent given the connecting detection variables, i.e. $\mathcal{Y}^{(t)} \perp\!\!\!\perp \mathcal{Y}^{(t+1)} \,|\, \mathcal{X}^{(t+1)}$.

These are rather strong independence assumptions and—of course—only an approximation to the real world. Later, we introduce an extension (eqs. (4.24) and (4.23)) that, on the one hand, improves the model accuracy but, on the other hand, weakens these relations and makes inference harder.

Taken together with the independence assumed further above, we obtain a model that no longer falls within the scope of either directed or undirected probabilistic graphical models but is a chain graph: a set of undirected graphical models, the CRFs $P^{(t)}(\mathcal{Y}^{(t)}|\mathcal{X}^{(t)},\mathcal{X}^{(t+1)})$ that depend on decisions $\mathcal{X}^{(t)}$ regarding the presence or absence of true objects.

4.2.2. Energy Representation

A (strictly) positive distribution factoring as a product of potentials can equivalently be restated as sum of energy factors by expressing the probability in eq. (4.15) as a Gibbs distribution

$$P(\mathcal{X},\mathcal{Y}) = \frac{1}{Z} e^{-E(\mathcal{X},\mathcal{Y})} \tag{4.16}$$

with

$$E(\mathcal{X},\mathcal{Y}) = \sum_{t=1}^{T} \sum_{i} E_{\text{det}}^{(t,i)}(X_i^{(t)}) + \sum_{t=1}^{T-1} \left(\sum_{i} E_{\text{out}}^{(t,i)}(X_i^{(t)}, \mathcal{Y}_{i\rightarrow}^{(t)}) + \sum_{j} E_{\text{in}}^{(t,j)}(\mathcal{Y}_{\rightarrow j}^{(t)}, X_j^{(t+1)}) \right) \tag{4.17}$$

where E_{det}, E_{out}, and E_{in} are the energy types corresponding to the factor types P_{det}, $\phi^{(t)}_{i\rightarrow}$, and $\phi^{(t)}_{\rightarrow j}$ respectively.

Switching to the energy domain has several advantages. First, it allows us to use inference algorithms such as integer linear programming that only work on factor sums but not on factor products. Second, in contrast to potentials which have to be nonnegative, energies can assume the full range of real numbers putting less restrictions on factor parametrization. Third, when implemented in software summing operations are in general more numerically stable compared to products.

4.3. Tracking as MAP Inference

The chain graph distribution of eq. (4.15) assigns a probability value to every configuration of assignment and detection variables. A tracking is obtained by finding the most probable or maximum a posteriori (MAP) configuration:

$$\text{MAP}: \operatorname*{argmax}_{\mathcal{X},\mathcal{Y}} P(\mathcal{X},\mathcal{Y}) \tag{4.18}$$

or equivalently

$$\text{MAP}: \operatorname*{argmin}_{\mathcal{X},\mathcal{Y}} E(\mathcal{X},\mathcal{Y})\,. \tag{4.19}$$

In Sec. 5.3.2 on page 50 we compare several energy minimization methods by their capability to find a MAP configuration. As it turns out, integer linear programming is—by a large margin—the best method and delivers good performance with a reasonable running time for practical applications. In particular there are two advantages. First, it allows us to use state-of-the-art ILP solvers that are able to determine globally optimal solutions of eq. (4.17). Any remaining shortcomings of our tracking results can thus be attributed to our model and are not a consequence of approximate optimization. Second, zero probability / infinite energy configurations can be naturally ruled out by adding hard constraints to the linear program.

Compared to other kinds of inference queries MAP inference has the advantage that there is no need to evaluate or estimate the partition function of the model since, for instance, in the energy domain the partition function just enters as a shifting constant with no influence on the argument minimum of the energy. Then again calculating marginals can provide valuable information about the certainty of the presented tracking. Another interesting approach that does not necessarily require an evaluated partition function is presenting the m-best solutions to the user (Batra et al., 2012) to convey an impression of the stability of the tracking solution: variables that wouldn't change state in different solutions would be considered more stable and certain than variables that change state.

4.4. Cell Nuclei Tracking

4.4.1. Basic Model

In this section we parametrize the chain graph model for cell nuclei tracking. Recall the energy representation of eq. (4.17). The first term allows to incorporate evidence from the raw data regarding the presence or absence of a cell at a location where an object candidate was detected. We use the Random Forest classifier (Breiman, 2001) to estimate the probability $\hat{P}_{\text{det}}^{(t,i)}$ of a detection being truly a cell, based on features such as intensity distribution and volume of each cell candidate found by the detection module (see Sec. 4.5.2 on page 37 for a detailed description). In keeping with the definition of the Gibbs distribution, we get

$$E_{\text{det}}^{(t,i)}\left(X_i^{(t)}\right) = \begin{cases} -\ln\left(\hat{P}_{\text{det}}^{(t,i)}\right) & , X_i^{(t)} = 1 \\ -\ln\left(1 - \hat{P}_{\text{det}}^{(t,i)}\right) & , X_i^{(t)} = 0 \end{cases} \tag{4.20}$$

The assignment energies E_{in} and E_{out} need to embody two kinds of prior knowledge: firstly, which trackings are consistent; and secondly, amongst all consistent trackings, which ones are plausible. To enforce consistency, zero probability / infinite energy is assigned to cells dividing into more than two descendants, cells arising from the merging of two or more cells from the previous slice, and cell candidates that are regarded as false positives.

To penalize implausible trackings, we need to parametrize our model appropriately. The model equations expose exactly five free parameters (C_{init}, C_{term}, C_{opp}, w, $\bar{d}$). C_{init} imposes a penalty for track initiation (i.e. when a cell (re-)appears in the data frame) and C_{term} for track termination (i.e. when a cell disappears from the data frame or is lost due to cell death). To discourage trivial solutions where too many objects are explained as misdetections, we exact a opportunity cost of C_{opp} for the lost opportunity to explain more of the data. Furthermore, each move is associated with a cost dependent on the squared distance traveled of that move. For cell divisions, finally, it is known that the descendants tend to be localized at an average distance $\bar{d}$ from the parent cell, and we punish deviations from that expected distance.

All of the above can be summarized in the following assignment energies:

$$E_{\text{out}}^{(t,i)}(X_i^{(t)}, \mathcal{Y}_{i\to}^{(t)}) = \left\{ \begin{array}{lll|l} \infty & , X_i^{(t)} = 1 \wedge \sum_j Y_{ij}^{(t)} > 2 & & > 2 \text{ children} \\ w\Big((d-\bar{d})^2 & & & \\ \quad +(d'-\bar{d})^2\Big) & , X_i^{(t)} = 1 \wedge \sum_j Y_{ij}^{(t)} = 2 & & \text{division} \\ wd^2 & , X_i^{(t)} = 1 \wedge \sum_j Y_{ij}^{(t)} = 1 & & \text{move} \\ C_{\text{term}} & , X_i^{(t)} = 1 \wedge \sum_j Y_{ij}^{(t)} = 0 & & \text{disappearance} \\ C_{\text{opp}} & , X_i^{(t)} = 0 \wedge \sum_j Y_{ij}^{(t)} = 0 & & \text{opportunity} \\ \infty & , X_i^{(t)} = 0 \wedge \sum_j Y_{ij}^{(t)} > 0 & & \text{tracked misdetection} \end{array} \right. \tag{4.21}$$

$$E_{\text{in}}^{(t,j)}(\mathcal{Y}_{\to j}^{(t)}, X_j^{(t+1)}) = \begin{cases} \infty & , X_j^{(t+1)} = 1 \wedge \sum_i Y_{ij}^{(t)} > 1 \\ 0 & , X_j^{(t+1)} = 1 \wedge \sum_i Y_{ij}^{(t)} = 1 \\ C_{\text{init}} & , X_j^{(t+1)} = 1 \wedge \sum_i Y_{ij}^{(t)} = 0 \\ 0 & , X_j^{(t+1)} = 0 \wedge \sum_i Y_{ij}^{(t)} = 0 \\ \infty & , X_j^{(t+1)} = 0 \wedge \sum_i Y_{ij}^{(t)} > 0 \end{cases} \begin{array}{l} > 1 \text{ parent} \\ \text{move} \\ \text{appearance} \\ \text{misdetection, no parent} \\ \text{misdetection with parent} \end{array} \tag{4.22}$$

Note that more informative features could be extracted from the raw data: for instance, besides the length of a move, one could consider the similarity of the associated cell candidates to obtain an improved estimate of their compatibility. However, it is then no longer possible to select appropriate values for the parameters with a simple grid search. Instead, a proper parameter learning strategy will be needed, which will be a subject of our future research.

4.4.2. Minimal Cell Cycle Length Extension

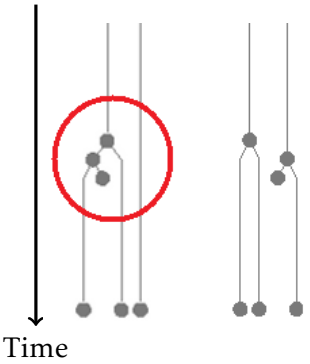


Figure 4.5.: Implausible division. The figure shows cell lineage trees with cell divisions marked by dots. They were obtained from the chain graph tracking model without the minimal cell cycle length extension. Several divisions in close temporal proximity—as marked by the red circle—are biologically impossible and should be ruled out by the method.

The model may be extended further by incorporating domain specific knowledge. For instance, cells must pass through specific cell cycle states (interphase, prophase, metaphase, anaphase, telophase) and thus, there is a biological lower bound for the duration of a cell cycle. In other words, it is impossible to observe a cell dividing twice within a number of subsequent slices as it is the case in Fig. 4.5. This biological constraint can be integrated in our cell tracking model by expanding the detection variables $X_i^{(t)}$ to obtain the number of time slices since the last division of an individual cell, i.e. $X_i^{(t)} \in \{0, 1, ..., \tau\}$, where $X_i^{(t)} = 0$ still stands for misdetection and τ is a parameter for the minimal duration between division events of an individual cell. Hence, the energy functions given above change only slightly in that $X_i^{(t)} = 1$ becomes $X_i^{(t)} \geq 1$ and $X_i^{(t)} = \tau$ for the division energy. Besides, another

two factors need to be introduced to incorporate the counting between successive detections. The corresponding energy functions are given by

$$E_{\text{cnt}\rightarrow}(X_i^{(t)}, \mathcal{Y}_{i\rightarrow}^{(t)}, \mathcal{X}_{i\rightarrow}^{(t+1)}) = \begin{cases} \infty & , Y_{ij}^{(t)} = 1 \wedge \sum_k Y_{ik}^{(t)} = 2 \wedge X_j^{(t+1)} \neq 1 \\ \infty & , Y_{ij}^{(t)} = 1 \wedge \sum_k Y_{ik}^{(t)} = 1 \wedge X_i^{(t)} < \tau \wedge X_j^{(t+1)} \neq X_i^{(t)} + 1 \\ \infty & , Y_{ij}^{(t)} = 1 \wedge \sum_k Y_{ik}^{(t)} = 1 \wedge X_i^{(t)} = \tau \wedge X_j^{(t+1)} \neq \tau \\ 0 & , \text{otherwise} \end{cases}, \tag{4.23}$$

$$E_{\text{cnt}\leftarrow}(X_j^{(t+1)}, \mathcal{Y}_{\rightarrow j}^{(t)}) = \begin{cases} \infty & , \sum_k Y_{kj}^{(t)} = 0 \wedge X_j^{(t+1)} \neq 0 \wedge X_j^{(t+1)} \neq \tau \\ 0 & , \text{otherwise} \end{cases}, \tag{4.24}$$

where $\mathcal{X}_{i\rightarrow}^{(t+1)}$ are the detection variables connected to $X_i^{(t)}$ through $\mathcal{Y}_{i\rightarrow}^{(t)}$. Technically, these rules are implemented as hard constraints on indicator variables each representing a possible state of the detection variables. It should be noted that with this modification, the detection variables are no longer independent. For that reason and due to the increased number of variables, computation time increases by a factor of ten compared to the chain graph model without this modification.

4.5. Determining Parameters

4.5.1. Exhaustive Search

The chain graph tracking reasoner as well as the joint optimal assignment reasoner and the method by Bise et al. (2011) (later used to compare our methods to in tracking experiments) all expose some free parameters. We chose to find parameter settings according to an optimal f-measure of successfully reconstructed move, division, appearance, and disappearance events relative to a manually obtained tracking. Based on sensible parameter ranges found by manual experimentation we set up a search grid of parameter combinations and evaluated the performance for all combinations. To speed up the procedure we first did a search on a coarse grid. Subsequently, we identified the region in parameter space containing the highest performance so far and conducted a second search with a finer grid in that region to find the final parameter settings. Furthermore, we evaluated several parameter combinations in parallel using the PARAWALKER software[4] which we developed previously (Kausler, 2010).

4.5.2. Chain Graph Priors: Discriminative Learning

The chain graph model contains prior factors that govern the state of the detection variables and which can be parametrized by an estimate of the probability $\hat{P}_{\text{det}}^{(t,i)}$ for a detected object being a true cell nucleus or not (see Sec. 4.4.1 on page 35).

[4] https://github.com/bekaus/parawalker

For a start we could just use the same value for all detection variables based on an estimate of the general noise level (say, roughly 10% of all objects are false positives) and, as a matter of fact, the results are acceptable in practice. However, a much better performance should be achievable if we use an individual estimate for every single object based on the prediction of a state-of-the-art discriminative classifier like Random Forest.

Lindner (2011) describes a set of object features calculated on three-dimensional connected components. Based on these features he conducted experiments to distinguish cell nuclei from other objects by training a Random Forest classifier. As training and test data he used segmentations of three-dimensional images of Zebrafish embryos in early stages of development recorded with a light sheet microscope. Based on a feature selection experiment he advises to select all the proposed features (pg. 61). Furthermore, he asserts that to achieve the best discriminative performance care should be taken to train on a balanced dataset with a similar number of true cell nuclei and other objects (pg. 64–65).

We take his advice and train a Random Forest classifier on balanced training datasets of cell nuclei and false positive objects using the following Lindner features calculated on the segmented object voxels:

volume	– total number of voxels in an object
bounding box	– ratio of object volume to bounding box volume
mean position	– mean voxel coordinates and higher central moments
center of mass	– intensity-weighted mean voxel coordinates and higher moments
maximum intensity	– coordinate of the highest intensity voxel
principal components	– eigenvalues of the object principal components
intensity	– mean voxel intensity and higher moments
intensity quantiles	– quantiles of the voxel intensity distribution
pairwise intensity	– several intensity distance measures on voxel pairs
statistical geometrical features	– texture sensitive features (Chen et al., 1995; Walker and Jackway, 1996)

Furthermore, all features except volume, bounding box, and mean position were also calculated on the voxels contained in an enlarged bounding box around the segmented object to capture some neighborhood context, too.

4.6. Summary

In this section we proposed a novel graphical model taking the form of a chain graph to solve the tracking-by-assignment problem and presented a parametrization of the model suitable for cell nuclei tracking. Furthermore we extended the model to consider an expected minimal temporal distance between cell divisions—demonstrating the flexibility of formulating tracking as a graphical model. In the following chapters we will subject the model to a thorough experimental evaluation and describe an implementation in software.

5. Evidence

In this chapter we present the experimental evidence for the capabilities of the chain graph tracking model presented in Chap. 4 and its performance when integrated with ilastik segmentation into a digital embryos recording pipeline (Chap. 2 on page 9). The experiments are based on two light sheet microscopy data sets: recordings of zebrafish early embryogenesis and Drosophila syncytial blastoderm. For both datasets we manually obtained a tracking as a benchmark for the automatic methods. We introduce *precision*, *recall*, and *f-measure* as quantitative tracking measures and show how they can be derived from set-theoretic principles.

In particular, we conducted experiments for the following aspects of our approach:

Segmentation Accuracy: To establish ilastik as a suitable segmentation method on the presented light sheet microscopy datasets.

Inference Method: To find the best performing inference methods amongst popular choices.

Tracking Performance: To evaluate performance of the chain graph model compared to others for cell tracking.

Minimal Cell Cycle Length: To show the impact of a prior term based on an expected minimal temporal distance between cell divisions.

Detection Performance: To investigate the capability of the chain graph model to improve false positive classification performance.

The results of the experiments will be put into context in a separate chapter with discussions (Chap. 7).

5.1. Datasets

For the quantitative evaluation of our methods we choose two light sheet microscopy datasets consisting of 3d+t recordings of the currently most popular model organisms in the field: zebrafish and Drosophila. In this section we document technical details of theses datasets.

In Sec. 2.2 on page 12 we describe two segmentation approaches—ilastik pixel classification and regularized graph cut (rgc)—that are both on par in terms of performance. Since pixel classification is easier to use and to parametrize we prefer it over rgc in general. Therefore, the Drosophila dataset was segmented with ilastik using all available pixel features for maximum performance. However, for the zebrafish dataset the quantitative evaluations in this chapter are based on a rgc segmentation (due to historical reasons). Future experiments should use ilastik alone.

5.1.1. Zebrafish Early Embryogenesis

Keller et al. (2008) recorded time-lapse volumetric images of developing zebrafish embryos (3d+t). The zebrafish dataset stems from their work and is a recording of 80 time slices of the animal pole from the 66 cells stadium[1] up to ca. 2400 cells. The embryo enters the 64 cell stage about two hours after fertilization and starts forming the blastula—a hemispherical cell body covering roughly one third of the spherical egg with the yolk covering the remaining two thirds. The embryo has a diameter of around 0.7mm. Development of the embryo is complete after ca. 48 hours and the animal enters the hatching period while continuing to grow. See Wolpert (2007, pg. 96 pp.) for more details.

One image stack is recorded every 60s, totaling 2 hours over the 80 time slices. The volume size is 1116x1111 pixels per image and 166 images per stack. The physical resolution per voxel is 0.3 µm and 1µm in lateral and axial direction, respectively. The dynamic range of the intensity is 8 bit. Since only the cell nuclei are stained with a fluorescent marker neither cell membranes nor yolk are visible in the microscopy images. A maximum intensity projection of the first volume in the sequence is shown in Fig. 5.1.

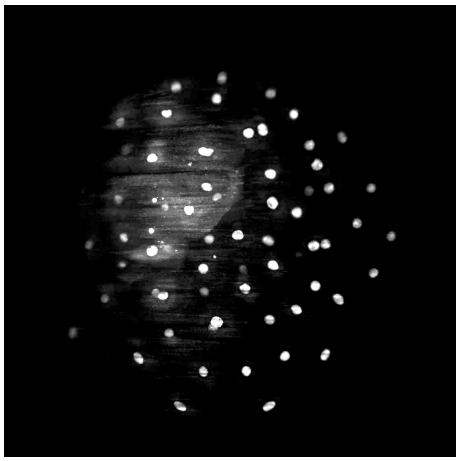


Figure 5.1.: Maximum intensity projection of the first volume in the zebrafish sequence consisting of 80 volumes.

We manually reconstructed the cell lineage for the first 25 time slices of the dataset comprising two full cell cycles. In particular we marked the center position of every nucleus in the raw images and the tracking-associations of the nuclei in consecutive time slices. A detailed tracking protocol is reproduced in Appendix A.

[1] Textbook knowledge claims synchronous cell divisions in the early zebrafish embryo and we would expect to see exactly 64 cell nuclei after six cycles. Due to unknown reasons there were two additional divisions.

5.1.2. Drosophila Syncytial Blastoderm

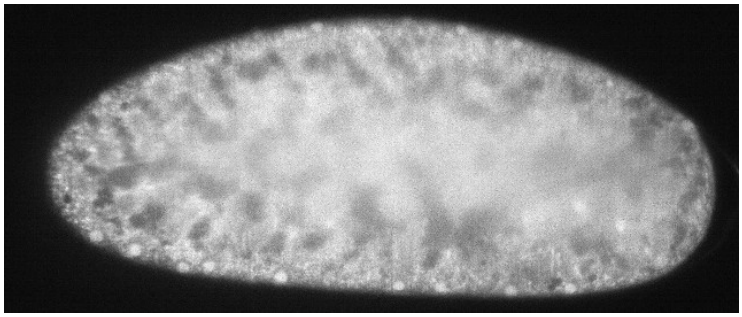


Figure 5.2.: One image of the first stack of the Drosophila dataset. Cell nuclei are residing near the border of the cigar-shaped embryo.

The other dataset consists of 3d image stacks of a fruit fly syncytial blastoderm over 40 time slices (see Fig. 5.2). The syncytial blastoderm is formed about 90 minutes after fertilization and persists for roughly 90 minutes. During that stage the cell nuclei reside at the periphery of the cigar-shaped embryo in a *syncytium*—the nuclei share a common cytoplasm without separating membranes. The embryo measures around 0.5 mm along its longest axis. See Wolpert (2007, pg. 32 pp.) for more details.

One image stack is recorded every 30s, totaling 20 minutes over the 40 time slices. The volume size is 2362x994 pixels per image and 47 images per stack. The physical resolution per voxel is 0.1625 µm and 2µm in lateral and axial direction, respectively. The dynamic range of the intensity is 8 bit. The embryo stems from a transgenic line of Drosophila that is autofluorescent (Mavrakis et al., 2008). Compared to the zebrafish dataset not only the nuclei but also the cytoplasm exhibits intensity in the microscopy images.

The data was recorded by the lab of Lars Hufnagel at the European Molecular Biology Laboratory (EMBL) in Heidelberg in autum 2011. They used an in-house developed light sheet microscope with two light sheets and two cameras later to be published in Krzic et al. (2012). Every data slice is therefore a fusion of four recorded images. Due to that new technology the contrast is homogeneous over the whole image range and the segmentation method was able to retrieve all visible cell nuclei (alas with more than 10% false positive objects).

We performed a manual tracking for all 40 time slices. Unlike the zebrafish dataset the manual tracking was executed on the segmented objects and not on the raw data.

Compared to the zebrafish dataset the fruit fly dataset has lower contrast resulting in more ill-shaped cell segmentations and false positive detections that are hard to distinguish from actual cells. Then again, cells reside only near the surface of the embryo, lowering the chance of wrong tracking assignments to neighboring cells compared to the zebrafish embryo.

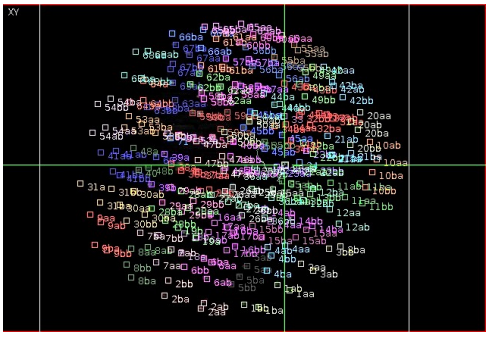
5.2. Methods

5.2.1. Detection Variable Priors via Object Classification

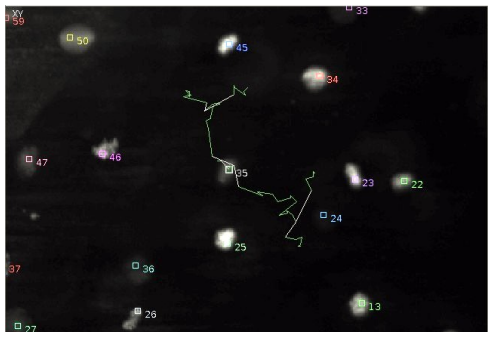
Tracking-by-assignment formulated as a chain graphical model (cf. eq. 4.15 on page 33) contains prior probability factors $P_{\text{det}}^{(t,i)}(X_i^{(t)})$ describing the likelihood whether an object candidate i at time slice t is a true positive ($X_i^{(t)} = 1$) or a false positive ($X_i^{(t)} = 0$). A simple approach for parametrization would be to assign a flat prior to every object according to an estimated noise level. For instance, if we expect about 10% false positive objects, we would set $P_{\text{det}}^{(t,i)}(1) = 0.9$ and $P_{\text{det}}^{(t,i)}(0) = 0.1$ for all detected objects. A smarter way is to individually assign a factor to each object based on object specific features.

Using connected component analysis we can extract objects in form of sets of distinct voxel coordinates from segmented microscopy images. Lindner (2011) describes a set of features suitable to characterize cell nuclei-shaped objects in 3d, comprising features such as quantiles of the intensity distribution or statistical geometric features (Walker and Jackway, 1996). We labeled a representative subset of the object candidates and trained a Random Forest classifier (Breiman, 2001) to predict the probability of an object being a false or true positive segmentation. Based on that we parametrized the detection priors in the chain graph model for every object individually. (See also Sec. 4.5.2 on page 37 for more details.)

5.2.2. Ground Truth Recording on Raw and Segmented Data



(a) 2D projection of one volume together with manually placed cell nuclei markers. Nuclei with common ancestors share the same marker color and principal number.



(b) Zoomed-in detail of the 2D projection for one volume. The lines in the center represent the future movements of nucleus 35 which were manually tracked. The three bifurcations of the line indicate cell divisions.

Figure 5.3.: View5D software for manual tracking. The two panels show XY projections of the zebrafish dataset generated by the software.

The image processing software Fiji (Schindelin et al., 2012) includes the View5D[2] plugin that allows the manual tracking of divisible objects. The user provides a point-like marker for every object at every time slice and connects the markers between consecutive time slices. Fig. 5.3 on the facing page shows one example volume of the zebrafish dataset loaded in View5D together with markers for all nuclei.

With its help we tracked the cell nuclei in both the zebrafish (sec. 5.1.1 on page 42) and the Drosophila dataset (sec. 5.1.2 on page 43). In case of the zebrafish dataset we placed the markers directly on the raw images, i.e. we did a manual identification and tracking of the nuclei. The full tracking protocol is reproduced in Appendix A. For the Drosophila dataset we tracked only the segmented objects and only distinguished between false positive and true positive objects. That is, with the Drosophila ground truth we can exactly judge the performance of the chain graph tracking model since we used the very same input information that is visible to the tracking algorithm; whereas the zebrafish ground truth is useful as a benchmark for a complete reconstruction pipeline consisting of a segmentation and a tracking step.

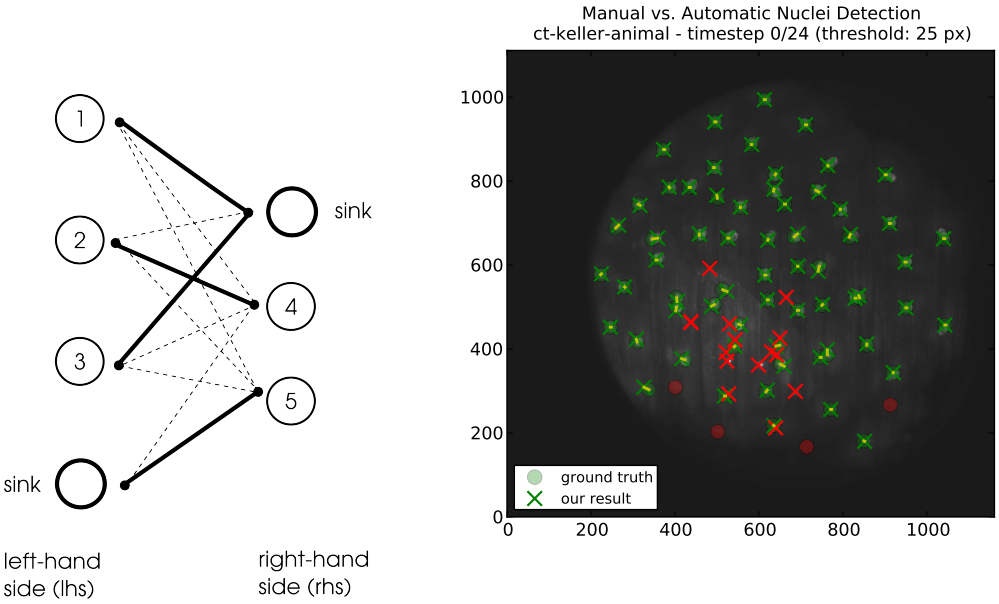
When placing the marker on the raw images directly we have to establish a correspondence between the markers and the segmented objects. We placed the markers on the pixel with maximum intensity inside a nucleus and calculated the maximum intensity position inside the segmented objects, too. Assuming a perfect segmentation we could simply match each marker to its nearest segmented object, then. Since we have to consider false positive (phantom nuclei) and false negative (missed nuclei) segmentations, we have to allow for markers and segmented objects that are not matched. This can be formulated as a *weak asymmetric bipartite matching* where markers resp. segmented objects are matched to their nearest neighbor segmented object resp. marker as long as the distance is below a certain threshold. Otherwise, they are matched with a hypothetical *sink* object resp. marker. Fig. 5.4a on the next page illustrates the idea.

This matching problem can be formulated as a binary integer program

$$\begin{aligned} \min \ & \sum_{i,j} c_{i\leftrightarrow j} \cdot x_{i\leftrightarrow j}, && i \in \text{lhs}, j \in \text{rhs} \\ \text{s.t.} \ & \sum_{j} x_{i\leftrightarrow j} = 1, \ \sum_{i} x_{i\leftrightarrow j} = 1, && \forall i, j \setminus \text{sink} \\ & x_{i\leftrightarrow j} \in \{0,1\}, && \forall i, j \end{aligned} \tag{5.1}$$

where $c_{i\leftrightarrow j}$ is the distance between the marker i and the segmented object j measured from the maximum intensity position. The sets lhs and rhs contain one index for every marker resp. segmented object and one additional index each representing the two sink nodes. The problem can be easily solved in practice with

[2] http://www.nanoimaging.uni-jena.de/View5D/View5D.html



(a) Assigning ground truth markers and segmented objects formulated as an asymmetric bipartite matching. Markers (lhs) and segmented objects (rhs) are matched with their respective nearest neighbors below a certain distance threshold. Otherwise, they are matched with a *sink* node.

(b) Example result of matching ground truth and segmentation. Circles indicate ground truth and crosses segmented objects. Successful matches are green. Red crosses are false positives and red circles false negatives.

Figure 5.4.: Matching ground truth nuclei locations with segmented object locations.

an off-the-shelve integer programming package. As a result we obtain three sets of objects: objects both present in ground truth and segmented data, phantom segmentations (false positives), and objects that were not segmented (false negatives). An example result is shown in Fig. 5.4b for the first time slice of the zebrafish dataset and a distance threshold of 25 pixels.

5.2.3. Measuring Tracking Performance

Performance of a *contestant* tracking relative to a *ground truth* tracking is measured in the following terms:

Recall: The number of tracking events that are found by the contestant and are also present in the ground truth relative to the number of all ground truth events.

Precision: The number of tracking events that are found by the contestant and are also present in the ground truth relative to the number of all contestant events.

Precision and recall make sense intuitively. Additionally, we will derive the two measures by reasoning over cardinalities of sets containing tracking events and derive the *f-measure* as a unified error measure.

Tracking events are defined in the sense of Sec. 3.2 but limited to types that describe transitions, i.e. *moves*, *divisions*, *appearances*, and *disappearances*. In particular, *true positive detections* and *false positive detections* are not tracking events. A tracking event can only be correct if all participating objects are true positive detections. Therefore, we do not consider true and false positive detections in the tracking precision and recall since they are already implicitly taken into account.

Formally, we have two sets: the set of tracking events present in the ground truth G and the set of tracking events extracted by the contestant tracking C. Furthermore, we can establish a matching between elements of the two sets when they describe the same tracking event type involving the same objects, i.e. the true positive tracking events.[3] That is, we split each set again in two distinct subsets G_m, $G_{\overline{m}}$, C_m, and $C_{\overline{m}}$ with m and $\overline{m}$ indicating match and no match.

Since they contain the successfully tracked events the sets G_m and C_m have the same cardinality. Consequently, the set $G_{\overline{m}}$ contains true tracking events that were not found correctly by the contestant tracking method and $C_{\overline{m}}$ contains the tracking events that didn't actually happen. In summary,

$$\begin{aligned} G &= G_m \cup G_{\overline{m}} \\ C &= C_m \cup C_{\overline{m}} \\ |G_m| &= |C_m| \,. \end{aligned} \tag{5.2}$$

This constellation is illustrated in Fig. 5.5 on the next page.

We can now formally define tracking *recall* and *precision* in terms of the ground truth and contestant sets:

$$\begin{aligned} \mathrm{rec}(G, G_m) &= \frac{|G_m|}{|G|} \\ \mathrm{prec}(C, C_m) &= \frac{|C_m|}{|C|} \end{aligned} \tag{5.3}$$

In empirical evaluations it is desirable to have only a single error measure. This allows a unique ranking of competing methods and provides a clear target objective for machine learning approaches. Of course, one could combine precision and recall in an *ad hoc* manner like the sum or the arithmetic mean. But by inspecting the definitions of recall and precision in terms of set cardinalities we can define an analogous performance measure covering both sets G and C:

$$\mathrm{perf}(G, G_m, C, C_m) = \frac{|G_m| + |C_m|}{|G| + |C|} \tag{5.4}$$

[3] This shouldn't be confused with true positive objects, which are a necessary but not sufficient precondition for true positive tracking events.

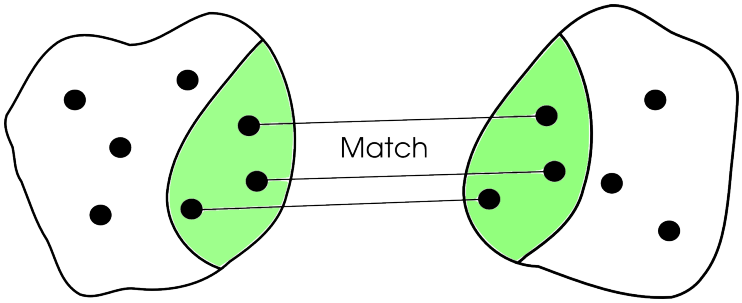


Figure 5.5.: Ground truth vs. contestant tracking method. The black dots represent tracking events like moves, divisions, and (dis-)appearances. There are two sets of tracking events, those in the ground truth and those found by the contestant tracking method. The events that were tracked correctly by the tracking method are present in both sets as indicated by the matching lines.

This combined performance score can also be expressed in terms of precision and recall. Using $|G_m| = |C_m|$ and expanding the fraction by $\frac{|C_m|}{|G|\cdot|C|}$, we obtain

$$\begin{aligned} \mathrm{perf}(G, G_m, C, C_m) &= \frac{2 \cdot \frac{|C_m|}{|C|} \cdot \frac{|G_m|}{|G|}}{\frac{|C_m|}{|C|} + \frac{|G_m|}{|G|}} \\ &= \frac{2 \cdot \mathrm{prec}(C, C_m) \cdot \mathrm{rec}(\mathrm{G}, \mathrm{G_m})}{\mathrm{prec}(\mathrm{C}, \mathrm{C_m}) + \mathrm{rec}(\mathrm{G}, \mathrm{G_m})} \end{aligned} \tag{5.5}$$

We now see that the combined error measure is actually the harmonic mean of precision and recall and is therefore equivalent to the well known *f-measure* (Costa et al., 2007).

5.3. Experiments

5.3.1. ilastik Segmentation Performance

In Sec. 2.2 on page 12 we described "segmentation" as the first step in a two-step cell tracking pipeline and selected ilastik pixel classification as our preferred segmentation method. This section documents its quantitative tracking performance on the zebrafish and Drosophila benchmark datasets.

Kaster (2011) already established ilastik as a viable segmentation method on data similar to our zebrafish dataset. He measures segmentation performance—among others—in terms of precision, recall, and f-measure defined as follows (Kaster, 2011, pg. 147).

Segmentation performance measures zebrafish (Kaster, 2011)

Precision: The percentage of segments in the computed segmentation that overlap at least one nucleus in the ground truth

	ilastik Segmentation Performance	
	Zebrafish Data	Drosophila Data
Precision	>0.99	0.87
Recall	0.7–0.8	1.0
F-measure	0.8–0.9	0.93

Table 5.1.: ilastik segmentation performance on zebrafish (Kaster, 2011, pg. 152) and Drosophila light sheet microscopy data. Note, that the measurements for the two kinds of data are defined in a comparable but not identical manner (see text for more information).

Recall: The percentage of nuclei in the ground truth that overlap with at least one segment in the computed segments.

F-measure: The harmonic mean of precision and recall.

He compares ilastik with a regularized graph cut approach and reports no significant difference between the two regarding segmentation performance. The typical performance on zebrafish data is given in Table 5.1.

The remainder of the section reports the segmentation performance of ilastik on the Drosophila dataset. This dataset is different from the zebrafish dataset since most of the exposed intensity does not belong to cell nuclei but to the internuclear space (compare Fig. 5.1 on page 42 and Fig. 5.2 on page 43). It is therefore not obvious that ilastik will perform equally well. As a performance measure we also choose precision, recall, and F-measure. Different from Kaster (2011) we use the matching of ground truth markers and segmented objects described in sec. 5.2.2 on page 44 as a basis using the following definitions.

Segmentation performance measures Drosophila

Precision: $TP/(TP+FP)$

Recall: $TP/(TP+FN)$

F-measure: The harmonic mean of precision and recall.

The true positives (TP) are segmented objects that are matched with a ground truth marker, the false positives (FP) are segmented objects with no matching ground truth marker, and the false negatives (FN) are ground truth marker with no matching segmented object. Note, that there are no true negatives. They would stand for non-existing nuclei that were correctly identified as such. That makes no sense in a segmentation context.

The chain graph tracking model is able to correct false positives, but it cannot recover false negatives. Therefore, we trained ilastik to achieve perfect recall—ensured by manually inspecting all volumes—at the cost of precision. As a consequence, we observe no false negatives. The segmentation of the Drosophila dataset

for the first 25 time slices contains 14303 segmented objects of which 12493 are true positives and 1810 are false positives. This is equivalent to a precision of 0.87 and a f-measure of 0.93 as summarized in Table 5.1 on the previous page.

Note, that for the zebrafish and the Drosophila dataset we used different definitions for precision and recall. The two definitions are equivalent in intention but will not lead to exactly the same performance in numbers for any given situation. Therefore when comparing the numbers for the two data sets no conclusions should be drawn from the relative ranking but only from their orders of magnitude.

5.3.2. Comparison of Inference Methods[4]

In sec. 4.3 on page 34 we argued that a tracking can be obtained from the maximum a posteriori (MAP) configuration of the chain graphical model and formulated the associated problem as a maximization of the joint probability or a minimization of the energy representation in terms of the random variables $\mathcal{X}$ and $\mathcal{Y}$:

$$\text{MAP}: \operatorname*{argmax}_{\mathcal{X},\mathcal{Y}} P(\mathcal{X},\mathcal{Y}) \tag{5.6}$$

or equivalently

$$\text{MAP}: \operatorname*{argmin}_{\mathcal{X},\mathcal{Y}} E(\mathcal{X},\mathcal{Y})\,. \tag{5.7}$$

In this section we compare a group of popular methods for MAP inference in graphical models regarding their optimization performance. We have chosen the Drosophila dataset for the comparison and constructed a chain graphical tracking model for it. Since not all of the investigated methods can work in the probability domain and to avoid an unnecessary and expensive evaluation of the partition function, we state the performance in the energy domain. Only linear programming based methods can naturally deal with infinite energy configurations by adding hard constraints. For all other methods we introduce soft constraints in form of very high energy values for infeasible configurations.

Polyhedral and Combinatorial Methods

A large class of algorithms solves a linear programming relaxation (LP) of the discrete energy minimization problem. Probably the most commonly used relaxation is the LP relaxation over the local consistency polytope (Schlesinger, 1976; Kumar and Torr, 2008). It can be solved with off-the-shelve LP-solvers (*LP-LP*). Other approaches are based on a relaxation of the dual instead of the primal formulation. Kappes et al. (2012) consider spanning trees as subproblems such that the relaxation is equivalent to the local consistency polytope relaxation. They propose—among others—bundle methods with a heuristic stepsize as a solving procedure which is guaranteed to converge to the optimum of the relaxed dual (*BUNDLE-H*). Related to polyhedral methods are Integer Linear Programs (*ILPs*). These include additional integer constraints and guarantee global optimality, contrary to the

[4] Content of section is based on the publication by Kappes et al. (2013).

Algorithm	Running Time	Minimal Energy	Lower Bound
BUNDLE-H	1068.11 sec	107,553,778.57	7501875.98
ILP	32.77 sec	**7,514,421.21**	**7514421.21**
LBP	30.47 sec	407,520,058.41	−inf
LBP-LF	308.83 sec	7,515,575.61	−inf
LP-LP	**3.26 sec**	7,516,359.61	7513851.52

Table 5.2.: Inference performance for different methods in the chain graphical model for tracking in the Drosophila dataset.

methods based on LP-relaxations. Solutions of ILPs are found by solving a sequence of LPs and either adding additional constraints to the polytope (cutting plane techniques), or branching the polytope into several polytopes (branch-and-bound techniques) (see Darby-Dowman and Wilson (2002) for a short review of integer programming).

Message Passing

Message passing methods like the mean field machine (Jordan et al., 1999) or belief propagation (Pearl, 1982) are simple to implement and can be easily parallelized, making them a popular choice in practice. Maybe the most popular message passing algorithm is loopy belief propagation (*LBP*). While LBP converges to a global optimum for acyclic graphical models, it is only a heuristic for general graphs. Then again it is known to perform reasonably well in practice (Murphy et al., 1999) and can be derived from first principles (Wainwright and Jordan, 2007).

Move-Making Methods

Another class of common methods are move-making algorithms with iterative conditional modes (ICM) as the most basic example (Besag, 1986). Starting from an arbitrary configuration the algorithm is iteratively searching through the labels of a each random variable to find a lower energy configuration. The Lazy-Flipper algorithm (Andres et al., 2012b) generalizes this principle to a greedy search over local subsets, converging to a configuration which is optimal within a hamming distance in label space. Lazy Flipper performance relies heavily on initialization. For the chain graph model trivial start configurations like setting all variables to 0 or 1 or a random value usually get stuck in some infeasible configuration. Therefore we initialize the lazy flipper with a solution from loopy belief propagation (*LBP-LF*).

Results

For each method we report runtime, objective value achieved by the final integer solution, and lower bound achieved. ILP found a globally optimal solution with an associated energy value of 107553778.57 . For all inference methods we used the implementations provided by OpenGm (Andres et al., 2012a)—an open-source C++ library for discrete graphical models. The results are summarized in Table 5.2. The methods BUNDLE-H and LBP violated soft constraints as indicated by the

very high minimal energy values whereas ILP, LBP-LF, and LP-LP found a feasible solution.

Impact of non-optimal solutions on tracking performance

The different inference methods are evaluated in terms of their energy minimization capabilities. Besides ILP no method was able to find a global minimum. This raises the question of how the tracking results are affected by approximate inference. In practice LP-LP is the best alternative to ILP, since only LP-LP achieves a competitive energy in similar or less time. Therefore we limit our analysis to LP-LP in comparison to ILP.

The energy difference of the methods is 1938.4 A typical energy quantum associated with flipping a single random variable is around 20 to 200.[5] The approximate LP-LP solution should therefore differ in roughly 8 to 80 random variables. And indeed, the Hamming distance between the two solutions is 20—10 flipped detection variables and 10 flipped assignment variables each. All 20 variables have state 1 in the optimal ILP solution and state 0 in the approximate LP-LP solution. This means that in the LP-LP solution ten objects more are marked as false positives and ten assignments less are present in the tracking.

5.3.3. Tracking Performance

We evaluate the tracking performance of the chain graph model in comparison to other state-of-the-art methods on the segmented version of the Drosophila and zebrafish dataset (cf. Sec. 5.1 on page 41). The tracking performance is measured in terms of precision, recall, and F-measure as detailed in Sec. 5.2.3 on page 46. The parameters of all methods were optimized using exhaustive search as described in Sec. 4.5.1 on page 37. If not otherwise stated we used the basic chain graph model without the minimal cell cycle length extension (cf. Sec. 4.4.2 on page 36).

Zebrafish dataset

In a first experiment we compare two reasoners to solve the cell nuclei tracking-by-assignment problem for the zebrafish dataset: the baseline method employing an optimal joint assignment (Sec. 3.3 on page 19) and our proposed method based on a chain graphical model (Chap. 4 on page 21). For the zebrafish dataset we already reported tracking results using the optimal joint assignment method (Lou et al., 2011). The dataset is challenging since the segmentation does not only comprise cell nuclei but also false positives caused by speckle artifacts and inhomogeneous contrast. The results for the optimal joint assignment and the chain graph model are summarized in Table 5.3 on the facing page.

Drosophila dataset

In a second experiment we investigate the tracking performance of different methods including the chain graph model on the Drosophila dataset. No quantitative

[5] Assuming no violations of consistency constraints; the energy range can be estimated from typical object movements of 5 to 15 voxels by squaring the moving distance.

	Optimal Joint Assignment	Chain Graph
precision	0.807	**0.897**
recall	**0.861**	0.850
f-measure	0.833	**0.873**

Table 5.3.: Tracking results on the Zebrafish dataset. The chain graph model shows improved performance in terms of f-measure and precision compared to optimal joint assignment model.

	Detections Given	Unconditioned Chain Graph	Chain Graph with $\tau = 3$	Bise et al. (2011)
precision	0.889	0.953	**0.956**	0.550
recall	0.933	0.957	**0.960**	0.718
f-measure	0.911	0.955	**0.958**	0.623

Table 5.4.: Tracking results on the Drosophila dataset. The conditioned chain graph model with previously filtered false positives (*detections given*) is inferior to the full chain graph model optimizing detection and assignment variables at once (*unconditioned chain graph*). The extended chain graph with the condition that division events in a cell lineage must at least be three time slices apart from each other shows an improved performance over the unconditioned (or basic) chain graph model (*chain graph with* $\tau = 3$). Finally, the method of Bise et al. (2011) shows decent recall but suffers in precision.

tracking performance was reported for this dataset before. In particular, we compare the following approaches:

- the basic chain graphical model,
- a chain graph model with fixed detection variables,
- a chain graph model with four-state detection variables, satisfying a minimal duration of three time slices between division events of a particular track (see Sec. 4.4.2 on page 36),
- a state-of-the-art cell tracking method published by Bise et al. (2011).

Chain graph with fixed detection variables. The chain graphical model is a holistic tracking approach that considers all time slices at once. That way it can reason about the state of detection variables and assignment variables simultaneously. In a less complex approach we could first decide about the states of the detection variables and then—given the detection variables—optimize the assignment variables. To show possible performance improvements gained by the higher complexity

of the holistic model we therefore compare with a chain graph model with fixed (resp. given) detection random variables. The states of the detection variables are determined by thresholding the Random Forest predictions at 0.5 probability to set the variables to 'active' or 'inactive'. That is, we filter out all objects classified by the random forest as false positives before the tracking. Since the assignment CRFs in the chain graph are conditioned on prior factors over the detection variables, this approach can be justified as observing the detection variables and reasoning over the assignment variables given the detection variables without the need to refactorize the distribution—a conditioned chain graph. Note, that we use the very same probabilities to parametrize the factor in the full chain graph model. The difference lies only in the inference scheme but not in the amount or quality of input information.

Chain graph with four-state detection variables. In Sec. 4.4.2 on page 36 we describe an extension to the basic chain graph model that incorporates our prior knowledge about a minimal temporal distance between two divisions in a single cell lineage. Again, we want to show a possible performance gain at the cost of higher model complexity compared to other variants like the basic or conditioned chain graph. For this experiment we set $\tau = 3$, that is, we believe that the temporal distance between two divisions is at least three slices. The dependence of the performance on τ is investigated in another experiment described in Sec. 5.3.4.

Cell tracking method by Bise et al. (2011). Finally, we evaluate the cell tracking method recently proposed by Bise et al. (2011) on the Drosophila dataset to set the chain graph model variants in proper context with another state-of-the-art approach. This method is quite similar to the chain graph model and is therefore a suitable candidate for a comparison. It is also a probabilistic model with similar random variables and is optimized over all time slices at once, too. However, since it is not a graphical model its factorization is ad hoc and differs from the chain graph factorization.

The tracking results for all four methods are presented in Table 5.4 on the preceding page. A full synopsis of all lineage trees is reproduced in Appendix B. A comparison of some lineage trees obtained by manual tracking, the extended chain graph, and the method by Bise et al. (2011) is shown in Fig. 5.6 on the next page. The last method shows convincing results in regions with high data quality (cf. Fig. 5.6). However, its f-measure is 33 percentage points worse compared to the chain graph model. This is most likely caused by the high false positive detection rate of 13% as is evident from the low precision of 0.550 and Fig. 5.6.

5.3.4. Minimal Cell Cycle Length

Cell cycles happening during embryogenesis have a minimal duration. However, cell lineages obtained with the basic chain graph model sometimes include consecutive divisions that are in shorter temporal distance than is biologically plausible. To remove this error source we designed an extension of the basic chain graph model that also factors in a minimal temporal distance between two divisions in a

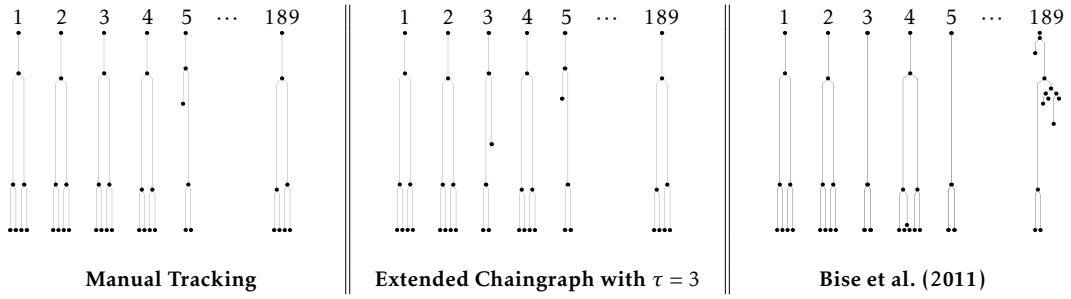


Figure 5.6.: Lineage trees for Drosophila spanning 25 time slices. Divisions are indicated by black dots. Time is running from top to bottom. Exemplary manually tracked lineage trees are compared to the extended chain graph model with $\tau = 3$ (that is, at least three time slices between two divisions in the same cell lineage) and the method by Bise et al. (2011). The examples include correct reconstruction (chain graph and Bise: 1, 2), false disappearances (chain graph: 3; Bise: 189), missed divisions (Bise: 3, 5), and false divisions (Bise: 4, 189).

single cell lineage (cf. Sec. 4.4.2 on page 36). In the experiment measuring general tracking performance (see Sec. 5.3.3 on page 52) we already compared the basic chain graph model with the extended variant. The extended variant showed the overall best performance in that experiment but only slightly better performance than the basic chain graph (f-measure of 0.955 vs 0.958). It is uncertain if the improved performance was caused by the extension or has to be attributed to chance. The experiment described in this section is a more detailed analysis of the extended chain graph's performance. We will investigate the performance of tracking division events separately from other events and control for the parameter τ which determines the minimal demanded temporal distance between divisions. The working dataset will be again Drosophila.

In the Drosophila dataset we can observe roughly twenty times more move events than division events. All events that contribute to the f-measure value have the same internal weighting and the overall performance is clearly dominated by the tracking method's ability to track move events correctly. Since the minimal cell cycle length extension was introduced to improve the tracking performance of division and not move events, we will measure a separate f-measure that only takes division events into account. That way the result will not be distorted by the move tracking performance and we can compare the basic chain graph with the extended variant solely in terms of the division tracking performance.

Furthermore, from biology we know that the minimal distance between two divisions in the same cell lineage in the Drosophila dataset is at least five time slices. We will show the dependence of the performance on the parameter τ by varying it between one and five (for $\tau = 1$ the model is equivalent to the basic chain graph and should show the very same performance).

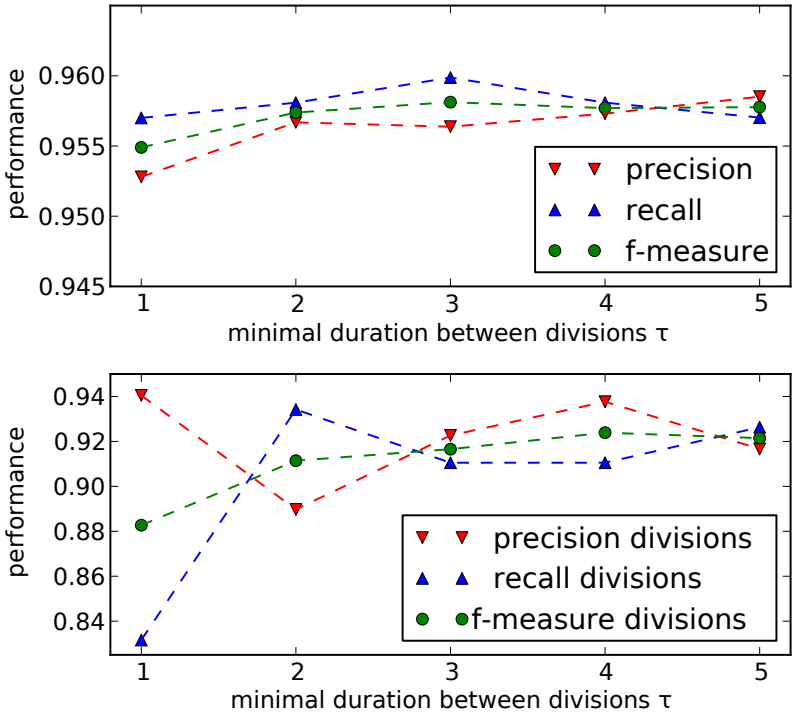


Figure 5.7.: Performance of the extended chain graph model which requires a minimal duration between division events. Performance results on the Drosophila dataset with varying τ are depicted. The upper panel shows the overall performance for all events (Moves, Divisions, Disappearances, and Appearances) and the lower panel shows the performance only for the division events.

Results

The basic chain graph exhibits a division performance of 0.941 precision, 0.832 recall, and 0.883 f-measure. The division performance for the extended model with $\tau = 3$ improves to 0.923 precision, 0.911 recall, and 0.917 F-measure. The dependence of the performance depending on τ is given in Fig. 5.7.

Furthermore, we measured the runtime impact of the extension and recorded the following values (in seconds): 100s ($\tau = 1$), 929s ($\tau = 2$), 1450s ($\tau = 3$), 1715s ($\tau = 4$), 3057 ($\tau = 5$) with a maximal allowed optimality gap of 5%.

5.3.5. Cell Nucleus Detection Performance

The chain graph is a holistic model that does tracking and false positive correction in one inference step. In other experiments we mostly investigated the tracking performance (which is of course influenced by the false positive correction performance). In this section we will concentrate on the performance in terms of the ability to find false positive objects. In particular, we compare three approaches with a manually generated ground truth on the Drosophila dataset:

	Classifier	Unconditioned Chain Graph	Chain Graph with $\tau = 3$	Human
true positive	12391	12330	12346	12493
true negative	1284	1607	1598	1810
false negative	102	163	147	-
false positive	526	203	212	-

Table 5.5.: Nucleus detection performance on Drosophila dataset. Compared to an object classifier the chain graph recovers less actual cells (*true positives*), but finds significantly more false positive segmentations (*true negatives*). The basic chain graph and the variant with a minimal cell cycle length don't differ significantly in detection performance (but in tracking performance: see Table 5.4 on page 53).

Discriminative Classifier: A Random Forest classifier is trained for the two classes *cell nucleus* and *other* and its predictions used to discriminate true cell nuclei from false positive objects. See Sec. 5.2.1 on page 44 for more details.

Basic Chain Graph Model: The detection variable priors are parametrized using the same Random Forest classifier. After the MAP inference the state of the detection variables is used to label cell nuclei vs. others.

Extended Chain Graph Model: The parametrization and procedure is identical to the basic chain graph model above. The τ parameter is set to three.

In total the segmented dataset contains 14303 object candidates of which 12493 are actual cell nuclei and 1810 are other objects. The detection performance in form of a contingency table is given in Table 5.5. It shows the results in terms of correctly and incorrectly identified cell nuclei for both variants of the chain graph in contrast to human annotations and the Random Forest classifier alone. Note, that the true/false positives/negatives refer to the detection performance itself and not the segmentation performance. That is, a *false positive segmentation* is either counted as a *true negative detection* (when correctly identified as a false positive object) or a *false positive detection* (when incorrectly identified as a cell nucleus).

5.4. Summary

In this chapter we presented several experiments to evaluate the proposed chain graphical model for tracking-by-assignment and the digital embryo recording

pipeline from different angles. Besides the pure tracking performance we also investigated the impact of different inference methods and documented experimental results concerned with object detection and segmentation. In Chap. 7 we will bring these results together in a holistic discussion.

6. Software

The methods developed in the thesis at hand are made available to the public in form of carefully crafted open source software packages to let other scientists make the most effective use of our work. The hypotheses graph concepts (Sec. 3.1 on page 15) together with the three reasoners *joint optimal assignment*, the method by Bise et al. (2011), and the *chain graph* (Ch. 3 and 4) are implemented in a C++ library called pgmLink. It enables the modeling of tracking-by-assignment problems and their solution with the provided or user supplied reasoners. Building upon pgmLink we implemented the second step of the two-step digital embryo recording pipeline (Ch. 2) as a tracking workflow with an easy to use graphical user interface in ilastik—the interactive learning and segmentation toolkit[1]. It constitutes the first freely available software[2] with a graphical user interface for the automatic tracking of many divisible objects in 3d+t data and is already in use for (cell-)tracking applications other than the ones presented in the thesis at hand. (Hand et al. (2009) and Meijering et al. (2009) give an extensive overview of other free and commercial cell tracking software.) Furthermore, we extended ilastik with a novel Python library called volumina. This library enables the display of and the interaction with the presented 3d+t data sets, which are too large to fit into the main memory of a typical workstation. It is now a core component of ilastik and used in many other ilastik workflows related and unrelated to our work.

The pgmLink library can be obtained from `https://github.com/bekaus/pgmlink` and the volumina library from `https://github.com/ilastik/volumina` . The tracking workflow is part of the official ilastik distribution. It can be found as source code at `https://github.com/ilastik/ilastik` or as a binary package at `http://ilastik.org` .

About ilastik

ilastik is developed in an collaborative effort by researchers world-wide. The project was initiated by Christoph Sommer and Prof. Fred Hamprecht at the Heidelberg Collaboratory for Image Processing (HCI). Its first official release was version 0.5 in 2011 (Sommer et al., 2011). The software is geared toward an audience from the life and material sciences and offers several image analysis workflows like pixel classification or interactive watershed segmentation. ilastik can be easily extended with other kinds of workflows—as we did in the work at hand—and software developers can leverage synergistic effects by sharing software components between workflows. The strong points of ilastik compared to other

[1] `http://ilastik.org`

[2] to our best knowledge

image analysis tools are manifold. It supports larger than main memory data sets employing streaming techniques. Data sets can have up to three spatial dimensions spanning several time slices and channels. The workflows can be interactively parametrized by the user employing active learning approaches and the need to specify opaque parameters is alleviated.

6.1. pgmLink: Tracking-by-Assignment Library

The pgmLink library is about tracking-by-assignment with probabilistic graphical models and other approaches. It is written in C++. The central data structures of the library are the `HypothesesGraph`, the `Traxel1Store`, and several implementations of the `Reasoner` abstract base class. Furthermore, there are several builder classes and factory functions that assist the user in constructing the central data structures. The graphical model used in the *chain graph* reasoner is implemented with the openGm library(Andres et al., 2012a). It provides a convenient data structure for factor graphs together with a comprehensive collection of MAP inference methods and enables effortless experimentation with different inference schemes for the chain graphical model.

Below we give an overview of the pgmLink high-level API. The mentioned classes are also wrapped for `Python`. The library can therefore be used from scripts—as we did to conduct the tracking experiments described in the thesis at hand—or by applications like ilastik that are written in `Python`.

Tracking

Tracking in pgmLink is conducted by a reasoner that labels the nodes and edges of a hypotheses graph as active or inactive (cf. Sec. 3.1 on page 15). The labeled HypothesesGraph is the output of the tracking. It can be converted to several other tracking-result formats with helper functions. Every reasoner implements three methods `formulate(const HypothesesGraph&)`, `infer()`, and `conclude(HypothesesGraph&)`. The first method allows the reasoner to build up its internal solving model, the second method executes the inference process, and the last method labels the hypotheses graph according to the results of the last inference. By splitting the process up into three steps, the user has a fine control over the tracking process. For instance, it is possible to rerun the inference with different parameters, storing each result separately, and—at the same time—avoid the rebuild of the internal solving model.

Constructing hypotheses graphs

A `HypothesesGraph` can be constructed from a collection of `Traxel` objects. Each traxel object has spatio-temporal coordinates and optional other features like volume, surface etc. They are stored in a `Traxel1Store` which allows filtering by features or field-of-view and serves as an input to a hypotheses graph builder class. A typical builder constructs a hypotheses graph from a traxel store according to some user-specified parameters like the maximum number of nearest-neighbor

traxels to consider but can also use more advanced heuristics like adding division hypotheses only when indicated by some local characteristic. In fact, a simple greedy nearest-neighbor tracking can be implemented by the builder alone without the need for a reasoner. That way, the builder is an important preprocessing tool to make the tracking problem tractable for the reasoners.

Code example

The following code example illustrates the central steps to track with `pgmLink`.

```
// construct a hypotheses builder given
// a traxelstore and some options
HypothesesBuilder hyp_builder(&traxelstore, builder_opts);

// construct the hypotheses graph
HypothesesGraph* graph = hyp_builder.build();

// configure a chaingraph model
// and construct a chaingraph reasoner
chaingraph::ModelBuilder b =
  chaingraph::ModelBuilder(appearance_energy,
                           disappearance_energy,
                           move_energy,
                           opportunity_cost,
                           forbidden_cost)
  .with_divisions(division_energy)
  .with_detection_vars(detection_energy,
                       misdetection_energy)
  ;

Reasoner* r = new Chaingraph( b );

// track
r->formulate( graph ); // build up graphical model
r->infer(); // graphical model inference
r->conclude( graph ); // label hypotheses graph
```

Note the special builder pattern which is used to parametrize the `ModelBuilder`. A library implementing data analysis methods usually has to handle a myriad of parameters leading to constructors or free functions with many calling arguments. Such code is difficult to read and can easily cause programming errors (think of 15 indistinguishable `double` arguments). Builder classes with method chaining help to avoid these problems. The calling order of the chained methods is irrelevant, the number of calling arguments is reined in, and their meaning is verbosely encoded in the name of the chained method. Furthermore, builder objects are light-weight (they are only a collection of parameters) and can be passed upstream in the class

hierarchy efficiently to allow the user a clear parametrization of all components of the algorithm.

6.2. volumina: Volume Slicing and Editing Library

The `volumina` library is concerned with displaying slice views of high dimensional image data and allowing user interactions with the slice views. It is written in Python and makes heavy use of the Qt graphical user interface library.[3] volumina implements the following features:

- handling of data larger than main memory employing tile-based streaming
- support for 3d+t volumes with multiple channels
- display of several volumes simultaneously rendered as a stack of layers
- slice views of any axes pair like x-y or z-time
- interaction modes like brushing labels or selecting objects
- ready-to-use viewer widgets

The library is structured into a low level and high level part. The low level part consists of a collection of loosely coupled components which can be combined with each other to design a variety of systems for slicing and edition multidimensional volumetric data. Besides others the central low level components are the pixel rendering pipeline and the interaction modes. The design of these components is governed by the Observer and Model-View-Controller design patterns. The high level part builds upon the low level components and provides ready-to-use classes and widgets for the end user; with the *volume editor* being the most important one.

6.2.1. Central Design Patterns: Observer and Model-View-Controller

The low level classes of the library are interacting with each other using the Observer pattern (Gamma et al., 1994). In Observer there are two types of classes: one observable and zero or more observers. Observers can be registered and deregistered at the observable at any time. The observable is exposing parts of its internal state to the observers and notifies them about changes. This is especially useful in an application where users can modify the program state in any order at any time. In practice, the pattern is implemented employing the signaling and event mechanisms provided by Qt. They promote loose coupling between the classes, in particular a frictionless passing of state updates between otherwise unrelated objects.

Class responsibilities are distributed according to the Model-View-Controller(MVC) pattern (Freeman et al., 2004). It separates data, data representation, and user

[3] `https://qt-project.org/`

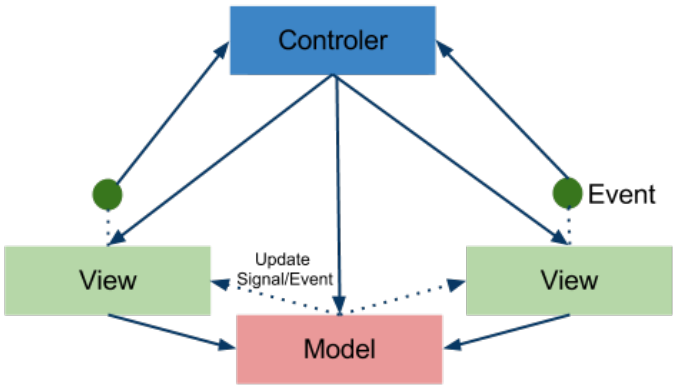


Figure 6.1.: The Model-View-Controller (MVC) design pattern. Controller (blue), View (green), and Model (red) are interacting with each other trough the Observer pattern. Observer relationships are indicated with arrows pointing from the observer to the observable.

interactions with the data from each other and builds upon Observer. Fig. 6.1 shows a schematic of MVC, laying out the observable–observer relationships. The Model encapsulates state and notifies its observables about state changes. In VOLUMINA there are several models that, for example, hold information about slicing and cursor position or the current state of the brushing tool. The model is observed by zero or more views, that is, components that represent the model state in a human readable form. Typical examples for views are graphical user interface (GUI) widgets like list views or image canvases. Finally, the controller serves as an entry point for user generated mouse and/or keyboard events (the events can also be machine generated, for example network communication events). The controller interprets the events and updates the model and/or views according to the user's intent. (Note, that subsequently the model notifies the views about the state change and the views update themselves accordingly.) The prime example is a left click event that can be interpreted in many different ways, one being a request to change the slicing position or another being the intent to select a clicked object.

Overall, the flexible architecture based on MVC allows to react quickly to new requirements and new interaction modes or layer rendering modalities can be added easily without rewriting already existing code.

6.2.2. Pixel Rendering Pipeline

The pixel rendering pipeline or short *pixel pipeline* renders 2d bitmap images from several multidimensional scalar data sources. It can produces several kinds of images, amongst others: grayscale, rgba, alpha-modulated, and colortable images. Each image type needs up to four scalar data sources as input, for example one

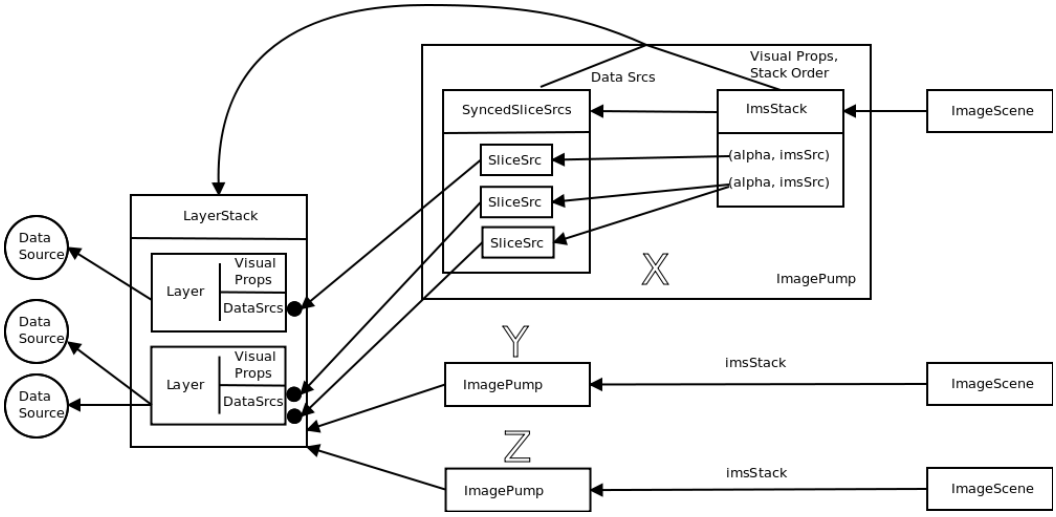


Figure 6.2.: Pixel pipeline architecture.

source for each channel (red, green, blue, and alpha) in case of a rgba image.

Furthermore, these images can be combined into a stack using alpha blending with an image of the rendered stack as output. A schematic of the pixel pipeline architecture is shown in Fig. 6.2. The visual properties of one image are contained in a *layer* object and a ordered stack of layers is organized in a *LayerStack*. They serve as a rendering blueprint used in *image pumps*. A image pump takes the blueprint together with the data sources and renders a 2d image. Internally, a image pump consists of subcomponents that take care of slicing through the correct axes of the multidimensional data sources and the blending of several layers into a stack image. Furthermore, several image pumps can be connected to the same layer stack and data sources allowing to render different slice views simultaneously.

The pixel pipeline is a lazy data flow processing pipeline, that is, subregions of images can be requested from the pipeline and only the calculations necessary to complete the request are actually performed.

6.2.3. Interaction Modes

Another central low level component are the *interaction modes* like navigation, brushing, or object selection. They are analog to the different editing tools available in pixel graphics programs like Gimp or PhotoShop. Interaction modes are implemented as controllers that interpret incoming user events. For example, the left click is interpreted as "recenter the cursor" in navigation mode but as "start to paint" in brushing mode.

Many single interactions are caused by a stream of user input events, for instance painting a single label involves a mouse click event, several mouse move events, and a mouse release event. volumina can switch between different interaction modes at any time. To prevent inconsistent states when switching between modes in the

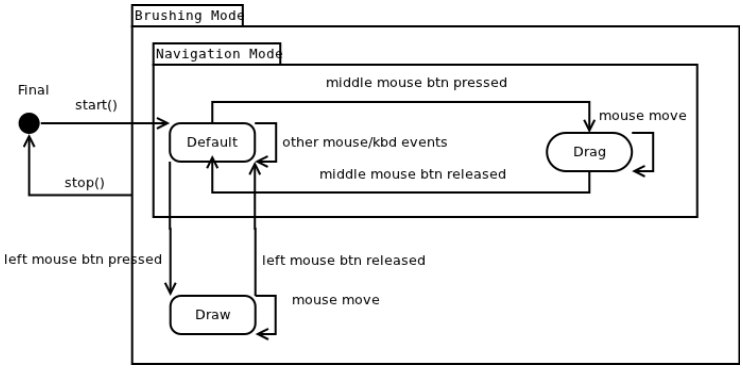


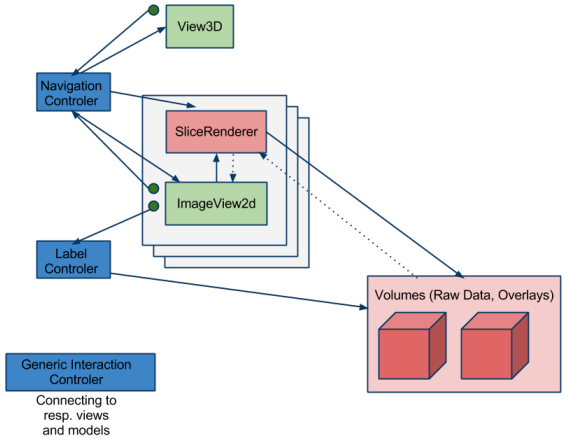
Figure 6.3.: The brushing interaction mode implemented as a state machine.

middle of an interaction and to manage the overall complexity, the interaction mode controllers are implemented in form of *state machines*. Fig. 6.3 shows the flow diagram of the brushing mode state machine as an example. A state machine is represented in form of a directed graph. For each distinct state a node is added to the graph. Possible transitions between states are encoded as edges. A state machine processes a stream of tokens (in our case, input events) that can cause state transitions if certain conditions are met. To start and end the processing special entry and exist nodes are used.

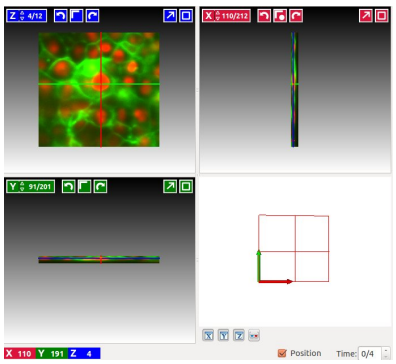
State machines can make use of other state machines as subprocessing units. For example, the brushing mode shares many interactions with the navigation mode (like the basic keyboard shortcuts for navigation). Therefore, the brushing mode controller internally uses a navigation mode controller for the processing of most events and only transitions into special states to process events directly concerned with brushing. This design promotes code reuse and avoids defects caused by code duplication issues.

6.2.4. Volume Editor Component

The central high-level component of the library is the volume editor. It is implemented in terms of the low level classes and exposes all the features listed in the introduction of this section in a single widget which can be easily incorporated in application code. Fig. 6.4a gives an overview of its architecture and Fig. 6.4b shows the widget as it appears to the end user. Other widgets are also included like a viewer for 2d images and a layer widget that allows to control the order and appearance of different layers in the slice views. Furthermore, a standalone viewer application is part of VOLUMINA, too. The viewer can be used in a Python command prompt to inspect multidimensional datasets and is launched with a simple `view(dataset)` call.



(a) Volume editor: high level architecture. Observer–observable relations are indicated by arrows. Models are red, views green, and controllers blue (as defined in the MVC-pattern). The small green dots represent user input events generated with mouse or keyboard.



(b) Volume editor: widget. The widget shows three slicings trough the spatial axes of a three dimensional dataset with two channels rendered as a red-green composite image at time slice 0 of 4.

Figure 6.4.: The volume editor is the most important component of the `volumina` library.

Code example

The following code example illustrates the central steps to display multidimensional data in several layers with `volumina`.

```
# construct some data sources from raw data
source = ArraySource(npy_array)
rgb    = ArraySource(npy_rgbarray)

# define layers and store them in a layerstack
layerstack = LayerStackModel()

layer1 = GrayscaleLayer( source )
layer1.name = "Raw Data"
layerstack.append(layer1)

layer2 = RGBALayer(red=rgb[..., 0],
                   green=rgb[..., 1],
                   alpha=rgb[..., 2])
layer2.name = "RGB Layer"
layer2.opacity = 0.5
layerstack.append(layer1)
```

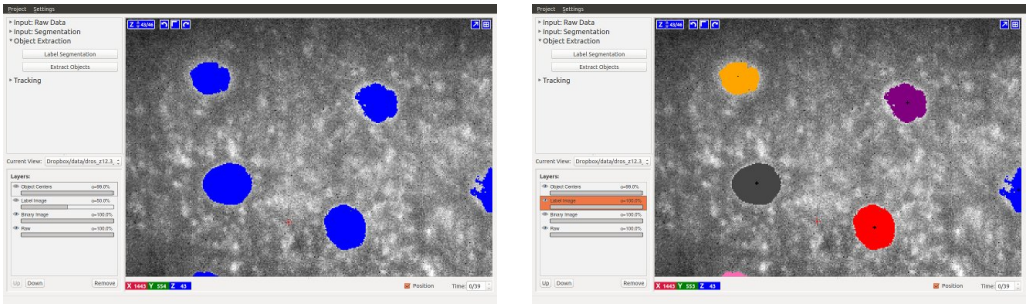


Figure 6.5.: Object identification and feature extraction step of the tracking workflow. The left screenshot shows a zoomed-in detail of the Drosophila dataset. The segmentation of the cell nuclei is overlayed in blue. The right screenshot shows the result of object identification and feature extraction. Objects are assigned an individual color and their centers are marked by small black crosses (3d crosses in a six voxel neighborhood).

```
# instantiate a volume editor and display the layers
self.editor = VolumeEditor(shape, layerstack)
self.widget = VolumeEditorWidget( self.editor )
self.widget.show()
```

6.3. ILASTIK Tracking Workflow

The ILASTIK tracking workflow allows to track objects given a foreground-background segmentation and raw data as input. The workflow consists of three steps which are executed by the user in consecutive order. First, individual objects have to be identified in the segmentation using connected component analysis. Second, features like the object centers have to be extracted. Third, based on the extracted features the objects are tracked with PGMLINK as the underlying engine. Finally, the user can export the result in different formats. For the thesis at hand the input segmentation is generated with the ILASTIK pixel classification workflow, which works well for the presented datasets as shown in Sec. 5.3.1 on page 48. Of course, the input can also be obtained from any other method that generates a binary segmentation.

At any step, the user gets visual feedback and can control the intermediate results. If necessary the user can then tweak parameters to improve the results. The object identification and feature extraction step inside the ILASTIK application is shown in Fig. 6.5. (Note, that only the z-slice view is shown to make the illustration clearer.) Object centers can be described by several measures like the center of (intensity) mass, the maximum intensity position or the geometrical mean. By visually inspecting the centers the user can quickly assess which descriptor fits best

to the given objects.[4]

In the tracking step the user can parametrize the tracking method itself and set necessary meta information like resolution anisotropy factors of the input data. Furthermore the user can select a spatio-temporal bounding box to track only in a subset of the data leading to a reduced inference time. The latter is useful to tune the algorithm quickly before applying it to the whole dataset spending a potentially large time on inference.

The tracking step is depicted in Fig. 6.6 on the facing page. The tracking result is presented to the user in form of a consistent coloring over time. One unique color is given to every object in the first time slice. Later, siblings originating from the same object share the color of their ancestor. The Figure shows four screenshots in total. Two screenshots from each of two consecutive division cycles, one shortly before and one shortly after the division event.[5] Groups of four nuclei sharing the same color are clearly visible in the last screenshots indicating the common ancestor with the same color in the first screenshots. Such a coloring can be seen as a digital *fate map*. Fate mapping is an important method in developmental biology to understand the origin of various tissues by chemically staining cells earlier in the development (Bildsoe et al., 2007). Compared to the traditional approach that has to decide about the time point and select the cells to be stained before the experiment, digital fate maps have the advantage to be produced after the experiment. This allows the temporal origin of the map to be shifted to any desired time slice, selecting any cells of interest.

6.4. Summary

The thesis at hand led to the development of three software packages that enable the application and evaluation of the presented methods. pgmLink implements the tracking methods and—amongst others—the hypotheses graph data structure as described in previous chapters. volumina enables the display of and interaction with the 3d+t datasets which are typically too large to fit into RAM. Finally, both packages are integrated into ilastik as a tracking workflow making the methods accessible to a broad audience.

[4] Well segmented cell nuclei in high-quality images are usually best tracked with the center-of-mass descriptor. In case of low signal-to-noise and/or insufficient shape segmentation the other features can be a more robust estimate of the nuclei centers.

[5] Divisions happen almost synchronously for all cell in the shown stages of the dataset.

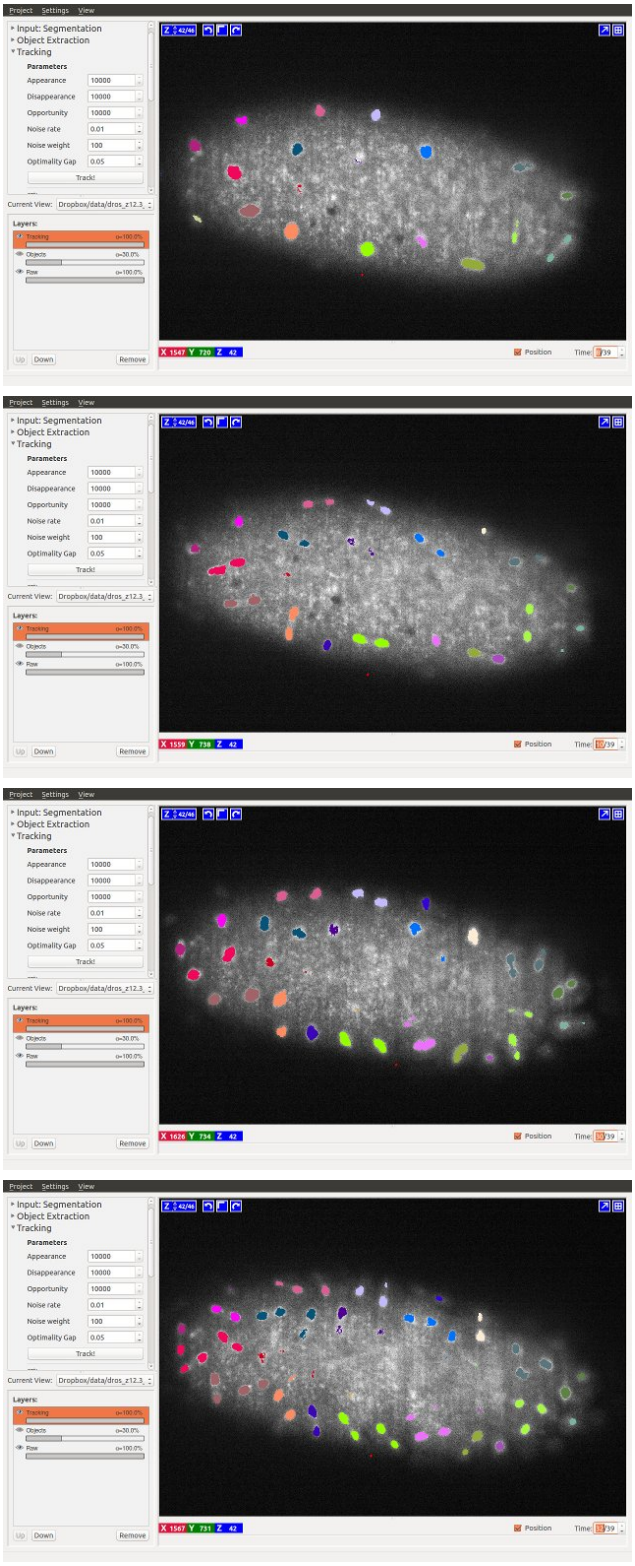


Figure 6.6.: Tracking step of the tracking workflow. The four screenshots show a successful tracking of the Drosophila dataset at four different browsing positions. The sequence displays time slices 8, 10, 30, and 32 (all at $z = 42$).

7. Discussion

In previous chapters we presented our methods to record digital embryos (Chap. 2, 3, 4, 6) and conducted several experiments (Chap. 5) to investigate different aspects of the approach. In this chapter we will interrelate both to substantiate our claim that the chain graph model in conjunction with the digital embryo recording pipeline is a viable improvement over the state-of-the-art in tracking-by-assignment in general and cell lineage reconstruction in particular.

7.1. A Holistic Model Over All Time Slices Helps Tracking

In this section we will show that a global model like the chain graph that is optimized over all time slices at once is superior compared to an approach that acts only on pairs of time slices. The two main advantages are:

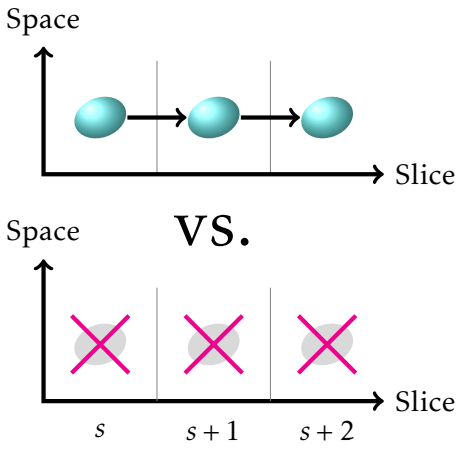
1. the ability to reason about track properties over several time slices
2. the ability to weigh local evidence against the overall likelihood of a track.

Furthermore, we argue that (temporal) long-range effects can be achieved by local factors only and consider advantages and disadvantages of incorporating higher order factors spanning several time slices in a chain graph model.

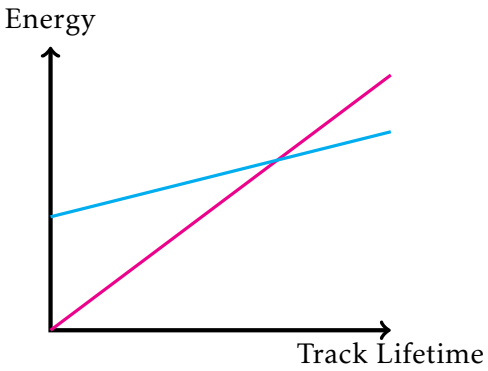
7.1.1. Encoding Long-Range Effects Locally

The MAP solution of the chain graph model is obtained from all time slices at once allowing the model to consider track properties over several time slices. This may sound counterintuitive since the factors in the chain graph model are only defined over variables from at most two consecutive time slices and do not consider tracks as a whole. For instance, in contrast to actual cell nuclei, false positive detections tend to suddenly appear and only exist for a comparatively short time span since they are mostly caused by noise clusters in the raw data. To mitigate this issue we could add higher order factors to the model to encode our belief of the expected lifetime of a track. We now show that the model in its current form already considers the above situation without the need to add any higher order factors.

Consider a highly idealized situation with only one detection per time slice, a small number of time slices n, and a fixed energy cost for all tracking events as defined in the following table:



(a) Two explanations for the same detections: either all true positives (in blue) or all false positives (in red).



(b) Schematic of the energy costs for the two explanations (same color coding as on the left).

Figure 7.1.: Local costs determine wide-range decisions. A series of detections is explained either as all true positives or all false positives (left). For short lifetimes the latter explanation is more likely in the model. To be considered a temporal sequence of true objects the track has to reach a certain minimal lifetime to exhibit less energy costs than the other explanation. The costs for both explanations are linear functions of the lifetime but with different slope and intercept (right).

Event	Energy
Appearance	25
Disappearance	25
Move	5
True Positive Detection	7
False Positive Detection	30
Opportunity	0

In case of $n = 1$ there is only one decision to be made: is the one detection a true positive or a false positive? The former has to be explained in terms of an appearance (25 energy units), a disappearance (25 energy units), and a true positive detection event (7 energy units) and has a total energy cost of 57. The latter can be explained in terms of just one false positive detection event and we only need to expend 30 energy units. We therefore explain the one detection as being a false positive. Now compare that with $n = 4$. The two most likely explanations are that all four detections are either the same object existing during four time slices or are all false positives. For the former explanation we would now need to expend in total 93 energy units (one appearance, four true positive detections, three moves, and one disappearance) and for the later 120 energy units (four false positive

detections). In this case we would pick the other explanation that the detections constitute a true track. The two possible decisions are illustrated in Fig. 7.1a on the preceding page.

In general, the energy costs for the false positive explanation are a linear function of n with a slope equal to the false positive detection energy and an intercept of zero. The true positive explanation costs are also a linear function of n with a less steep slope equal to the sum of true positive detection and move energy but a positive intercept equal to the sum of appearance and disappearance energy. That is, a object or track needs a minimal lifespan to overcome the track initialization and termination costs and the costs for the alternative explanation of being all false positives. Fig. 7.1b on the facing page illustrates the idea. To conclude, this effect is exploited in the chain graph model to reason about minimal track lengths given evidence encoded in the energy values.

Another more obvious way to consider wide-range effects with local factors is the minimal cell cycle length extension. It controls the minimal temporal distance between divisions—not directly with higher order factors but indirectly using the counting trick (see Sec. 4.4.2 on page 36)—and is another example of how the chain graph model can reason globally using local factors..

We argue that local factors are the better choice compared to global factors because they allow to build up a model from the same basic elements for any data input. In particular, time slices can be easily added to the chain graph by just repeating the same construction used for all other slices. Higher order factors over time would involve newly added variables for the new slice and already added variables from older slices and some factors could only be added after the model has reached a certain size. This would complicate both the theoretical formulation of the model and its implementation in software, increasing the chance of errors. However, when using a linear programming based inference method global factors can nevertheless be useful. There might be some constraints (i.e. zero probability configurations) that are only rarely violated in a typical solution. Such cases can be handled efficiently with an iterative *cutting planes* (Kelley, 1960) approach for which a few global constraints make more sense than many local ones (again, for simplicity reasons but also because of possible performance gains).

7.1.2. Weighing Local and Wide-Range Evidence Against Each Other

A full chain graph model implements local factors to influence wide-range decisions as described in the previous section. This mechanism is combined with local-evidence dependent energies and consistency constraints to reason about tracks spanning multiple time slices (in the previous section we assumed fixed energy costs).

A consequence of the wide-range reasoning is the possibility to override local evidence encoded in the detection variable priors (the following points are illustrated by Fig. 7.2 on the following page showing some exemplary false and true positive detections):

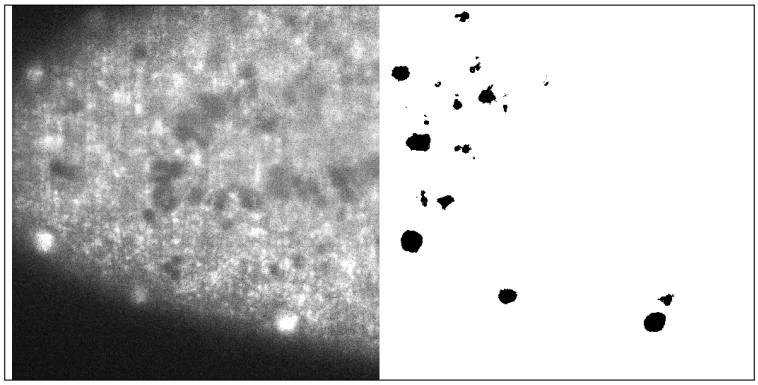


Figure 7.2.: Drosophila dataset: synopsis of raw data and corresponding segmentation; shown is a detail of stack slice 33 at time slice 7. The image illustrates that false and true positives are hard to distinguish by only looking at object features in a single time slice.

1. On the one hand objects that exhibit a low likelihood of being a true positive can nevertheless be activated if they increase the overall likelihood of the track in which they are integrated.

2. On the other hand false positive detections can deceptively look like true positives due to mere chance (smoothly shaped ellipsoid in case of cell nuclei) but if they are not part of a track with a certain lifetime, they will nevertheless be deactivated.[1]

The described behavior is evident in the tracking results. Fig. 7.3 on the next page shows a visualization of the chain graph tracking in the Drosophila dataset (cf. Sec. 5.3.3 on page 52) for eight consecutive time slices with a clipped data frame. We choose this sequence because it contains both false positive detections that have a shape similar to cell nuclei and true positive nuclei whose shapes were distorted in the segmentation step. The model tries hard to make sense out of the noisy upper part of the sequence which is worst around time slice 10 and bridges it in a meaningful manner. That is, only tracks that are well grounded in higher quality regions are interpolated through the noise. No additional phantom tracks are caused by the noise objects and quite the contrary most noise objects are filtered out.

In contrast, Fig. 7.4 on page 76 illustrates the behavior of the *joint optimal assignment* method (Sec. 3.3 on page 19) in presence of false positive detections. The method cannot distinguish between false and true positive detections and erodes the quality of the tracks mostly by claiming non-existing division events

[1] This is analogous to the strategy a human expert may use to obtain a manual tracking. Tracks are counterchecked against the visual appearance of the segmented objects by browsing back and forth in time before arriving at a decision that the human expert considers the most likely explanation of the data.

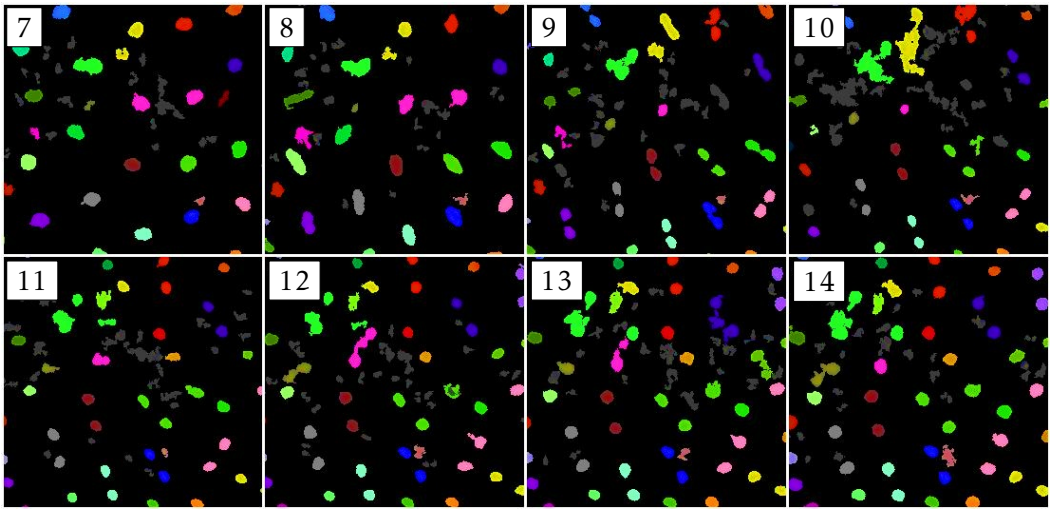


Figure 7.3.: Sample tracking sequence obtained with the chain graph model from the Drosophila dataset. The eight consecutive time slices show a detail of the data projected to 2d. The colored spots are segmented objects and objects sharing the same color are assigned to each other. Dark gray objects are misdetections as indicated by 'inactive' detection variables. Ill-shaped objects will be treated as active if they are bridging two tracks in a sensible manner. On the contrary, cell-shaped objects may be labeled inactive, if they do not constitute a track of a certain length.

to explain the suddenly appearing clutter detections. In terms of tracking true positive objects the method is on par with the chain graph, though. This is no wonder because optimal joint assignment can be seen as a special case of the chain graph model where the true positive-probability of all detections is equal to one. In quantitative terms, we expect a similar recall for both methods since recall is influenced by the ability to track true positive objects and a higher precision for the chain graph because an increased precision indicates a better handling of false positive detections. In fact, the above interpretation is supported by the results: on the zebrafish dataset the chain graph loses only 1.1 percentage points (pp) in recall but gains 9pp in precision compared to *optimal joint assignment* leading to a overall better tracking performance (see Tab. 5.3 on page 53). In practice this makes all the difference because the remaining errors are not distributed equally over the data range but are mostly located in regions with particularly low data quality. In the other regions (which constitute the majority) the tracking is very reliable and no longer distorted by the noise detections as with optimal joint assignment.

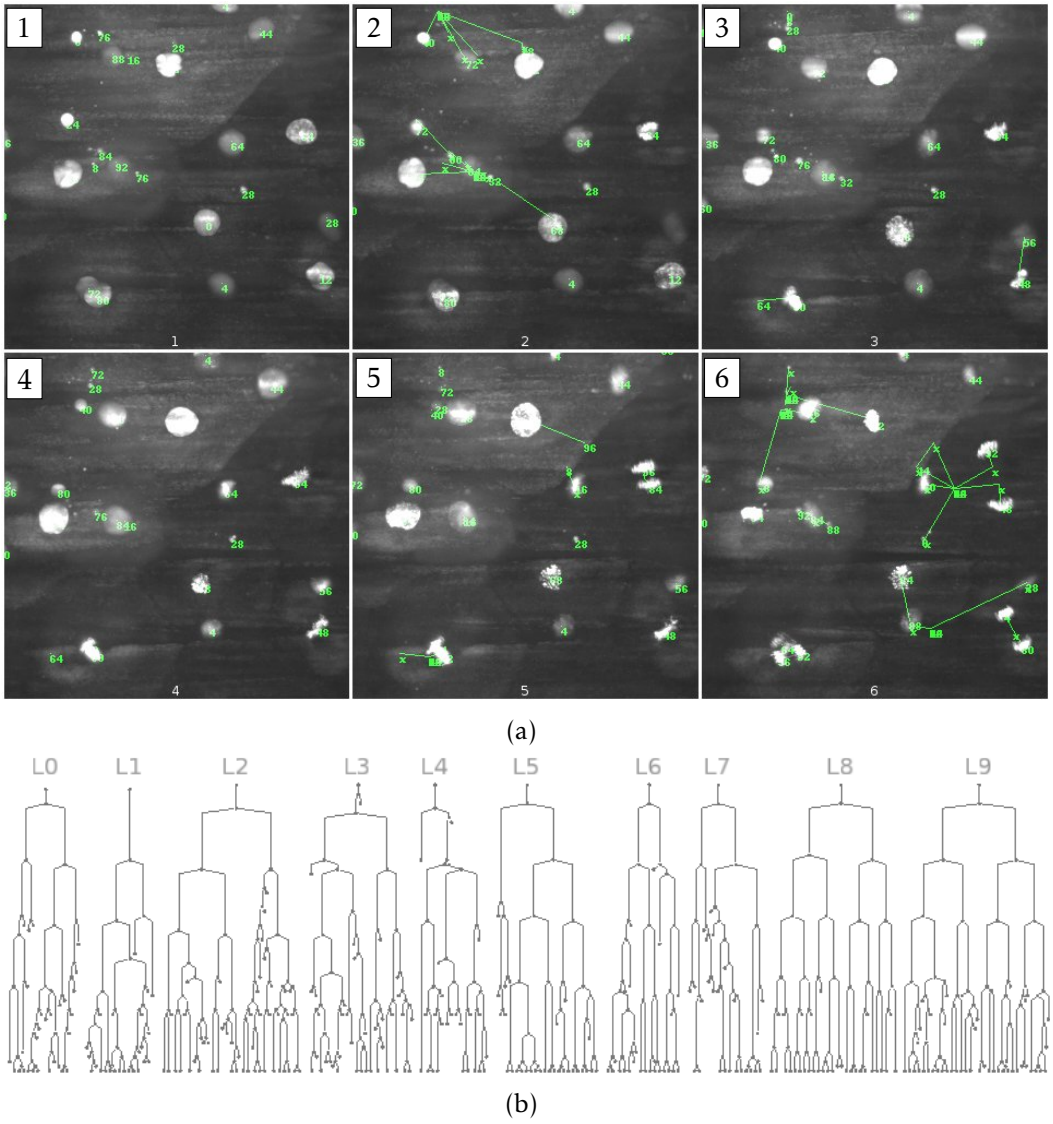


Figure 7.4.: Tracking with the *optimal joint assignment* method in the zebrafish dataset. The panel (a) shows a clipping of the full tracking result for six consecutive time slices. Centers of tracked objects are marked with their associated numeric track id. Division origins are indicated with *x* markers and the translation vectors of the children with green lines. The images are 2d maximum intensity projections of the 3d data. Panel (b) shows some representative lineage trees based on the tracking result (time running from top to bottom; divisions indicated by round dots). The trees are distorted by many erroneous divisions caused by false positive detections.

The reader may argue that this is an unfair comparison since the performance of *optimal joint assignment* can certainly be improved by using the Random Forest predictions (which are used in the chain graph model) to filter out false positives in a preprocessing step. In particular, it is entirely possible that the increased performance must be attributed mostly to the powerful predictions of the non-linear classifier and not to the global model structure. But the results prove otherwise. Table 5.4 on page 53 clearly shows a significant performance increase of the full chain graph model, which weighs in the Random Forest predictions during inference, in contrast to filtering false positives in a preprocessing step and treating the detection variables as given. Compared to the chain graph with previously fixed detection variables, the full (unconditioned) model gains 2.4pp and 4pp in terms of recall and precision, respectively. Since both approaches are otherwise identical, the performance increase has to be caused by the long-range effects which can be only considered in a holistic model like the chain graph (as explained above).

7.2. The Chain Graph Model is a Segmentation Regularizer

In this section we shift the focus from seeing the chain graph model as a tracking method to viewing it as temporal regularizer for a local prediction-based segmentation. A popular application of graphical models in image analysis acting as regularizers is foreground–background segmentation. The prime example is a Markov random field taking the form of a regular grid with one binary variable per pixel (Boykov and Funka-Lea, 2006). Two types of factors are added: single-site potentials over one variable each encoding the likelihood of a pixel being foreground or background and second order potentials between neighboring variables resp. pixels imposing spatial smoothness as exhibited in natural images. The MRF therefore acts as a regularizer of the pixel-wise predictions.

Interpreting the chain graph model in an analog manner, the chain graph detection priors are single site potentials trying to include only true objects in the final segmentation whereas the remaining factors represent the smoothing potentials. The difference is that the smoothing operates along the temporal dimension, that is, segmented objects should be present in all time slices in temporal vicinity as expected in a natural time lapse sequence.

Table 5.5 on page 57 summarizes the results of the experiment investigating the nucleus detection performance of the Random Forest classifier alone compared with the chain graph (in the Drosophila dataset). Compared to the classifier the (basic) chain graph retrieves 61 less cells (less true positives), but labels 323 more noise detections correctly (more true negatives). (The chain graph also controlling the minimal cell cycle length exposes the same behavior with nonsignificant differences in performance.) This supports the interpretation that the chain graph regularizes the single-site detections. It can prevent many wrong detections from entering the final segmentation at a small decline in the true positive rate.

This effect is related to the precision-recall trade-off in tracking performance of the chain graph compared with optimal joint assignment (see Tab. 5.3 on page 53). Since correct cell identifications are a precondition for a successful recovery of move, division, appearance, and disappearance events, the higher object identification rate should directly transfer to the tracking performance. Indeed we see a slight decrease in recall of tracking events that could indirectly be caused by the decrease in *detection* recall, whereas the significant gain in tracking precision is certainly connected with the increased rate of correctly identified noise detections.

In summary, the chain graph model can even add value when the application target is not tracking but segmentation since it acts as a temporal smoother analog to the above MRF that acts as a spatial smoother. Furthermore it could be an interesting direction for future research to try a combined tracking and segmentation directly at the level of single pixels. One way to achieve that could be to combine the mentioned MRF with the chain graph in a single model.

7.3. The Cell Cycle Length Extension is a Trade-Off

In Sec. 4.4.2 on page 36 we describe an extension of the basic chain graph model that controls the minimal temporal distance between two divisions. The desired minimal distance can be set with the parameter τ. As we showed in the previous sections it is comparatively inexpensive in terms of energy to mark tracks as inactive that are either very short or consist mainly of objects with a low detection probability. But it can be even cheaper to link a short noise track with a nucleus track using a division event if the noise objects are in close vicinity of true cells. The goal of the extension is to make such wrong division more unlikely.

On the Drosophila dataset the extension causes an increase in f-measure by 0.3 percentage points for a minimal temporal distance of three ($\tau = 3$) compared to the basic chain graph model (see Tab. 5.4). At a first glance the gain seems insignificant. But one has to take into consideration that the f-measure puts the same importance on every type of tracking event. However, the number of move events exceeds the number of division events by a factor of 20. The f-measure is therefore dominated by the move events (appearance and disappearance events are very rare). To bypass this issue we will only consider the performance calculated on division events alone (see Sec. 5.3.4 on page 54).

In that regard the f-measure for divisions improves from 0.883 (precision 0.941, recall 0.832) to 0.917 (precision 0.923, recall 0.911) for $\tau = 3$. This is a significant gain and very important for an accurate assessment of cell ancestry since a single mistracked division can spoil the whole lineage downstream in time. In Fig. 5.7 on page 56 we plot the dependence of the performance on τ and it turns out that the f-measure stays roughly the same for $\tau \geq 3$. This may be surprising since the average number of time slices between divisions in the Drosophila dataset is around ten and one would expect an ever increasing performance up to $\tau = 10$. This can be explained as follows since the extension operates symmetrically in time. Assume

that a track divides at time slice $s = 0$ and then again at $s = 9$ and the model tracks both divisions correctly. Then the extension prevents divisions of the same track at time slices 0, 1, 2 , 7, 8, and 9, blocking already 60% of all time slices and making erroneous divisions quite unlikely. (They have to happen in the right time window and must have a low energy compared to other explanations of the data.)

Unfortunately, the extension is no free lunch since inference time increases at least by a factor of ten compared to the basic model on the presented datasets. Furthermore, as a direct consequence the maximal number of objects and time slices the model can handle is also reduced since in practice inference time increases exponentially in the size of the input. The reason is of course the enlarged state space that is caused by the increased number of states of the detection variables. In practice, the extension should therefore only be used when the problem size at hand still permits inference in reasonable time and/or the utmost achievable tracking quality is needed.

In summary, the extension is a trade-off between inference time and tracking accuracy. On a higher level it shows the flexibility of formulating tracking as a data structure (in particular, a graphical model) instead of a procedural algorithm. The extension can be easily added to or removed from the model without the need to change other parts of the pipeline. This is especially useful in the actual software implementation of the approach. Future extensions could try to impose the whole cell cycle consisting of several consecutive stages (prophase, metaphase, anaphase, and telophase). That is, the cell in a track has to undergo the phases in the correct order preventing wrong assignments to objects that are "out-of-phase".

7.4. Linear Programming Approaches are Best for MAP Inference

One of the advantages of graphical models is the separation of representation and inference. It is usually enough to develop a graphical model representation of a given problem (as we did with tracking in the work at hand) without the need to develop a specialized solver for the problem. Instead, we can choose from a large collection of general purpose inference methods. For our evaluation of different MAP inference approaches for the chain graph model we selected five promising candidates (see Sec. 5.3.2 on page 50).

Branch-and-cut combined with the simplex method (ILP) implemented in CPLEX showed the overall best performance. As the only method, ILP found a confirmed global energy minimum in only 33 sec (and serves as the energy ground truth). Given these results modeling the local consistency constraints imposed by tracking as hard constraints in an integer linear program seems to be a natural fit. Two of the methods—loopy belief propagation (LBP) and bundle methods (BUNDLE-H)—couldn't neither find an optimal energy value nor fulfill the consistency constraints. Additionally, BUNDLE-H took the longest time to convergence. LBP took only 30 secs to converge (on par with ILP) but got obviously stuck in a local minimum.

Surprisingly, this deficiency can be fixed by taking the LBP solution as the start configuration of a lazy flipper run (LBP-LF). The combination of the two can achieve a competitive energy, even though the running time is longer by a factor of ten compared to ILP. Finally, the local polytope relaxation of the integer program solved with the simplex method alone (LP-LP) achieves a competitive energy and is the fastest method with only three seconds inference time. Since LP-LP can output only fractional solutions in general and the final integer solution is obtained by rounding, there is no guarantee to fulfill all local consistency constraints (in contrast to ILP). However, the energy landscape of the chain graph model seems to exhibit a structure well suited for this kind of approximation.

To conclude, ILP should be used by default since it performs best overall. LP-LP is an alternative when inference time is more important than an exact solution. This is for instance the case in an interactive end user application like the presented ilastik tracking workflow (Chap. 6). Another aspect when choosing a method is the requirement for a fast (integer) linear programming solver. As of the time of writing unfortunately only commercial solvers such as CPLEX or GUROBI are fast enough for the presented application and the user needs to acquire a potentially expensive software license. If such a license shouldn't be in reach, LBP-LF is an interesting alternative since it is freely available[2] and can solve the tracking problem with adequate quality, even though inference takes its time.

7.5. ilastik and the Chain Graph are a Viable Cell Tracking Pipeline

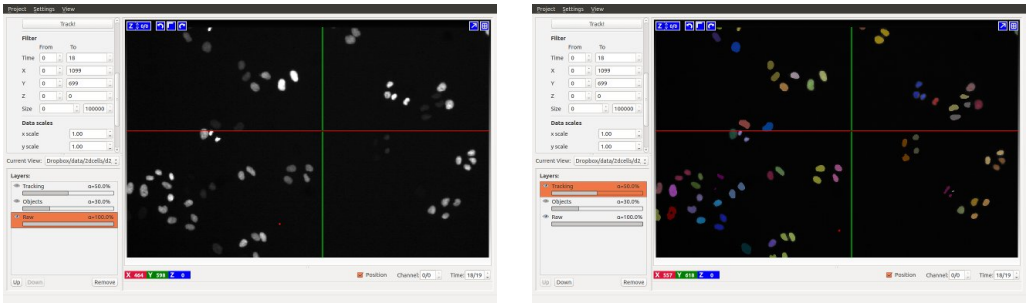


Figure 7.5.: Tracking of an unrelated, freely available 3rd party dataset in 2d+t (Neumann et al., 2010) to show the general applicability of the presented cell tracking pipeline and software (common color indicates common ancestry).

The majority of this work is concerned with describing and evaluating the chain graph tracking model. The motivation behind is constructing a reliable tracking

[2] It is part of the open source package opengm which is distributed under a free license here: `https://github.com/opengm/opengm`

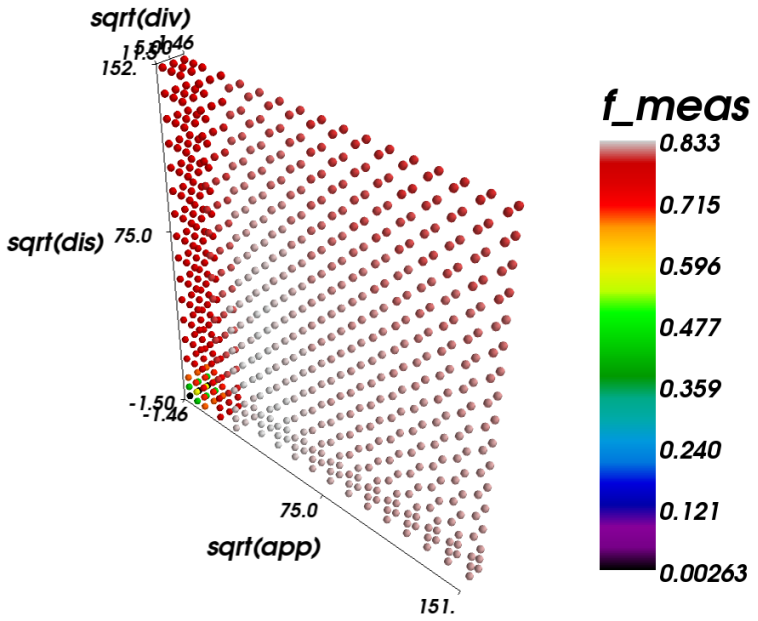


Figure 7.6.: Dependence of tracking performance on the choice of parameter values (zebrafish dataset; optimal joint assignment). The three axes correspond to the three parameters controlling division (`div`), appearance (`app`), and disappearance (`dis`). The axes are scaled down by the square root (`sqrt`).

tool for the second step of the two-step pipeline for cell tracking in general and digital embryo recording in particular (as outlined in Chap. 2). We cast the pipeline into software with a graphical user interface (Chap. 6) to serve as a means to an end for life scientists enabling them to conduct quantitative research in multidimensional images of proliferating cells.

Naturally, one main application of the presented pipeline is cell tracking in 3d+t recordings of embryos. However, the application was designed with any general (cell) tracking use case in mind. To prove the point we arbitrarily chose a freely available 2d+t dataset of dividing cell nuclei (Neumann et al., 2010). After a segmentation with ilastik pixel classification, we conducted a tracking using the ilastik tracking workflow obtaining a (by visual inspection) high quality result (see Fig. 7.5 on the preceding page). This was done without any modifications to the original pipeline. The 2d images are internally embedded in a 3d space with a singleton third dimension. That way, the tracking code for 3d can handle the 2d case transparently. Care has to be taken in case of overlapping or occluding objects, because this phenomenon cannot happen in 3d and is consequently not considered in the chain graph model. Events to describe occlusions could be added easily, but for recovering the identities of overlapping objects new methodology would have to be developed.

To successfully apply the method, the user has to set parameters like appearance

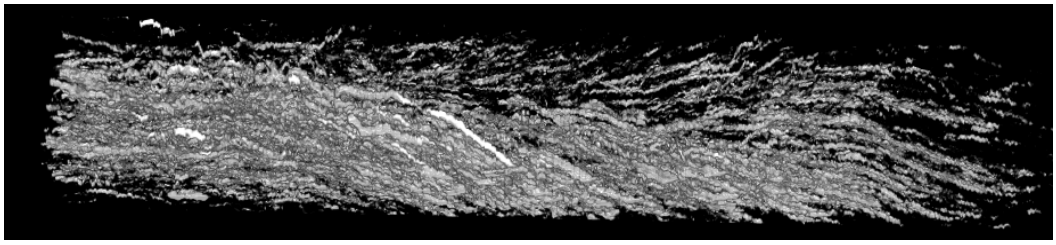


Figure 7.7.: Spacetime plot of the Zebrafish dataset. The 3d volumes were converted to 2d slices by maximum intensity projections and stacked in consecutive order. The image shows a volume rendering of the obtained 3d spacetime volume, time running from left to right. Only roughly 50% of the intensity is shown to make single tracks more pronounced.

or disappearance costs. In our quantitative experiments we found optimal parameters by conducting a fairly costly exhaustive search through parameter space taking many hours. This is not an option in an end user application. However, the tracking performance turns out to be fairly robust against the exact choice of parameters in practice as long as certain orders of magnitude are adhered to. This is evident from Fig. 7.6 on the preceding page which shows a plot of the *joint optimal assignment* performance on the zebrafish dataset depending on its three free parameters.[3] The performance changes only slowly in parameter space or is even invariant. The average move distance during division, C_{term}, and C_{init} could in principle even be calculated from the data. (In fact, we determined the former by measuring a set of manually selected divisions.) The latter two are directly connected to the likelihood of cell death and the signal-to-noise ratio of image acquisition, which influences the accuracy of the cell detector. Furthermore, the application uses default parameters that lead to a viable result out-of-the-box in most cases.

In future work the parameters could be learned from user annotations of the data. Lou and Hamprecht (2011, 2012) pioneered cell tracking by structured learning in two consecutive time slices. The approach can in principle be applied to the chain graph model, too. Instead of setting parameters, the user could then label some characteristic tracking events and refine the result further in an interactive manner as it is already the case with the ilastik pixel classification workflow.

7.6. Outlook

We have shown in several experiments that the two-step pipeline is a very capable approach when implemented with ilastik pixel classification and the chain graph model. The segmentation step (Sec. 5.3.1 on page 48) and the tracking step

[3] We cannot use the chain graph model for such a plot since it depends on five free parameters. However, the result is also conclusive for the chain graph model, since joint optimal assignment is essentially equivalent to the inter time slice assignment MRFs and the chain graph model shows the same invariance to the exact parameter values in practice.

(Sec. 5.3.3 on page 52) both produce high quality results, especially since the chain graph model can correct errors made in the segmentation step. But there are alternative pipeline designs which could be topics of future research. We showed that there are performance gains in a holistic approach where decisions can influence each other compared to a hard conditioning on previous results (see Sec. 7.1.2 on page 73). Therefore further improvements may be made by unifying the two-step pipeline into a single-step pipeline.

One idea is a combined segmentation and tracking graphical model. Currently in the ilastik segmentation step a foreground–background probability is assigned to each voxel and a segmentation is obtained by thresholding the predictions at 0.5 probability. Each connected component from the segmentation is then represented by a single detection variable in the chain graph. To generalize this, one could generate different segmentation hypotheses (for example by varying the threshold or using a different method such as superpixel segmentation (Achanta et al., 2012)) and add a random variable for each hypothesis. That way, the model could weigh segmentation and tracking against each other finding the overall most likely explanation of the data. Of course, this would increase the state-space of the model making inference harder and requires new consistency constraints to prevent the simultaneous activation of contradicting segmentation hypotheses. Funke et al. (2012) describe a related idea in the context of segmenting tubular structures in 3d, proving that such an approach can be practical.

Another idea is treating the 3d+t dataset as a single volume and conducting a segmentation in 4d. The result should be a forest of 4d spacetime tubes and tracking could be extracted by skeletonizing the segmentation (see Tschirren et al. (2002) for an example of the approach in 3d). Fig. 7.7 on the facing page shows a spacetime visualization of the zebrafish dataset and, indeed, tracks are visible in form of nuclei-tubes. However, they are not particularly smooth in time and broken up into many pieces because of an insufficient temporal sampling rate (0.5–2 volumes per minute in the presented datasets). Especially cells mitosis is not visible in the plot since the relative speed of the nuclei is much higher than during interphase. Furthermore due to noise there are also several phantom track fragments. Nevertheless, light sheet microscopy is improving fast and there are already microscope prototypes that can record up to 30 volumes per minute with a high contrast. Therefore, in the near future 4d segmentation could be an elegant alternative to the presented pipeline.

To conclude, the presented pipeline is a viable approach for digital embryo recordings and cell tracking in current microscopy datasets, and can be a foundation or inspiration for future developments.

7.7. Summary

In this chapter we argued that a holistic tracking model conducting inference over all time slices simultaneously can produce performance gains compared to models

that consider only a few time slices at once or are conditioned on irreversible decisions. Furthermore, we showed that the chain graph can also be seen as segmentation regularizer which provides value even in a setting that is not primarily concerned with tracking. Finally we discussed the minimal cell cycle length extension assessing that it trades off inference time against tracking accuracy and portrayed the digital embryo recording pipeline as a viable general purpose cell tracking approach.

8. Conclusions

We presented a novel tracking-by-assignment approach formulated as a probabilistic graphical model taking the form of a chain graph. It can handle a variable and large number of divisible objects and is highly robust against false positive detections and clutter. We established its performance on challenging datasets from developmental biology with false positive rates of more than 10% and showed an improvement over the state-of-the-art on these datasets. In particular, the robustness is evident in the increased precision at a stable recall compared to other methods. We demonstrated the modeling flexibility of the approach by extending it with a term controlling the minimal cell cycle length, examining the properties of different inference schemes for the same model, and separating preprocessing steps from inference with the hypotheses graph formalism. This makes it easy to control the performance influence of the single components by testing them separately and gives several entries point for future improvements. The method is implemented in software with a graphical user interface geared towards non-expert users. This enables other researchers—in particular from the life sciences—to build upon our method to advance the state-of-the-art in their respective fields.

We are now in a position to reliably, but not yet perfectly, segment and track cell nuclei in datasets like the presented ones. Light sheet microscopy is advancing at a fast pace and it is reasonable to assume a near perfect segmentation recall of cell nuclei for future datasets, alleviating the need to further improve the segmentation in that respect. Furthermore, the chain graph model can correct most of the false positive segmentations. In practice, the method can handle several thousand objects for dozens of time slices which satisfies the requirements to reconstruct early embryogenesis of typical model organisms before the gastrulation stage.

Tracking performance of our model is diminished mostly by merged segmentations of nuclei in close contact with each other which are treated as a single nucleus and by complex movements like several divisions in immediate vicinity, which can't be sufficiently described by the squared distance feature. Finding a way to correct the undersegmentations and introducing more features to describe tracking events could lift the results to a level on par with human performance. Then again, more features lead to more parameters and a better approach for parametrization is required. Lou and Hamprecht (2011, 2012) pioneered cell tracking by structured learning in two consecutive time slices. Their approach could be generalized to learn the parameters in global graphical models like the chain graph.

To go beyond early embryogenesis, the tracking model needs to handle movements of cell groups and motion of the organism as a whole. Currently, our energy functions are assuming independent movements of the single cells. Group move-

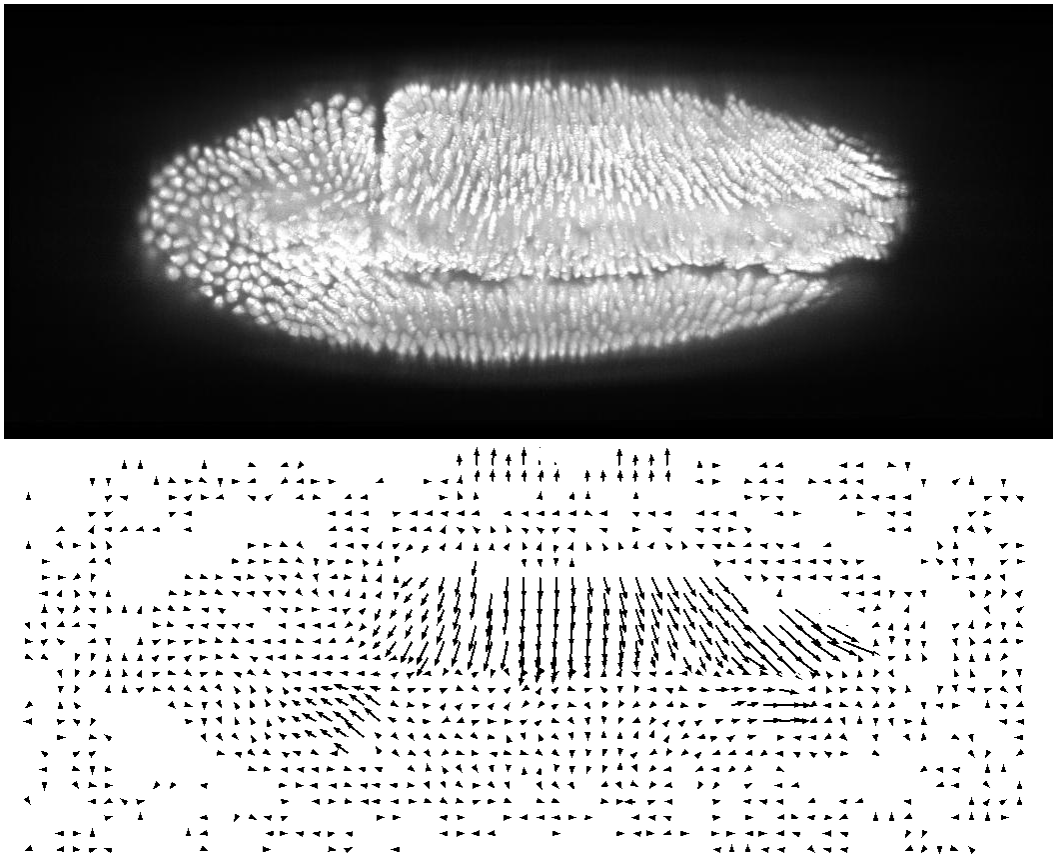


Figure 8.1.: Group movements of nuclei. The upper part is a projection of three consecutive time slices to visualize coordinated group movements of the cell nuclei (shown is early gastrulation in Drosophila). The lower part shows a cross-correlation estimate of the group movements in 2d. Such estimates could serve as motion priors in future extensions of the tracking model.

ments could be modeled in form of local velocity priors that could be estimated in advance, for instance, by using optical flow or cross correlation (cf. Fig. 8.1). The current model can most likely not be extended to handle global organism motion. New methodology needs to be developed to track contortions of the whole embryo. Methods like the chain graph could then be applied in a local coordinate system calculated relative to the global embryo model.

In summary, the very first steps towards complete digital embryos from the zygote to the post-embryonic stage are made and we hope that the thesis at hand will be considered a positive contribution to the field, when the grand goal is finally met in the future.

A. Ground Truth Cell Lineages and Manual Tracking Protocol

We manually tracked the cell nuclei in both presented datasets (zebrafish and Drosophila) for benchmarking purposes. Unfortunately, the datasets are too large to be presented here. However, they are part of the supplementary materials of Kausler et al. (2012) and can be acquired from there. Alternatively, please contact either
fred.hamprecht@iwr.uni-heidelberg.de or bernhard@kausler.net.

For the Drosophila dataset we did the manual tracking directly on the segmentation, limiting our view to the type of information the automatic tracking methods can access. In contrast, on the zebrafish dataset the tracking was conducted on the raw data to be able to benchmark the pipeline as a whole. The latter required a strict protocol to avoid biases and document disputable decisions and is reproduced below.

Zebrafish dataset manual tracking protocol

```
Input
-----
as-tiff_ct-keller-animal-spacetime_0-100: 0000.tiff - 0024.tiff
downsampled from 16bit to 8bit
scale: 1111x1161x25 (30.80MB) (100%)

coordinates start at 0 -> time 5 is the 6th time slice

Software
--------
Java 5D image viewer, Version V1.3.0 by Rainer Heintzmann
ImageJ 1.43l
Java 1.6.0_18 (64-bit)

View5D settings
---------------
Tracking Mode: Max
Tracking direction: Z
Finish Mode: Stop
Finish Metric: Integral
```

```
Threshold 0.2
Use automatic maximum finding: 3.0 X,Y neighbors
Subpixel positions turned off
Repulsion turned on

General Protocol
----------------
We use a maximum intensity snap-in. If this fails to center on a cell,
we do a manual correction. We don't record this in the protocol,
though. Therefore, in most cases the marker sits on the maximal
intensity, but not always. Sometimes, we mark positions by
extrapolating the expected track of a cell, as long as there is the
slightest evidence (like a very small intensity gradient) at the
expected position. This might be impossible to pick up by a
segmentation or tracking algorithm that works only locally and/or at
one point in time.  In a few cases, we even connect cells over several
time slices, even if there is no evidence in between. These are
borderline decisions and recorded in the remarks below.

Remarks
-------
interleave: unsure how to separate the overlap; separable interleaves
are not recorded explicitly

* 9ab vanishes at t22
* 21bb interleave with 21aa (t 17-24)
* 22ab interleave with 23bb (t 17-24)
* 23a interleave with 35a (t 9-17)
* 23aa, 23ab split in z direction; interleaving (t 17-24)
* 24 very low intense cell (7-20); shows characteristic speed
increases at the two mitosis points in time, but cannot find any offspring
* 27aa low intensity (17) at t 18; disappears in t 19; seems to
reappear in t 21
* 27ab very low intensity (11) at t 18; disappears in t 19; seems to
reappear in t 20
* 28 interleave with 29, but easily distinguishable
* 29b (t 5): manually corrected snap-in
* 29ab: evidence very weak (int 10-22) over whole track;
 shape irregular, noisy
* 30b (t 7) very weak evidence (int maximum 10; background 7)
(same at t8, t9 -> int 11); reappears at same position with much
better intensity at t10
* whole 31 lineage: very weak evidence over the whole track
* 31a split at t20; no evidence for other daughter cell (small
```

disturbances at the expected positions -> too inconclusive)
* 33 cell is not splitting, but has good intensity over t0-24
* 34a interleave with 34b (t 5-13)
* 37a and 37b interleave at t7; split probably happened 1-2 time slices before, but there is only a interleaving blob visible
* 37bb, t18 manually corrected snap-in
* 27aa, 37bb, 37aa t19-24 heavy interleave
* 43b interleave with 49b (t 6,7,8)
* 44 did possibly split in stack direction at t5 (not visible though); at t7 an interleaved fragment appears and morphs into a full cell later
* 44aa interleave with 44bb (t 18)
* 46b split probably as early as t16, but interleave till t17 (where the split is visible)
* 46ba interleave with 50 bb (t 21)
* 50aa gone at t 22,23, reappears at t24; some doubts left
* 50ba gone at t 21, 22 ; reappears at t23; not sure, if it is actually the same cell; could also be 59aa
* 51b probably splits at t 19; due to interleave visible only at t 22 and beyond
* 54ab very weak evidence (int 3-5, background 2)
* 58bb vanishes after t 23
* 58ab interleave with 57ab
* 59aa vanishes after t 20; there is a possible candidate, that is assigned to 50ba; some doubts here
* 59a: transition t7 -> t8 may be a cell disappearance and appearance, since the translation distance is very long
* 64a: recorded split at t21; very likely happened at t20, but no suitable candidate for second daughter cell found
* 64ab: is not visible at t20, but has to be there due to typical division pattern observed in latter
* 66bb vanishes at t20 and reappears at t21

Speckles:

Speckles are -- compared to cells -- a little bit smaller and have in general a higher intensity; they could be confused with cells by a tracking algorithm. There are even smaller splitters that we don't track here.

Candidates:
40,69,70,71
* 40 object doesn't split, but has good snr at t0-24; very borderline; might be a cell, but is a little bit too small
* 70 vanishes at t5
* 71 appears t7, might be identical to 70 (distance quite large, though)

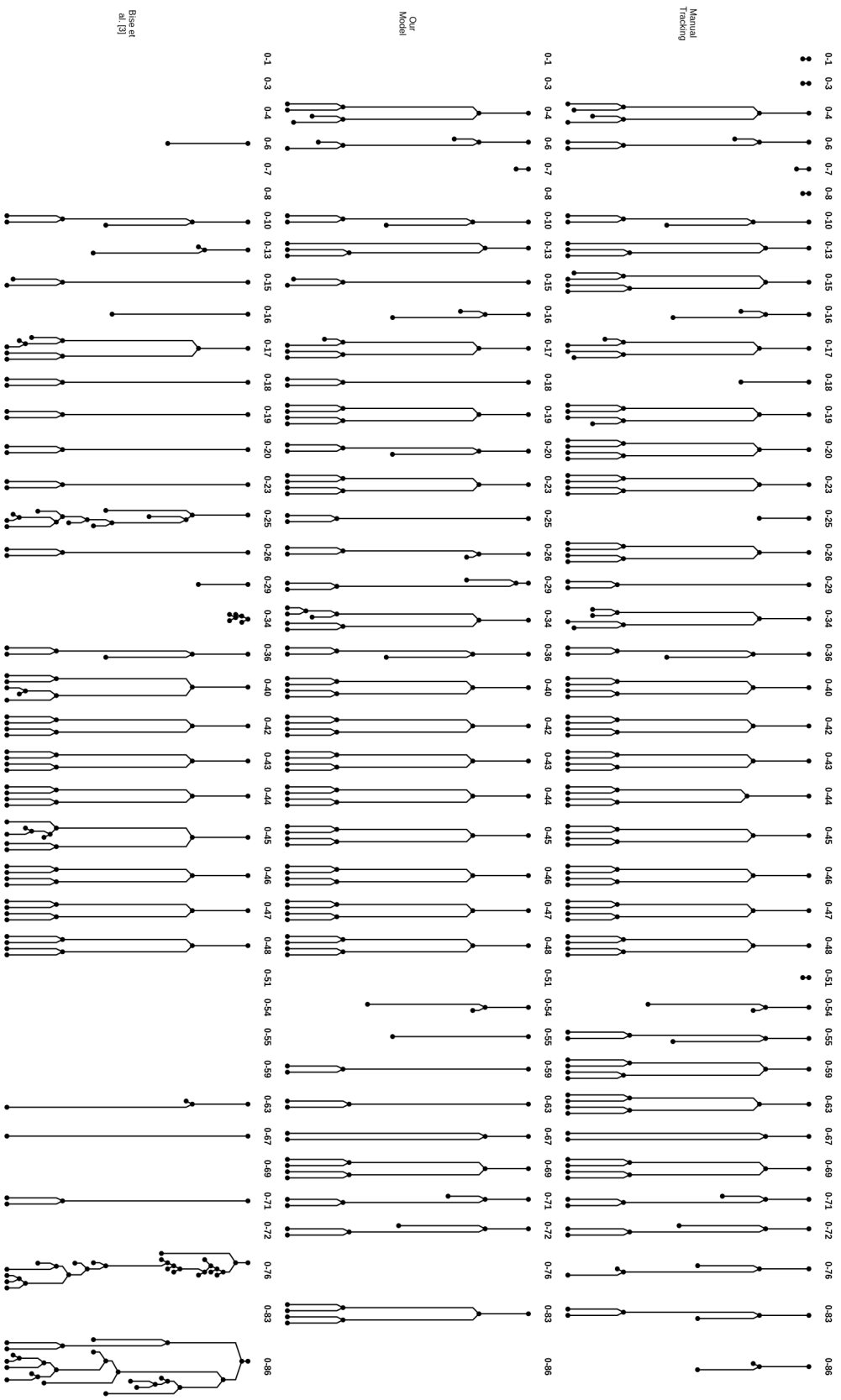
B. Drosophila Dataset: Complete Lineage Synopsis

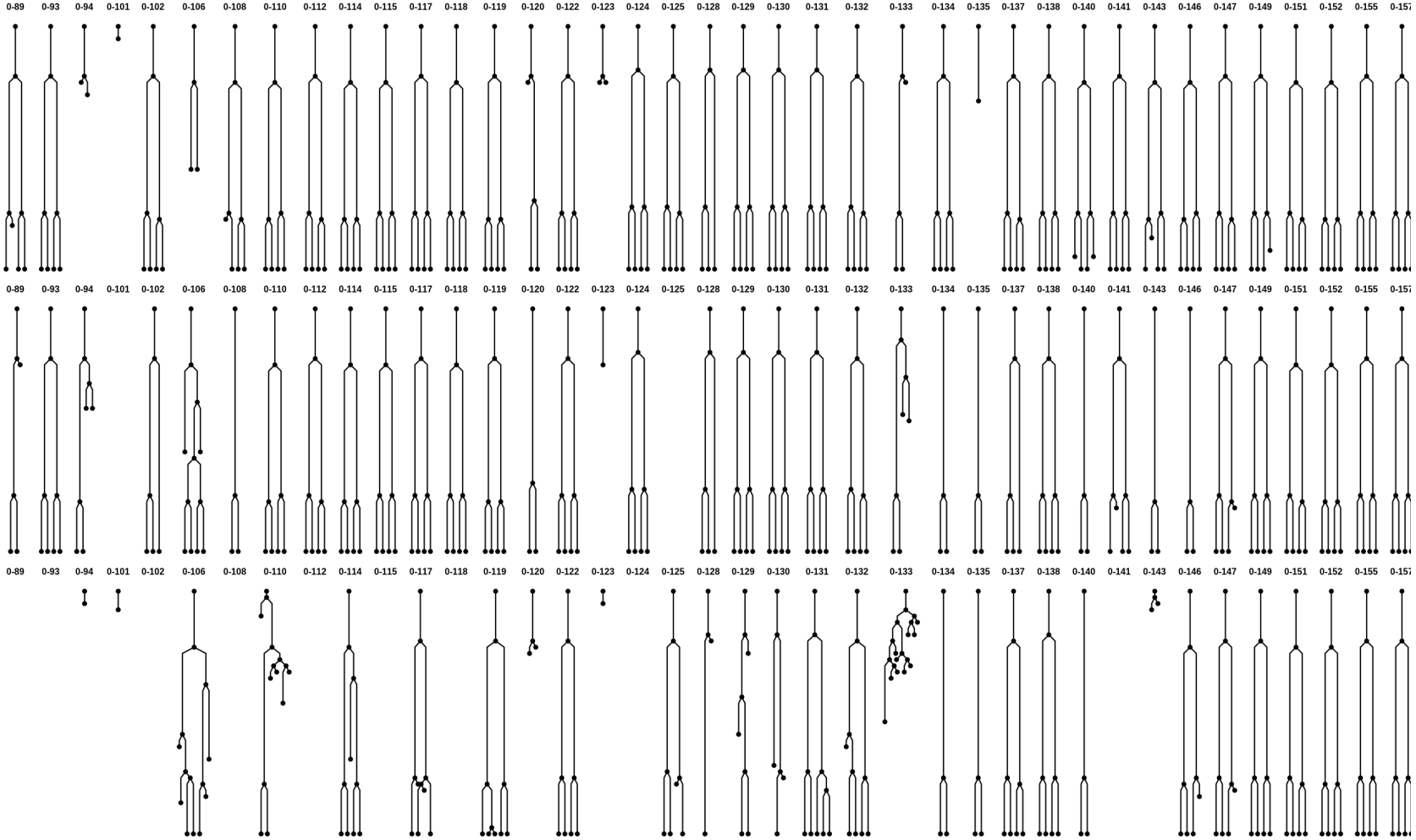
In this chapter we reproduce three complete lineage tree reconstructions of the Drosophila dataset obtained as follows: tracked by human (ground truth), by the chain graph model, and by the method of Bise et al. (2011).

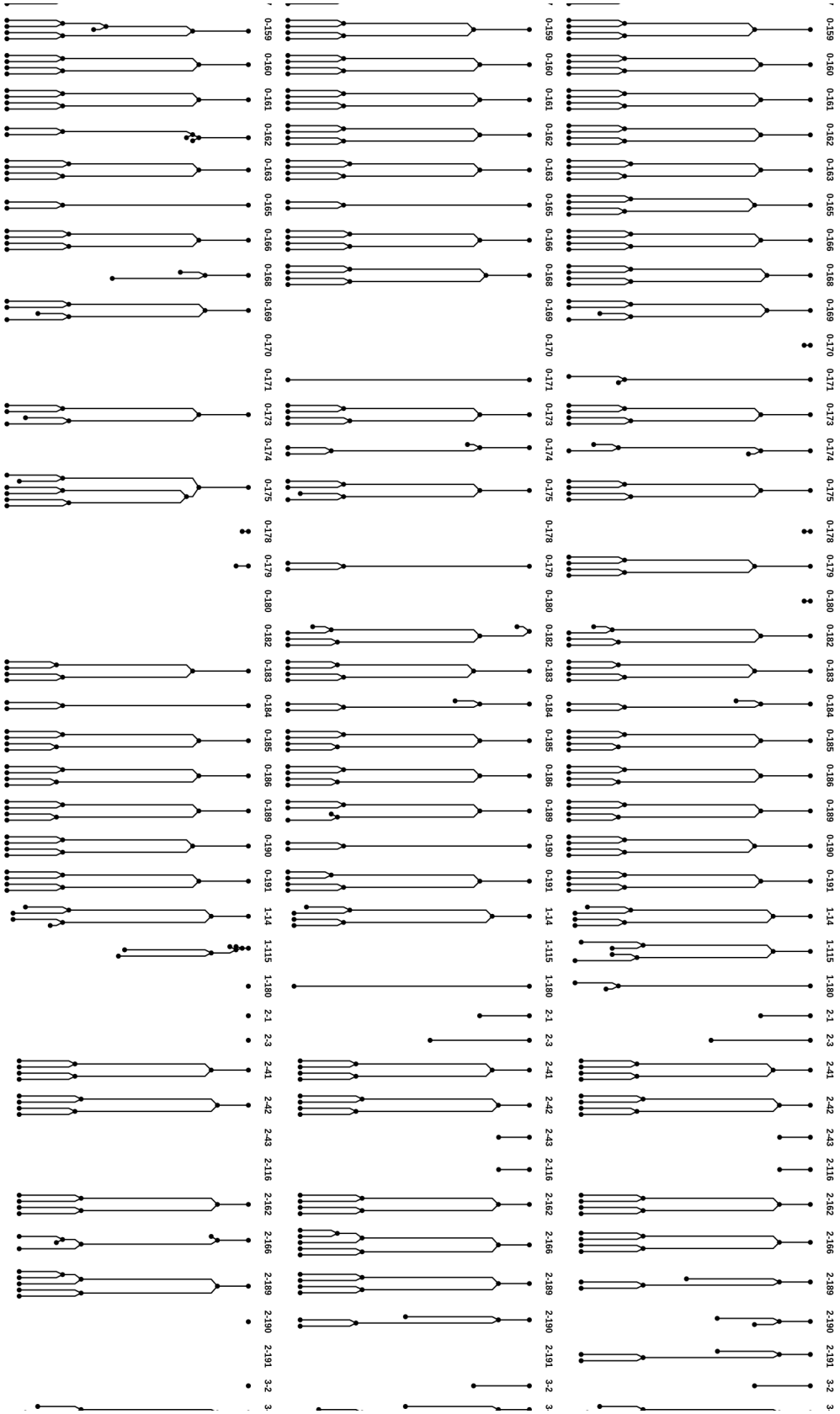
Appearing, disappearing, and dividing cells are denoted by black circles; cells only involved in movements are omitted for reasons of clarity.

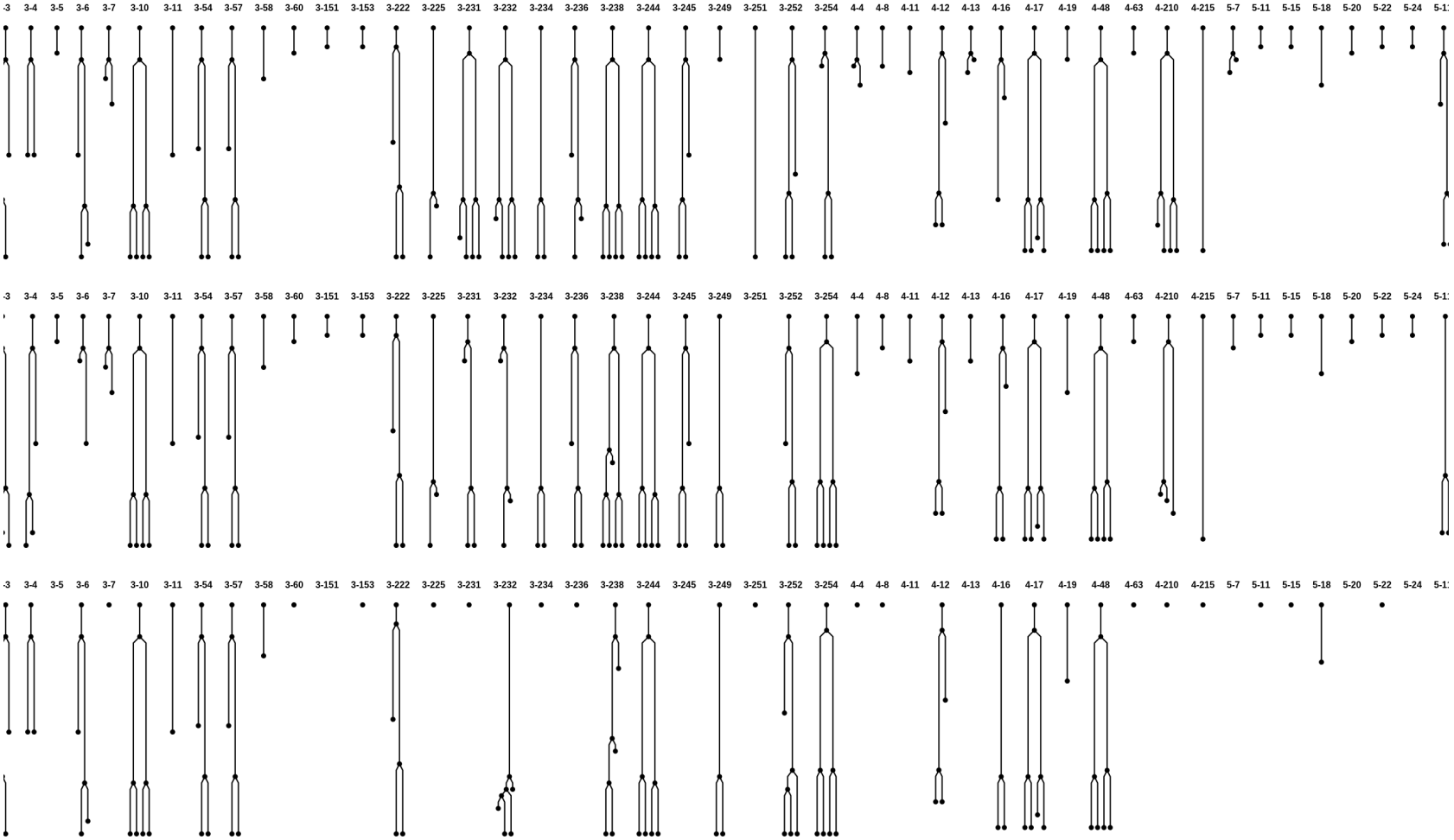
The leading number of the IDs indicates the time step at which the lineage tree begins; the second number displays the label of the corresponding cell.

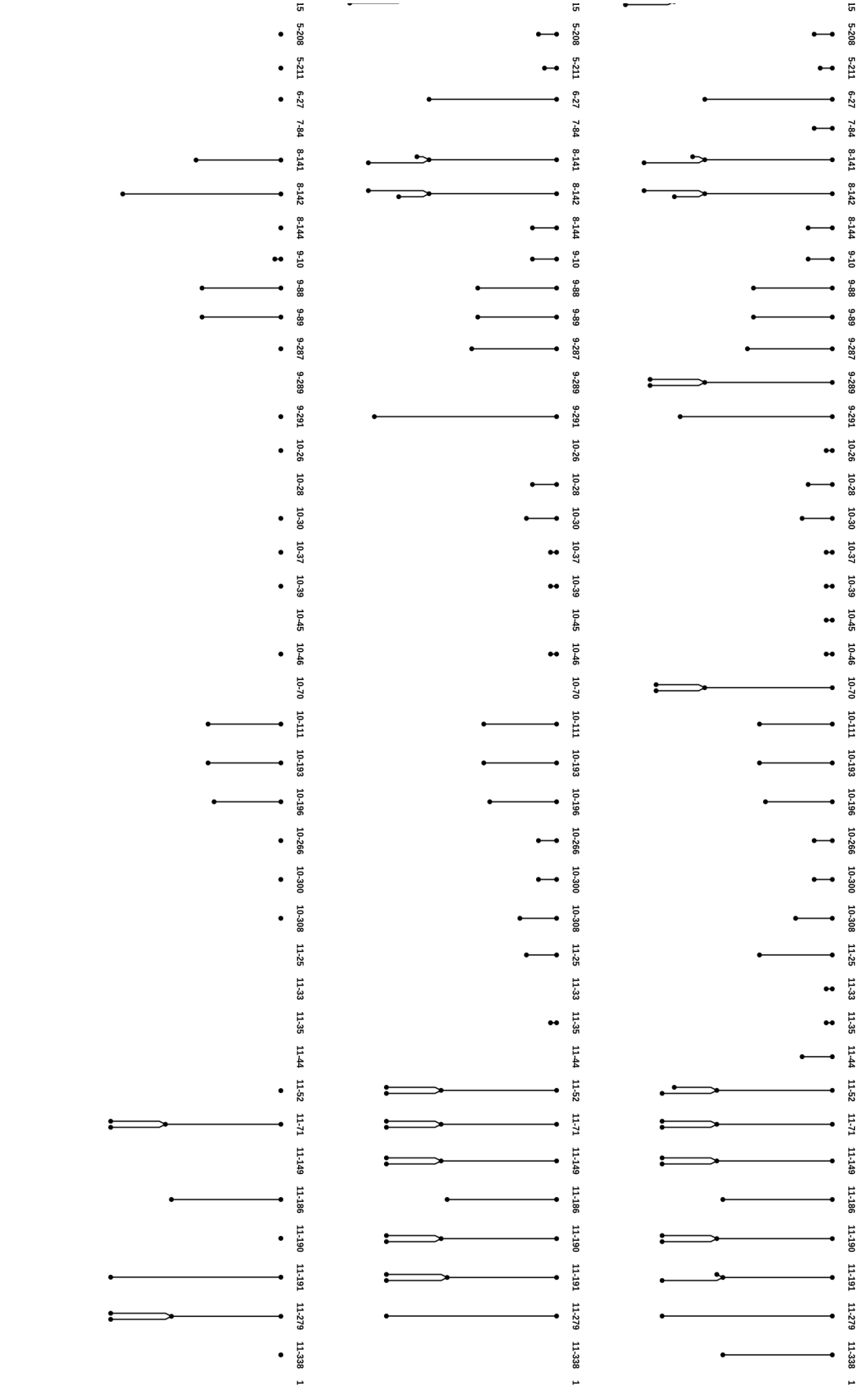
It should be noted that, on the one hand, equal lineage trees are not necessarily spanned by the same cells, and, on the other hand, missing lineage trees might be present at a later time slice (e.g. only the root cell was not tracked) rather than missing completely. This also includes the subtrees of (temporarily) falsely disappearing cells. Furthermore, similarity in the labels does not imply proximity in space. [Lineage tree graphic starts on the next page.]

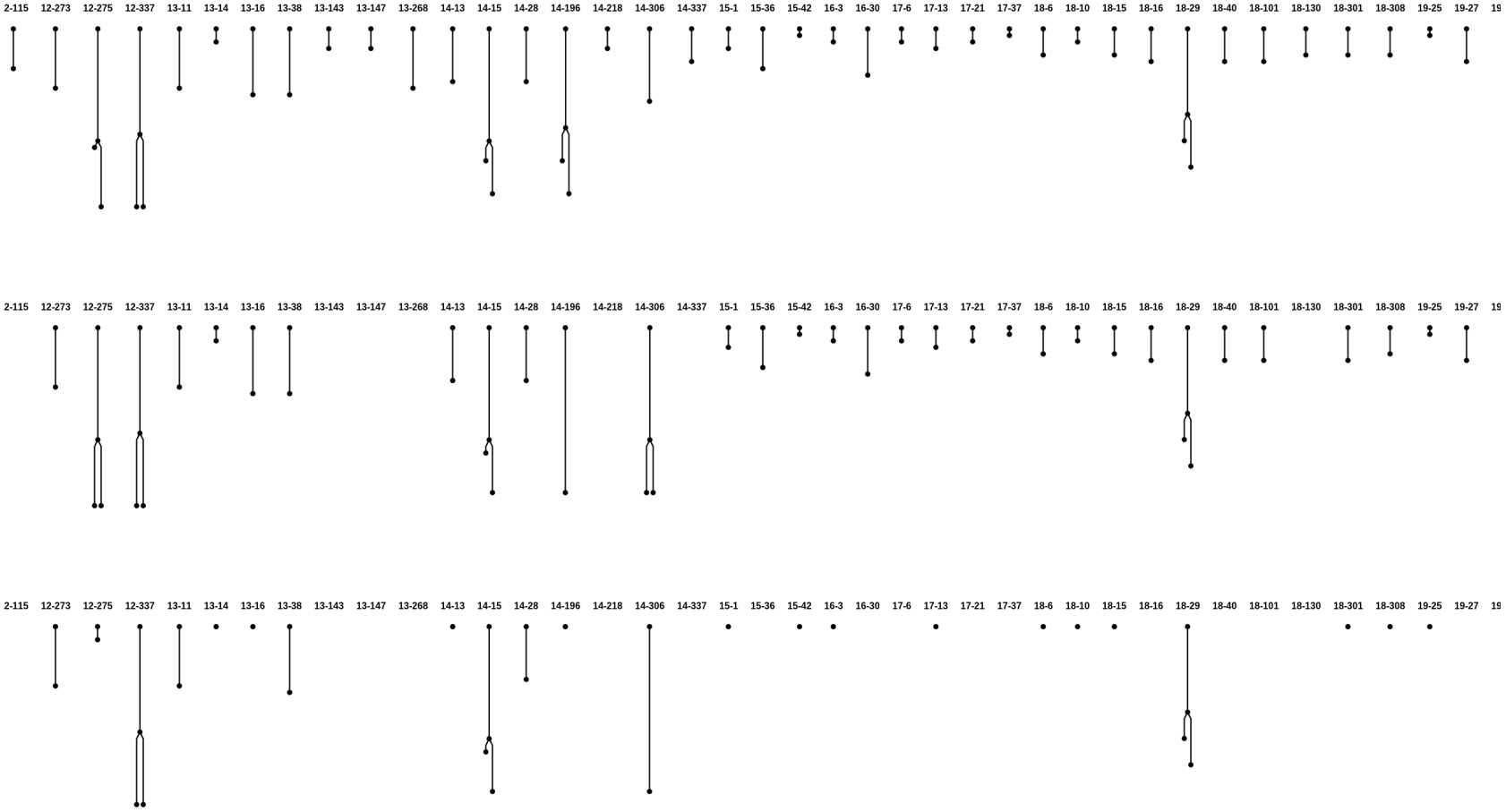

Manual Tracking
Our Model
Bise et al. [3]
0-1 0-3 0-4 0-6 0-7 0-8 0-10 0-13 0-15 0-16 0-17 0-18 0-19 0-20 0-23 0-25 0-26 0-29 0-34 0-36 0-40 0-42 0-43 0-44 0-45 0-46 0-47 0-48 0-51 0-54 0-55 0-59 0-63 0-67 0-69 0-71 0-72 0-76 0-83 0-86

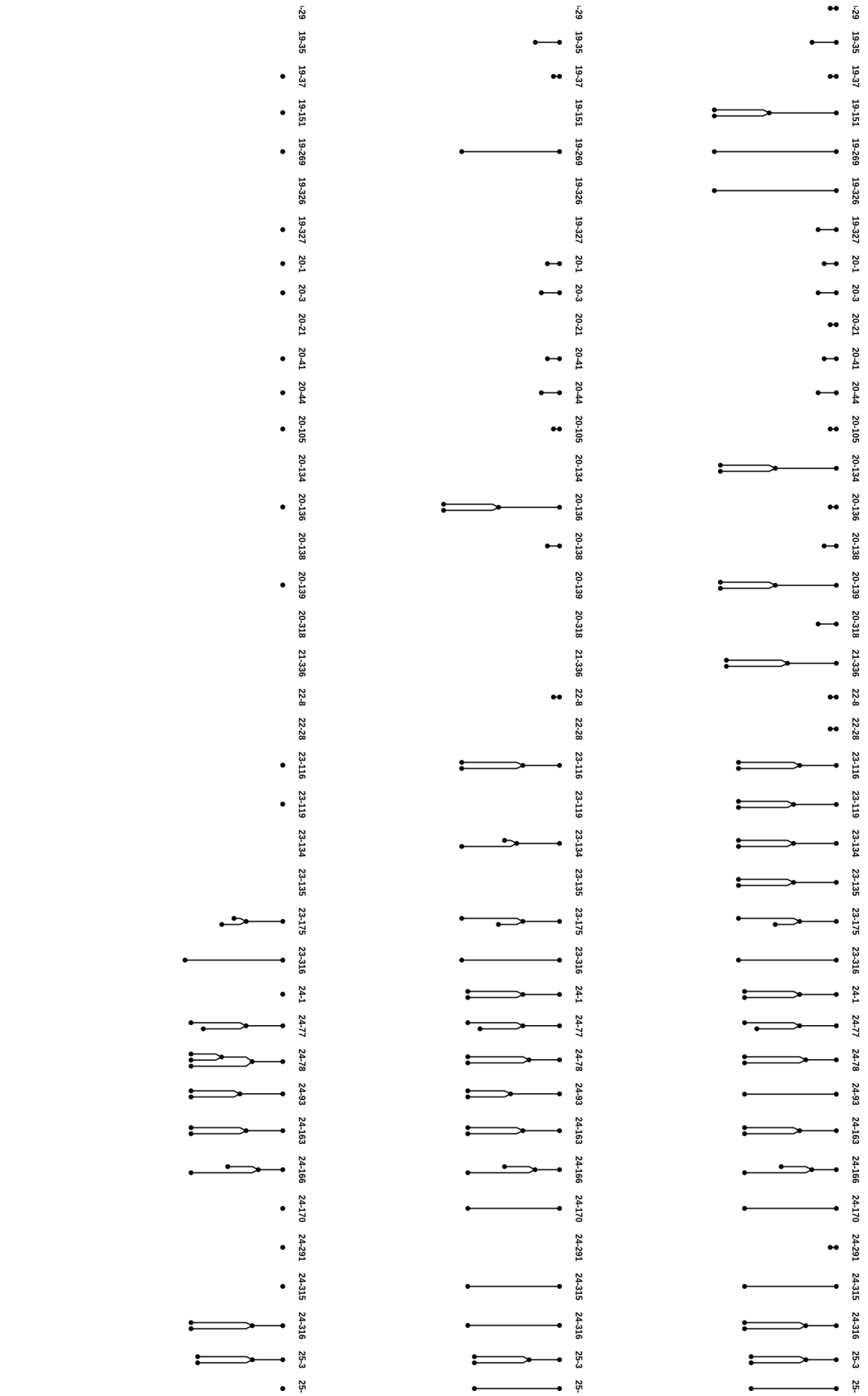

0-89 0-93 0-94 0-101 0-102 0-106 0-108 0-110 0-112 0-114 0-115 0-117 0-118 0-119 0-120 0-122 0-123 0-124 0-125 0-128 0-129 0-130 0-131 0-132 0-133 0-134 0-135 0-137 0-138 0-140 0-141 0-143 0-146 0-147 0-149 0-151 0-152 0-155 0-157
0-89 0-93 0-94 0-101 0-102 0-106 0-108 0-110 0-112 0-114 0-115 0-117 0-118 0-119 0-120 0-122 0-123 0-124 0-125 0-128 0-129 0-130 0-131 0-132 0-133 0-134 0-135 0-137 0-138 0-140 0-141 0-143 0-146 0-147 0-149 0-151 0-152 0-155 0-157
0-89 0-93 0-94 0-101 0-102 0-106 0-108 0-110 0-112 0-114 0-115 0-117 0-118 0-119 0-120 0-122 0-123 0-124 0-125 0-128 0-129 0-130 0-131 0-132 0-133 0-134 0-135 0-137 0-138 0-140 0-141 0-143 0-146 0-147 0-149 0-151 0-152 0-155 0-157

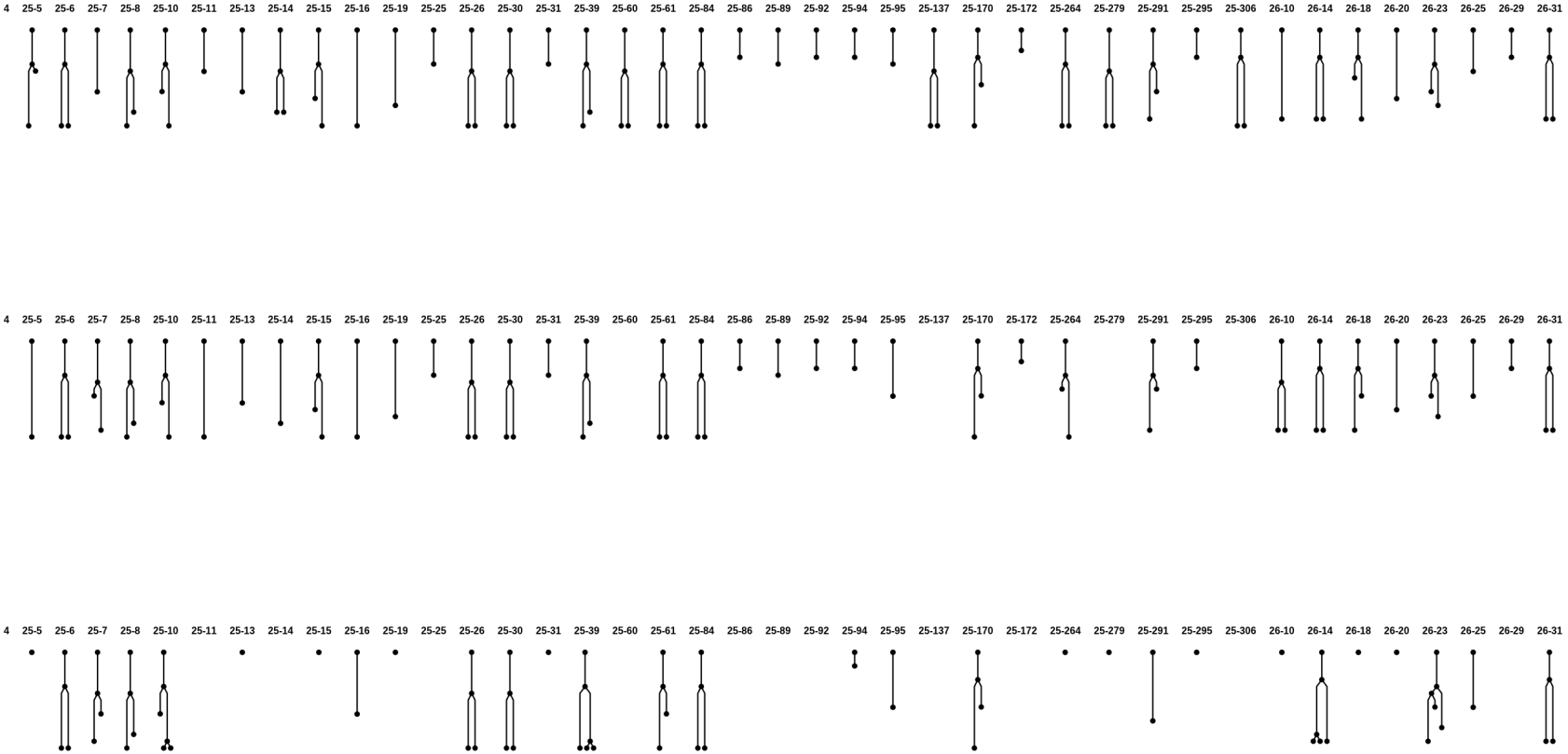


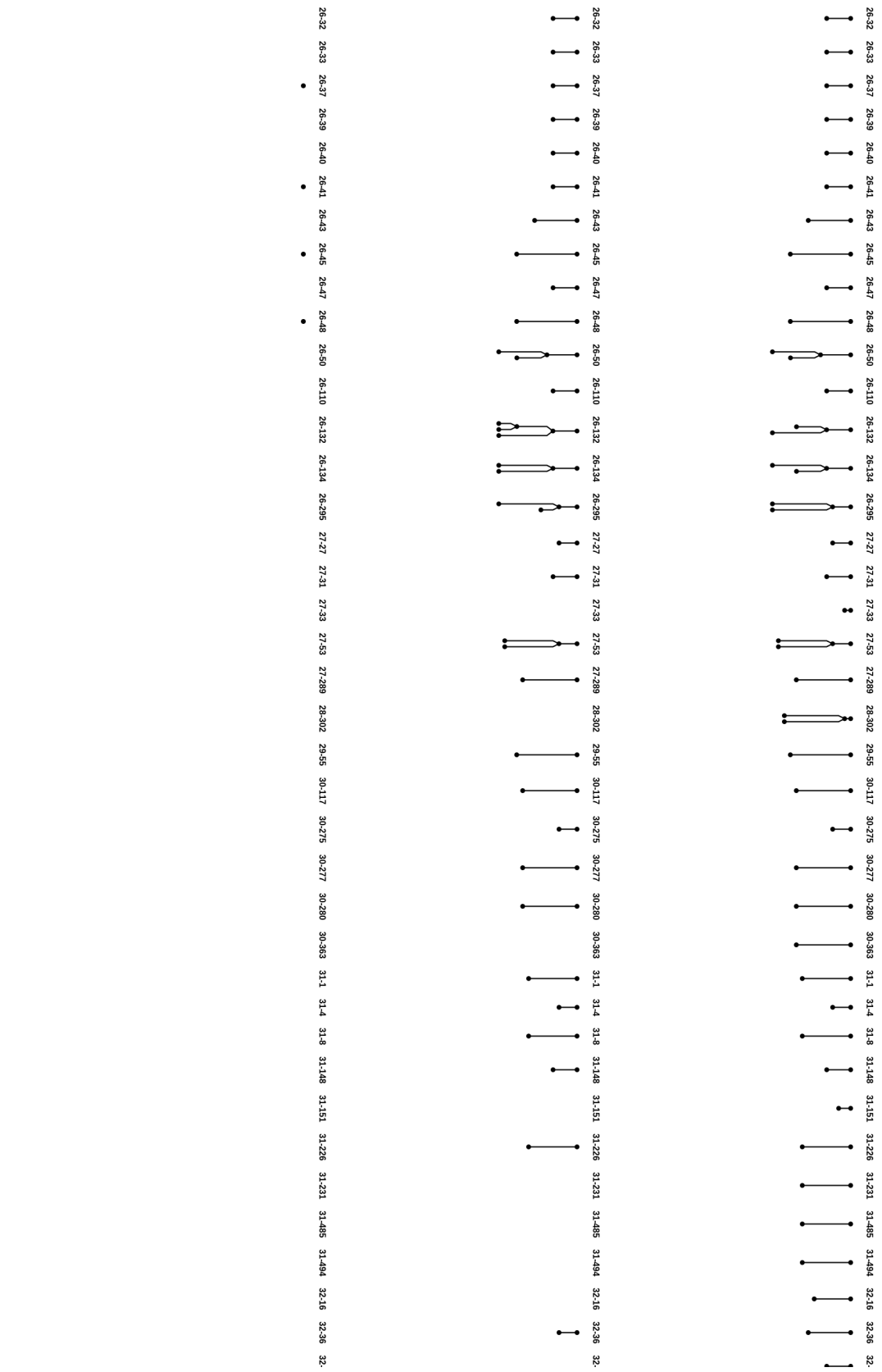

3-3 3-4 3-5 3-6 3-7 3-10 3-11 3-54 3-57 3-58 3-60 3-151 3-153 3-222 3-225 3-231 3-232 3-234 3-236 3-238 3-244 3-245 3-249 3-251 3-252 3-254 4-4 4-8 4-11 4-12 4-13 4-16 4-17 4-19 4-48 4-63 4-210 4-215 5-7 5-11 5-15 5-18 5-20 5-22 5-24
3-3 3-4 3-5 3-6 3-7 3-10 3-11 3-54 3-57 3-58 3-60 3-151 3-153 3-222 3-225 3-231 3-232 3-234 3-236 3-238 3-244 3-245 3-249 3-251 3-252 3-254 4-4 4-8 4-11 4-12 4-13 4-16 4-17 4-19 4-48 4-63 4-210 4-215 5-7 5-11 5-15 5-18 5-20 5-22 5-24
3-3 3-4 3-5 3-6 3-7 3-10 3-11 3-54 3-57 3-58 3-60 3-151 3-153 3-222 3-225 3-231 3-232 3-234 3-236 3-238 3-244 3-245 3-249 3-251 3-252 3-254 4-4 4-8 4-11 4-12 4-13 4-16 4-17 4-19 4-48 4-63 4-210 4-215 5-7 5-11 5-15 5-18 5-20 5-22 5-24


15 5-208 5-211 6-27 7-84 8-141 8-142 8-144 9-10 9-88 9-89 9-287 9-289 9-291 10-26 10-28 10-30 10-37 10-39 10-45 10-46 10-70 10-111 10-193 10-196 10-266 10-300 10-308 11-25 11-33 11-35 11-44 11-52 11-71 11-149 11-186 11-190 11-191 11-279 11-338 1


2-115 12-273 12-275 12-337 13-11 13-14 13-16 13-38 13-143 13-147 13-268 14-13 14-15 14-28 14-196 14-218 14-306 14-337 15-1 15-36 15-42 16-3 16-30 17-6 17-13 17-21 17-37 18-6 18-10 18-15 18-16 18-29 18-40 18-101 18-130 18-301 18-308 19-25 19-27 19
2-115 12-273 12-275 12-337 13-11 13-14 13-16 13-38 13-143 13-147 13-268 14-13 14-15 14-28 14-196 14-218 14-306 14-337 15-1 15-36 15-42 16-3 16-30 17-6 17-13 17-21 17-37 18-6 18-10 18-15 18-16 18-29 18-40 18-101 18-130 18-301 18-308 19-25 19-27 19
2-115 12-273 12-275 12-337 13-11 13-14 13-16 13-38 13-143 13-147 13-268 14-13 14-15 14-28 14-196 14-218 14-306 14-337 15-1 15-36 15-42 16-3 16-30 17-6 17-13 17-21 17-37 18-6 18-10 18-15 18-16 18-29 18-40 18-101 18-130 18-301 18-308 19-25 19-27 19


-29
19-35
19-37
19-151
19-269
19-326
19-327
20-1
20-3
20-21
20-41
20-44
20-105
20-134
20-136
20-138
20-139
20-318
21-336
22-8
22-28
23-116
23-119
23-134
23-135
23-175
23-316
24-1
24-77
24-78
24-93
24-163
24-166
24-170
24-291
24-315
24-316
25-3
25-


4 25-5 25-6 25-7 25-8 25-10 25-11 25-13 25-14 25-15 25-16 25-19 25-25 25-26 25-30 25-31 25-39 25-60 25-61 25-84 25-86 25-89 25-92 25-94 25-95 25-137 25-170 25-172 25-264 25-279 25-291 25-295 25-306 26-10 26-14 26-18 26-20 26-23 26-25 26-29 26-31
4 25-5 25-6 25-7 25-8 25-10 25-11 25-13 25-14 25-15 25-16 25-19 25-25 25-26 25-30 25-31 25-39 25-60 25-61 25-84 25-86 25-89 25-92 25-94 25-95 25-137 25-170 25-172 25-264 25-279 25-291 25-295 25-306 26-10 26-14 26-18 26-20 26-23 26-25 26-29 26-31
4 25-5 25-6 25-7 25-8 25-10 25-11 25-13 25-14 25-15 25-16 25-19 25-25 25-26 25-30 25-31 25-39 25-60 25-61 25-84 25-86 25-89 25-92 25-94 25-95 25-137 25-170 25-172 25-264 25-279 25-291 25-295 25-306 26-10 26-14 26-18 26-20 26-23 26-25 26-29 26-31


26-32
26-33
26-37
26-39
26-40
26-41
26-43
26-45
26-47
26-48
26-50
26-110
26-132
26-134
26-295
27-27
27-31
27-33
27-53
27-289
28-302
29-55
30-117
30-275
30-277
30-280
30-363
31-1
31-4
31-8
31-148
31-151
31-226
31-231
31-485
31-494
32-16
32-36
32-

52
32-54
32-56
32-57
32-61
32-64
32-65
32-66
32-73
32-75
32-184
32-188
32-215
32-439
33-18
33-31
33-43
33-46
33-47
33-60
33-198
33-384
33-443
33-449
33-516
33-530
33-540
33-541
33-553
34-7
34-65
34-69
34-75
34-101
34-185
34-237
35-5
35-23
35-26

35-51
35-58
35-63
35-206
35-317
35-318
35-546
36-2
36-11
36-38
36-41
36-43
36-66
36-277
36-297
36-364
36-428
37-37
37-100
37-127
37-131
37-195
37-228
37-230
37-241
37-308
37-433
38-16
38-43
38-282
38-286
38-295
38-360
38-449
38-477
38-555

Bibliography

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. Slic superpixels compared to state-of-the-art superpixel methods. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 34 (11), 2274–2282.

Andres, B., Beier, T., Kappes, J. H., 2012a. OpenGM: A C++ Library for Discrete Graphical Models. Tech. rep.

Andres, B., Kappes, J. H., Beier, T., Ullrich, K., Hamprecht, F. A., 2012b. The Lazy Flipper: Efficient Depth-limited Exhaustive Search in Discrete Graphical Models. In: 12th European Conference on Computer Vision (ECCV). pp. 154–166.

Arulampalam, M., Maskell, S., Gordon, N., Clapp, T., 2002. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. Signal Processing, IEEE Transactions on 50 (2), 174 –188.

Bao, Z., Murray, J. I., Boyle, T., Ooi, S. L., Sandel, M. J., Waterston, R. H., 2006. Automated cell lineage tracing in caenorhabditis elegans. Proceedings of the National Academy of Sciences of the United States of America 103 (8), 2707–2712.

Batra, D., Yadollahpour, P., Guzman-Rivera, A., Shakhnarovich, G., 2012. Diverse m-best solutions in markov random fields. In: 12th European Conference on Computer Vision (ECCV). Springer.

Bertele, U., Brioschi, F., 1972. Nonserial Dynamic Programming. Academic Press, Inc., Orlando, FL, USA.

Besag, J., 1986. On the Statistical Analysis of Dirty Pictures. Journal of the Royal Statistical Society. Series B (Methodological) 48 (3), 259–302.

Bildsoe, H., Franklin, V., Tam, P. P., 2007. Fate-mapping technique: using carbocyanine dyes for vital labeling of cells in gastrula-stage mouse embryos cultured in vitro. Cold Spring Harbor Protocols.

Bise, R., Yin, Z., Kanade, T., 2011. Reliable cell tracking by global data association. In: IEEE International Symposium on Biomedical Imaging (ISBI 2011).

Blake, A., Kohli, P., Rother, C. (Eds.), 2011. Markov Random Fields for Vision and Image Processing. MIT Press.

Boykov, Y., Funka-Lea, G., 2006. Graph cuts and efficient N-D image segmentation. International Journal of Computer Vision 70, 109–131.

Breiman, L., 1996. Bagging predictors. Machine Learning 24 (2), 123–140.

Breiman, L., 2001. Random forests. Machine Learning 45, 5–32.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees. Wadsworth and Brooks, Monterey, CA.

Brendel, W., Amer, M., Todorovic, S., 2011. Multiobject tracking as maximum weight independent set. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2011). pp. 1273 –1280.

Caruana, R., Niculescu-Mizil, A., 2006. An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd International Conference on Machine Learning (ICML 06). pp. 161–168.

Ceriani, L., Verme, P., 2012. The origins of the gini index: extracts from variabilità e mutabilità (1912) by corrado gini. The Journal of Economic Inequality 10 (3), 421–443.

Chakraborty, C., Hsu, C. H., Wen, Z. H., Lin, C. S., Agoramoorthy, G., 2009. Zebrafish: a complete animal model for in vivo drug discovery and development. Current Drug Metabolism 10 (2), 116–124.

Chen, X., Zhou, X., Wong, S. T., 2006. Automated segmentation, classification, and tracking of cancer cell nuclei in time-lapse microscopy. IEEE Transactions on Biomedical Engineering 53 (4), 762–766.

Chen, Y. Q., Nixon, M. S., Thomas, D. W., 1995. Statistical geometrical features for texture classification. Pattern Recognition 28 (4), 537–552.

Costa, E., Lorena, A., Carvalho, A., Freitas, A., 2007. A review of performance evaluation measures for hierarchical classifiers. In: Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop. pp. 1–6.

Darby-Dowman, K., Wilson, J. M., 2002. Developments in linear and integer programming. The Journal of the Operational Research Society 53 (10), 1065–1071.

Dechter, R., September 1999. Bucket elimination: A unifying framework for reasoning. Artificial Intelligence 113 (1–2), 41–85.

Detrich, III, H. W., Westerfield, M., Zon, L. I. (Eds.), 2009. Essential Zebrafish Methods. Academic Press.

Díaz-Uriarte, R., De Andres, S. A., 2006. Gene selection and classification of microarray data using random forest. BMC Bioinformatics 7, 3.

Doucet, A., Johansen, A., 2011. A tutorial on particle filtering and smoothing: Fifteen years later. In: Handbook of Nonlinear Filtering. Oxford University Press.

Dzyubachyk, O., Van Cappellen, W., Essers, J., Niessen, W., Meijering, E., 2010. Advanced level-set-based cell tracking in time-lapse fluorescence microscopy. IEEE Transactions on Medical Imaging 29 (3), 852–867.

Fox, E., Choi, D., Willsky, A., 2006. Nonparametric Bayesian methods for large scale multi-target tracking. In: Signals, Systems and Computers, 2006. ACSSC'06. Fortieth Asilomar Conference on. pp. 2009–2013.

Freeman, E., Robson, E., Bates, B., Sierra, K., October 2004. Head First Design Patterns. O'Reilly Media.

Frey, B. J., 2003. Extending factor graphs so as to unify directed and undirected graphical models. In: Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence (UAI 03). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 257–264.

Frydenberg, M., 1990. The chain graph Markov property. Scandinavian Journal of Statistics 17, 333–353.

Funke, J., Andres, B., Hamprecht, F. A., Cardona, A., Cook, M., 2012. Efficient automatic 3d-reconstruction of branching neurons from em data. In: 2012 IEEE Conference onComputer Vision and Pattern Recognition (CVPR). IEEE, pp. 1004–1011.

Gamerman, D., Lopes, H., 2006. Markov chain Monte Carlo: stochastic simulation for Bayesian inference. Taylor & Francis.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1994. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, USA.

Guo, L., Ma, Y., Cukic, B., Singh, H., 2004. Robust prediction of fault-proneness by random forests. In: 15th International Symposium on Software Reliability Engineering (ISSRE 2004). pp. 417–428.

Hand, A., Sun, T., Barber, D., Hose, D., Macneil, S., 2009. Automated tracking of migrating cells in phase-contrast video microscopy sequences using image registration. Journal of Microscopy 234 (1), 62–79.

Höckendorf, B., Thumberger, T., Wittbrodt, J., Dec. 2012. Quantitative Analysis of Embryogenesis: A Perspective for Light Sheet Microscopy. Developmental Cell 23 (6), 1111–1120.

Huisken, J., Swoger, J., Del Bene, F., Wittbrodt, J., Stelzer, E. H., 2004. Optical sectioning deep inside live embryos by selective plane illumination microscopy. Science 305 (5686), 1007–1009.

Jemielita, M., Taormina, M. J., DeLaurier, A., Kimmel, C. B., Parthasarathy, R., 2012. Comparing phototoxicity during the development of a zebrafish craniofacial bone using confocal and light sheet fluorescence microscopy techniques. Journal of Biophotonics.

Jiang, H., Fels, S., Little, J. J., 2007. A linear programming approach for multiple object tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007).

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., Saul, L. K., 1999. An Introduction to Variational Methods for Graphical Models. Machine Learning 37 (2), 183–233.

Kachouie, N. N., Fieguth, P. W., 2007. Extended-Hungarian-JPDA: exact single-frame stem cell tracking. IEEE Transactions on Bio-Medical Engineering 54.

Kalman, R. E., Bucy, R. S., 1961. New results in linear filtering and prediction theory. Journal of Basic Engineering 83 (3), 95–108.

Kanade, T., Yin, Z., Bise, R., Huh, S., Eom, S., Sandbothe, M. F., Chen, M., 2011. Cell image analysis: Algorithms, system and applications. In: IEEE Workshop on Applications of Computer Vision (WACV).

Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Lellmann, J., Komodakis, N., Rother, C., 2013. A comparative study of modern inference techniques for discrete energy minimization problem. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Kappes, J. H., Savchynskyy, B., Schnörr, C., 2012. A Bundle Approach To Efficient MAP-Inference by Lagrangian Relaxation. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1688–1695.

Kari, G., Rodeck, U., Dicker, A. P., 2007. Zebrafish: an emerging model system for human disease and drug discovery. Clinical Pharmacology & Therapeutics 82 (1), 70–80.

Kaster, F., 2011. Image analysis for the life sciences - computer-assisted tumor diagnostics and digital embryomics. Ph.D. thesis, Heidelberg University.

Kausler, B., Schiegg, M., Andres, B., Lindner, M., Koethe, U., Leitte, H., Wittbrodt, J., Hufnagel, L., Hamprecht, F., 2012. A discrete chain graph model for 3d+t cell tracking with high misdetection robustness. In: 12th European Conference on Computer Vision (ECCV).

Kausler, B. X., 2010. Modeling of spectral peaks for mass-spectrometry-based proteomics. Master's thesis, Universities of Karlsruhe and Heidelberg.

Keller, P. J., March 2013. In vivo imaging of zebrafish embryogenesis. Elsevier Methods.

Keller, P. J., Dodt, H.-U., 2012. Light sheet microscopy of living or cleared specimens. Current Opinion in Neurobiology 22 (1), 138–143.

Keller, P. J., Schmidt, A. D., Santella, A., Khairy, K., Bao, Z., Wittbrodt, J., Stelzer, E. H., 2010. Fast, high-contrast imaging of animal development with scanned light sheet-based structured-illumination microscopy. Nature Methods 7 (8), 637–642.

Keller, P. J., Schmidt, A. D., Wittbrodt, J., Stelzer, E. H., 2008. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. Science 322, 1065–1069.

Kelley, J. E., 1960. The cutting-plane method for solving convex programs. Journal of the Society for Industrial & Applied Mathematics 8 (4), 703–712.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. The MIT Press.

Kolmogorov, V., Zabih, R., 2002. What energy functions can be minimized via graph cuts? In: Proceedings of the 7th European Conference on Computer Vision (ECCV 02). Springer, London, UK, pp. 65–81.

Krzic, U., Gunther, S., Saunders, T. E., Streichan, S. J., Hufnagel, L., Jun. 2012. Multiview light-sheet microscope for rapid in toto imaging. Nature Methods 9, 730–733.

Kschischang, F., Frey, B., Loeliger, H.-A., 2001. Factor graphs and the sum-product algorithm. Information Theory, IEEE Transactions on 47, 498 –519.

Kumar, M. P., Torr, P. H. S., 2008. An Analysis of Convex Relaxations for MAP Estimation of Discrete MRFs. Journal of machine learning research (JMLR) 10, 71–106.

Kunz-Schughart, L. A., Freyer, J. P., Hofstaedter, F., Ebner, R., 2004. The use of 3-d cultures for high-throughput screening: the multicellular spheroid model. Journal of Biomolecular Screening 9 (4), 273–285.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., Huang, F., 2006. A tutorial on energy-based learning. Predicting Structured Data.

Li, K., Chen, M., Kanade, T., Miller, E. D., Weiss, L. E., Campbell, P. G., 2008. Cell population tracking and lineage construction with spatiotemporal context. Medical image analysis 12 (5), 546.

Li, Y., Huang, C., Nevatia, R., 2009. Learning to associate: HybridBoosted multi-target tracker for crowded scene. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2009).

Lindner, M., 2011. A Machine Learning Approach to Improve Digital Embryo Analysis. Diploma thesis, University of Heidelberg.

Lou, X., Hamprecht, F. A., 2011. Structured learning for cell tracking. In: Twenty-Fifth Annual Conference on Neural Information Processing Systems (NIPS 2011).

Lou, X., Hamprecht, F. A., 2012. Structured learning from partial annotations. In: The 29th International Conference on Machine Learning (ICML 2012).

Lou, X., Kaster, F. O., Lindner, M. S., Kausler, B. X., Koethe, U., Jaenicke, H., Hoeckendorf, B., Wittbrodt, J., Hamprecht, F. A., 2011. DELTR: Digital embryo lineage tree reconstructor. In: IEEE International Symposium on Biomedical Imaging (ISBI 2011).

Lou, X., Koethe, U., Wittbrodt, J., Hamprecht, F. A., Jun. 2012. Learning to segment dense cell nuclei with shape prior. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2012). IEEE, pp. 1012–1018.

Luengo-Oroz, M., Rubio-Guivernau, J., Faure, E., Savy, T., Duloquin, L., Olivier, N., Pastor, D., Ledesma-Carbayo, M., Debarre, D., Bourgine, P., Beaurepaire, E., Peyrieras, N., Santos, A., 2012. Methodology for reconstructing early zebrafish development from in vivo multiphoton microscopy. IEEE Transactions on Image Processing 21 (4), 2335–2340.

Madigan, D., Perlman, M. D., Triggs, C. M., Levitz, M., 1995. c-separation in chain graphs.

Maizel, A., Von Wangenheim, D., Federici, F., Haseloff, J., Stelzer, E. H. K., 2011. High-resolution live imaging of plant growth in near physiological bright conditions using light sheet fluorescence microscopy. The Plant Journal 68 (2), 377–385.

Mavrakis, M., Rikhy, R., Lilly, M., Lippincott-Schwartz, J., 2008. Fluorescence imaging techniques for studying drosophila embryo development. Current Protocols in Cell Biology 39, 4.18.1–4.18.43.

Meijering, E., Dzyubachyk, O., Smal, I., van Cappellen, W. A., 2009. Tracking in cell and developmental biology. Seminars in Cell & Developmental Biology 20.

Melani, C., Peyrieras, N., Mikula, K., Zanella, C., Campana, M., Rizzi, B., Veronesi, F., Sarti, A., Lombardot, B., Bourgine, P., 2007. Cells tracking in a live zebrafish embryo. In: IEEE Engineering in Medicine and Biology Society (EMBS 2007).

Menze, B. H., Kelm, M. B., Masuch, R., Himmelreich, U., Bachert, P., Petrich, W., Hamprecht, F. A., 2009. A comparison of random forest and its gini importance with standard chemometric methods for the feature selection and classification of spectral data. BMC Bioinformatics 10 (1), 1–16.

Mikula, K., Peyriéras, N., Remešíková, M., Stašová, O., 2011. Segmentation of 3d cell membrane images by pde methods and its applications. Computers in Biology and Medicine 41 (6), 326–339.

Miura, K., 2005. Tracking movement in cell biology. In: Rietdorf, J. (Ed.), Microscopy Techniques. Vol. 95 of Advances in Biochemical Engineering. Springer Berlin Heidelberg, pp. 267–295.

Murphy, K., Weiss, Y., Jordan, M. I., 1999. Loopy belief-propagation for approximate inference: An empirical study. In: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI). pp. 467–475.

Murray, J. I., Bao, Z., Boyle, T. J., Waterston, R. H., 2006. The lineaging of fluorescently-labeled caenorhabditis elegans embryos with starrynite and acetree. Nature protocols 1 (3), 1468–1476.

Neumann, B., Walter, T., Heriche, J. K., Bulkescher, J., Erfle, H., Conrad, C., Rogers, P., Poser, I., Held, M., Liebel, U., Cetin, C., Sieckmann, F., Pau, G., Kabbe, R., Wünsche, A., Satagopam, V., Schmitz, M. H., Chapuis, C., Gerlich, D. W., Schneider, R., Eils, R., Huber, W., Peters, J. M., Hyman, A. A., Durbin, R., Pepperkok, R., Ellenberg, J., 2010. Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes. Nature 464, 721–727.

Olivier, N., Luengo-Oroz, M. A., Duloquin, L., Faure, E., Savy, T., Veilleux, I., Solinas, X., Débarre, D., Bourgine, P., Santos, A., Peyriéras, N., Beaurepaire, E., 2010. Cell lineage reconstruction of early zebrafish embryos using label-free nonlinear microscopy. Science 329 (5994), 967–971.

Ong, L., Jr., M. A., Asada, H., 2010. Tracking of cell population from time lapse and end point confocal microscopy images with multiple hypothesis kalman smoothing filters. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 71–78.

Padfield, D., Rittscher, J., Roysam, B., 2011. Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis. Medical Image Analysis 15 (4).

Padfield, D., Rittscher, J., Thomas, N., Roysam, B., 2009. Spatio-temporal cell cycle phase analysis using level sets and fast marching methods. Medical Image Analysis 13 (1), 143–155.

Papadimitriou, C. H., Steiglitz, K., 1998. Combinatorial optimization: algorithms and complexity. Dover Publications.

Pearl, J., 1982. Reverend bayes on inference engines: A distributed hierarchical approach. In: AAAI. pp. 133–136.

Pearl, J., 1987. Evidential reasoning using stochastic simulation of causal models. Artificial Intelligence 32 (2), 245–257.

Planchon, T. A., Gao, L., Milkie, D. E., Davidson, M. W., Galbraith, J. A., Galbraith, C. G., Betzig, E., 2011. Rapid three-dimensional isotropic imaging of living cells using Bessel beam plane illumination. Nature Methods 8 (5), 417–423.

Richards, J. L., Zacharias, A. L., Walton, T., Burdick, J. T., Murray, J. I., 2013. A quantitative model of normal caenorhabditis elegans embryogenesis and its disruption after stress. Developmental Biology 374 (1), 12–23.

Rokach, L., 2010. Ensemble-based classifiers. Artificial Intelligence Review 33 (1-2), 1–39.

Santi, P. A., 2011. Light sheet fluorescence microscopy - a review. Journal of Histochemistry & Cytochemistry 59 (2), 129–138.

Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., Preibisch, S., Rueden, C., Saalfeld, S., Schmid, B., Tinevez, J.-Y., White, D. J., Hartenstein, V., Eliceiri, K., Tomancak, P., Cardona, A., 2012. Fiji: an open-source platform for biological-image analysis. Nature Methods 9 (7), 676–82.

Schlesinger, M., 1976. Sintaksicheskiy analiz dvumernykh zritelnikh singnalov v usloviyakh pomekh (syntactic analysis of two-dimensional visual signals in noisy conditions). Kibernetika 4, 113–130.

Schrijver, A., 1998. Theory of linear and integer programming. Wiley.

Shafer, G., Shenoy, P., 1990. Probability propagation. Annals of Mathematics and Artificial Intelligence 2 (1–4), 327–351.

Shenoy, P., Shafer, G., 2008. Axioms for probability and belief-function propagation. In: Yager, R., Liu, L. (Eds.), Classic Works of the Dempster-Shafer Theory of Belief Functions. Vol. 219 of Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, pp. 499–528.

Siedentopf, H., Zsigmondy, R., 1902. Über Sichtbarmachung und Größenbestimmung ultramikoskopischer Teilchen, mit besonderer Anwendung auf Goldrubingläser. Ann. Phys. 315 (1), 1–39.

Smal, I., Meijering, E., Draegestein, K., Galjart, N., Grigoriev, I., Akhmanova, A., Van Royen, M., Houtsmuller, A., Niessen, W., 2008. Multiple object tracking in molecular bioimaging by Rao-Blackwellized marginal particle filtering. Medical Image Analysis 12 (6), 764–777.

Sommer, C., Straehle, C., Koethe, U., Hamprecht, F. A., 2011. ILASTIK: Interactive learning and segmentation toolkit. In: IEEE International Symposium on Biomedical Imaging (ISBI 2011).

Studeny, M., Bouckaert, R. R., 1998. On chain graph models for description of conditional independence structures. Annals of Statistics, 1434–1495.

Sulston, J., Schierenberg, E., White, J., Thomson, J., 1983. The embryonic cell lineage of the nematode caenorhabditis elegans. Developmental Biology 100 (1), 64–119.

Sulston, J. E., December 8 2002. C. elegans: the cell lineage and beyond. Nobel Lecture.

Sutton, C., McCallum, A., 2012. An introduction to conditional random fields. Foundations and Trends in Machine Learning 4 (4), 124.

Thiele, T. N., 1880. Sur la compensation de quelques erreurs quasi-systematiques par la methode des moindres carres. University of Michigan.

Tomer, R., Khairy, K., Amat, F., Keller, P. J., Jun. 2012. Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy. Nature Methods 9, 755–763.

Tomer, R., Khairy, K., Keller, P. J., 2011. Shedding light on the system: Studying embryonic development with light sheet microscopy. Current Opinion in Genetics & Development 21 (5), 558–565.

Truong, T. V., Supatto, W., Koos, D. S., Choi, J. M., Fraser, S. E., 2011. Deep and fast live imaging with two-photon scanned light-sheet microscopy. Nature Methods 8 (9), 757–760.

Tschirren, J., Palágyi, K., Reinhardt, J. M., Hoffman, E. A., Sonka, M., 2002. Segmentation, skeletonization, and branchpoint matching–a fully automated quantitative evaluation of human intrathoracic airway trees. In: Medical Image Computing and Computer-Assisted Intervention 2002 (MICCAI). Springer, pp. 12–19.

Tserevelakis, G. J., Filippidis, G., Megalou, Evgenia V.and Fotakis, C., Tavernarakis, N., 2011. Cell tracking in live caenorhabditis elegans embryos via third harmonic generation imaging microscopy measurements. Journal of Biomedical Optics 16 (4).

Voie, A. H., Burns, D. H., Spelman, F. A., 1993. Orthogonal-plane fluorescence optical sectioning: three-dimensional imaging of macroscopic biological specimens. Journal of Microscopy 170 (Pt 3), 229–236.

Wainwright, M., Jaakkola, T., Willsky, A., 2005. Map estimation via agreement on trees: message-passing and linear programming. IEEE Transactions on Information Theory 51 (11), 3697–3717.

Wainwright, M. J., Jordan, M. I., 2007. Graphical Models, Exponential Families, and Variational Inference. Foundations and Trends in Machine Learning 1, 1–305.

Walker, R., Jackway, P., 1996. Statistical geometric features-extensions for cytological texture analysis. In: Proceedings of the 13th International Conference on Pattern Recognition (ICPR). Vol. 2. pp. 790–794.

Weber, M., Huisken, J., 2011. Light sheet microscopy for real-time developmental biology. Current Opinion in Genetics & Development 21 (5), 566–572.

Wolpert, L., 2007. Principles of Development. Oxford University Press.

Wu, Y., Ghitani, A., Christensen, R., Santella, A., Du, Z., Rondeau, G., Bao, Z., Colón-Ramos, D., Shroff, H., Oct. 2011. Inverted selective plane illumination microscopy (iSPIM) enables coupled cell identity lineaging and neurodevelopmental imaging in Caenorhabditis elegans. Proceedings of the National Academy of Sciences of the United States of America 108 (43), 17708–13.

Yang, X., Li, H., Zhou, X., 2006. Nuclei segmentation using marker-controlled watershed, tracking using mean-shift, and kalman filter in time-lapse microscopy. IEEE Transactions on Circuits and Systems I: Regular Papers 53 (11), 2405–2414.

Yedidia, J. S., Freeman, W. T., Weiss, Y., 2005. Constructing free energy approximations and generalized belief propagation algorithms. IEEE Transactions on Information Theory 51, 2282–2312.

Yilmaz, A., Javed, O., Shah, M., 2006. Object tracking: A survey. ACM Computing Surveys (CSUR) 38.

Zeiss, October 2012. Carl zeiss introduces lightsheet z.1 light sheet microscope system. Zeiss Press Release.

Zhang, N. L., Poole, D., May 1994. A simple approach to bayesian network computations. In: Proceedings of the Tenth Biennial Canadian Artificial Intelligence Conference (AI 94). pp. 171–178.