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

presented by

Systems Biologist (MSc.)	Yin Cai
born in	Shanghai, PR China
Oral examination on	13. Jan. 2016

Mapping the Dynamic Protein Network of Dividing Cells in Space and Time

Referees: Dr. Wolfgang Huber

Prof. Dr. Ursula Kummer

Some of the contents of this thesis were used in the manuscript Cai Y, Hossain MJ, Hériché J-K, Politi AZ, Koch B, Wachsmuth M, Nijmeijer B, Mergenthaler J, Ellenberg J: ''Towards a model of the canonical human dividing cell,'' *in preparation.*

Part of the work presented in this thesis was jointly performed by Dr. M. Julius Hossain and Dr. Jean-Karim Hériché.

Acknowledgments

This dissertation benefited from the fruitful comments and support from many people. First and foremost, I want to thank my adviser Dr. Jan Ellenberg for his continued support and encouragement. I am grateful that he gave me the opportunity for working on this unique project.

Dr. Julius Hossain, through years of common research, has become both an intellectual companion and a friend to me. I want to thank him for our successful joint research and many hours of spirited and entertaining discussions during long hours at work.

I conducted my work in the environment of the MitoSys team. I benefited greatly from support by and comments from all members of the team, especially from Dr. Jean-Karim Hériché in data analysis, Dr. Antonio Politi and Dr. Malte Wachsmuth in automatic imaging, and Dr. Birgit Koch in cell line production and culturing.

I am especially grateful to my Thesis Adviser Committee made up of Dr. Wolfgang Huber, Dr. Lars Hufnagel, Dr. Ursula Kummer and Dr. Daniel Gerlich for the valuable feedback. Special thanks go to Dr. Wolfgang Huber and Dr. Bernd Fischer, former member of the Huber group, for many helpful discussions of my work.

Part of the project was performed at the Advanced Light Microscopy Facility. I appreciate support by and discussions with Beate Neumann and Volker Hilsenstein.

I also want to thank our collaborators Christoph Sommer, Rudolf Hoefler from the Gerlich lab for their kind support in CellCognition development, Ina Poser from the Hyman lab for providing the BAC cell lines, and Andreas Christ from the Thery lab for producing micropatterns. Many thanks to Markus Winter and Birgit Koch for the revision of the thesis.

I have been privileged to be a member of the *EMBL International PhD Programme*. In this context, I would like to thank the program for financial support. I also want to thank Helke Hillebrand, Milanka Stojkovic, Matija Grgurinovic, and Meriam Bezohra for making this place the nourishing environment that it is.

The time of my dissertation would not have been half as much fun without my fellow grad students and the Ellenberg Lab members Stephanie Alexander, Nathalie Daigle, Manuel Eguren, Mayumi Isokane, Vilma Jemenez Sabinina, Moritz Küblbeck, Bianca Nijmeijer, Oyvind Odegard, Shotaro Otsuka, Judith Reichmann, Maria Julia Roberti, Nike Walther, Wanqing Xiang, Adele Rickerby, Thomas Walter, Anna Szymborska, Aicha Metchat, Robert Mahen, Andrea Boni, Julia Mergenthaler, Petr Strnad and Sylvia Schattschneider. Thank you for countless stimulating discussions, great retreat trips and many hours of laughter over lunch and Christmas party.

Finally, I want to express my deep gratitude to my family for their support and encouragement. My husband Volker Tjaden has always been there for me through the ups and downs of this project. My little daughter Viola Yida Tjaden has motivated me with her smiles. Thank you!

Abstract

Live cell imaging is a powerful tool for studying the distribution and dynamics of proteins. However, due to the difficulties in absolute quantification and standardization of data obtained from individual cells, it has not been used to map large sets of proteins that carry out dynamic cellular functions. Cell division is a good example of this challenge for an essential cellular function, as rapid changes in protein localization and protein interactions result in dramatic changes to subcellular structures and cellular morphology, which in turn influence the behavior of the enclosed proteins.

Here, I report an integrated experimental and computational pipeline to map the dynamic protein network of dividing human cells in space and time. Using 3D live confocal microscopy, I imaged human cell lines that stably expressed fluorescently tagged mitotic proteins throughout mitosis. To obtain the absolute quantities of protein abundance with high subcellular resolution over time, the microscopy pipeline was calibrated by fluorescence correlation spectroscopy (FCS).

Cell and chromosome volumes were segmented as references of cellular context for temporal and spatial alignment based on fluorescent landmarks. Together with my colleague Julius Hossain, we computationally generated a canonical model of mitotic progression for both kinetics (''mitotic standard time'') and morphology (''mitotic standard space'') by averaging and kinetically and geometrically parametrizing many registered dividing cells. The resulting model enabled us to subdivide the mitotic process into 20 characteristic kinetic steps and integrate our complete proof of concept dataset of 13 mitotic proteins imaged in over 300 dividing cells, represented as the 3D protein localization probability of each protein over time.

To measure localization similarities between different proteins and make predic-

tions about their dynamic interactions, the integrated data was then mined using supervised as well as unsupervised machine learning. The power of this approach was demonstrated by our ability to automatically identify the major subcellular localizations of all proteins in the dataset and quantify protein fluxes between subcellular compartments and structures. Due to the quantitative nature of our imaging data, we were able to estimate the abundance of each protein in mitotic structures and complexes such as kinetochores, centrosomes, and the midbody, and determine the order and kinetics of their formation and disassembly.

The integrated computational and experimental method I present in my thesis is generic and scalable and makes many dynamic cellular processes amenable to dynamic protein network analysis even for large numbers of components. The pipeline provides a powerful instrument for analyzing large sets of quantitative live imaging data of fluorescently tagged proteins. It allows the systematic mapping and prediction of dynamic protein networks that drive complex cellular processes such as mitosis, thus promoting our understanding of the mechanisms by which many molecules together achieve spatio-temporal regulation.

Zusammenfassung

Mikroskopie an lebenden Zellen ist eine leistungsfähige Methode zur Untersuchung der Verteilungsdynamik von Proteinen. Wegen der Schwierigkeiten bei der Quantifizierung und Standardisierung von Daten, die von einzelnen Zellen erhoben wurden, konnte die Technik aber noch nicht für die Abbildung größerer Eiweißgruppen verwendet werden, die zusammen dynamische Funktionen innerhalb der Zelle erfüllen. Ein gutes Beispiel für die damit verbundenen Herausforderungen ist die Zellteilung, eine essenzielle Zellfunktion. Während der Zellteilung ändert eine Reihe von Eiweißen durch ihre räumlichen Verteilungen und Interaktionen miteinander dramatisch die Morphologie der Zellen und der intrazellulären Strukturen, die wiederum das Verhalten der entsprechenden Eiweiße beeinflussen.

Ich stelle hier über eine Reihe integrierter experimenteller und numerischer Verfahren vor, die das dynamische Proteinnetzwerk einer sich teilenden Zelle sowohl räumlich als auch zeitlich abbilden. Mit Hilfe eines konfokalen Mikroskops habe ich lebende humane Zellen, die durch genetische Modifikationen fluoreszierende mitotische Eiweiße stabil beinhalten, während der Zellteilung in hoher Auflösung in 3D aufgenommen. Die Aufnahme wurde durch Fluorescence Correlation Spectroscopy (FCS) kalibriert, damit die absolute Anzahl der Proteine bestimmt werden konnte. Anhand der fluoreszierenden Markierungen wurden die zellulären und chromosomalen Volumina segmentiert und als Vorlage für das zelluläre Umfeld für die räumliche und zeitliche Registrierung genutzt. Zusammen mit meinem Kollegen Julius Hossain habe ich numerisch ein kanonisches Modell über den zeitlichen („Standardzeit für die Mitose“) und den morphologischen („Standardraum für die Mitose“) Verlauf einer Zellteilung aufgebaut. Dafür haben wir aus allen aufgenommenen Zellen eine

Durchschnittszelle berechnet, und sie wiederum parametrisiert. Das Modell erlaubt es uns, den Verlauf der Zellteilung in 20 charakteristische Abschnitte zu teilen und den gesamten Proof-of-Concept Datensatz von 13 mitotischen Eiweißen aus über 300 sich teilenden Zellen zu integrieren. Für jedes Protein konnten wir über den Verlauf der Teilung eine Reihe dreidimensionaler Karten seiner räumlichen Verteilung erstellen.

Durch die Messung der Ähnlichkeit in der Dynamik unterschiedlicher Proteine haben wir versucht, deren dynamische Interaktionen vorherzusagen. Dafür wurden die integrierten quantitativen Verteilungsdaten der Proteine mit Maschinenlernverfahren analysiert. Wir konnten zeigen, dass unsere Methoden die intrazellulären Lokalisierungen aller getesteten Proteine automatisch erkennen. Auch die Strömungen der Eiweiße zwischen verschiedenen subzellulären Strukturen konnten quantifiziert werden. Da unsere Aufnahmen kalibriert waren, konnten wir sowohl die Anzahl der Proteinmoleküle in mitotischen Strukturen und anderen Komplexen, wie z.B. Kinetochore, Centrosomen oder Midbody, als auch die Reihenfolge und Kinetik deren Auf- und Abbaus schätzen.

Diese integrierten numerischen und experimentellen Methoden in meiner Arbeit hier sind allgemein für die Analyse dynamischer Proteinnetzwerke vieler zellulärer Prozesse mit vielen Komponenten anwendbar. Die Pipeline ist ausreichend leistungsfähig, um große Mengen quantitativer Mikroskopiedaten von mit Fluorophore markierten Proteinen in lebenden Zellen zu verarbeiten. Diese Verfahren können die Struktur der Proteinnetzwerke, die komplexe zelluläre Prozesse wie Zellteilung steuern, vorhersagen und abbilden, und uns damit helfen zu verstehen, wie diese Prozesse sowohl räumlich als auch zeitlich reguliert werden.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

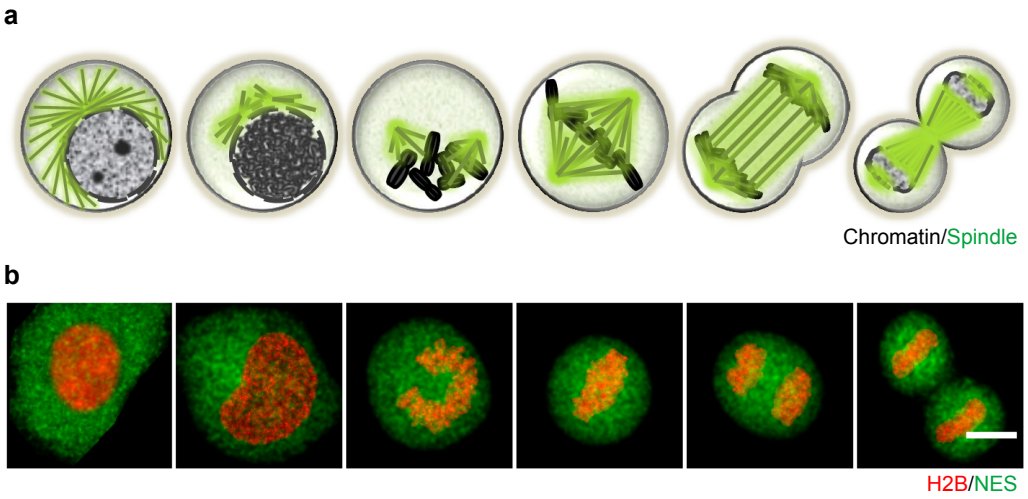
1.1 Systems biology of mitosis

1.1.1 Mitosis is an essential and complex process of life

Cell division, also called mitosis, is one of the fundamental processes of life (Morgan, 2007). It drives the development of all multicellular organisms, where a single fertilized cell undergoes a series of divisions to develop into a complex organism with cells that can have very different functions (Herbert et al., 1995). The regulated division of cells is also crucial for the regeneration of tissues, such as blood and epithelium, throughout the entire life of the organism (Goranov and Amon, 2010; Thummel et al., 2008). Defects in mitosis can therefore cause severe diseases. For example, deficient control of cell division in vertebrates has been related to tumorigenesis in cancer (Weaver and Cleveland, 2005), and errors in division during early development can cause infertility or abortions (Kim and Kao, 2005; Vorsanova et al., 2005). Moreover, a number of neurological diseases have been linked to the consequences of aberrant mitosis (Noatynska et al., 2012). Thus, research on mitosis is vital for our basic understanding of life and will make important contributions to the understanding and treatment of several diseases.

During mitosis, cells undergo dramatic morphological changes to pass the duplicated genome equally to the two daughter cells (Figure 1.1). The division of animal cells is traditionally sub-divided into five stages based on the nuclear morphology: prophase, prometaphase, metaphase, anaphase and telophase (Alberts et al., 1997), as defined in Campbell and Reese (2003): prophase starts with the condensation of the duplicated chromatin within the intact nucleus and ends when the nuclear envelope starts to break down. During prometaphase the centrosomes move towards the opposite ends of the cell and organize the formation of the mitotic spindle. In metaphase,

Figure 1.1: Mitosis in animal cells.



(a) Schematic illustration of a dividing animal cell. From left to right, interphase, prophase, prometaphase, metaphase, anaphase and telophase are shown respectively. (b) Confocal fluorescence image of a dividing HeLa Kyoto cell in different mitotic stages. Maximum projection of z-stack images, filtered and segmented, scale bar: 10 µm. A different cell was selected for interphase as the mitotic cell.

the chromosomes reach their maximum condensation and congress in the middle of the cell to form the metaphase plate. The separation of the sister chromatids at the centromeres marks the onset of Anaphase, in which the chromosomes travel to the opposite sides of the cell. And finally, telophase defines the time when the chromosomes de-condense and the nuclear envelope reforms. In parallel to these classic five phases of nuclear division and genome segregation, cytokinesis, the separation of the cytoplasm, starts with the ingression of the cleavage furrow and ends with the complete disconnection of the two daughter cells by abscission. In most cells the mitotic division, also called M-phase, occupies a minor part of the cell cycle, and the time between two divisions is referred to as interphase (Reece et al., 2011). In the most common human cultured cell models, such as the HeLa cells employed in this study, the cell cycle lasts between 20 and 24 hours, and the duration of mitosis from microscopically detectable chromatin condensation until the end of telophase takes about an hour (Cooper Geoffrey, 2000).

Cell division is a complex process driven by a large number of proteins acting in a coordinated manner (Dephoure et al., 2008; Nigg, 2001). Important mitotic subcellular structures, such as the spindle, centrosomes and kinetochores, are formed

by many specific proteins in a highly organized way (Karsenti and Vernos, 2001; Satisbury, 1995; Doxsey et al., 2005; Rieder and Salmon, 1998). These structures in turn provide platforms for the localization and functions of many regulatory proteins, which can influence the morphology of the dividing cell and position other subcellular structures (Werner et al., 2013; O'Connell and Wang, 2000; Lázaro-Diéguez et al., 2015). The correct formation of these subcellular structures and the timed localization of proteins to these structures are crucial for the success of the division.

As in the example of kinetochores, a large number of proteins need to be correct localized to ensure their function of controlling the temporal progression of the cell division in a coordinative and collaborative manner. To prevent the unequal distribution of sister chromosomes in the two daughter nuclei, chromosome segregation is delayed by the spindle assembly checkpoint (SAC) until the biorientation of all sister chromatid pairs to spindle microtubules is complete (Musacchio and Salmon, 2007; Hauf and Watanabe, 2004). Upon the nuclear envelope break down, active Mps1 directs checkpoint proteins to kinetochores which are not attached by microtubules (Lan and Cleveland, 2010). The core SAC machinery undergoes enzymatic and/or conformational activation at kinetochores to form the mitotic checkpoint complex (MCC) composed of Mad2, Bub3, BubR1 and Cdc20 (Musacchio and Salmon, 2007; Lara-Gonzalez et al., 2012). The activation of Mad2 stabilizes the cytoplasmic anaphase-promoting complex/cyclosome (APC/C) inhibitory complexes including Cdc20 (Maciejowski et al., 2010) and prevents the activation of both the APC/C and the E3 ubiquitin ligase. Once all sister chromosomes are attached on microtubules and bi-oriented, SAC releases Cdc20 which activates the APC/C to digest both securin and Cyclin B. The degradation of the checkpoint protein stops the inhibition of separase which then digests the cohesin tying up the sister chromosome pairs and promotes the chromosome segregation. The reduction of the Cyclin B allows the activation of Cdk1 which drives the mitotic exit (Musacchio and Salmon, 2007).

1.1.2 Understanding mitosis at the systems level

Research on mitosis is carried out by a large number of laboratories, but has traditionally focused on a specific step or only a few components of the mitotic machinery. This has led to fragmentation of research results and made it difficult to integrate the data obtained. Only a fraction of the proteins required for mitosis in human

cells were known until a few years ago, and a systematic molecular understanding of mitosis is still missing. In 2010, the MitoCheck project screened, for the first time, the entire human genome to identify mitotic proteins. After inactivating the expression of each gene by RNA interference, they scored mitotic chromosome segregation errors by live cell imaging. Using an automatic segmentation and classification pipeline, different classes of phenotypes were identified, and genetic effects could be compared by phenotypic similarity with each other. Almost 600 genes were validated as mitotic hits because knocking down their expression caused a reproducible and significant defect in chromosome segregation (Neumann et al., 2010). This was the first large scale study of human mitosis at the systems level. But while the data provided researchers with a starting point for the mitotic functions of unknown genes based on well characterized genes in the same phenotypic group (Lo et al., 2015), they rarely allowed to predict molecular mechanism. Further bioinformatics data integration and mining methods, developed based on the MitoCheck data set, allowed the prediction of additional genes involved in particular mitotic processes such as chromosome compaction (Hériché et al., 2014). However, for the large majority of the now identified mitotic genes the molecular mechanism behind their phenotype remains unknown.

Thus, systematic analysis of the proteins encoded by mitotic genes is essential for understanding the molecular mechanism that explains their functional requirement for chromosome segregation. Due to the very dynamic nature of the multistep cell division process, knowing the changes in abundance and subcellular localization and interactions of the identified mitotic proteins during mitosis is the key to understand mitosis as a complex multistep regulatory system, where hundreds of components are regulated dynamically in space and time. A first effort in this direction was made as part of the Mitocheck project, based on the stable expression of fluorescence and affinity tagged mitotic genes in bacterial artificial chromosomes (BAC) (Kittler et al., 2005) in HeLa cells (Poser et al., 2008). Mass spectroscopy was used to identify interaction partners of a number of known mitotic proteins in interphase and M-phase, and immunofluorescence microscopy of fixed HeLa cells and manual annotation of the intracellular location of around 100 mitotic proteins in a subset of mitotic stages was used to predict functional clusters for these proteins (Hutchins et al., 2010). However, this initial effort lacked the temporal resolution required to track the rapid formation

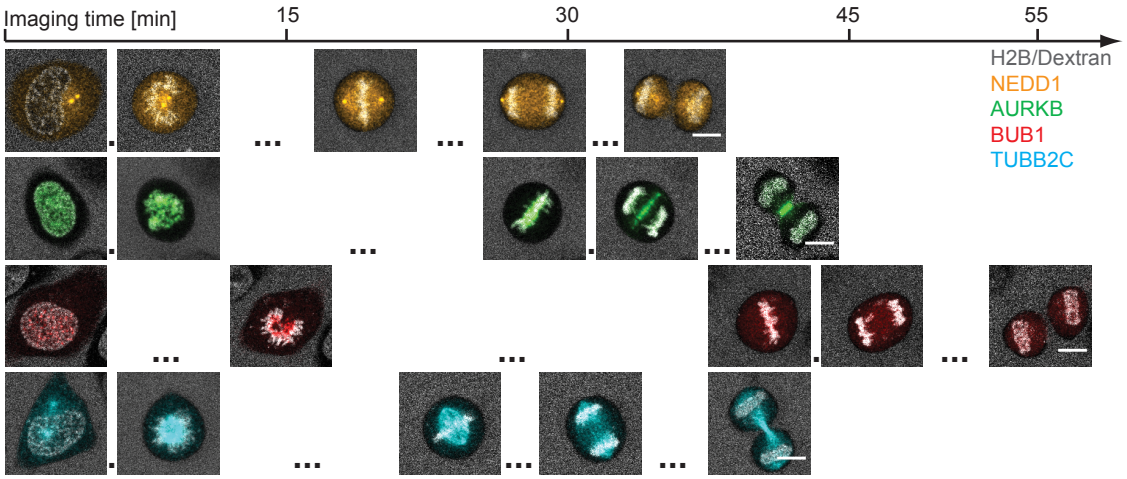
and disassembly of mitotic protein complexes that occurs on the seconds to minutes time scale and, due to the limited throughput of the annotation pipeline, did not cover the whole mitotic proteome identified in the RNAi screen. Furthermore the qualitative localization data and its manual annotation made integration with data from other laboratories impossible and precluded a computational analysis of the data. A quantitative dynamic study of the mitotic proteome in live cells could in principle address these shortcomings, but remained elusive due to technical challenges in high-throughput and high-resolution quantitative imaging of dividing cells as well as in computational data integration and mining.

1.1.3 Challenges in dynamic proteomics

Systems biology of the proteome can be done at different levels. At the cellular level, the structure of the proteome including the interaction network and the subcellular localization of each protein has to be mapped. Next, the dynamics within the structure, e.g. the kinetics of protein interactions and potential changes in localization should be measured before the experiments are designed for analyzing the logic behind the network (Kitano, 2007). However, understanding mitosis at the proteomic systems level is particular challenging since the structure of the mitotic proteome is known to be extremely dynamic. During mitosis, for many proteins, both their localization and abundance as well as their interaction partners are constantly changing. One prominent example is the ‘‘chromosome passenger complex’’ of Aurora B kinase and its regulators that relocalizes from the cytoplasm to the kinetochore, then to the midplane, and finally back to the cytoplasm (Murata-Hori et al., 2002). Moreover, the morphology of the cell and almost all major mitotic subcellular structures is changing dramatically (De Souza and Osmani, 2007; Ando et al., 2008) which puts changing spatial constraints on the behavior of all proteins (Cadart et al., 2014; Charnley et al., 2013; Kiyomitsu and Cheeseman, 2013).

The key to access such a dynamic proteome is to perform measurements for all proteins under exactly the same spatio-temporal constraints, i.e. at the same mitotic time and in the same mitotic space, and therefore dynamic three-dimensional data with good subcellular resolution are required. However, individual cells divide with different spatial and temporal dynamics (Figure 1.2). Traditionally, proteins are compared in cells assigned to the same of the five classic mitotic stages as described

Figure 1.2: Spatio-temporal variation of mitotic HeLa Kyoto cells.



Scale bar: 10 µm.

in 1.1.1, where staging is done manually, and therefore subjective to bias, and too coarsely to capture important kinetic transitions. The lack of cell standardization methods in space and time, which is necessary to integrate data on many proteins from many different individual cells, has for a long time been the bottleneck for a comprehensive systems biology of mitosis.

1.2 Methods for protein interaction and network studies

1.2.1 Mass spectrometry

With the progressive reduction in cost for mass spectrometers and their accompanying increase in capability, mass spectrometry has become one of the most widely used methods to quantitatively study protein interactions. Proteins of interest can simply be purified from the cell using affinity approaches, and all of their binding partners that are co-extracted can then be detected by mass spectrometry (Pandey and Mann, 2000; Aebersold and Mann, 2003). The technology is not only powerful for analyzing the binding partners for few proteins of interest, but also used as the gold standard for analysis of large scale protein interactom (Kito and Ito, 2007). This needs a large number of proteins being tagged with affinity baits. Although in yeast, affinity baits can be introduced to hundreds of proteins easily, and the ensemble of protein interactions can be analyzed at once since many years (Ho et al., 2002).

However, tagging a series of proteins with affinity tags in mammalian cells is not trivial. Only in recent years, until the technology of both mass spectrometry and generation of mammalian cell lines expressing engineered fusion proteins have been further improved, large scale studies on protein abundance (Beck et al., 2011) and interactions (Hutchins et al., 2010) could be performed in human cell lines.

Although mass spectrometry is an effective technology to quantitatively study the proteome and interactome, several disadvantages of the technique prevent its application to the proteome of a very dynamic process such as mitosis. Since mass spectrometry needs a relatively large amount of protein to be detected, proteome scale data at the single cell level are still impossible to obtain, except for very special giant cells such as amphibian oocytes (Wang and Bodovitz, 2010). Therefore thousands of mitotic cells have to be collected to generate one sample at one stage of mitosis, and studies typically use nocodazole to artificially arrest cells in prometaphase. As nocodazole disrupts spindle formation it is unclear in how far this state still resembles a natural mitotic stage. Arresting the cell at a specific stage obviously also fails to cover all mitotic transitions in proteome structure. Furthermore, transient and weak interactions, which are typical for dynamic regulatory systems such as those between phosphorylation partners, are often not preserved in the necessary biochemical purification procedures. It has even been questioned in principle if cell extracts can reflect the interaction environment present inside a living cell (Vasilescu et al., 2004; Tagwerker et al., 2006). Thus, for studying the dynamic interactome in mitotic cells, other technologies than affinity purification followed by mass spectrometry need to be developed.

1.2.2 Immunostaining based imaging

An alternative approach to investigate the structure of a protein interaction network is through the localization of its components. Interactions are often assumed as the cause for co-localization of protein pairs or groups (Dunn et al., 2011; Fay et al., 1997; Manders et al., 1992; van Steensel et al., 1996). Imaging of immunostained fixed cells was successfully applied to mammalian cells for the prediction of cell growth rates (Kafri et al., 2013).

To investigate cell division, Hutchins et al. (2010) immunostained over 100 different cell lines with a GFP tagged protein and manually assigned both the localization of

the protein as well as the mitotic stage of the cells. However, as discussed before, the pipeline was manual, too coarse in time resolution, and biased and non-quantitative in spatial mapping. A more recently developed powerful alternative would be to detect the co-localization of many proteins directly within the same cell using multiple rounds of immunostaining (unpublished data, Pepperkok lab, EMBL Heidelberg). However, the limited availability of validated antibodies against all proteins does currently make this approach not easily scalable for hundreds of components, and it suffers from the same lack of temporal resolution as Hutchins et al. (2010).

1.2.3 Live cell imaging

The best way to study dynamic protein interactions in their physiological environment is currently to use live cell imaging. The co-localization of proteins can be studied directly by live imaging of cells co-expressing proteins fused to different fluorophores (Tanaka et al., 2010). Interactions between proteins can even be more directly assessed using fluorescence resonance energy transfer (FRET) or fluorescence cross-correlation spectroscopy (FCCS). FRET detects the Förster energy transfer when two protein molecules bring their attached fluorephores within less than 10 nm from each other (Stryer and Haugland, 1967). However, since FRET not only depends on the proximity but also on the orientation of the two fluorophores used, it suffers from a large false negative rate if generic fluorescence tagging approaches are used. That makes it unsuitable for large scale interaction studies. FCCS measures the correlation in the diffusive movement of two proteins, each labeled with a different fluorophore, through the subfemtoliter confocal volume of a laser scanning microscope (Schwille, 2001; Bacia et al., 2006). It only requires the putative interactors to be fluorescently labeled and is therefore well-suited and has been applied to larger scale as well as single cell time-resolved studies (Wachsmuth et al., 2015). FCCS measurements are quantitative, reflect the dynamics of the interaction, and have been used for analyzing small protein networks (Boeke et al., 2014). Both FRET and FCCS can be applied to live cells and the read outs can be interpreted in the context of a dynamic cellular process (Sekar and Periasamy, 2003; Michelman-Ribeiro et al., 2010; Pramanik, 2004).

Live imaging methods suffer from the fact that only a very limited number of fluorophores suitable for protein tagging in living cells are available, limiting such

studies practically to one or at most two interaction pairs per cell. In addition to capture cellular heterogeneity, each interaction should be sampled in tens of cells for statistical robustness and richness of the data. The bottleneck of using live cell imaging for proteomic studies of mammalian cells is therefore the low throughput in the production of cell lines that express two fluorescent fusion proteins that replace the endogenous unlabeled copies and ensure functionality and physiological expression as well as maximal signal for an interaction. For the mitotic proteome with about 600 relevant proteins, nearly 180,000 cell lines would be needed to cover all possible binary protein interactions. Thus, resolving the coarse structure of the dynamic proteome based on dynamic localization of individual proteins and predicting likely interactors in clusters is essential to guide the production of cell lines for FCCS measurements of the dynamics of each interaction cluster.

1.3 Standardization of cellular context

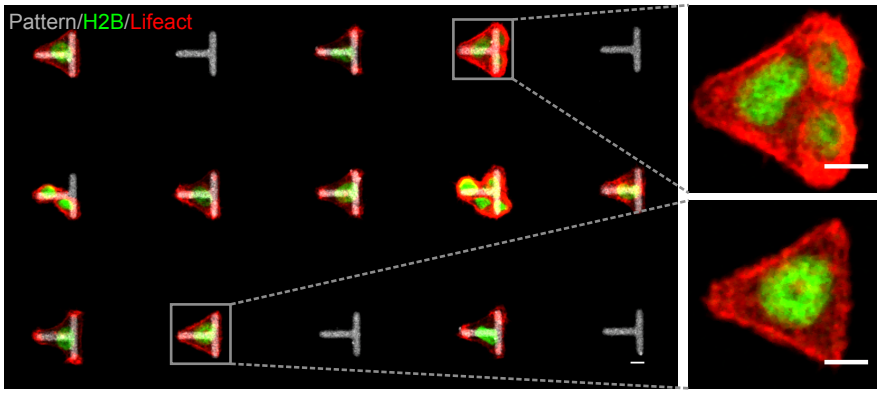
1.3.1 Micropatterns constrain the cellular geometry

In order to make protein distribution data in different individual cells comparable, the cellular geometry has to be standardized. Since cellular morphology of adherent cells is strongly influenced by the extracellular matrix, micropatterns were reported to be a solution for the morphological standardization of the mammalian cell (Whitesides et al., 2001). The shape of an individual cell, which grows on a micrometer-scaled fibronectin pattern printed on glass, is constrained by the shape of the pre-designed pattern (Théry, 2010; Théry and Piel, 2009). Using this method, Schauer et al. (2013, 2010) have followed the distribution of organelles in multiple cells via the expression of tagged marker proteins (e.g. CD63 and Rab6). Integrating each protein's localization in single cells into a probability density map allowed for a statistical comparison between the distributions of organelles under different conditions. Micropatterns were also used for constraining the mitotic morphology, such as the size, orientation and even the temporal progression of the dividing cell (Fink et al., 2011). However these studies also made it clear that even on micropatterns a lot of variability remains at the submicron scale of organelles, which makes averaging between individual cells very challenging, and led only to very low resolution maps for Golgi associated proteins.

Furthermore, no micropattern has been reported to be able to constrain the orientation of the assembling mitotic spindle in early prometaphase (Théry et al., 2005). The positions of the centrosomes relative to condensing chromosomes appears to be random at the moment of nuclear envelope breakdown, which leads to prophase and prometaphase having the most cell to cell shape variation. Other disadvantages of the micropattern technique are the restriction to one cell type per optimized pattern requiring cells to be able to grow in isolation, and that only adherent cell types can be used (Fink et al., 2007; van Dongen et al., 2013). Based on preliminary tests that I carried out as part of my project, the standard human cell line used in Mitocheck, HeLa Kyoto, was not very suitable for single cell micropatterns due to the tendency to form colonies with tightly adhering neighboring cells (Figure 1.3).

In conclusion, while micropattern technology is in principle a useful approach to help standardization of the cellular morphology, its resolution and applicability to standard mitotic cell lines were thought to be not yet sufficient for the systematic study of the mitotic proteome.

Figure 1.3: HeLa Kyoto cells on micropatterns.



A significant fraction of fibronectin coated 2D micropatterns (provided by the Thery lab) could not be used for cell division assays due to the occupancy by multiple cells (upper box) instead of by a single cell (lower box). Scale bar: 10 µm.

1.3.2 Cell synchronization with chemicals

Temporal standardization of mammalian cells relies on reversible mechanical or chemical perturbations. The most commonly used agent is nocodazole, a chemical inhibiting microtubule polymerization and thus the assembly of the mitotic spindle

(Deysson, 1968). Cells cultured in a nocodazole solution are consequently arrested in prometaphase. Inhibitors of DNA synthesis such as aphidicolin, thymidine or hydroxyurea arrest cells at the G1/S transition (Bootsma et al., 1964; Ikegami et al., 1978)), and are therefore often used to synchronize the cell cycle. After several hours of inhibition, a release from these arrests results in a cell population enriched M-phase cells. However, all of these methods harm the cell, and none preserve the natural state of cellular functions (Samuels et al., 1964; Zieve, 1984). Moreover, the achieved efficiency of the synchronization is insufficient to obtain a highly synchronized population (Dulla and Santamaria, 2011; Ma and Poon, 2011).

Mitotic shake-off is an often used mechanical method for obtaining a mitotic cell population (Elvin and Evans, 1984). But if the method is not used in combination with chemical synchronization, then shake-off cells can be in any mitotic stage between late prometaphase and telophase.

The major issue with current cell synchronization methods is the fact that the treatment can only arrest cells at one particular and possibly unnatural stage. While they are able to aid in the investigation of the cell cycle, their drawbacks outweigh their usefulness for the aim of this project.

1.4 Computer vision methods in cell biology

1.4.1 Bioimage informatics helps us to see

Computational analysis has become essential in image based biological research due to both the increase in the amount of imaging data as well as the demand for quantitative measurements of biological processes, (Danuser, 2011; Peng, 2008). Information technology not only helps biologists to handle and process large amounts of data in an unbiased, reproducible and quantitative way, but also in extracting useful information and thereby gaining knowledge from the data (Myers, 2012; Johnston et al., 2006; Sommer et al., 2013). Moreover, model-based algorithms are not only able to detect significant changes in experimental data, but can also often identify the mechanistic reasons behind them (Sprague et al., 2003; Ji et al., 2008). Besides the classic tasks such as segmentation (Nunez-Iglesias et al., 2014), registration (Kukulski et al., 2012), tracking (Parthasarathy, 2012), computer vision has also been used for visualization (Mickoleit et al., 2014), data integration (Ronneberger et al., 2012)

or automatic classification (Carpenter et al., 2006; Harder et al., 2009; Loo et al., 2009), where the later one has been used particularly often for analyzing data from large scale image based screens (Wawer et al., 2014; Laufer et al., 2013)Many tests reported a higher consistency and even higher accuracy of computational pipelines compared to human analysis (Danuser, 2011; Sommer and Gerlich, 2013; Zhong et al., 2012). Two prominent applications of computer vision relevant to my research will be discussed in the next two subsections.

1.4.2 Automatic assignment of mitotic stages

Temporal alignment has been a challenge for the analysis of time-resolved imaging data in research on mitosis. Traditionally, either videos of mitotic events are aligned to one of the two sharp M-phase transitions, i.e. the breakdown of the nuclear envelope (NEBD) or the onset of anaphase (Dultz et al., 2008; Dumont et al., 2010), or frames of cells in the predefined mitotic stages are manually picked and then aligned (Zhu et al., 2005; Uzunova et al., 2012). The precision of the alignment is strongly influenced by the temporal resolution of imaging, and often many valuable data points go unused. When a large number of mitotic events need to be manually annotated into mitotic stages, then the annotation does not only limit the throughput of the approach but is also subject to each expert's individual bias (Zhong et al., 2012).

A commendable case for the automatic annotation of mitotic stages using supervised computer vision methods was the MitoCheck project (Neumann et al., 2010). Billions of imaged nuclei were segmented individually and tracked over the course of up to two cell cycles. Experts annotated a subset of these cells into pre-defined mitotic stages. Hundreds of numerical features were extracted from each of the segmented nuclei. Using the support vector machine algorithm (Chang and Lin, 2011), boundaries separating differently staged cells were determined in the higher dimensional feature space based on the human annotation. The large majority of non-annotated nuclei were then annotated automatically using the learned model (Walter et al., 2010). This approach was key to the success of the MitoCheck project as it allowed tens of millions of mitotic events to be analyzed, something that would have been near impossible using a purely manual approach. The derived algorithms were however not only used for the analysis of the siRNA screening data to provide important

insights into whether and how a gene is relevant to cell division (Horton et al., 2015; Yamaguchi et al., 2015). They were also integrated into open source software such as CellCognition that allows for a more general analysis of mitotic screens (Held et al., 2010; Schmitz et al., 2010), and Micropilot that is employing machine learning based image analysis for automating microscopy experiments (Conrad et al., 2011; Wachsmuth et al., 2015).

The drawback of the approach used in MitoCheck is its reliance on human supervision. Since then, researchers have thought about fully automating the annotation of mitotic cell stages without any previous knowledge. Zhong et al. (2012) reduced the dimensionality of the feature vector describing segmented nuclei using principle component analysis and, taking the temporal evolution of the feature vector along the nucleus track into account, clustered all data points into a pre-defined number of mitotic stages without supervision. The algorithm relied on temporally constrained combinatorial clustering followed by Gaussian mixture modeling and hidden Markov modeling. In a test using the five traditional mitotic stages, its performance was similar to that of the supervised approach (Zhong et al., 2012).

El-Labban et al. (2011) took an entirely different approach to annotate the mitotic time of dividing cells. Only a few numerical features were used to describe the nucleus of the cell. However, the entire mitotic progression was represented as a trace in this low dimensional feature space and analyzed as a whole. A subset of the tracks were annotated into seven mitotic stages by locally squeezing or stretching the feature trace. The averaged feature trace based on the training set was then used as template to which unknown tracks were aligned using dynamic time warping (Müller, 2007). The achieved accuracy of the mitotic assignment was comparable to that of the approach used in MitoCheck. Unlike the latter, it also has the potential to be extended to a continuously rather than discretely defined mitotic staging.

1.4.3 Automatic annotation of subcellular protein localization

For the past 15 years, machine learning approaches have been widely used for the annotation of subcellular protein localization (Murphy, 2010). In pioneering work, differently targeted two-dimensional immunostained images of proteins purely localized in one subcellular compartment were automatically annotated using a classification approach (Boland and Murphy, 1999, 2001), which over the following

years has been improved through a variety of feature extraction, selection and classification methods (Chebira et al., 2007; Huang et al., 2003; Nanni et al., 2010; Hamilton et al., 2007; Tahir et al., 2012; Conrad et al., 2004). Similar pipelines can also be used for classifying images with multiple cells, and a robust version was reported in Coelho et al. (2013). More recently, three-dimensional cell images or time-resolved videos of interphase cells were successfully annotated using the classification approach with high precision (Chen and Murphy, 2004; Hu et al., 2010). The only assumption in the time-resolved case was a static protein localization over time, which can obviously not be used for dividing cells.

At the single cell level, one could also visually model each protein localization first and then classify unknown images by finding the most similar model. Zhao and Murphy (2007) built models in 2D for nine vesicle based subcellular compartments by fitting the images with multi Gaussian distribution and parameterizing the size, position and intensity of the signal. Using the parameters of the model as features, real images can be classified with high accuracy (Zhao and Murphy, 2007), and the method could also be extended to 3D (Peng and Murphy, 2011). The weakness of this approach is its limitation to vesicular localization due to the use of the Gaussian fit. For subcellular localization of other morphologies, such as microtubules, completely different models need to be developed (Shariff et al., 2010).

One challenge in the automatic annotation of subcellular protein localization has not yet been extensively studied, although it is very relevant for most cases: one protein can localize to multiple subcellular structures and dynamically exchange between them. The first work addressing this challenge was reported in Peng et al. (2010), where a linear regression model was trained based on a data set with labeled lysosomes and mitochondria in known signal proportion. Subsequent work then used latent Dirichlet allocation to identify the underlying basic patterns in the data in an unsupervised manner (Coelho et al., 2010). However, both methods were only tested on two vesicle-like protein localizations, and their general applicability remains unclear.

Automated protein distribution annotation has already shown its power in many projects, such as the correction and annotation of the Human Protein Atlas Project (Li et al., 2012). In yeast, where the small cell size causes difficulties in resolving subcellular structures, classification was used successfully for determining protein

localization at the system level (Handfield et al., 2013; Chong et al., 2015). Nevertheless, proteins with multiple subcellular localizations, or proteins which change their localization over time, have never been automatically annotated and certainly not in the dynamic cellular context of the dividing cell.

Chapter 2

Aim and approach of the thesis

The aim of my thesis is to develop generic computational methods for the systematic study of protein networks in dynamic cellular processes using live cell imaging technologies. As an example for dynamic cellular processes, I chose cell division due to the availability of an already identified reference proteome in MitoCheck, and because cell division exhibits changes both in protein distribution as well as in cellular context while the function is being carried out. This should ensure that the solution I developed is generically applicable to many biological functions that may present only one or the other challenge.

To achieve this, I have to establish an integrated experimental and computational pipeline that standardizes the acquisition and analysis of dynamic mitotic protein imaging data. This pipeline aims to integrate the data on different proteins observed in distinct cell lines, and to allow comparisons in a single digital model of the human dividing cell. As output, subcellular localizations and interaction clusters of these proteins have to be objectively identified, and kinetics of these clusters' formation or disassembly will be quantified. The pipeline should be robust and scalable up to thousands of protein components and provide tools for achieving a systematic understanding of the properties of the dynamic mitotic protein network. And it should be applicable to other cellular protein networks in the future.

The approach I chose is FCS calibrated automatic 3D live cell confocal microscopy in order to acquire absolute quantitative data on mitotic protein distribution in HeLa cell lines that express fluorescent knock-in versions of these proteins. These proteins are imaged relative to cellular landmarks of the genome and cell surface, which should allow me to create a canonical model of the human dividing cell, to which the absolute protein distribution data can be registered in both space and time. To mine the integrated data I chose machine learning approaches that aim to

automatically identify localization clusters to which these proteins can be assigned. As proof-of-concept, a small data set with well-known mitotic proteins is generated in order to test and validate the integrated experimental and computational pipeline.

Chapter 3

Result

3.1 References of mitotic cellular morphology

3.1.1 Chromosomal volume as spatial-temporal landmark

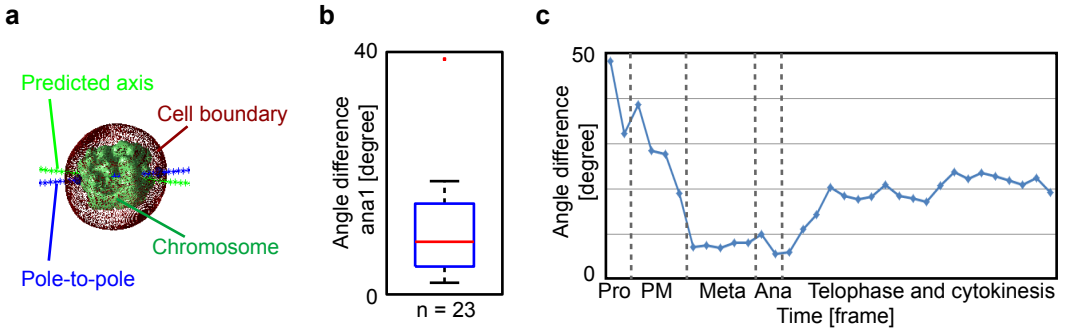
Before generating the spatial-temporal model for the standardization of dynamic protein distributions through mitosis, landmarks defining the mitotic cellular context need to be selected. The selection criteria are as follows: it should enclose a functional volume that is connected to various subcellular structures. It should have a distinguishable morphology between different mitotic stages. And finally, it could be fluorescently labeled without changing the cell fitness or the dynamic of division.

As a logical consequence, I selected the chromosomal volume as the first landmark. As one major goal of cell division is to separate the duplicated chromosome ensemble into two equal partitions, functional processes and morphological changes of many subcellular structures are centralized to the chromosomal volume, such as the breakdown of the nuclear envelope, Golgi and ER network (Shima et al., 1998; Zaal et al., 1999), the formation and deformation of the spindle (Kitajima et al., 2011), and the remodeling of the centromere/kinetochores (Cleveland et al., 2003). Moreover, the morphology and the texture of the chromosomal volume has been used to stage mitosis temporally for more than a hundred years (Paweletz, 2001), and recently, also as marker for the automatic annotation of mitotic progression (Neumann et al., 2010).

I have used different HeLa Kyoto cell lines with fluorescently (mCherry, EGFP or mCerulean3) tagged histon H2B generated by the cDNA transfection technology (see 5.2.3). The overexpressed labeled H2B has fully decorated all chromosomes within the cell (data not shown). The volume encloses all chromosomes or, where the duplicated genome has already been separated into two partitions, the volumes

in each of the daughter cells can be segmented using the computational pipeline developed by Dr. Julius Hossain (5.2.15).

Figure 3.1: Comparison between the real and predicted division axis.



(a) Schematic illustration of the real division axis (blue) and the predicted axis (green) in the first anaphase frame of a cell. The real division axis was determined by the position of centrosomes based on segmented Nedd1 signal. The predicted axis was calculated based on the geometry of the chromosomal volume segmented on H2B signal. (b) Boxplot of the difference between the predicted division axis and the pole-to-pole axis in the first anaphase frame. (c) Averaged differences between pole-to-pole and predicted axes throughout mitosis. Cells were annotated into one of the five mitotic stages and aligned manually. The analysis was performed by J. Hossain. The illustration was done by J. Hossain and myself.

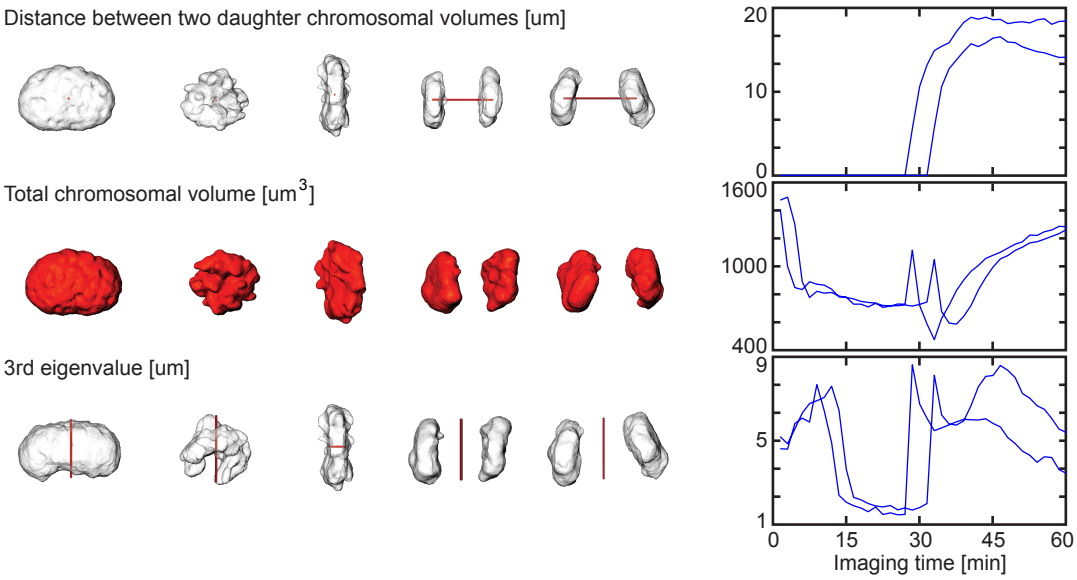
In order to validate whether the chromosomal volume can be used to characterize the mitotic morphology, Julius Hossain and I have examined the difference between the actual division axis, i.e. the pole-to-pole axis of the mitotic spindle at the moment of chromosome segregation, and the symmetrical axis of the chromosomal volume. Birgit Koch and I acquired an imaging data set from the HeLa Kyoto BAC cell line H2B-mCherry/NEDD1-LAP provided by Ina Poser (see 5.1.2). Using confocal live cell imaging each cell was imaged every five minutes through mitosis with a spatial resolution of 150 nm in x-y and 500 nm in z. Julius Hossain segmented the chromosomal volume based on the H2B and the centrosome volume based on the Nedd1 signal in 3D. We compared the predicted division axis based on the binary chromosomal volume geometry (see 5.2.16) with the vector connecting both centrosomes in the first frame of anaphase (Figure 3.1 a). The difference between the direction of the two vectors is negligible (Figure 3.1 b).

Furthermore, we looked at the correlation between the geometry of the chromosomal volume and the spindle position. The third chromosomal eigenvector was identified as a good approximation for the symmetrical axis towards the division direction for late metaphase until the beginning of chromosome segregation. For the remaining frames,

one of the eigenvectors was selected based on the rules described in 5.2.16. Figure 3.1 c shows that, initially, the difference between the chromosomal symmetrical axis and the spindle pole-to-pole direction is large when the mitotic spindle forms in early mitosis. But both vectors converge in late prometaphase and stay similar, until the disassembly of the mitotic spindle starts in late telophase. Thus, chromosomal volume could well be used for the estimation of the mitotic spindle orientation, as long as the spindle is completely formed.

I then examined whether simple geometrical features of the chromosomal volume can be used for characterization of the mitotic temporal progression. I extracted three features from the chromosomal volume: the total volume, the distance between the two daughter genome partitions, and the third eigenvalue of the binary chromosomal volume. All three features gave very characteristic and reproducible behavior throughout cell division (Figure 3.2).

Figure 3.2: Features of the chromosomal volume.



Three features were extracted based on the chromatin signal. Visual illustrations of the features are shown on the left in red (illustrated by J. Hossain), while on the right the feature values of two mitotic cells are plotted over time.

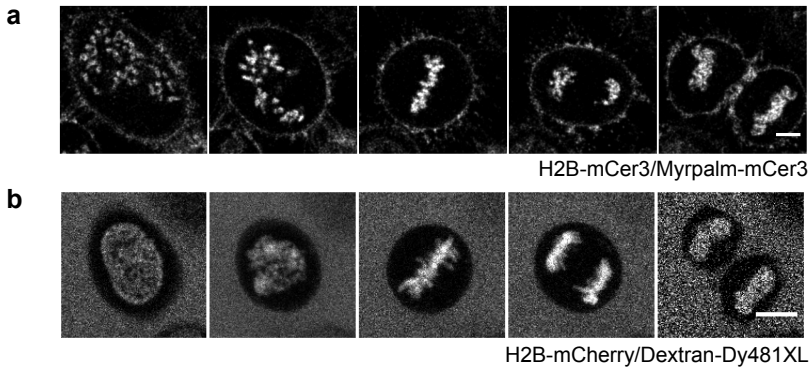
All of these results show that chromosomal volume is a good landmark for both the temporal and morphological progression of the mitosis. It is highly predictive of the division axis and has a tight correlation to the spindle orientation, another major mitotic subcellular structure. Finally, geometrical components of the chromosomal

volume are characteristic for different mitotic stages.

3.1.2 Cell volume defines the border of reactions

The cellular boundary defines the border of the cell. Moreover, it actively contributes to the mitotic process by connecting the extracellular environment with cytoskeleton structures such as the spindle and actin filaments (Xu and Saunders, 2008). The shape of the cellular volume is very dynamic throughout mitosis as well: at the beginning of division, it rounds up from its adherent flat shape. In telophase, the so-called cleavage furrow, a neck in the middle of the cell, emerges and defines where the mother cell should separate. Thus, I took the cell boundary as an additional landmark in addition to the chromosomal volume.

Figure 3.3: Labeling strategies of cellular landmarks.

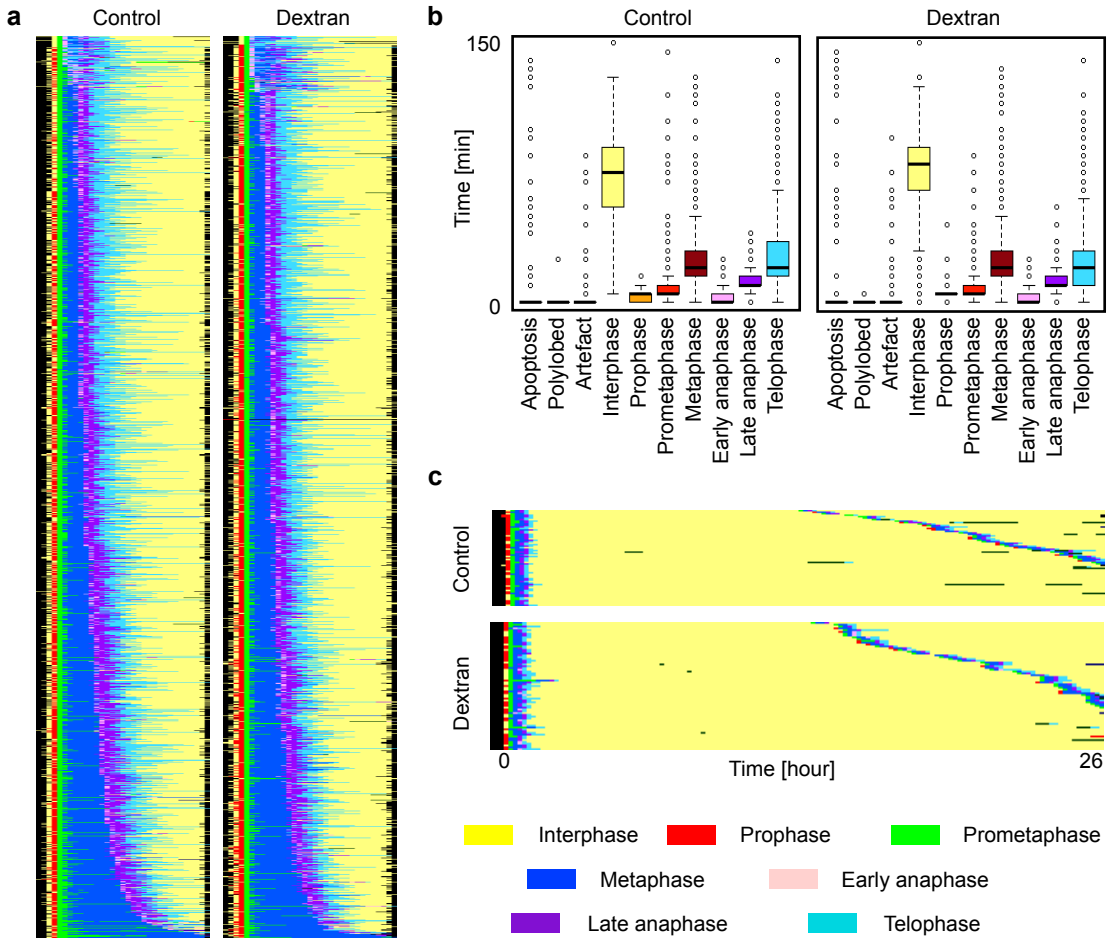


Confocal 3D live cell imaging was performed on HeLa Kyoto H2B-mCer3/Myrpalm-mCer3 stable cell (a) or H2B-mCherry stable cell cultured in Dextran-Dy481XL solution (b). Single imaging z-plane was taken for the cell in (a) and an averaged projection through 6 z-planes over 3.75 µm was used for (b). Scale bar: 10 µm.

There are different ways to visualize the cellular boundary under a fluorescence confocal microscope. I first tried to use HeLa Kyoto cells with fluorophore tagged myrPlam, a cell membrane marker (Figure 3.3 a, cell line generated by Bianca Nijmeijer). The cells behaved well, but the segmentation results for the cell membrane were not precise enough due to the irregularity of the membrane signal.

I then used 500 KD dextran labeled with the fluorescent dye Dy481XL (chemical provided by M. Julia Roberti) to stain the culture medium in such a way that the cell boundary can be recognized by negative contrast (Figure 3.3 b). I tested the staining protocol with different concentrations of dextran, and confirmed that a

Figure 3.4: Toxicity of dextran to cell mitosis and cell cycle.

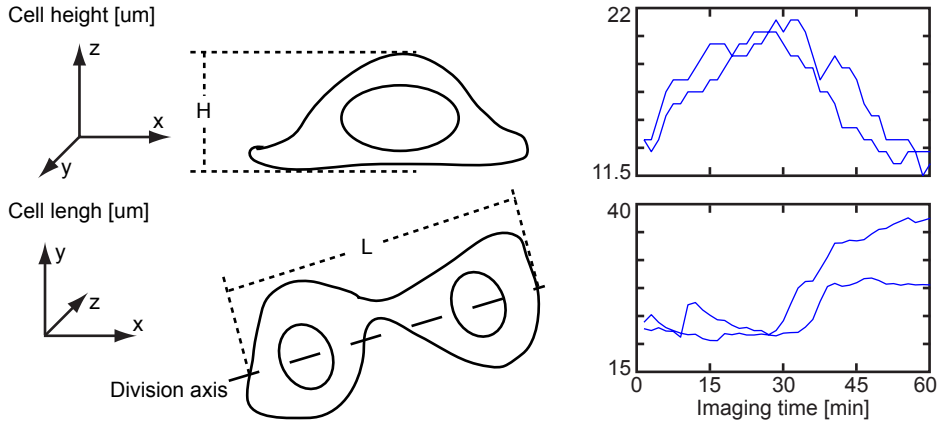


HeLa Kyoto cells were immersed in imaging medium containing 3.29 µM 500 KD dextran-Dy481XL. The mitotic kinetics and the cell cycle time were measured as described in session 5.2.11. (a) For each condition about 1500 mitotic events were automatically annotated and ranked by the total duration of early mitotic stages (prometaphase to metaphase). The statistics for the duration of all stages are shown in (b). (c) HeLa Kyoto cells were imaged with a confocal microscope for 40 hours, and about 70 cells were tracked for 26.6 hours after their first division. About 60% of the cells divided a second time under both conditions.

dextran concentration of 3.29 µM as I later used for imaging did influence neither the cell division (Figure 3.4 a, b) nor the cell cycle kinetics (Figure 3.4 c, see material and methods). The advantage of using labeled dextran instead of producing a cell line with a labeled cell boundary or volume was the high flexibility in experimental design. The color of the label can be chosen freely according to the fluorescent tags on the chromosome and the proteins of interest.

Some simple features of the cellular boundary had characteristic behaviors through-

Figure 3.5: Features of cellular volume.



Cell height and length, illustrated on the left, of two cells over the course of division are plotted on the right.

out cell division. An example is shown in Figure 3.5, where both the height and the length of the cell at the symmetrical axis of the chromosomal volume are changing with a reproducible course. It turns out that taking both the chromosomal and cellular volume as landmarks produces good coverage of the dynamic morphology of a dividing cell without overcomplicating the experimental design.

3.1.3 Acquisition of landmark data set for modeling of the Mitotic Standard Cell

HeLa cells stably expressing H2B-mCherry were cultured in an imaging medium stained with 0.35 µM 500 KD dextran-DY481XL and imaged live overnight at lower resolution on a ZEISS 780 fluorescence confocal microscope. Cells in prophase were automatically detected using classification and imaged tomographically in a z-stack every 90 seconds for an hour at a higher x-y resolution (details see material and methods). Both cellular and chromosomal volumes were segmented using the computational pipeline developed by Julius Hossain (see 5.2.15). For modeling of the Mitotic Standard Cell, 132 successful divisions were acquired.

3.2 Modeling of the Mitotic Standard Time

3.2.1 Alignment of 2 feature sequences using modified multidimensional discrete dynamic time warping

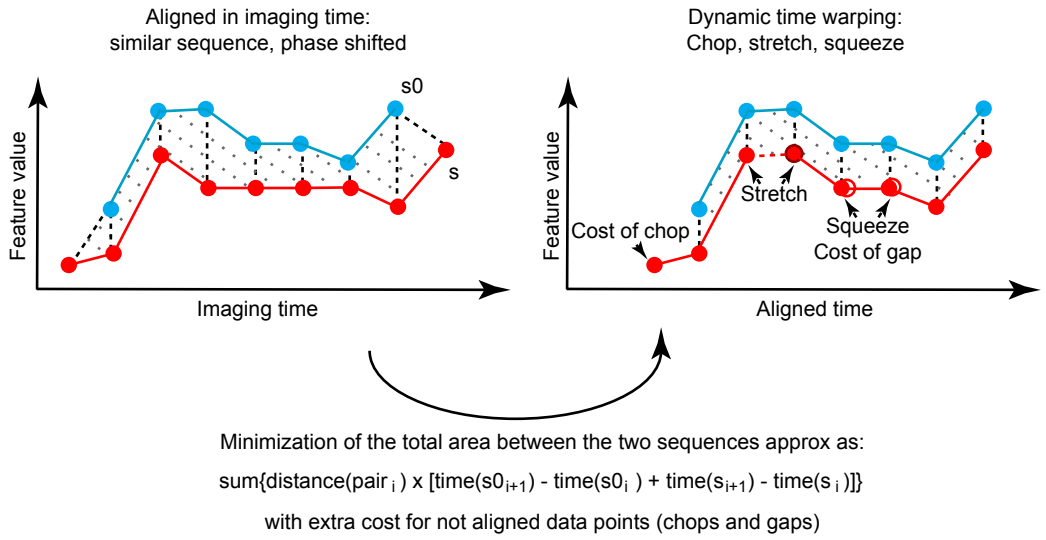
Successful divisions of HeLa cells usually take less than an hour, but the duration of the division can differ strongly between cells. Moreover, the progression varies massively between cells, i.e. cells with identical mitotic duration can have different lengths for prometaphase, metaphase, etc. (Figure 1.2). In order to model the averaged temporal progression of mitosis, I had to find a way to align all of the cells in time first.

Traditionally, biologists annotated the mitotic stages manually, mostly based on the morphology and texture of the chromosomal ensemble. More recently, supervised computer vision methods were increasingly being used where the mitotic stages were assigned by a classifier trained by experts. However, the annotation was not consistent between different persons (Zhong et al., 2012), and thus not always reproducible. The annotation was also limited to a very few characteristic stages.

I decided to use discrete dynamic time warping, an unsupervised approach widely used in speech recognition (Myers and Rabiner, 1981). It optimizes the temporal mapping between two digital sequences by locally stretching (duplicate signal points) or squeezing (deleting signal points) the sequence such that the total distance between the two signals is minimized over the entire time (Figure 3.6). The method is well suited for the alignment of mitotic image sequences for the following reasons: the imaging sequence is discrete in time; the mitotic progress can be described numerically as feature value over time; and the optimization runs very fast.

I modified the standard implementation of the multidimensional discrete dynamic time warping reported in Müller (2007) and Ten Holt et al. (2007) based on the specific features of mitotic imaging sequences. I changed the calculation of the distance between the two sequences which is used as the cost function for optimization. In the standard implementation, it was implemented as the sum of the Euclidean distances between all aligned pairs of data points. Thus, sampling the signal more frequently will cause the cost function to increase, which would be a problem for later steps (see next subsection). Thus, I weighted the distance for each aligned pair for the time this aligned pair lasted (Figure 3.6 and appendix). Moreover, I introduced a penalty

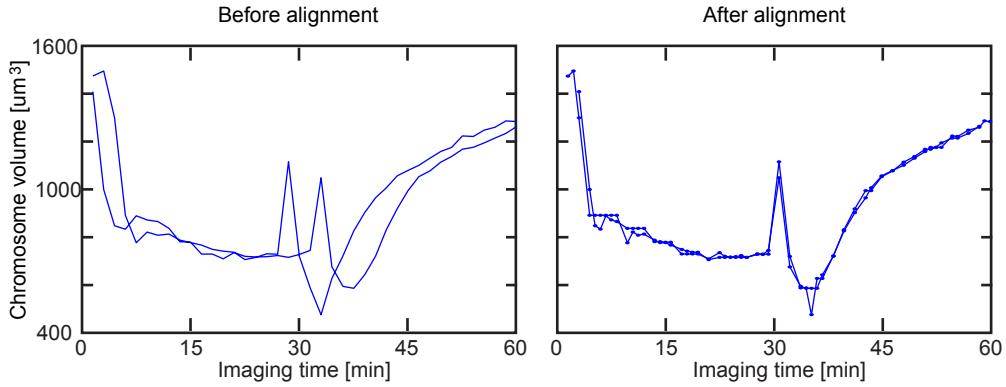
Figure 3.6: Illustration of discrete dynamic time warping.



for fusion alignment, i.e. one data point being aligned to multiple points. The reason for this was that without it, in order to reach the minimum total distance, parts of one sequence would sometime be extremely stretched or squeezed. However, mitosis is always progressing, where each stage's duration has variation but also constraints, and the penalty pushes the alignment towards similar progression. Furthermore, alignment gaps at the beginning and end of the sequences and as well as a gap ofgaps of one data point in the middle of the sequence were allowed but with a penalty. Since data acquisition has a set duration, image sequences do not start and end at the same stage. Some stages, such as anaphase, may also only last a very short time, and therefore the temporal resolution of the imaging protocol might miss certain stages of cell division. A detailed implementation is attached in 6.3.5.

Next, I had to find a good numerical description for the temporal progression of mitosis. Since dynamic time warping works best if the time course of the feature has characteristic and reproducible shapes with multiple ups and downs, I used the combination of the three morphological features of the chromosomal volume described in Figure 3.3: the chromosomal volume, the distance between two daughter nuclear partitions, and the third eigenvalue of the chromosomal volume. All features were normalized to zero-mean-unit-deviation. In addition to this, as the shape of

Figure 3.7: Discrete dynamic time warping of two sequences.



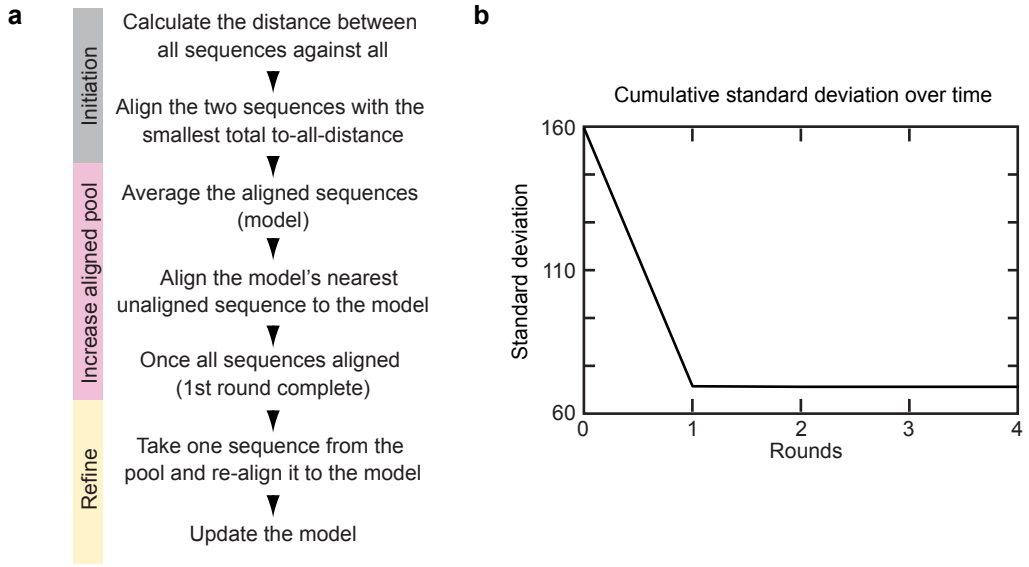
The time-resolved chromosomal volume feature sequences of two cells were aligned using the algorithm described in Figure 3.6. The timing of the aligned sequences was determined by the averaged progression of the original sequences.

the feature curve was as important as the absolute value, first derivatives of each of the three features were also included. To do this, the feature curves were first smoothed using non-parametric fitting by a sliding window before the derivatives were calculated (‘‘smoothingspline’’ and ‘‘differentiate’’ function in the MATLAB curve fitting toolbox). Also the derivative features were normalized to zero-mean-unit-deviation. Figure 3.7 shows the corresponding feature sequences of Figure 3.3 after the customized discrete dynamic time warping was performed. The alignment was very effective.

3.2.2 Generation of the model by multi-sequence alignment

In order to generate a model representing the standard mitotic temporal progression, the average of 132 mitotic cells aligned in the feature space would need to be taken. However, discrete dynamic time warping is not suitable for aligning more than two sequences at once. A framework for multi-sequence alignment had to be developed. I implemented a modified Barton-Sternberg algorithm (Barton and Sternberg, 1987) as follows: To initiate the alignment, the sequence with the smallest total Euclidean distance to all remaining sequences in the pool was selected and aligned pair-wise first to the next closest sequence using dynamic time warping as described in the last subsection. The average of the aligned pool was updated and the next unaligned

Figure 3.8: Algorithm of multi-sequence alignment



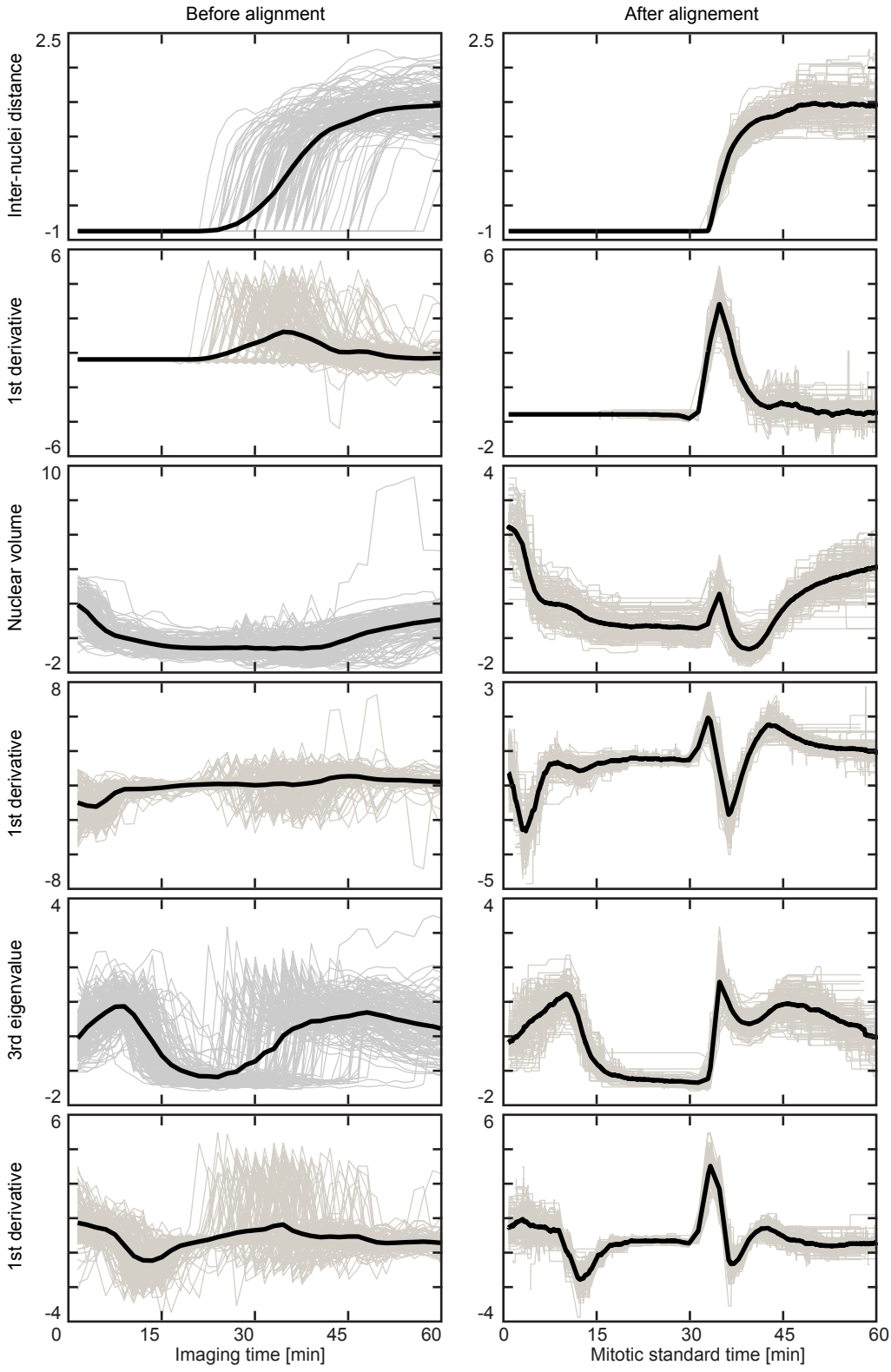
(a) Schematic pipeline of modified Barton-Sternberg algorithm. (b) The convergence of the alignement of 132 landmark feature sequences measured by the total standard deviation.

sequence was selected and aligned against the aligned average. Here, an asymmetric penalization was used where any gaps in the alignment of the selected sequence had a higher cost compared to those of the averaged sequence. Since the average presented many mitotic sequences, it was unlikely that any frames in the selected sequence could not find the corresponding aligning partner in the averaged sequence. This step was repeated until all sequences were aligned. To refine the alignment, sequences were taken from the pool one-by-one and re-aligned to the current average sequence over several rounds (Figure 3.8 a). The convergence of the alignment evaluated by the total standard deviation was reached after 2 rounds (Figure 3.8 b).

Among all 132 mitoses, sequences with significantly larger (mean + standard deviation) distance to the averaged sequence were sorted out and the average of the remaining sequences was used as the model of mitotic temporal progression in the feature space (Figure 3.9). The averaged progression time between two sequential frames of the averaged sequence was calculated as the total increase in the imaging time of all data points aligned to divided by the number of data points. The mitotic

Figure 3.9: 132 cells (grey lines) were aligned in the 6-dimensional feature space, and the average (black line) of the aligned sequences was used as the mitotic standard time model in the feature space.

Figure 3.9: Generation of the mitotic standard time model.



time line was derived from the progression time. The model had a very high sampling rate that was hard to interpret biologically as it would introduce additional noise. Thus, I performed sub-sampling with a constant time lapse of 15 seconds. The feature vector between two sampling points was linearly interpolated. In the end, the temporal progression of mitosis was modeled as a process that lasts 60 minutes with 237 sampled frames.

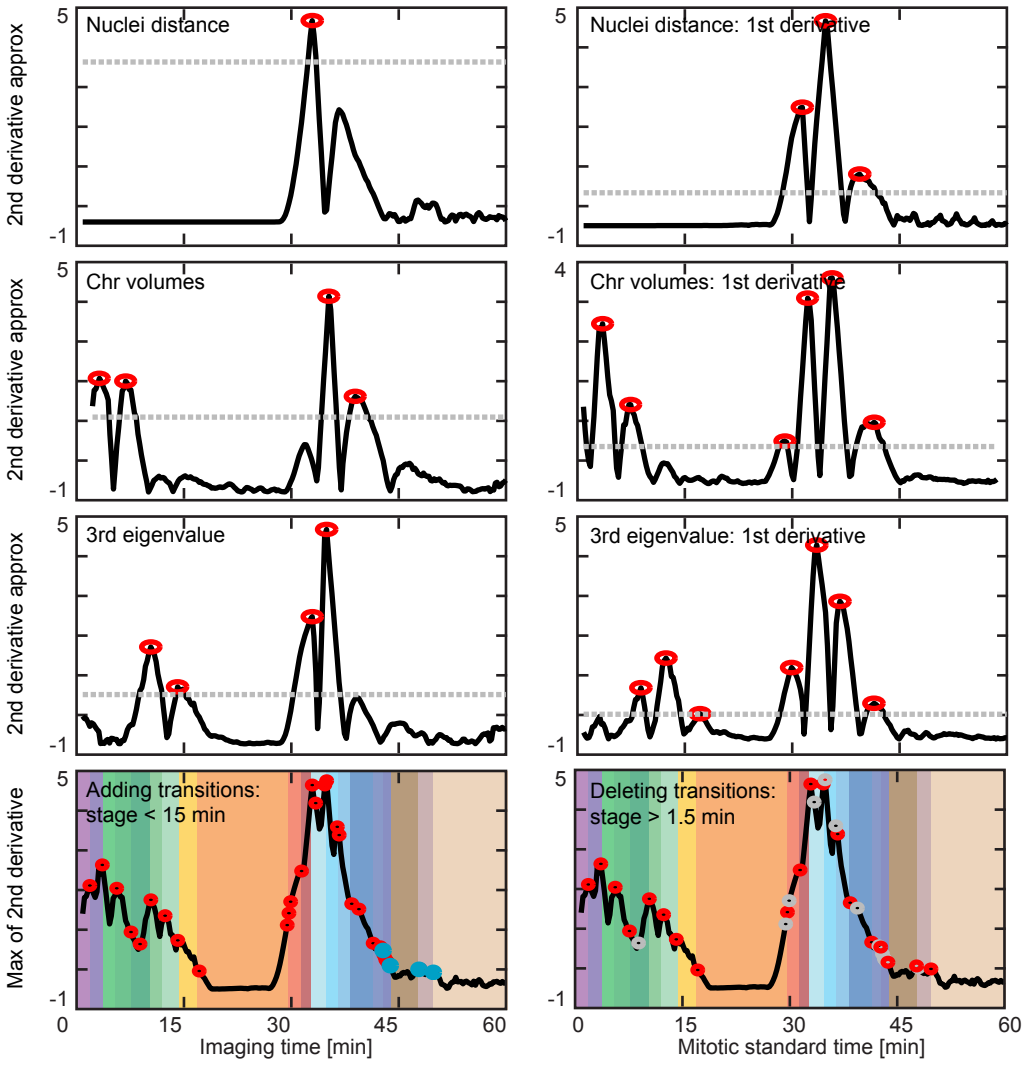
3.2.3 Objective finding of mitotic transitions

Next, I wanted to objectively stage mitosis based on the numerical transitions of the model in the feature space. A transition was defined as a point in time, before and after which the slope of the feature sequence changed strongly. Since the averaged feature sequence was noisy, the change of slope was calculated based on the mean of a window of 4.5 minutes and normalized to zero-mean-unit-deviation in each of the feature dimensions. Points which had a change of slope value (approximated second derivative) larger than a pre-defined threshold (see 6.3.6) in all feature dimensions were identified as transitions. Since the slopes were calculated through a window of 4.5 minutes, only the peaks in the second derivative were used as final transitions. When selected transitions in all feature dimensions were merged, sections longer than 12 minutes without any transitions were detected, and additional transition points were set to the position with the maximum change of slope (Figure 3.10). Finally, if multiple transitions were selected within a mitotic duration of 1.5 minutes, only the transition with the highest total change of slope among all feature dimensions was kept (Figure 3.10). In the end, a total of 19 clearly distinguishable transitions were identified this way.

3.2.4 Generation of the virtual mitotic cell

To validate that the mitotic standard time adequately represents real biological stages, we reconstructed for each stage between two standard mitotic transitions a virtual mitosis by choosing the 3D stack with the shortest distance to the average feature values among all frames in the mitotic standard time model of that stage (Figure 3.11). Although the images picked for each stage are from different cells, their computationally assigned sequential order reconstitutes a perfect mitosis (Figure 3.12) in which we see known transitions such as nuclear envelope breakdown (between

Figure 3.10: Objective definition of mitotic standard transitions.



Transitions (red circles in the upper six panels) were identified as peaks above an automatically determined threshold (grey line) on the approximated second derivative (black line) in each feature dimension. The transitions were merged (lower two panels) and transitions were added (blue dots) or deleted (grey circles) such that the duration between two transitions was kept between 1.5 and 15 minutes. The final stages defined as the time between two transitions are color-coded in the lower panels.

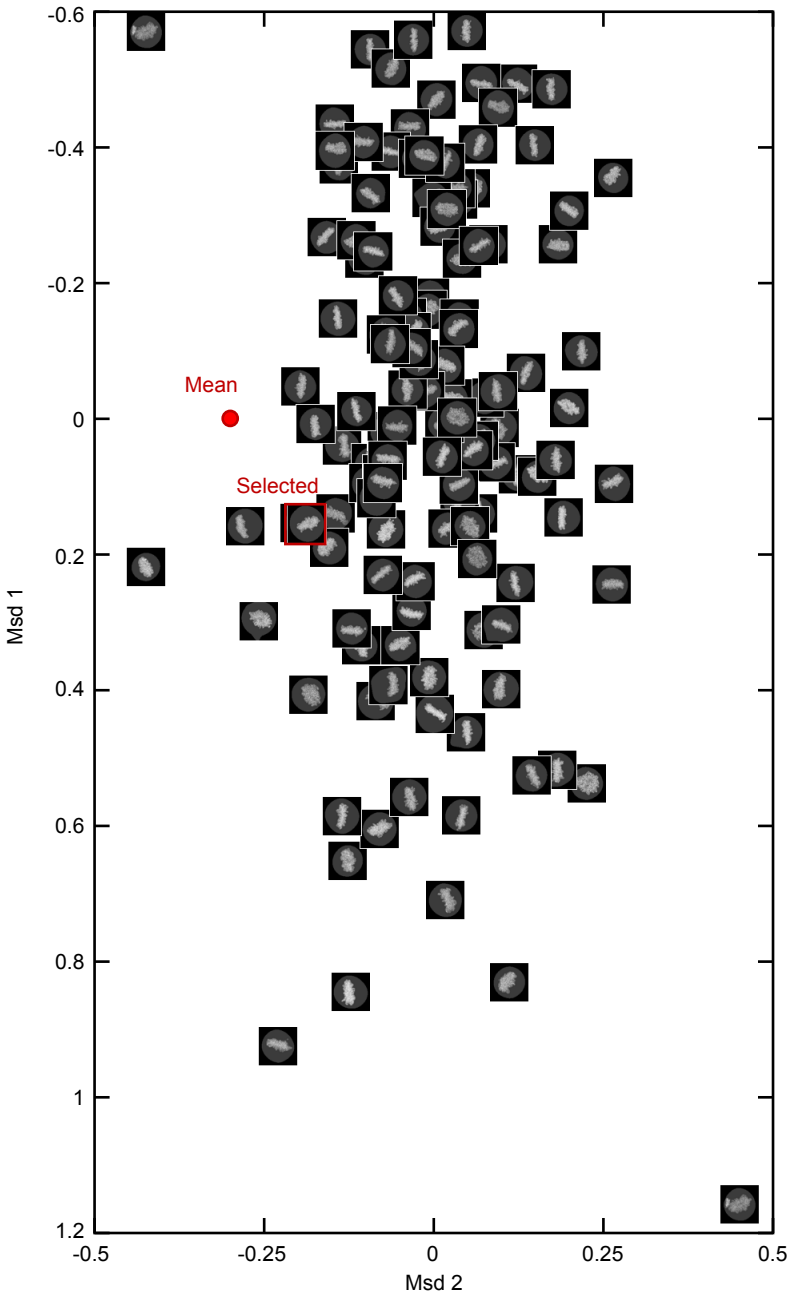
frame 2 and 3), anaphase onset (between frame 11 and 12), and the start of telophase (between frame 15 and 16). But we also recognize previously hard-to-define stages such as the formation of the metaphase plate in late prometaphase (between frame 6 and 7), and are able to differentiate between anaphase and telophase stages (frames 12 to 18). Thus the mitotic standard time provides a quantitative and objective way to define mitotic stages. The results show that the mitotic standard time provides a quantitative and objective way to define mitotic transitions.

3.2.5 Assign mitotic stages using the Mitotic Standard Time model

Any mitosis imaged in HeLa Kyoto cells with the chromosomal volume as landmark could now be temporally registered to the Mitotic Standard Time. After the segmentation, corresponding parameters were extracted and aligned to the model resolved every 15 seconds using our discrete dynamic time warping implementation. In order to avoid having frames being unaligned, or multiple frames being aligned to the same point in time, the penalization parameters were adjusted (see 6.3.7). Very often, a single frame in the sequence to be analyzed was aligned to multiple frames in the model. The final annotation took from these aligned model frames the one closest to the frame being analyzed. Based on the mitotic time assigned to the sequence being analyzed, all frames could be automatically annotated with the objective mitotic stages.

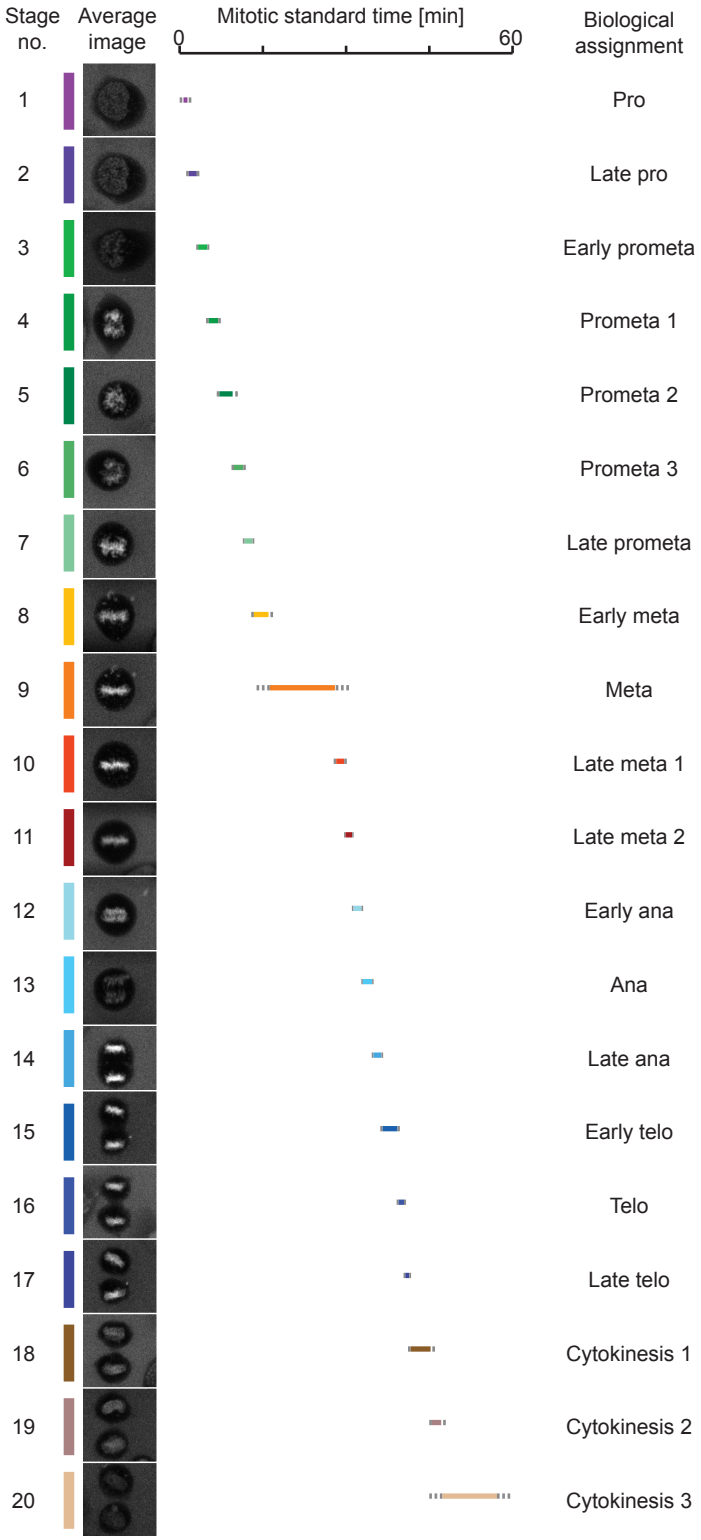
In order to validate the temporal registration accuracy, I used the pipeline to annotate 96 mitotic HeLa sequences acquired in the same way and with the same landmarks and compared the annotation results with my manual annotation blindly done before. It was very difficult for me to annotate a mitotic sequence into 20 mitotic stages, and thus I first compared whether three major transitions, i.e. nuclear envelop breakdown, anaphase onset, and post-mitotic de-condensation of the chromosomal volumes, were identical in the manual and automatic annotation. Both annotation methods agreed on 67% of the sequences. I then tried my best to assign each frame into one of the 20 mitotic stages and the annotation would be identical to the automatic annotation in 96.4% of all z-stacks. This result shows that even using a completely unsupervised approach, our Mitotic Standard Time model provides a very good template for the temporal registration of new mitotic data.

Figure 3.11: Selection of the representative cell.



The six-dimensional feature space was projected into two dimensions using the standard implementation of multi-dimensional scaling in MATLAB. The averaged feature vector of the mitotic stage no. 10 is marked with a red dot, and the cell with the shortest distance marked in red was selected as the representative cell for this stage.

Figure 3.12: Mitotic Standard Stages as a virtual mitotic cell



3.3 Modeling of the Mitotic Standard Space

The standard mitotic time model could register image sequences of any mitotic cell into the standard time and assign each frame into one of the mitotic standard stages. Using the images assigned to, Julius Hossain was able to generate a model of the averaged cell geometry for all symmetrical stages, i.e. from late prometaphase to cytokinesis (stage 8 to 20). The following text in this section was modified based on the text formulated by Julius Hossain.

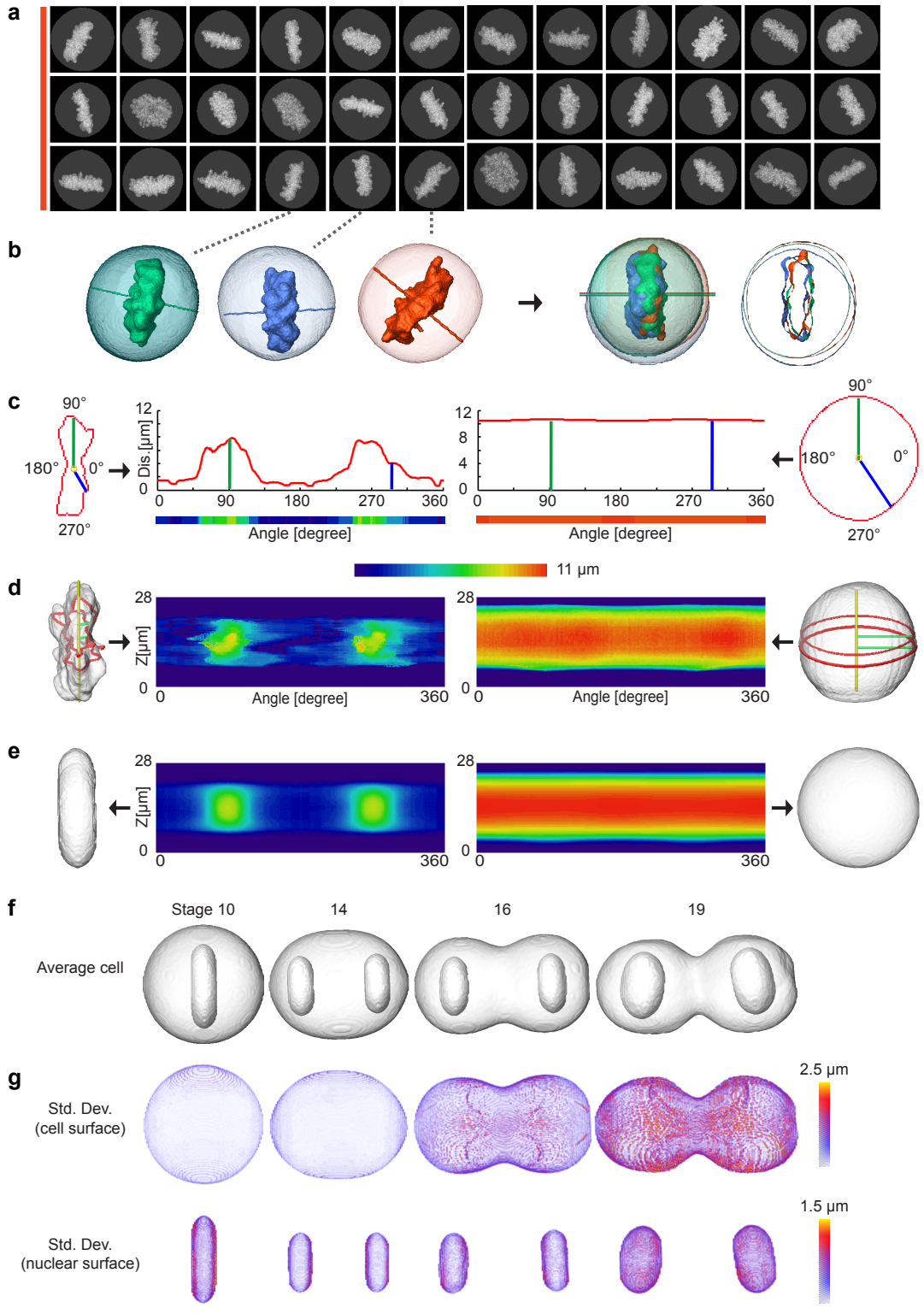
3.3.1 Cylindrical representation of the cellular and chromosomal volumes

All cells in the same mitotic standard stage were registered to the division axis. To do this a virtual coordinate system was defined first within a volume where the center of the volume defines the origin of the virtual coordinate system. Landmarks from all the cells were registered to the virtual coordinate system by applying translation and rotation in 3D where transformation function was estimated from the predicted cell axis and the desired axis in the virtual coordinate system. Bicubic interpolation was used during the rotation (Keys, 1981). Same transformation function was applied to both the landmarks so that the inter relationship between the landmarks was preserved in the registered image stacks shown in Figure 3.13 a, b.

Registered landmarks were represented using a cylindrical coordinate system (Szymanski, 1989), which has the advantage of polar coordinate systems (Brown and Gleason, 2004) where the object boundary in 2D is represented as a radial distance with respect to a reference point (centroid) inside the object as shown in 3.13 c. Cylindrical representation of 3D landmarks combines the polar based representation of the landmarks in all 2D planes. The centroids of each plane form an axis along z which was termed as cylindrical axis (Figure 3.13 d). However, after chromosome segregation it got difficult to represent both chromosomes accurately with a single vector representing cylindrical distances. To address this, two separate cylindrical representations were used to encode each of the chromosomes separately. Also, from

Figure 3.12: The standard cell representing each mitotic standard stage, defined in 3.2.3, was selected as described in Figure 3.11. The timing and duration is shown as colored bars, and the grey dashed bars show the standard deviation of the duration of each stage. The biological assignment was done manually based on the definition described in 1.1.1.

Figure 3.13: Generation of standard mitotic space.



late anaphase on the cell boundary starts to transform from one sphere into two spheres, and a cleavage forms in between them that makes it difficult to represent the cell boundary. Since the cylindrical axis was set along z, the radial distance was measured along xy, and thus the top and bottom part of the shape was not estimated as accurately as the middle slices were in order to generate the mean shape. As a result the top and bottom part of the mean shape got slightly elongated. To address this each of the landmarks was represented using two cylindrical distances, one by setting the cylindrical axis along z, and the other along the predicted cell axis. These two axes were orthogonal to each other due to the registration. Thus, by using a cylindrical system each cell is represented by 6 vectors that describe the cell boundary and chromosomal volumes 1 and 2 (if they exist) by separately using either z or the cell axis as cylindrical axis.

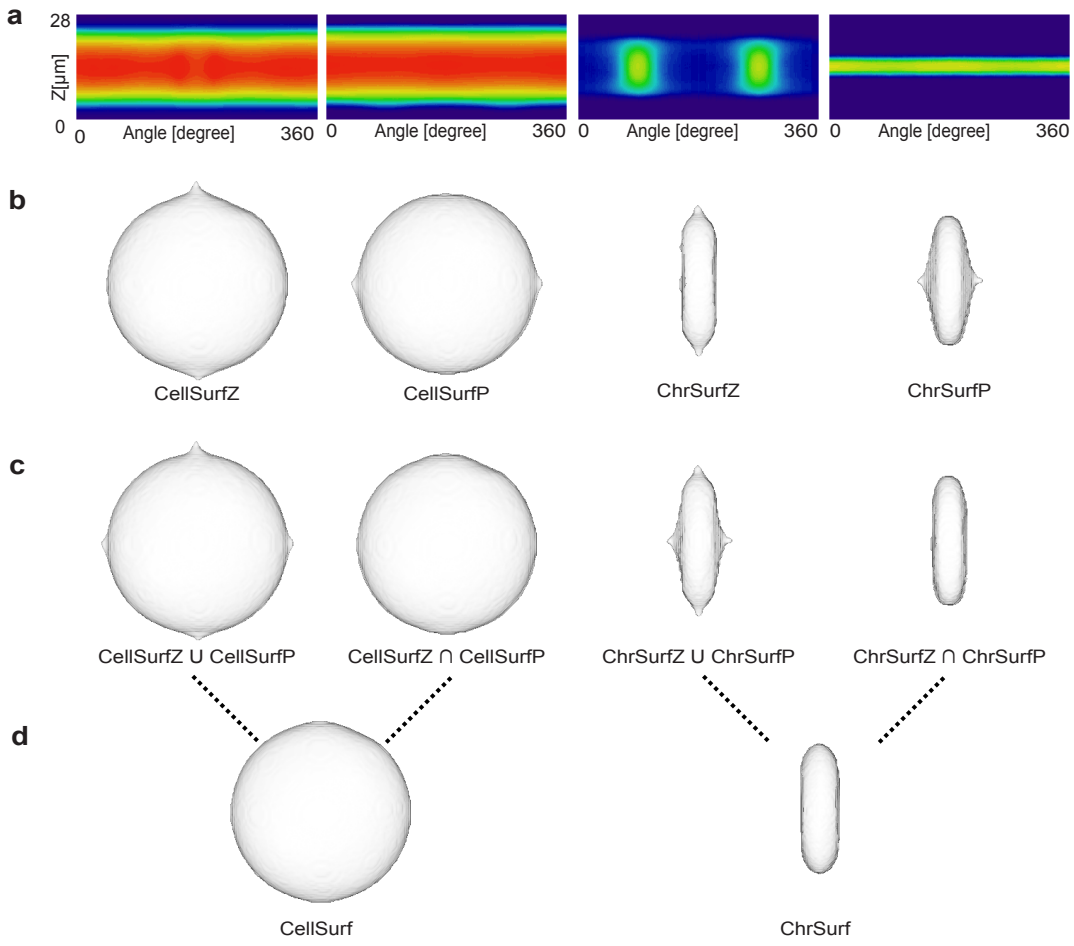
3.3.2 Reconstruction of the canonical model of mitotic morphology

Average cell shape was computed in two iterations by combining the vectors representing the cylindrical distance. In the first iteration, all cells within a particular stage were selected and 6 mean vectors were calculated from the corresponding vectors representing the landmarks. Mean vectors were then transformed back to a Cartesian coordinate system in order to obtain six corresponding binary image stacks, two for each of the mean cell and both chromosomal volumes (Figure 3.13 e). Mean cell region was obtained by combining two corresponding pairs of binary image stacks (one using z and the other using the cell axis as cylindrical axis). This was done by first taking the common space (intersection) between two binary images which was then extended radially until the mean volume of all the cells belonging to the stage was reached (Figure 3.14).

The mean shape computed from all the time points within a particular stage might be influenced more by cells that have a large number of time points than by those with fewer time points. This was addressed in the second iteration which takes only one time point (if it exists) from each cell. When a cell contained more than

Figure 3.13: (a) Examples of single cells taken from mitotic stage 10; (b) From left to right: 3D reconstruction of landmarks and registration of selected cells using predicted cell division axis; (c) Representation of chromosome and cell boundaries using polar coordinate system; (d) Representation of 3D landmarks using cylindrical coordinate system; (e) Average morphologies of cellular landmarks in cylindrical representation. (f) Average cell and chromosome surfaces in different mitotic stages; (g) Standard deviation of cell and chromosome surfaces. The figure was constructed by Julius Hossain with my help and input.

Figure 3.14: Merging two cylindrical representations.



(a) Example cellular and chromosomal surfaces with cylindrical axis marked in red. Vertical and horizontal axes define the predicted cell axis and z axis, respectively, as cylindrical axis. (b)-(c) Average cellular and chromosomal surfaces in cylindrical (b) and Cartesian (c) coordinate systems. (d) Union and intersection of two volumes occupied by average landmarks obtained from two cylindrical representations. (e) Combining the results obtained from two representations to generate final cellular and chromosomal surfaces. The result was obtained by Julius Hossain.

one time point in a stage then it selected the one most similar to the mean shape obtained in the first iteration. Here similarity between two shapes was measured from the deviation between two surfaces, and the corresponding vectors representing the cylindrical distance were used for this measurement. The standard deviation of all the cells that belong to the stage was also estimated in the same way. Thus, mean shape and standard deviation were generated for all the stages (8-20) for each of the landmarks (Figure 3.13 f, g). If the number of chromosomes varied in different cells that were at the same stage (due to the legging chromosomes, both daughter nuclei could be segmented as a connected component even in progressed anaphase), then a single chromosome was divided into two by fitting a plane orthogonal to the cell division axis passing through the origin. The mean shape was then generated as described above by treating each cell as having one of the chromosomes.

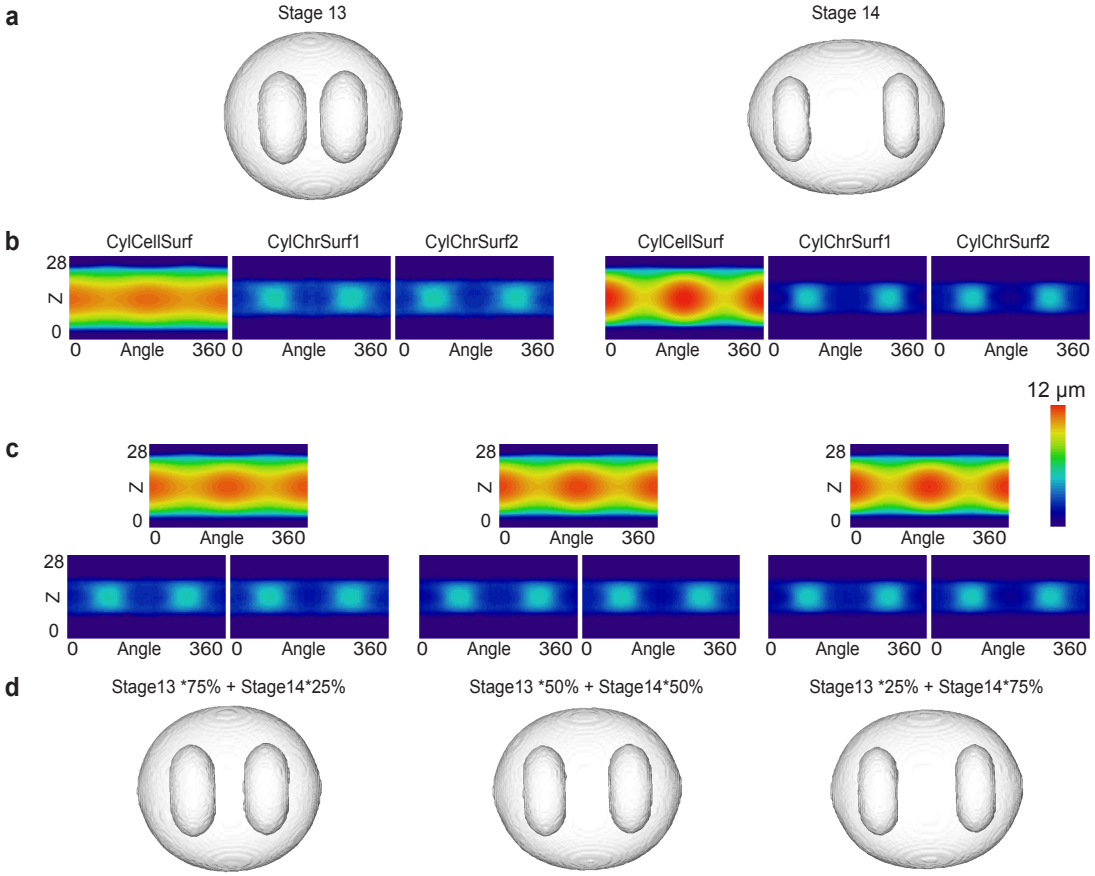
Once mean shapes for all the stages were obtained, any number of intermediate shapes between two neighboring stages could be generated by interpolation. To do this, mean shapes of two neighboring stages were represented using both cylindrical representations as described before. Then a combined vector was generated for each of the landmarks by taking a weighted average of two vectors representing the mean shapes where the weight of a cluster determined the similarity of that cluster to the interpolated shape. Theses vectors were then transformed back to a Cartesian coordinate system to generate an interpolated mean space in a similar way to (Figure 3.15).

3.4 Generation of the 3D protein density map through mitosis

3.4.1 Acquisition of the proof of concept data set

In order to test and show the power of the integrated experimental and computational pipeline, I generated a proof of concept data set with 12 well-known mitotic proteins. Due to a delay in cell line production using genome editing methods such as zinc finger nuclease (ZFN), transcription activator-like effector nuclease (TALEN), or Clustered regularly interspaced short palindromic repeats (Crisp R) techniques, I based my selection of cell lines only on the correct localization of the proteins and the goodness of fit of the cells and not on the technologies used for fusion protein

Figure 3.15: Interpolation of mitotic morphology.



(a) Standard mitotic spaces: stages 13 and 14. (b) Cellular landmarks: cell surface, chromosome surfaces 1 and 2 in cylindrical representation. (c) Interpolated landmarks in cylindrical representation, from left to right – 75% of stage 13 and 25% of stage 14, 50% of stage 13 and 50% of stage 14; and 25% of stage 13 and 75% of stage 14. (d) Corresponding interpolated cellular and chromosomal surfaces. The result was obtained by Julius Hossain.

expression (Table 3.1).

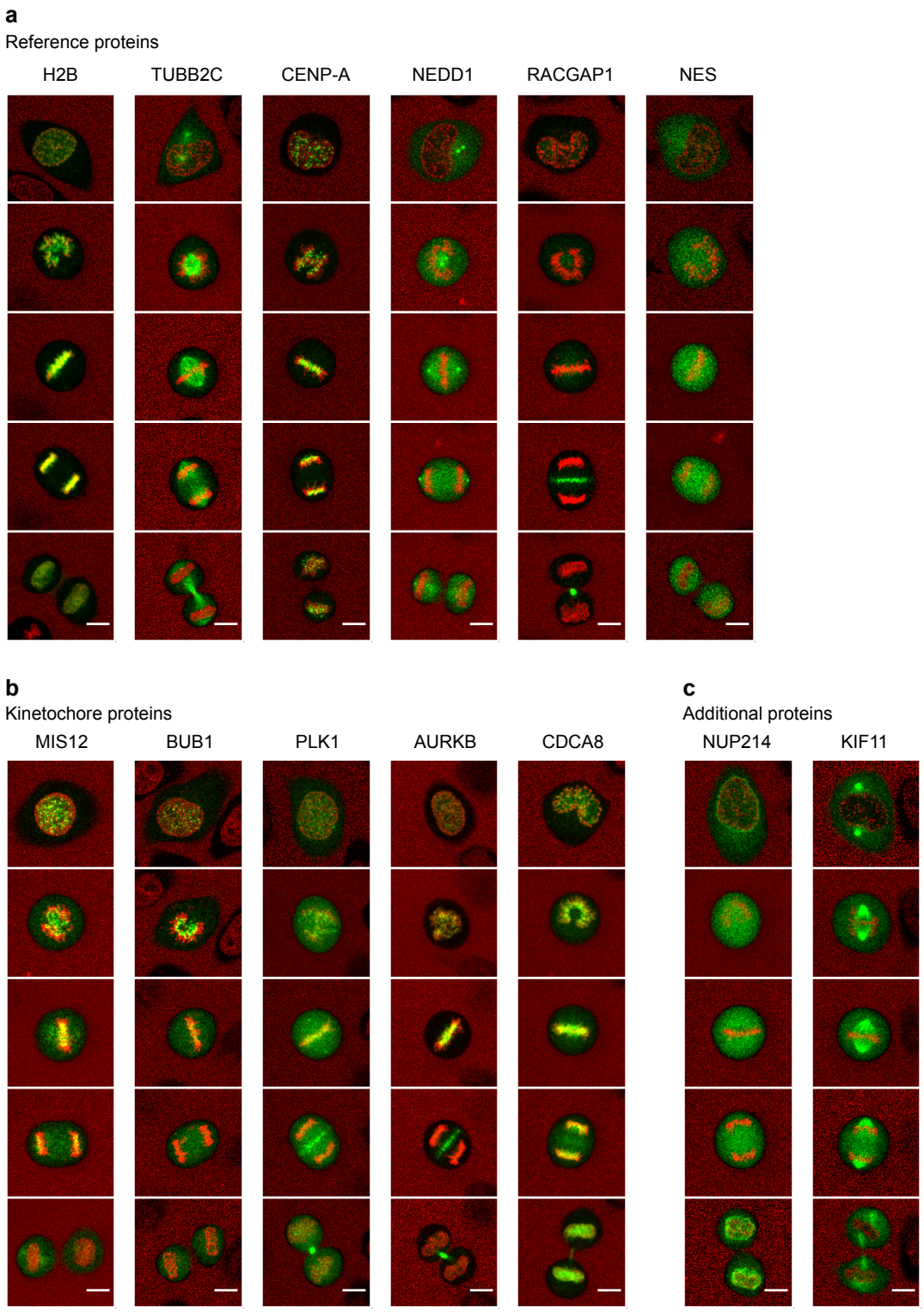
Table 3.1: The proof of concept data set

Gene	Technique	Division imaged	Division QC+
KIF11	BAC	17	16
MIS12	BAC	36	33
TUBB2C	BAC	36	29
RACGAP1	BAC	34	15
CDCA8	BAC	16	13
NEDD1	BAC	33	28
AURKB	BAC	22	17
AURKB	ZFN	17	15
NUP214	BAC	44	37
PLK1	ZFN	20	17
CENPA	cDNA	27	21
BUB1	CrispR	17	12
NES	cDNA	23	16
H2B	cDNA	35	33

First, only proteins that localized on typical mitotic subcellular structures were selected, such as the TUBB2C as structural element on the spindle, CENPA as component of the centromere/kinetochores, RACGAP1 on the miplane/midbody, and nuclear export sequence (NES) in the cytoplasm. For the centrosome, I chose the best available marker NEDD1, which also localizes to the pole area of the spindle. In addition, the chromatin marker H2B that was used for landmark labeling can also be used as a reference for the localization of proteins of interest (Figure 3.16 a).

In the proof of concept, I wanted to focus in particular on the kinetochore proteins in order to test the analysis in dynamics quantification. The reason for this are the distinct localization on the kinetochores, a reasonable expression level, and the large amount of existing knowledge about the structure, interaction network, assembly and disassembly of the kinetochores. I selected five proteins with kinetochore localization (Figure 3.16 b) but different dynamics: MIS12 is a protein localized in the outer kinetochore connecting both microtubule binding proteins KNL1 and NDC80 (Jia et al., 2013; Petrovic et al., 2010). As reported in Gascoigne and Cheeseman (2013), MIS12 dissociates from the kinetochore complex after the chromosome segregation. BUB1 is an important mitotic checkpoint serine/threonine-protein kinase (Roberts

Figure 3.16: Proof of concept data set with 13 proteins



et al., 1994). It bridges knl1 to spindle assembly checkpoint (SAC) proteins such as MAD1/MAD2 before the binding of microtubules to the kinetochores (Yu, 2002; Petrovic et al., 2010). BUB1 localizes in the cytoplasm during interphase and accumulates on the kinetochore at the very beginning of prophase (Grabsch et al., 2004; Johnson et al., 2004), which is important for the localization and recruitment of many other proteins such as PLK1 and AURKB (Qi et al., 2006; Boyarchuk et al., 2007). The polo-like kinase (PLK1) partially localizes to kinetochores from prophase to anaphase, with one of its phosphorylation targets being the anaphase-promoting complex (APC) (Hansen et al., 2004). In addition to this, PLK1 has functions on the spindle pole and migrates during anaphase to the central region of the spindle (Eot-Houllier et al., 2010; Glotzer et al., 2007). Aurora B kinase (AURKB) is a component of the chromosomal passage complex (Carmena et al., 2012) and plays a central role in ensuring the biorientation of the spindle attachment on the kinetochores (van der Waal et al., 2012). AURKB targets CENPA on the inner centromeres (Zeitlin et al., 2001) and maintains the correct localization of the SAC components (Lens et al., 2003). After anaphase, AURKB moves to the midplane/midbody and exerts functions in cytokinesis (Nguyen et al., 2014). The last protein I selected is Borealin (CDCA8), also a CPC component interacting with AURKB (Carmena et al., 2012). It is supposed to co-localize with AURKB throughout mitosis (Gassmann et al., 2004).

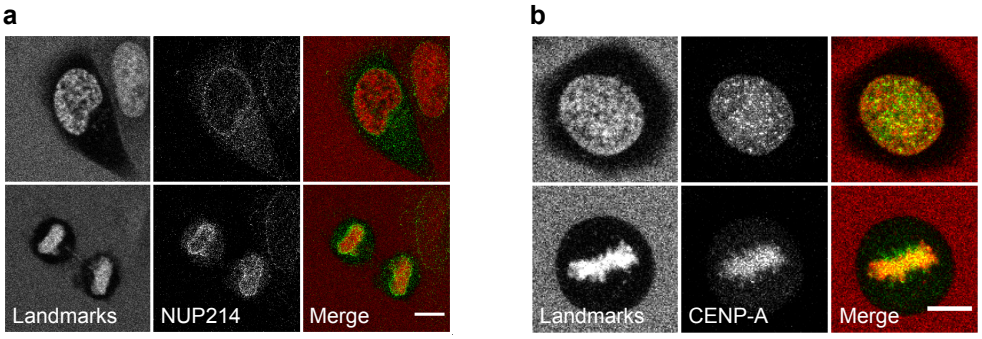
Bearing in mind that the proof of concept data set might be enlarged in the future such that structures other than kinetochores could be analyzed in a dynamic way, I added kinesin 5 (KIF11) and nuclear pore protein NUP214 to my list (Figure 3.16 c). KIF11 has been reported as an important component during the separation of the centrosome pair during prometaphase, and ensures the bipolar orientation of the mitotic spindle (Raaijmakers et al., 2012). The major localization of KIF11 is the spindle/spindle pole. NUP214 is a structural protein in the nuclear pore complex and localizes, under confocal resolution, on the nuclear envelope before NEBD, and then again in telophase (Strambio-De-Castillia et al., 2010; Xylourgidis et al., 2006). I aimed to test whether a new localization could be detected using our analysis pipeline.

The technology and source used for cell line production as well as the number

Figure 3.16: Example cells from the proof of concept data with the protein of interest in green and landmarks in red. Scale bar: 10 µm

of mitotic cells that were taken using the automatic calibrated confocal imaging platform are summarized in Table 3.1. Due to the variable expression levels in the cDNA and BAC expression systems, some of the proteins had challenging variations in their localization. For example, CENPA showed distinct kinetochore localization but also clear spread over chromosomes in many cells (Figure 3.17 b), and most cells with NUP214 show only very dim nuclear rim localization (Figure 3.17 a). This variation could not be excluded completely even though I performed a manual quality control by sorting out cells with very high or low protein expression.

Figure 3.17: Variation of protein localization



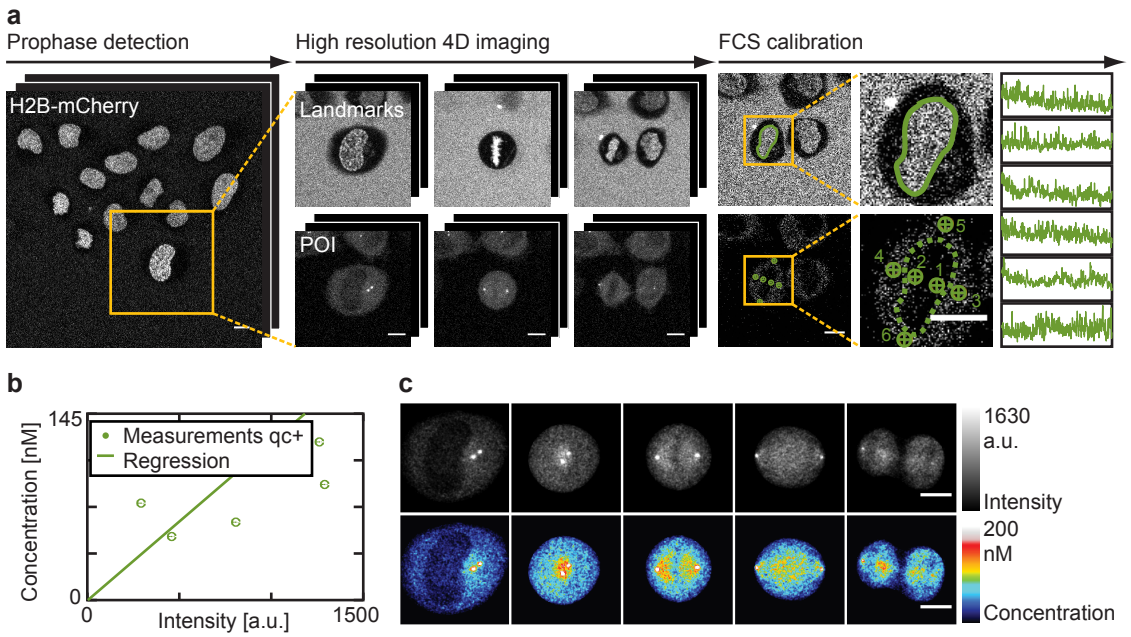
Example cells from the proof of concept data with a high background protein localization. Scale bar: 10 µm

3.4.2 Getting the absolute protein abundance using calibrated imaging

In order to measure the absolute protein abundance within the dividing cell, I used a calibrated automatic confocal live cell imaging pipeline that was developed in-house (A. Politi). Alexa 488 solution at known concentration was used for determining the confocal volume before each imaging session. Cells were cultured in Dextran-DY481XL solution, and pre-selected positions were imaged every 5 minutes for three planes at a low spatial resolution. Using Micronaut, the extension software from CellCognition for microscope online image analysis (Gerlich lab, IMBA), cells in prophase were identified using a previously trained classifier. The prophase cell was then imaged every 90 seconds for an hour at a higher spatial resolution at 31 z-planes covering the entire cell volume. After an hour, a selected z-plane was imaged using the same setup followed by six FCS measurements in the POI channel positioned in and around the nucleus closest to the center. In addition to that, FCS measurements

were also performed in HeLa cells expressing monoclonal EGFP for every session such that the brightness of each EGFP molecule could be determined. Details can be found in the Material and Methods section (Figure 3.18 a).

Figure 3.18: Calibrated automated confocal live mitotic cell imaging



(a) Automated mitotic cell imaging pipeline. Prophase cells are found in the low resolution imaging mode (left) and followed through mitosis in the landmark (upper line, middle) and GFP protein of interest (lower line) channels. At the end of imaging, six FCS measurements are performed in the GFP channel (right). (b) The calibration factor, calculated using linear regression of the FCS measurements, can be used for the translation of the imaging intensity into the absolute concentration map of the protein (c). Scale bar: 10 µm

All FCS measurements were processed by Fluctuation Analyzer (developed by Malte Wachsmuth at EMBL), and identical starting values were used for the numerical fitting (see 6.2). The number of molecules from the fitting was then used to calculate the local concentration of the protein by dividing by the confocal volume. The concentration was corrected if the protein molecule was brighter than the monoclonal EGFP. The GaASP detector intensity at the spots of FCS measurements was determined from the image taken before. The averaged intensity of the GFP channel in cell-free areas was considered as background. The ratio between the local absolute protein concentration and the background-corrected imaging intensity was calculated for each measurement point, and the averaged ratio of all measurement points after passing multiple quality control steps was used to generate the density map of the

protein by multiplying the ratio with the intensity value of all pixels in the high-resolution 3D image (Figure 3.18 b, c). Based on the concentration map the absolute protein abundance could be calculated by summing up the map over the cell volume.

3.4.3 Registration of protein distribution into the Mitotic Standard Cell

The chromosomal and cellular volume of the cells were first segmented using the computational pipeline developed by Julius Hossain as described in material and methods. Three features were extracted from the chromosomal volume for modeling the Mitotic Standard Time. The first derivatives of the feature sequences were calculated and all features were normalized as described before. The six-dimensional feature sequences were then aligned to the Mitotic Standard Time model and mitotic stages were assigned to the image sequence as reported in subsection 2.2.5.

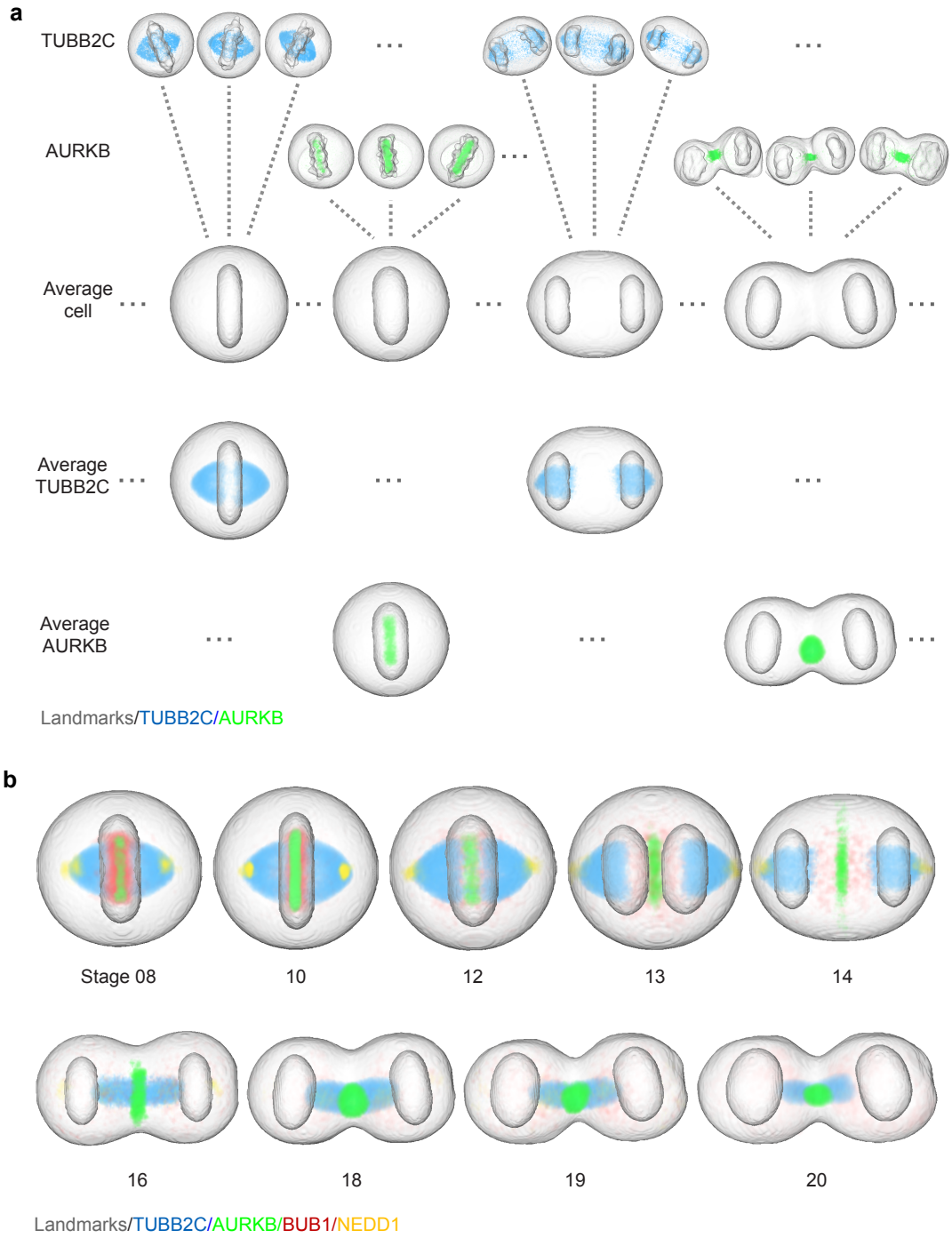
Julius Hossain then implemented algorithms to register the cells into the standard mitotic space. He wrote: ‘‘To do this, cells having same protein belonging to a particular mitotic stage were registered first to the average model of that stage by making use of the predicted cell axis. This brought all individual protein image stacks into the same coordinate system to generate the average probability map of that protein by simply taking the mean. The results of cellular region segmentation for individual cells were also used to take the mean precisely. Following the same way the average probability map of all the proteins was estimated one by one. Then, the standard mitotic space could integrate any number of average probability maps for visualization or analyzing co-localization (Figure 3.19).’’

3.4.4 The 3D protein density map

The 3D protein density maps represent the average of the protein distribution in 12 to 37 cells. By looking at each single map, the subcelluar localization of well known proteins in the proof of concept data set fully matched our expectations. All proteins had generally correct subcellular localization displaying the characteristic shape of the subcellular structure. Although detailed patterns got lost during the averaging such that single fibers of the spindle or single kinetochores could not be identified, 3D maps were able to provide important information about the protein localization.

Interesting observations were made for NEDD1. The protein is usually considered to be localized to centrosomes and spindle poles. However, a clear reduction of

Figure 3.19: Averaging protein distributions in the standard mitotic spaces.

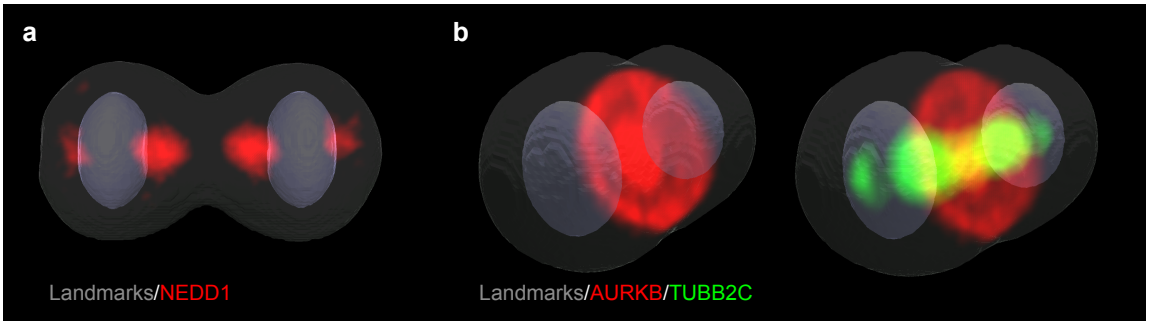


(a)-(b) Example cells with two proteins that are registered to the corresponding standard mitotic spaces shown in (b); (c) Average distributions of TUBB2C and AURKB; (d) Average distributions for 4 proteins are shown in different mitotic stages. This figure was provided by Julius Hossain.

the protein abundance on the spindle pole was shown from anaphase onward. In telophase, the protein distributed clearly to the central spindle volume (Figure 3.20 a). Previously, this observation was indicated in a few cells during manual inspection of the data. Based on the map, I was able to determine that this localization pattern was representative for the NEDD1 cells in my data set as a whole.

Concurrently visualizing multiple maps in the standard mitotic cell geometry allows the exploration of the co-existence and the co-localization of protein groups. The small size of the proof of concept data set made it difficult to generate any new findings, however it was interesting to see that AURKB formed a central plate and outer ring in the midbody area where TUBB2C indicating microtubules went through the central mass (Figure 3.20 b). This observation demonstrates the power of the 3D map in uncovering details at sub-structure resolution.

Figure 3.20: Example protein maps in the mitotic standard cell.



(a) Protein distribution map of NEDD1 in telophase. (b) AURKB distribution map in late anaphase (middle) co-visualized with the TUBB2C map (right). The figure was generated using the beta version of a web-based visualization app designed by Jean-Karim Hériché.

3.5 Quantitative data mining

3.5.1 Concept of a quantitative and automatic protein localization annotation

The 3D protein density maps include valuable and direct information on a protein's localization, and they enable the comparison of multiple proteins within the standard geometry. However, they are lacking a quantitative and biologically annotated description of protein localization. Thus, additional data mining techniques were developed for the quantification of the protein distribution.

The goal of the analysis is the detection of all underlying cellular structures to which the protein localizes and the determination of the amount of protein in each location. Traditionally, researchers would segment the area of a particular subcellular structure manually, and use image processing software to then quantify the protein amount within the area of interest. However, doing this for potentially hundreds of proteins would be very difficult. The number of landmarks for known subcellular structures is also limited during imaging, which means that the segmentation has to happen directly in the protein of interest channel. Since the contrast can vary strongly over all proteins, and the localization pattern can be a combinations of any number of subcellular structures, developing an objective automated pipeline for this purpose would be almost impossible this way.

As introduced in 1.4.3, machine learning approaches were widely used for solving similar problems. First, each image was translated into a numeric vector describing the features of this image, such as the texture, granularity, or in cell-specific cases also the signal distribution relative to cellular landmarks (Zhao and Murphy, 2007). For supervised approaches, a set of feature vectors of images with known annotations was used for the training, in which parameters of the model defining the boundary between different patterns in the feature space were optimized for achieving the least confusion in assigning the images present in the training set. This model was then used for the annotation of the remaining images. For clustering, the entire data set could be processed at once, and images close by in feature space would be assigned to the same cluster.

However, images with more than one basic pattern without any physical separation in space between the patterns were very challenging for classification based approaches, which assign each image into one of the pattern classes. These problems had to be solved using a regression (Peng et al., 2010) or factorization (Coelho et al., 2010) based approach. The former tried to improve the model by not only defining the boundary but by also fitting the quantity of each pattern during training. The latter tried to identify basic clusters and assign all data points as a quantitative combination of all clusters.

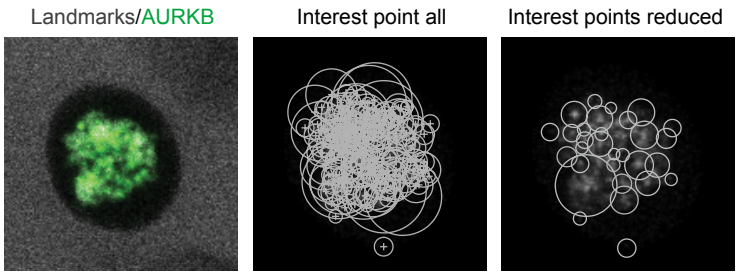
Therefore, the feature vector had to preserve the quantitative nature of the image data and reflect the quantitative distribution of the proteins to multiple locations, i.e. the combined localization could be represented as the linear combination of

multiple basic localization elements where the weight vector corresponds to the protein fraction in each localization element. Traditional image features such as the Haralick texture feature (Haralick et al., 1973) were not suitable. Thus, I decided to use a Bag-of-Words approach (Csurka et al., 2004). Each image (the ‘‘article’’) was represented by a series of local image spots (‘‘words’’), where each of them had a ‘‘meaning’’ assigned by a ‘‘dictionary’’ generated by clustering a set of ‘‘words’’ described by a common feature vector. The feature vector for the entire image composed as the frequency of each ‘‘meaning’’ should be completely linear additive. Similar methods were already used for the classification of cell images (Coelho et al., 2013), but not yet for the quantification of the underlying patterns.

3.5.2 Dissecting a protein signal into a series of interest points

To select ‘‘words’’ in the image, I chose the detector of the Speed Up Robust Features (SURF) algorithm reported in (Bay et al., 2006) and implemented in the Computer Vision Toolbox in MATLAB, to detect spots with a higher contrast relative to their local environment. Here, the protein z-stack concentration map was processed using a Gaussian filter at 0.5 micrometer followed by projection along the z-axis and normalization to the theoretical saturation intensity. SURF interest points were then detected at four scales, sized from single kinetochores up to the whole cell (default implementation in MATLAB, see 6.3.9).

Figure 3.21: Detection and selection of interest points



Interest points (grey circles) were extracted using SURF detectors in the protein of interest channel and selected further in order to avoid overlapping.

Interest points were subsequently ranked by their local contrast, with those displaying insufficient contrast being discarded. Even so, overlapping interest points remained and further selection was required (Figure 3.21, middle panel). An interest

point was kept if its center and more than 60% of its intensity were not overlapping with any other interest points, or if the mean intensity of its surrounding environment was higher than that of other overlapping interest points. After this process, the cell image was dissected into a series of interest points where the signal was reasonably well covered by the selected interest points but only a small fraction of signals was represented in more than one interest point (Figure 3.21, right panel).

Each of these interest points was then described by a numerical vector. Due to the low contrast nature of live cell fluorescence images, values calculated by the SURF descriptor did not give the best performance in further analysis steps. Thus, I implemented descriptors quantifying features in the following four categories: the location features, the correlation features, the intensity distribution relative to the center, the orientation of the intensity (explanation see below, implementation see 6.3.9).

For the location features, the original 3D coordinates of each pixel taken in the maximum projection were compared with the 3D segmentation of the cellular and chromosomal volumes. The relative position of all pixels in the interest points were summarized and percentage thresholds were set to categorize the interest points into cell boundary, cytoplasm, nuclear boundary and nuclear regions.

The correlation features describe the relations between the interest points and the chromatin signal and between the interest points and the midplane volume. For the former, the 3D image of chromatin was projected using the pixel coordinates taken for the protein channel maximum projection. The co-variance between the intensity values in the protein channel and the chromatin channel within the interest point area was used as feature value. For the later one, midplane/midbody was predicted as the symmetrical plane between two daughter chromosomal volumes (see material and method) and dilated into a volume. The intensity fraction located within the midplane volume was calculated for each interest point as a separate feature.

The feature vector derived from the intensity distribution relative to the center was implemented based on spin image features in (Lazebnik et al., 2005). Interest points were cropped into squares and re-sized to a standard of 73 by 73 pixels. A two-dimensional intensity histogram of five distance bins relative to the center was calculated based on the frequency of one of six intensity levels. Here, the ‘‘soft histogram”-approach was used such that each pixel could contribute into multiple

bins weighted by the intensity and location distance relative to the bin definition. The contribution of a pixel with a distance relative to the center of x and an intensity of I was defined as:

$$exp(-\frac{(x-d)^2}{50} - \frac{(I-i)^2}{0.005}),$$

for the bin (d, i). The two-dimensional histogram was then unfolded into a vector describing whether the signal was homogeneous, randomly distributed within the interest point or accumulated at a particular distance to the center, i.e. as a bright spot or a ring.

Furthermore the features describing the intensity orientation were calculated on the cropped square interest points. For each pixel in the interest point, pixels in the neighborhood at a distance of two pixels were assigned a 1 if they were brighter than the center pixel and 0 otherwise. The interest point was then summarized into a uniform Local Binary Patterns vector (uLBP) (Heikkilä and Pietikäinen, 2006) by counting the frequency of random pixels (with more than one 0-1 transitions in the neighborhood) and of each type of uniform pixel (0 to 8 brighter neighborhood pixels with none or only one 0-1 transition). In order to increase the signal to noise ratio, I further summarized the vector into four numbers: first, the Euclidean distance to the uLBP vector of a simulated interest point with a completely random signal; second, the frequency of uniform pixels with three to four brighter pixels in the neighborhood, indicating stripes; third, the ratio of pixels with a neighborhood of only darker pixels to those with random pixels; and fourth, the dominant direction where the local signal orientation was calculated as the direction, numbered one to eight, separating the neighborhood signal of an uniform pixel into two equal intensity fractions. Thus, only a four-dimensional vector was used to describe the local signal orientation of an interest point.

In total, each interest point was numerically defined by a 41-dimensional feature vector as the concatenation of all four categories of features.

3.5.3 Generation of the dictionary of interest points

The next step in a ‘‘bag-of-words’’ approach is the generation of a dictionary for the ‘‘words’’. The concept here is to compare a training set of ‘‘words’’ in their feature

space and distinguish clusters of ‘‘words’’ with similar meaning. Later, any ‘‘words’’ can then be assigned to a particular meaning cluster based on its feature vector.

I randomly selected 5% of the interest points in the complete data set to construct the training set. The structure of the training set was not obvious, and I therefore decided to perform clustering at multiple layers. First, all training interest points were separated into 16 clusters based on their localization feature and their contrast to the environment. For each location category, i.e. cell boundary, cell cytoplasm, nuclear boundary, nuclear inside, the interest points were clustered into four metric levels determined using median values.

I then checked the maximum metric value in each cluster first. If the maximum metric value was below a pre-defined threshold, interest points in that cluster were not separated further. Otherwise, interest points in each of the location-contrast clusters were further divided based on their correlation features. Using the pre-defined thresholds, data points were clustered into anti-correlated, no-correlation, and positive-correlated groups based on the intensity correlation to chromatin for interest points localized on the nuclear boundary or in the chromosomal volume and based on the correlation to the midplane volume for cellular localized interest points.

Next, I used dbscan (Ester et al., 1996) to cluster the interest points in each of the location-contrast-correlation clusters further if there were more than 15 data points in the current cluster. A principle component analysis (PCA) (Jolliffe, 2002) was performed on the data points in the feature space defined by 34 intensity distribution and orientation features. Components covering 85% of the variance were used to reconstruct the data set in a reduced feature space for clustering. Based on the density of the data in the feature space, parameters for the dbscan clustering algorithm were automatically calculated (see 6.3.10) and used for the clustering. The algorithm used was implemented by Tran et al. (2013) and tries to identify clusters in which the data points could be linked with each other at a distance below the pre-defined threshold.

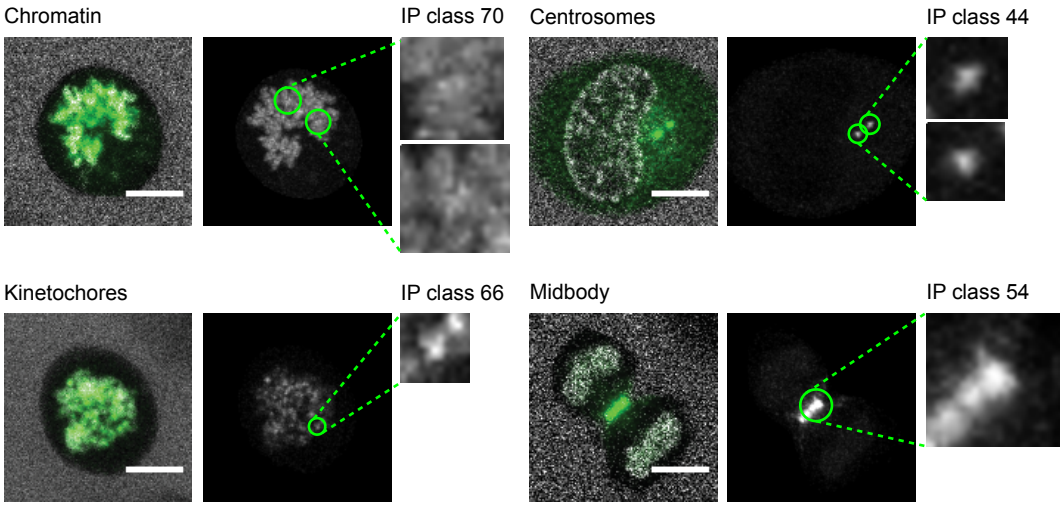
The final clustering step was performed only to the clusters with the highest contrast value in their location category. Using the 30 intensity distribution to center features, the median intensity value for each distance bin was calculated. Three classes were defined by pre-defined thresholds, i.e. dim interest points, where in all distance bins more than half the pixels were dimmer than a threshold; homogeneous bright interest points where the averaged median intensity throughout all distance

bins was high and the standard deviation was low; and mustered interest points where the standard deviation of the median intensity was high throughout all distance levels.

The total number of clusters was not deterministic since the training set was randomly generated, which meant that the number of clusters calculated by dbscan was inherently unpredictable. However, after multiple runs it turned out that for our data set the algorithm stably yielded between 70 and 90 clusters, which gave us a good dictionary size to work with.

I manually checked the interest points in each of the clusters. For most of them, a clear biological pattern could be identified among all data points (Figure 3.22). However, many of the biological patterns were represented in multiple clusters, and needed to be considered for further processing steps.

Figure 3.22: Interest point classes of subcellular structures



Interest point (green) classes defined by the dictionary often represent one particular biological pattern. Scale bar: 10 µm.

3.5.4 Numerical representation of a protein image

With the dictionary generated in the previous subsection, a ‘‘text’’ consisting of a series of ‘‘words’’, in our case an image as a list of interest points, can be interpreted based on the frequencies at which each ‘‘meaning’’ appears. Each interest point

was assigned to one of the clusters in the dictionary based on their feature vector compared to the separation thresholds and, in the case of clusters generated by dbscan, to the same class of the closest interest point in the training set. The fraction of intensity in each of the clusters composed the feature vector for the image (Figure 3.23).

Since some of the meaning clusters represent the same or very similar data points, similar interest points can be assigned to different clusters and thereby cause instability in the feature representation of similar images. Thus, in some of the feature dimensions, the numerical value over the division of a cell could display dramatic ups and downs that are biologically meaningless and technically difficult to handle. I therefore smoothed the feature vector over time for each cell.

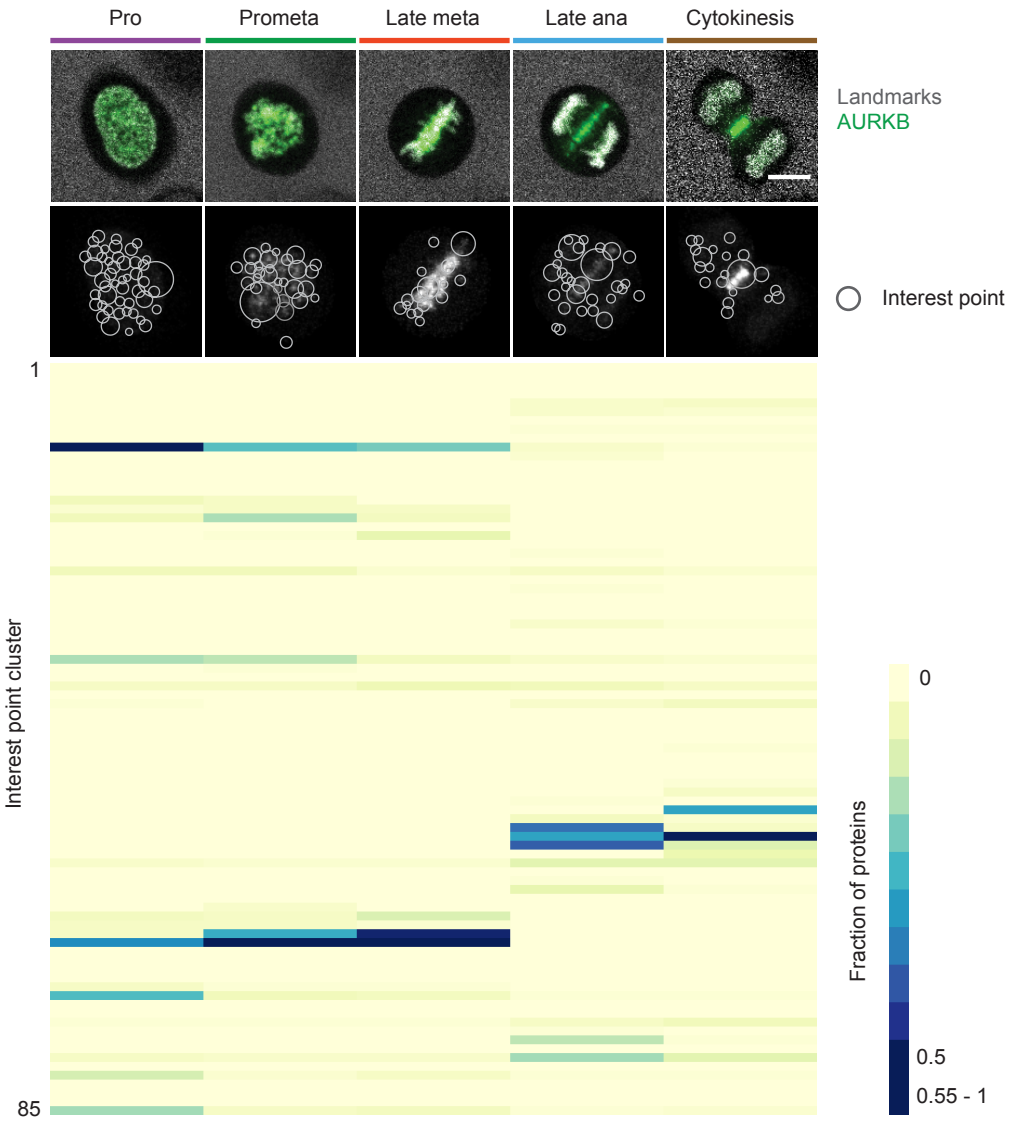
Smoothing turned out to be a nontrivial problem as proteins often change their localization dramatically and accordingly cause strong and fast changes in the feature vector. This behavior in the data can potentially be of particular biological interest, and should therefore not be filtered away. Thus, the size of the window for smoothing has to be determined based on the data.

I assumed that very similar images, i.e. having a high intensity correlation with each other, should have very similar feature vectors. For each mitotic image video, the intensity correlation between two adjacent frames was calculated and mapped with a pre-defined function into a correlation factor, a value normalized to between zero and one (Figure 3.24, see 6.3.12). For each frame to be smoothed (center frame), the contribution of all neighboring frames was calculated as the product of all correlation factors from the neighboring frame to the center frame. Finally, the total contribution was normalized to one. Figure 3.24 shows the smoothing function and its effect for an example sequence. Using this smoothing procedure, flips in the time-resolved feature vectors were under control, whereas genuine dramatic changes in the protein localization were still preserved in the trace through feature space.

3.5.5 Recognition of protein networks using non-negative tensor factorization

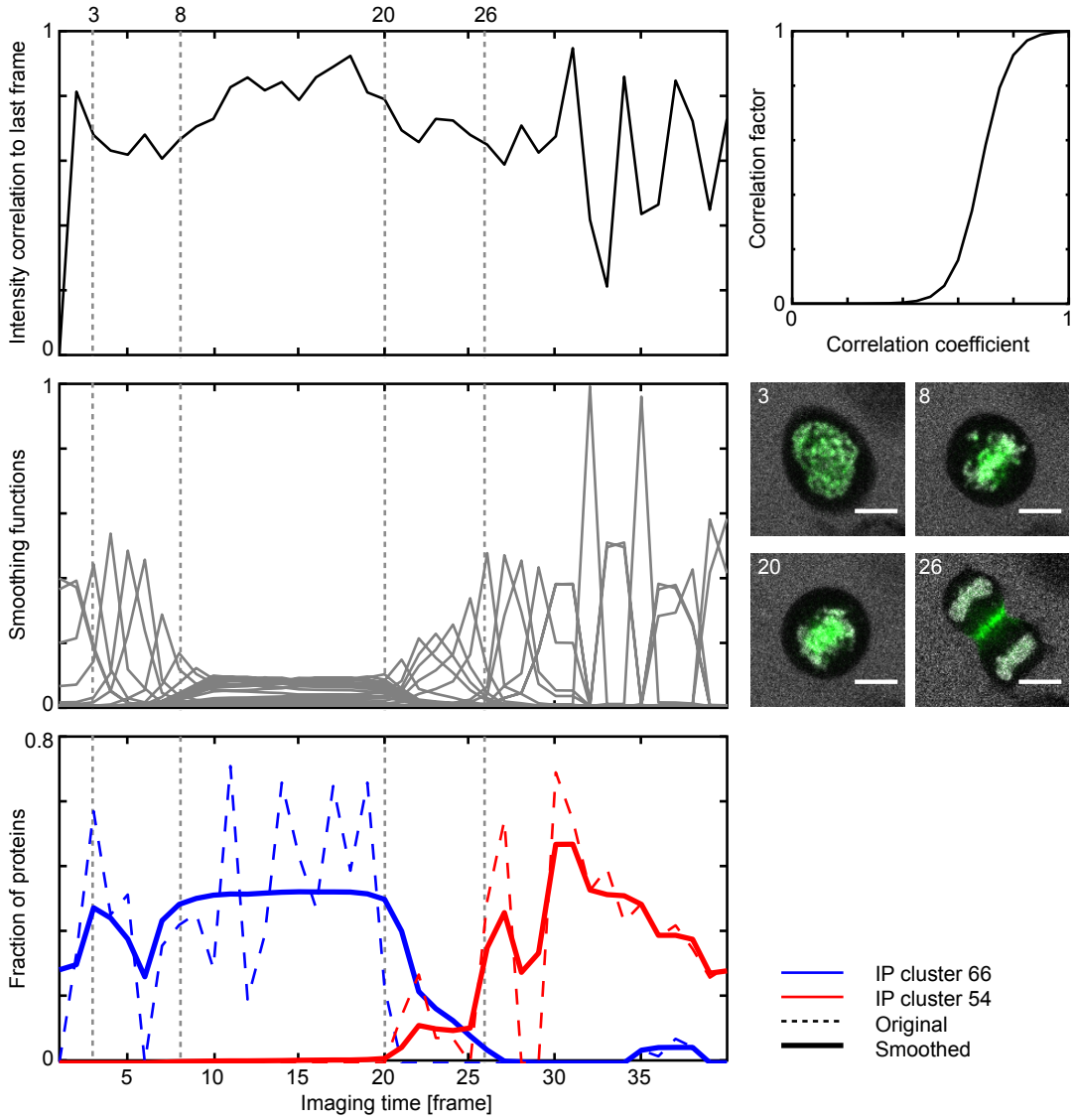
After the feature extraction, each image was represented as a 85-dimensional feature vector conserving the relative quantity of each of the interest point cluster. With the prospect of having data with a large amount of unknown proteins, the first thing I

Figure 3.23: Images as feature vectors



Using the ''bag-of-words'' approach, the feature vector over time as a heatmap for one example cell with AURKB as the protein of interest. The heatmap was generated using BlueGecko. Scale bar: 10 µm.

Figure 3.24: Time-dependent smoothing of the feature vector trace over time.



The implementation for smoothing the feature vector trace is shown based on the cell in Figure 3.23 and the interest point classes shown in Figure 3.22. The correlation coefficient (upper left) was translated into correlation factor by a pre-defined sigmoid function (upper right). Based on the correlation factor over time, the smoothing function was calculated for each point of time (middle left). The trace of interest point classes describing kinetochores and midbody before and after smoothing are illustrated in the lower panel.

wanted to do was to uncover a structure within the data by grouping similar proteins. Traditionally, time-resolved data were analyzed for each time point individually without considering the connection between points of time. I wanted to expand this approach to analyze the entire time-series using unsupervised techniques.

The approach to solving this problem was implemented and carried out by Dr. Jean-Karim Hériché, with whom I had many fruitful discussions on the subject. The following text in this subsection is based on notes on the topic formulated by him.

All images were temporally registered and assigned to one of the 20 mitotic standard stages. Thus, a cell could be represented by several vectors at a given mitotic stage. These duplicates were replaced by their average, resulting in each cell being represented by only one vector per mitotic stage. Each protein was then represented at each mitotic time point by the average of all its vectors present at that time point. The resulting data set was a 3-dimensional tensor $\mathbf{X}$ of 13 proteins (including H2B) times 85 features times 20 mitotic stages.

We viewed canonical subcellular localization as latent features of the data, which meant that we assumed that, at any time point, the observed vector for a protein was generated by a combination of the different canonical subcellular localizations the protein occupied at this mitotic stage. A protein vector x could then be expressed as the product of a subcellular localization membership vector z and a matrix $\mathbf{A}$ of canonical subcellular localization features. Therefore, we wished to model our data tensor $\mathbf{X}$ such that for each frontal (temporal) slice:

$$\mathbf{X}_t = \mathbf{Z}_t\mathbf{A} + \mathbf{E}_t$$

where $\mathbf{Z}_t$ was a matrix whose rows were localization membership vectors and $\mathbf{E}_t$ was a matrix containing the errors.

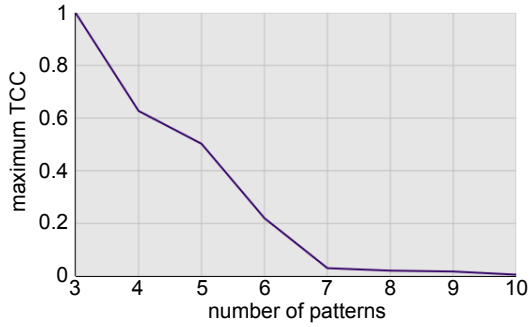
Given that all feature values were non-negative, a candidate solution for each time point could be found by non-negative matrix factorization (NMF) of individual matrices $\mathbf{X}_t$ (Lee and Seung, 1999). However, processing time independently results in loss of information with the undesirable effect that different canonical localizations are learned for different mitotic stages. Simultaneous non-negative factorization of a set of matrices is a special form of non-negative tensor factorization (NTF) that could be reduced to a standard NMF using column-wise unfolding of the data tensor $\mathbf{X}$ (Cichocki et al., 2009):

$$\mathbf{X} = \mathbf{ZA} + \mathbf{E}$$

where $\mathbf{X}$ was formed by vertically stacking the $\mathbf{X}_t$ matrices and $\mathbf{Z}$ was formed by the correspondingly stacked $\mathbf{Z}_t$ matrices and $\mathbf{E}$ contained the errors. $\mathbf{Z}$ and $\mathbf{A}$ were then found using multiplicative updates (Lee and Seung, 1999) to minimize the objective function $\|\mathbf{X} - \mathbf{ZA}\|$ where $\|.\|$ indicates the Frobenius norm. As a final step, the rows of $\mathbf{Z}$ were normalized to sum 1. Values in $\mathbf{Z}$ could be interpreted as fractions of the amount of protein (captured by the features) present at each canonical localization.

The method requires to chose the number k of canonical subcellular localizations we wanted to represent our data with. There was no good strategy for finding this number a priori as increasing k corresponds to a higher resolution of the localizations description, e.g. a low k would result in lumping all chromatin proteins together, while a higher k would resolve kinetochore proteins from other chromatin proteins. Thus, the optimal number of subcellular localizations was partly subjective, depending on the level of granularity desired. However, we could use heuristics to help guide the choice of k. If the number of selected canonical localizations was too low, many proteins would share the same temporal profile, i.e. their corresponding vectors in $\mathbf{Z}_t$ would have been highly similar for all mitotic stages. As more canonical localizations were added, we could expect more proteins to resolve into distinct profiles, i.e. the similarity between their corresponding vectors would decrease until eventually adding more canonical localizations wouldn't improve resolution and similarity would stop decreasing.

Figure 3.25: Tucker's congruence coefficient of the NTF of the data.



Tucker's congruence coefficient was calculated during the NTF of the data for determining the right number of basic patterns objectively. The figure was provided by Jean-Karim Hériché.

Similarity between vectors across points of time could be measured using Tucker's

congruence coefficient (TCC) (Tucker, 1951). Therefore, for each value of k from 3 on, we plotted the maximum value of TCC between the proteins (Figure 3.25). The first value of k for which the maximum TCC reached a low value plateau, which was 7 for our proof of concept data, indicated that there were enough canonical localizations to describe each protein individually, and therefore this value of k represented an upper bound on the number of canonical localizations. Because the NMF algorithm could converge to a local minimum of the objective function, 10 runs with random initialization of the matrices were performed, and the run with the lowest objective function value was kept.

3.5.6 Unsupervised dynamic clustering of proof of concept proteins

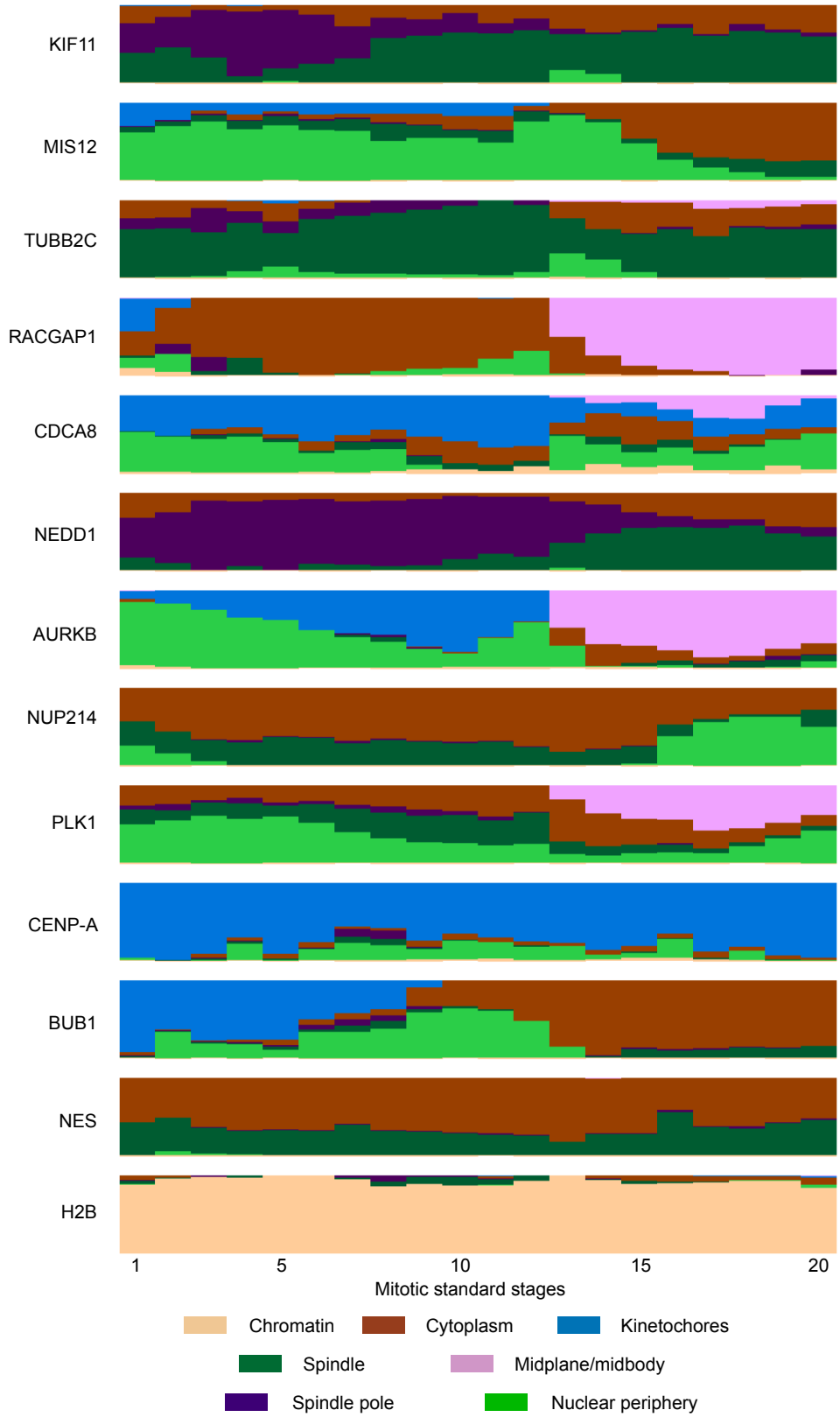
Based on the TCC value, we identified seven classes in our data set. The fractional affiliations to the clusters in all mitotic stages were visualized in Figure 3.26. According to the proteins assigned to each of these clusters, we could assign the classes to major mitotic subcellular structures: chromatin (H2B major localization), kinetochore (CENPA major localization), midbody (RACGAP major localization from anaphase onset), spindle pole (NEDD1 major localization until anaphase), cytoplasm (NES localization), spindle (TUBB2C localization), and chromatin periphery (NUP214 localization after anaphase).

Besides those proteins with a single major subcellular localization that were used for the interpretation of the classes, other proteins with mixed localizations were correctly assigned into the expected classes as well. For PLK1 before anaphase the localization on both kinetochores and spindle poles was identified (Figure 3.26), and KIF11 was assigned to both spindle pole and spindle (Figure 3.26). Moreover, the migration of proteins between subcellular components could be clearly distinguished. While AURKB, PLK1 and CDCA8 displayed a clear reduction of their kinetochore and nuclear localization and an immediate accumulation on midbody at anaphase onset, BUB1 and MIS12 displayed a more graduate loss of kinetochore/nuclear localization in metaphase and anaphase, respectively (Figure 3.26).

The quantitative nature of the feature extraction and NTF allowed us to successfully assign fractions of proteins to each of the localization patterns as evidenced by

Figure 3.26: Each protein was quantitatively assigned to one of the seven localization patterns determined by the NTF through all mitotic stages. The results were obtained by Jean-Karim Hériché.

Figure 3.26: Output of the NTF analysis on the proof of concept data.



comparison to existing knowledge. However, the lack of cytoplasmic localization in most of the proteins suggests that most of the classes might not represent a single subcellular structure. For example the spindle class might represent a combination of spindle and cytoplasm since NES, NUP214 (between NEBD and telophase) and other proteins without clear spindle localization could be found in that class. The nuclear periphery might also be a composite of signals on the boundary of the chromosomal volume, including both nuclear periphery and cytoplasm. In spite of these uncertainties, the NTF managed to automatically identify the major sub-cellular structures without supervision and allowed quantitative subcellular localization assignment in our proof of concept data set.

3.5.7 Determine protein fractions in pre-defined subcellular structures

Although the NTF could assign the distribution of proteins into different localization clusters quantitatively, depending on the data quality and the proteins in the data set, the automatically identified localization clusters might correspond to a mixture of multiple subcellular structures or incomplete structures, making their interpretation difficult. Moreover, since the algorithm is completely unsupervised, it needs data with little noise and good structures. Thus, the averaged feature vector through all cells was taken for each protein in each mitotic stage in the analysis. However, the assembly or dissociation of some protein clusters might happen in the order of tens of a seconds. Also the variations between cells could be potentially interesting in itself. Thus, methods need to be developed that allow researchers to analyze the dynamics of cellular structures specifically at higher temporal resolution for each single cell.

Considering how many researchers are interested in the dynamic nature of specific well-defined cellular structures, I decided to build a supervised framework for the automatic quantitative assignment of protein localization. In order to keep the annotator as generic as possible and the definition of subcellular localizations constant over time, one annotator was built for the entire mitosis. This means that for each of the pre-defined classes representing one subcellular structure, images of proteins with the corresponding localization in all mitotic stages, if feasible, were selected for the training of the annotator. In our proof of concept test, structures with their representative proteins are listed in Table 3.2.

Different models could be used for the annotator. Since the quantitative detection

Table 3.2: Reference structures

Localization	Gene	Stages
Cytoplasm	NES	1-20
Chromatin	H2B	1-20
Kinetochores	CENPA	1-20
Centrosomes	NEDD1	1-16
Spindle	TUBB2C	4-20
Midplane/Midbody	RACGAP1	13-20

of complex protein localizations on multiple cellular structures was needed, regression with models allowing a good resolution between 0 (not localized) and 1 (pure localized) would provide the best fit. Thus, classification algorithms such as support vector machine (Chang and Lin, 2011) were not suitable due to their binary outputs. I tested both linear and logistic regression instead and chose a linear model at the end due to its simplicity, good results, and the linearity of our 85-dimensional feature vector in the localization quantities.

In Peng et al. (2010), a specific data set of images with known distribution over two subcellular compartments was generated for the training of the unmixing model. However, generating such a data set for many subcellular localizations in living mitotic cells is impossible with existing techniques. Since proteins with clear single localization were selected for training, I estimated the objective function for the regression as follows: I performed thresholding on the 3D image stack based on the algorithm developed by Otsu (1975) (see 6.3.8) where the foreground signals f were considered as the fraction of proteins localized on the structure of interest. $f^{2/3}$ was taken as the approximation for the foreground in 2D as the feature extraction was performed on 2D images. The remaining signal fraction $(1 - f^{2/3})$ was assigned to the cytoplasmic location. For numerical reasons, the objective function was normalized, for extreme long-tail distributions log-normalized, to the range between 0 and 1 for each localization (see 6.3.12).

I performed a quality control step before the linear regression on the feature space. Interest point clusters which did not have more than a 5% signal in any of the cells of the entire data set were deleted, as were clusters which were not present in more than 20% of the cells with any proteins. Two global features were added. The first

measured the fraction of proteins in the segmented chromosomal volume in 3D. The second calculated the accumulation of the protein in the estimated spindle volume, which was defined as the convex hull volume of the intersecting points of the division axis on the cell boundary and the chromatin volume (see 6.3.8). All features were normalized to the range from 0 to 1.

For each of the six pre-defined localizations, up to 400 cell images were randomly selected to construct the training set. Linear regression models were trained using elastic net (Zou and Hastie, 2005) on each localization using a one-against-all approach. To balance the positive and negative data points for each of the location classes, an equal number of images with and without the corresponding location were used for training. The training algorithm was implemented in MATLAB as function ''lasso''. A ten-fold cross-validation was used to validate the quality of the linear model, and parameters achieving the best fit with a relaxation of one standard deviation were chosen. The predicted values were then back-calculated to the 3D foreground.

Judging by the correlation value R-square defined as,

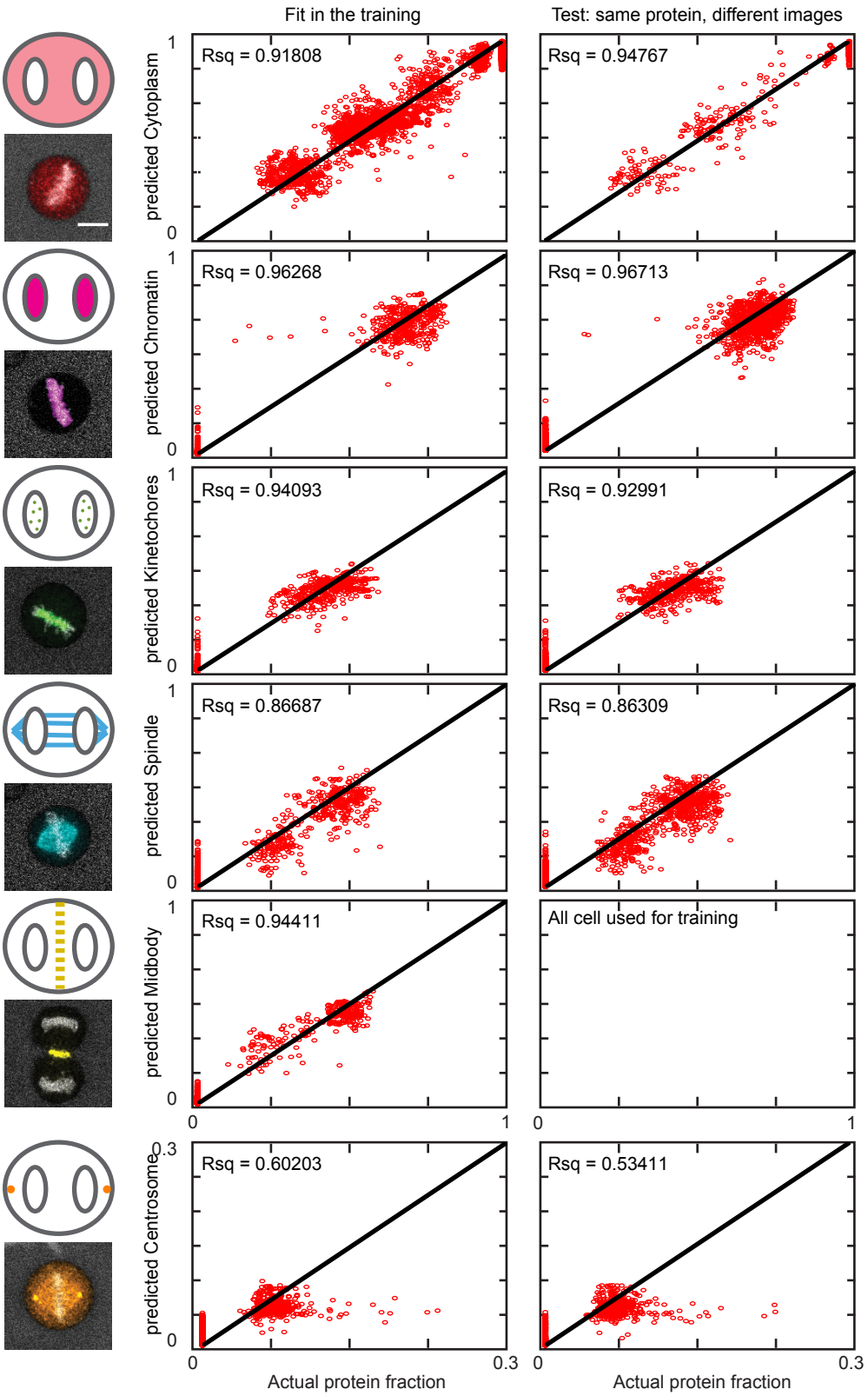
$$R^2 = 1 - \frac{\sum(p_i - f_i)^2}{\sum(f_i - mean(f_i))^2},$$

where p was the predicted and f was the actual fraction of proteins in a particular localization (Figure 3.27), the linear correlation of the protein fraction localized on each of the reference mitotic structures between the training objective function and the predicted value was very good. The lower accuracy of the prediction for the centrosome localization might be due to the fact that the reference protein NEDD1 distributes on the spindle pole as well, and the extremely low fraction of the protein localized to the structure. The prediction for images with reference proteins that were not shown to the classifier gave overall satisfactory results too (Figure 3.27). Thus, the classifier was used to assign all images in the proof of concept data set.

In order to validate the prediction for proteins with more than one mitotic localization, a number of images were selected and processed by manual segmentation for each of the mitotic structures. The comparison between the manual segmentation output and the prediction of the classifier is shown in Figure 3.28. Protein localization with

Figure 3.27: Six reference proteins built a training set for the pre-defined subcellular localizations, and the linear model was constructed using an elastic net algorithm. The correlation between the objective function and the prediction for the training set (on the left) as well as to a testing set that was not used for training (on the right) are shown. The R-square value was used for the evaluation of the model.

Figure 3.27: Prediction output for the training and testing set.



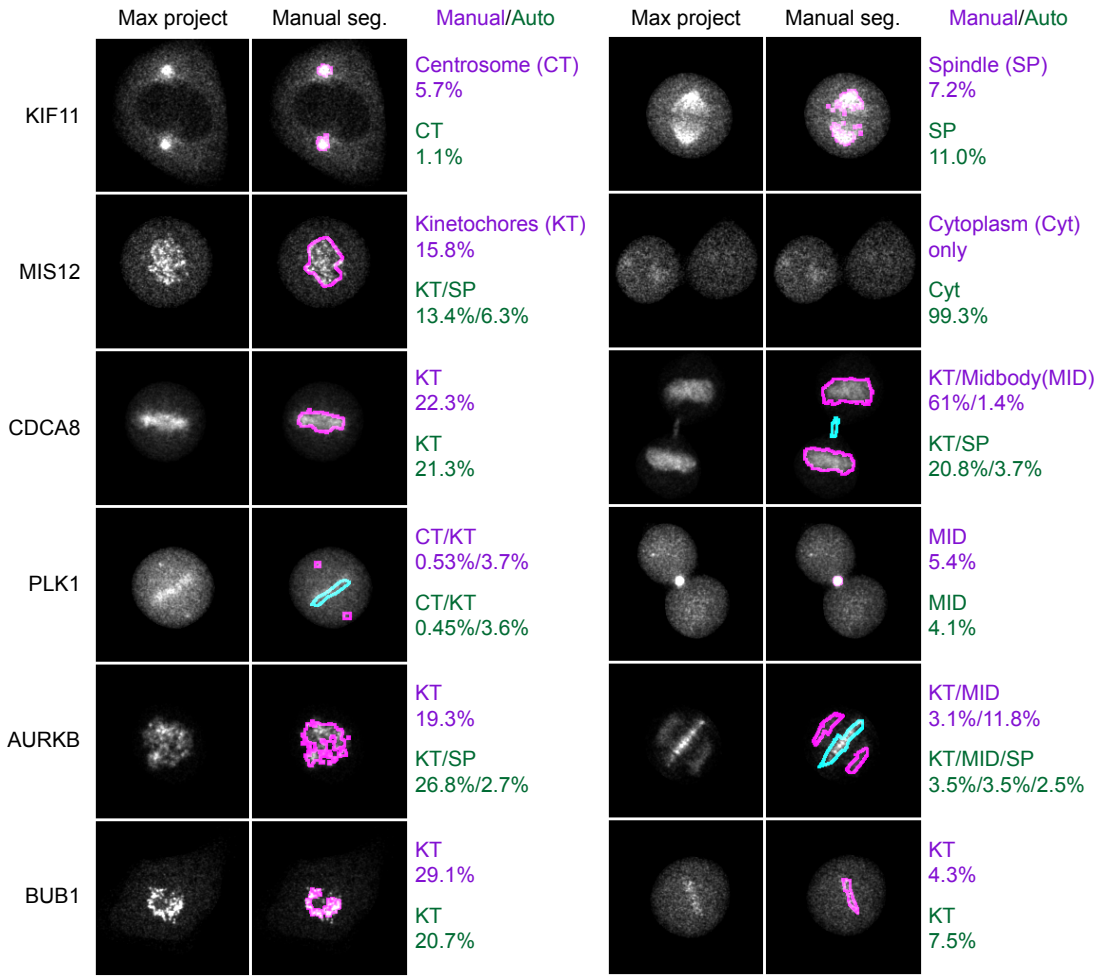
very low contrast or protein fraction (below 2%) might be missing in the estimation. Structured protein localization with very high protein fraction (beyond 50%) were usually under-estimated. The spindle localization was generally over-estimated, and the central spindle confused slightly with the midplane/midbody localization. For the remaining cases, the protein fraction on each mitotic structure could be relatively well predicted with an error smaller than 10% of the total protein.

3.5.8 Kinetic readout using supervised framework

The good subcellular localization prediction at single cell single frame level reassured me of the power of the analysis. More interestingly, I looked at predicted protein fluxes between mitotic structures by connecting the predicted localization over time. As an example the fraction of AURKB localized on each of the mitotic structures over time was predicted and temporally registered as shown in Figure 3.29. As observed at the single-cell level, the predicted localization value is noisy. I then took the time information into account and smoothed the prediction for each cell throughout mitosis. Since genuine sudden changes in the localization behavior should not be replaced, I only smoothed predicted value on peaks, i.e. both neighboring data points had either lower or higher predicted value than the central point and the difference in predicted values between the central and neighboring points was high (implementation see 6.3.13). In such cases the mean of both neighboring points was taken for the central point. The result after smoothing is shown in Figure 3.29 on the right.

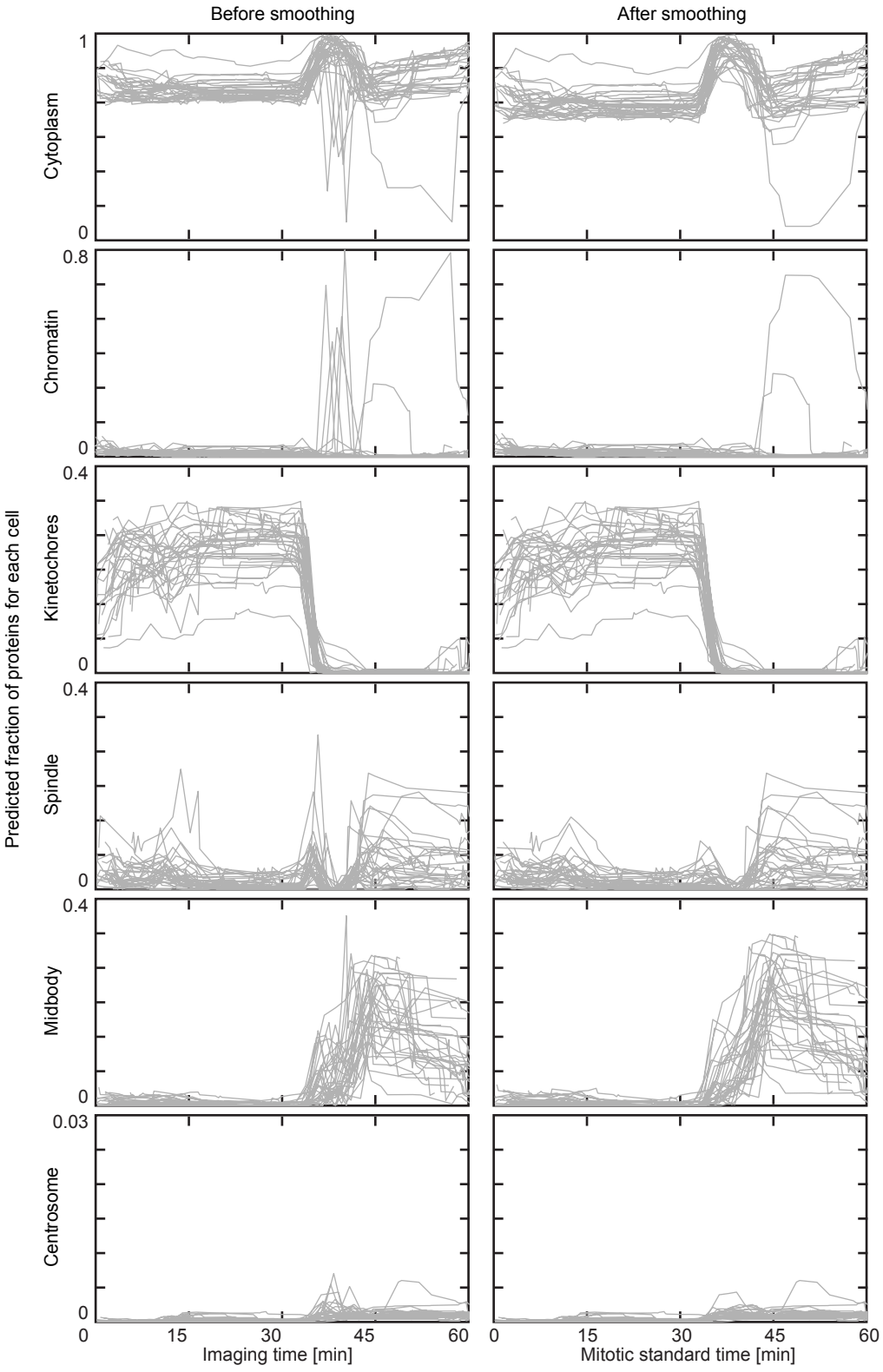
As a temporal resolution of 90 seconds for image acquisition is still low for some stages during mitosis, especially anaphase, combining information from multiple cells would give us a more complete picture about the protein movement dynamics. Based on the temporal registration of each cell, the time resolved localization predictions were aligned to the mitotic standard time, where only 40 out of the 237 frames were filled with data. Using linear interpolation, the missing predictions could be estimated for the remaining frames. Furthermore, due to the quantitative nature of our imaging data, the predicted fraction of proteins for different structures could be multiplied with the absolute total number of proteins in the cell such that the number of molecules on each of the mitotic structures was predicted. By averaging through 12 to 37 cells with the same protein, I obtained a very smoothed prediction

Figure 3.28: Prediction VS manual annotation.



Six cells with different proteins were randomly selected for validation. Two frames of each cell were manually processed by adjusting the intensity threshold to a value that preserved the characteristic localization pattern of the protein in 3D. Fractions for the protein in each subcellular structure were calculated based on the image after thresholding (lilac) and compared to the automatic annotation (green).

Figure 3.29: Prediction output for AURKB at the single cell level.



about the absolute number of each protein on six major mitotic structures throughout division with a temporal resolution of 15 seconds (Figure 3.30).

At the qualitative level, the prediction was very similar to our expectation. As an example, the accumulation of RACGAP1, AURKB and PLK1 each on the midbody after anaphase onset was well predicted and the disassembly of AURKB, BUB1 and MIS12 from the kinetochores was nicely estimated. Also a small fraction of TUBB2C was assigned to the kinetochore in prometaphase and at the start of anaphase. More challenging localizations, such as the centrosome localization of KIF11, the nuclear localization of RACGAP1 in prophase and the kinetochore distribution of PLK1 were also predicted as per our existing knowledge. However, some of the predictions were probably less accurate. The spindle localization dominating in TUBB2C and KIF11 matched the expectation, but its existence in all proteins was definitely due to confusion with a cytoplasmic localization. In the early anaphase, the midplane localization seemed to confuse with the cytoplasmic localization as well to a certain degree. The reason could be the low contrast of the RACGAP1 midplane in the training set. And unfortunately the midbody localization of CDCA8 was underestimated. But overall, the quality of the annotation was good. Some of these problems could probably be solved by using better contrasted images of endogenously tagged cells for training of the annotator.

Since both the validations at the single cell image level and on the protein fluxes show reliable results, especially for the kinetochore localization, I wanted to use the annotator to analyze the kinetics of kinetochore disassembly quantitatively. In the data set, six kinetochore proteins were present, and the post-mitotic disassembly of five of them has been reported previously (Sharp-Baker and Chen, 2001; Kelly and Funabiki, 2009; Shao et al., 2015; Gascoigne and Cheeseman, 2013). However, none of the research has analyzed all of them in a single study such that a quantitative comparison of the kinetics between different kinetochore components becomes possible. Although these five proteins were separately imaged in different cells using our computational pipeline, the data were completely integrated into the standard mitotic time and space.

I analyzed the predicted absolute abundance of all six kinetochore proteins through

Figure 3.29: The model estimates the fraction of AURKB localized in each pre-defined subcellular structure at the single cell level over time (grey lines). An additional smoothing step deleted spikes in the time-resolved prediction traces.

Figure 3.30: Predicted time-resolved localization profiles.

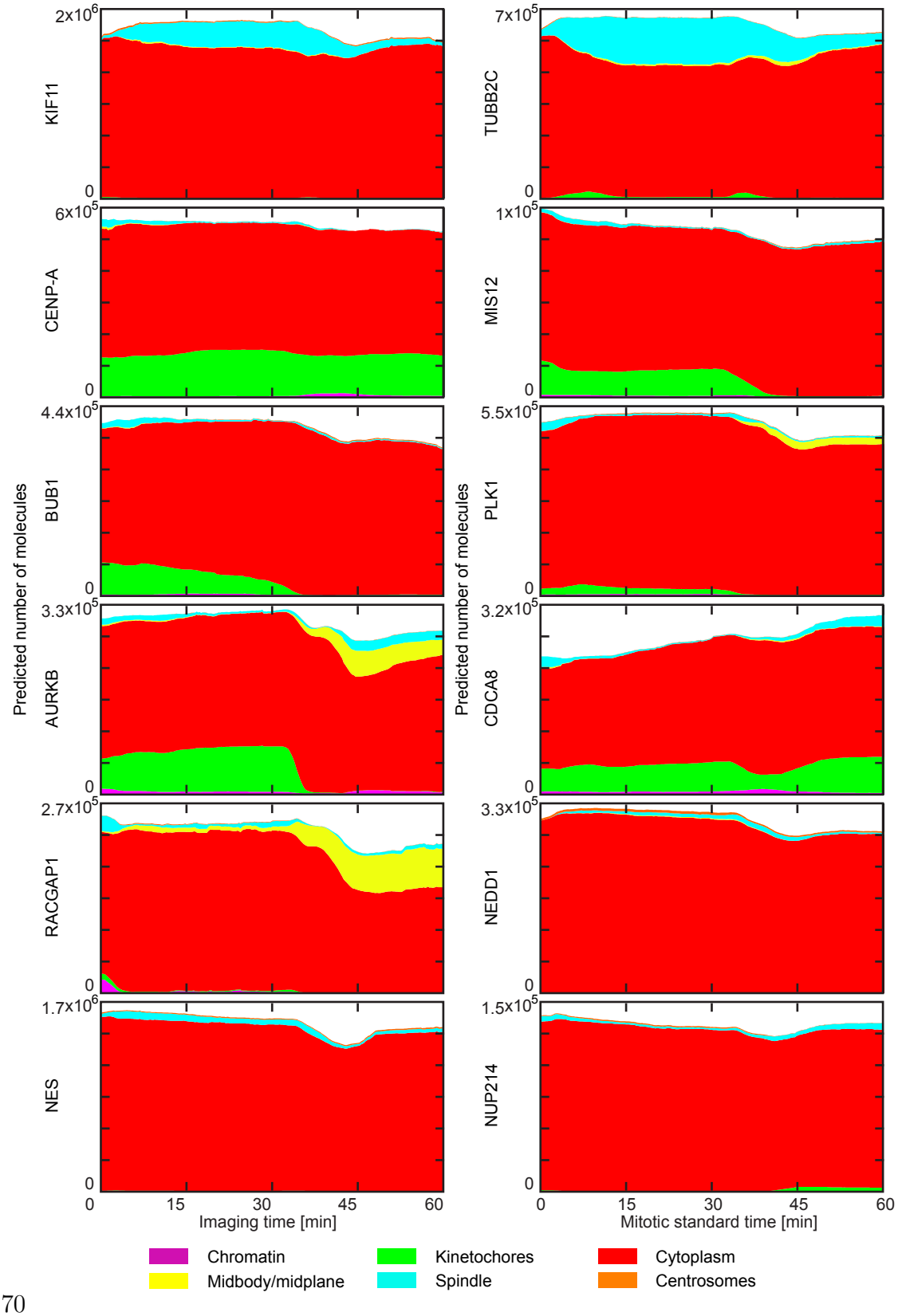


Table 3.3: Kinetochore disassembly kinetics

	Pre-anaphase dissociation			Anaphase dissociation			Post-anaphase accumulation		
Gene	Mitotic time [min]	Fraction [%]	Rate [10^4/ min]	Mitotic time [min]	Fraction [%]	Rate [10^4/ min]	Mitotic time [min]	Fraction [%]	Rate [10^4/ min]
BUB1	17.5	54	0.22	32.6	45	0.67			
PLK1	8.4	100	0.18	34.4	44	0.38			
AURKB	-	-	<0.1	35.0	95	3.2			
CDCA8				36.3	50	0.75	45.9	67	0.42
MIS12	2.4	44	0.17	36.9	72	0.20			

mitosis (Figure 3.31). The only protein staying on the kinetochore for the entire time is CENPA, and the number of CENPA molecules measured using biochemical methods matched with my prediction (Hasson et al., 2013; Scott and Bloom, 2014). BUB1, which was reported for disassociation from the kinetochores in metaphase, clearly showed two disassembly processes with different kinetics, one before and one after anaphase onset. Thus, I used the additive of two logistic functions:

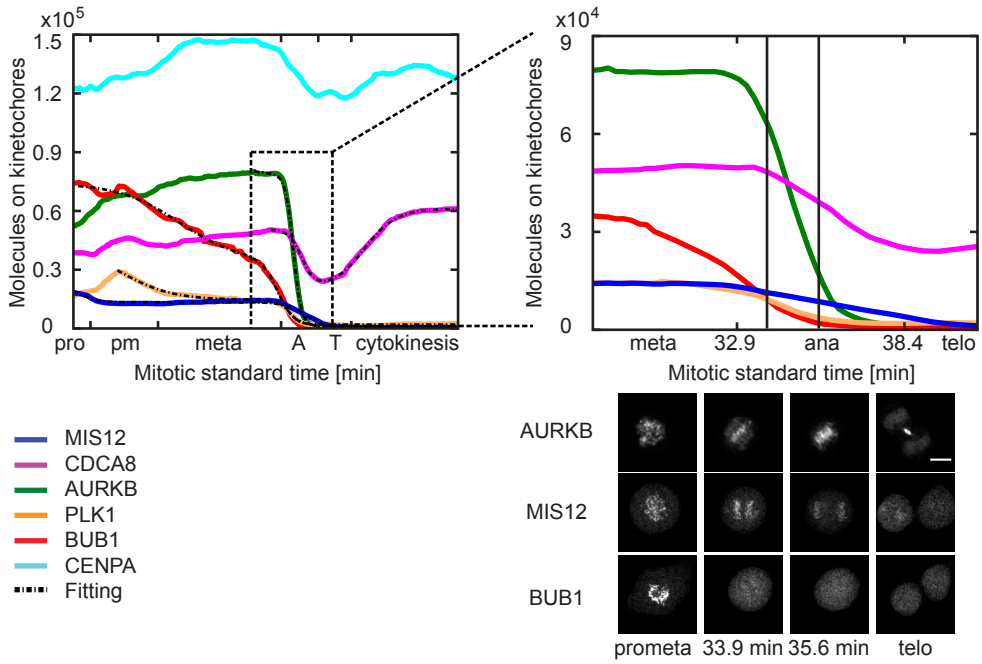
$$f = offset + \frac{fraction_1}{1 + exp(\tau_1 \cdot (t - time_1))} + \frac{fraction_2}{1 + exp(\tau_2 \cdot (t - time_2))},$$

to fit the kinetic curves of all five disassociating kinetochore proteins where $offset$, $fraction$, τ and $time$ are variables. The starting time for the fitting was selected as the maximum kinetochore accumulation before anaphase. The output of the fittings are summarized in Table 3.3. A model for kinetochore disassembly based on the findings is illustrated in Figure 3.32.

Both AURKB and CDCA8 were identified as having only one dissociation event starting at almost the same time in anaphase. Surprisingly, CDCA8, which had been thought to be part of CPC like AURKB, has a slower disassociation rate and returns to the kinetochore in telophase. PLK1 has a slower but otherwise similar anaphase disassembly as AURKB, and displays a clear drop after NEBD. Different disassociation kinetics were shown for BUB1 where half of the proteins are slowly released from kinetochores through prometaphase and metaphase, and the remaining half disassembles rapidly from the kinetochore as the earliest protein just after anaphase onset. Different from the kinetics of other kinases, the majority of MIS12

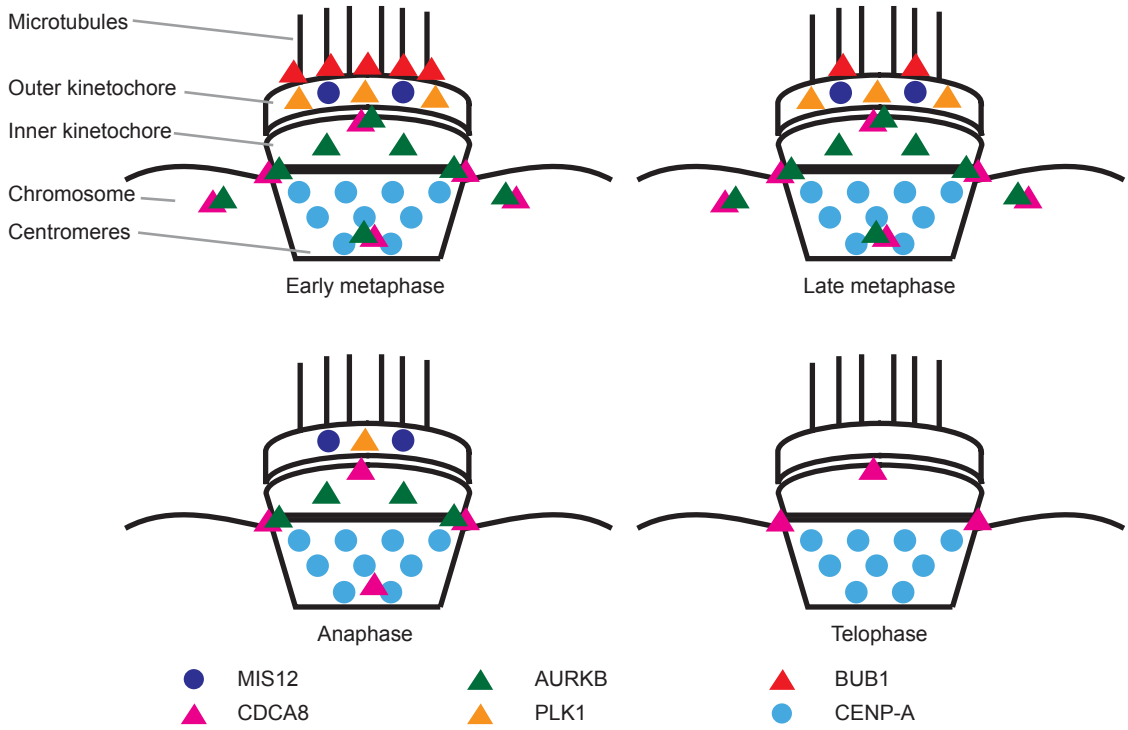
Figure 3.30: The predicted fraction of proteins was multiplied with the absolute total number of proteins within a cell to obtain the molecule number in each mitotic subcellular structure. The averaged prediction throughout 12 to 37 cells was taken for each protein. H2B could not be plotted due to the missing calibration.

Figure 3.31: Predicted dynamics of kinetochore disassmebly.



The predicted numbers of proteins on kinetochores were averaged through 12 to 37 cells for six kinetochore proteins. The predicted dynamic of the post-mitotic kinetochore disassembly could be assured by images of single cells. The kinetics could be quantified by fitting with a logistic function (dashed lines). The results were shown in Table 3.3. Scale bar: 10 µm

Figure 3.32: Illustration of kinetochore disassmebly.



Based on Figure 3.31 and Table 3.3, the predicted kinetochore disassembly dynamics are illustrated.

unbinds slowly from the outer kinetochores in late anaphase and telophase. Some of these kinetics matches with what was reported previously (Sharp-Baker and Chen, 2001; Kelly and Funabiki, 2009; Shao et al., 2015; Gascoigne and Cheeseman, 2013), others were surprising but do not disagree with previous reports as these proteins were analyzed in an integrated manner. Looking at many cell videos in our data set, the prediction seems to match what we see qualitatively. The biological discoveries need to be validated quantitatively using specifically designed experiments. Further experiments can be suggested to study the mechanism and relevance of our predicted kinetochore kinetics.

Chapter 4

Discussion

4.1 The Mitotic Standard Cell model can be used for mapping the dynamic protein network throughout mitosis

In this work, a spatio-temporal canonical model of a standard mitotic human cell was built where the cellular and chromosomal volumes were used as references for the cellular context. The model represents the average of more than 300 mitotic HeLa cells and allows an objective view of the mitotic progression in terms of dynamic morphology changes. By indicating the existence of 19 distinct mitotic transitions it promises to provide a much more detailed mechanistic view of mitosis, and it will be very interesting to test the value of the identification of these additional transitions once the canonical model developed here has been populated with the full complement of mitotic proteins. The proteins whose abundance and/or subcellular location changes at a transition will be the basis to predict the functional significance of each transition. Moreover, for the first time, the standard mitotic cell model developed here provides a quantitative spatio-temporal average of mitosis in human cells, as well as the cell-to-cell variations of all measured parameters.

Using this model, any image sequence of mitotic HeLa cells with landmarks showing the chromosomal and cellular boundaries can be registered into the model both in time and space. The automatic temporal registration was compared with manual annotation and showed a very high accuracy. Thus, the standard mitotic cell model allows a direct quantitative comparison of different cells within a standardized cellular context. By integrating multiple cells expressing the same protein into the model, a 3D protein distribution map can be generated for any mitotic stage from late prometaphase onward, as defined objectively by the mitotic standard time. As the

distribution maps of all proteins were generated within the same standard model, multiple maps can be virtually ''co-expressed'' and thereby allow the quantitative exploration of protein co-localization. Compared to the 3D protein map previously reported by Schauer et al. (2013), our method did not need the support of adhesive micropatterns and can therefore be applied to cells and processes which cannot be studied on micropatterns. In order to quantitatively compare the similarities in protein localization, temporal registered image sequences were analyzed using different data mining methods that employ both supervised and unsupervised machine-learning based identification of subcellular localization and abundance patterns.

For validation, I generated a proof-of-concept data set of well-known mitotic proteins. The integrated nature of the model offers the potential identification of new functional structures that are difficult or impossible to identify by eye alone. The power of the averaged 3D map in uncovering details in subcellular structures was demonstrated for AURKB where a central mass and outer ring in the midbody area were resolved, with TUBB2C, that likely corresponds to microtubules, seemingly going through the center of the midbody (Figure 3.20 b).

Annotations by experts are usually biased. Results against expectation and existing knowledge are often classified as ''spurious results'' during manual inspection of data. However the objective nature of the model allows the integration of even rarely seen features, putting them on a much more robust footing. This was impressively demonstrated for NEDD1 where a spurious observation made in only a few cells during manual inspection, that NEDD1 might localize on the central spindle in telophase, could be clearly identified and followed in the integrated model (Figure 3.20 a). Furthermore, due to the unsupervised nature of the standardization and clustering methods we used, protein localization patterns could be defined completely unbiased. The power of this is demonstrated by the cluster ''chromatin periphery'' that was robustly detect despite the very weak fluorescence signal of NUP214.

The quantification of transient and therefore rarely seen features has been challenging for a very long time. It was almost impossible to integrate data on dynamic cellular processes from experiments performed separately. Using the standard cellular context that provides high spatio-temporal resolution, my model allows analysis of fast cellular processes that occur over seconds to a few minutes, such as the kinetochore disassembly during anaphase. The dissociation kinetics of different components of

the kinetochore could be quantified and compared. Clear differences in the disassembly process were identified for several proteins including CENPA, MIS12, CDCA8, AURKB, PLK1 and BUB1 (Table 3.3). That CDCA8 and AURKB show different kinetics was unexpected from the literature and will serve as a motivation for more detailed mechanistic studies in the future.

This is the first time that such a model has been developed for a dynamic cellular process, where both the spatial and temporal progressions were integrated into one standardized framework. The model was not only used for simply assigning subcellular localizations to proteins but furthermore allowed to mine those for analyzing protein dynamics during mitosis. While the current version of the integrated model of a standard mitotic cell is based on only 13 proteins, it already gives a tantalizing glimpse of what a model populated with data from all 600 mitotic proteins could offer.

One could think of further validations of the model's functionality, for example proteins currently expressed in two separate cell lines could be co-expressed tagged with differently colored fluorophores, and the output from the computational integration could be compared against the biologically integrated readout. The estimated kinetics of the kinetochore disassembly based on the predicted kinetochore localization could also be further validated by comparison with results generated using a manual kinetochore segmentation and quantification pipeline. Nevertheless, I believe that our methods are generic, and can be used widely for analyzing live cell microscopy data of mitotic cells by mapping the observed proteins into our standard model of the mitotic cell.

4.2 Comparison of data mining methods

In this work, three data mining methods were used to analyze the localization dynamics of mitotic proteins within the standardized cellular environment. The first one was the unsupervised direct integration and visual comparison as a 4D protein density map, the second one was the unsupervised detection of protein localization clusters using non-negative tensor factorization (NTF), and the third one was the supervised classification/regression of protein distribution at single cell level using a linear model trained by reference proteins with known localizations.

The advantage of the 4D map was the direct and completely unprocessed nature of the data mining. Biologists could look by eye at each of the integrated and standardized protein maps, or even multiple maps at the same time, as they would usually do on the microscope. They could use their human experience and explore the data for any question they were interested in. The protein density map allows a continuous presentation of the protein distribution without any pre-defined and sometimes subjective borders for subcellular structures. The disadvantage of the method is the loss of resolution of fine structure patterns in subcellular localizations, such as the kinetochore proteins that are visualized as a diffusive cloud within the nuclear volume due to the averaging of multiple cells. One could think of using different computational techniques to better preserve the structured patterns during registration and averaging, however the large variation on the patterns of all mitotic subcellular structures make this quite challenging, and it would need to be tested extensively before implementation. In addition, looking at more than three to five differently colored protein maps at the same time becomes challenging quickly for finding interesting correlations between them. For exploring larger amounts of data at the system level, quantitative parameters that characterize protein density map similarities are still missing.

Using NTF to analyze protein distributions after parametrization in the quantitative feature space of interest point clusters is a good way to identify clusters of proteins with similar localization patterns within the data. The method was unsupervised, and thus it can process proteins that localize in any combination of subcellular structures, and thereby identify even completely novel distribution patterns. Furthermore, the method preserved the quantitative nature of the image features and could assign fractions of proteins into each of the automatically identified localization clusters instead of only performing binary annotations. The way in which the entire data set was decomposed at once, using time as an explicit dimension instead of frame by frame, allowed the definition of clusters over time, and confirmed the nature of subcellular structures. The disadvantage here was the unclear biological meaning of some of the clusters that could represent mixtures of multiple subcellular structures. Moreover, the requirement of high quality data with a high signal to noise ratio hardly allows analysis of single cell data. Thus, the interpretation of the NTF results could be challenging.

The supervised method based on reference proteins made it possible to assign even single images into different pre-defined subcellular localizations, and allowed analysis to be performed at the single cell level with high temporal resolution. The linear regression method I used was able to quantitatively assign protein fractions into a range of mitotic structures and accurately distinguish mixed patterns. The advantage of the supervision was the clear biological interpretability behind each of the localization classes. However, the disadvantage was the need for existing knowledge, and the inability of finding novel localization patterns. This method is particularly suitable for biologists with a clear target in mind, that want to look into the details of the kinetics of a particular process of interest at known subcellular structures, as shown for example in the case study on kinetochore disassembly.

In summary, all three methods used for data analysis were powerful and complemented each other in their ability to mine the integrated data. Each of them gave researchers new means to explore 4D localization data of many proteins. Combining all methods would be a good way to have an extensive and complete view of the data, and understand mitotic processes both at system and detailed kinetics level.

4.3 Spatio-temporal modeling as a general concept for studying other cellular dynamic processes

The concept of generating a spatio-temporal model for standardizing the environment of cellular processes for integration and analysis of experimental data is generic. The method we used to generate the model was based solely on registration and averaging and almost free of assumptions, and thus could be easily adapted to other dynamic cellular processes as well.

For the temporal modeling, the requirement was a reproducible and characteristic time course of features. In the case of mitosis, the chromosomal geometry was sufficient for the feature extraction. However, not all cellular processes are coupled to such dynamic morphological changes as undergone by chromatin. But other morphological or molecular clocks could be found for these processes. For example in cell migration, the morphology of the cell surface, or the focal adhesion kinase/paxillin ratio could be a good starting point for following the process of focal adhesions. For the entire cell cycle rather than just M-phase, different cyclin levels or DNA replication

markers could provide a good cell internal clock. The method could also be used for processes at particular structures of the cell, such as the centrosome. Initial results generated by Dr. Robert Mahen in the Mitosys group show that a few proteins could be sufficient as good temporal markers for the centrosome maturation process (data not shown). At the multicellular level, many researchers interested in Drosophila embryonic development were using expression patterns of transcriptional regulators or the number of nuclei to stage development. Such temporally highly controlled shape changes or expression patterns are found in many multicellular processes and could be used as the temporal landmarks.

For the geometric modeling, the symmetric nature of mitotic cells was crucial for the success of the biologically meaningful registration of protein localization. Other processes with clear symmetry or polarization could be analyzed in the same way, since the registration reference would be well correlated to the molecular process of interest. In directed cell migration, the front of the cell could be used as a reference, while in T cell maturation, the interaction interface to the B cell could be the best registration reference. For multicellular organisms, the polarization of the process would often be more obvious, such as the dorsal-ventral axis of the Drosophila embryo, or the apical-basal orientation of epithelium.

However, the method has limitations for non-polarized asymmetric processes with little molecular temporal control, such as un-directed cell migration or growth, since the correlation between the spatio-temporal registration landmarks and the actual molecular process of interest would be unclear. For these kind of processes a simple morphological registration marker, such as the nucleus, and a large amount of data could be a good starting point for exploration. The identification of novel characteristic distribution patterns of particular molecules during the analysis could then be used to refine the spatio-temporal registration.

4.4 Future perspectives

In the reported work, the computational pipeline was validated by a set of proof-of-concept data with 13 known proteins (incl. H2B in the landmarks channel). As mentioned before, for further validation the pipeline could process image sequences of pairs of two proteins co-expressed within one cell line and directly compare the

computational integration with the biological co-expression data.

The pipeline can and is planned to be used for the integration of a large amount of protein live cell imaging data in dividing human cells. To understand mitosis at the system level requires the analysis of the hundreds of mitotic proteins, for example those identified in Neumann et al. (2010). The absolute quantitative nature of our calibrated confocal live cell imaging pipeline allows us to assess the absolute abundance of a protein in each of the subcellular structures. Thus, generating cell lines with homozygous fluorescently tagged endogenous mitotic proteins is essential for getting the physiological picture of mitosis, and this effort is under way in the Mitosys team at EMBL.

For the computational pipeline, further development could be done to improve its usability. A user interface for uploading and processing data would make the software more user friendly. Distribution of the software would allow different labs to share the experimental burden of cell line and imaging data generation. Another computational challenge might arise with an increase in data volume that populates the standard model. With NTF or supervised annotation, trajectories through different protein clusters or subcellular localizations were output for each of the proteins. With just 13 proteins, all trajectories could be compared by eye. However, once the number of trajectories increases dramatically, it would be difficult to identify correlations between proteins in the data. Methods for mining the trajectories need to be developed to identify both similar overall paths as well as path sections.

One could think of further usages for the results generated by this thesis. The temporal model of the standard mitotic human cell progression could be used for guiding the intelligent acquisition of data. As different processes take place at different times and with different speed during mitosis, such as the kinetochore disassembly starting around the metaphase-anaphase transition, it would be more efficient to acquire data in very high resolution only from late metaphase to telophase. Based on the probability distribution generated by the temporal model for the duration of each mitotic stage, software could be developed to control microscope image acquisition by on-the-fly processing of the last acquired images. By identification of the exact mitotic stage of the cell, the timing for acquiring the next frame could be adjusted to adequately sample this particular kinetic phase of mitosis.

As the dynamics of different components of the kinetochore were integrated in our

model, the quantitative data of kinetochore disassembly could be used for building a small model of this process. Using the huge amount of existing knowledge about kinetochores, assumptions could be made for building a kinetic model to quantitatively explain the dynamic observations made in this work. Predictions of the kinetic model can be further tested by analyzing additional proteins or repeating the current analysis in conditions perturbed by silencing different components.

To understand the mitotic proteome in an integrated quantitative way, a database needs to be developed as an online platform for data visualization and exploration. This will guide and motivate researchers that analyze mitotic proteins both at a detailed level as well as on a system scale. This important work to disseminate the proof of concept data presented here and further data that will be generated in the future are underway in the Mitosys team at EMBL.

Chapter 5

Material and methods

5.1 Materials

5.1.1 Cell culture

Table 5.1: Cell culture instruments

Name	Size	Provider
Water baths	-	GFL
Cell culture incubator	-	Heraeus Hera Cell
Laminar flow hood	-	Safe 2020 ThermoScientific
Pipettes	P2, 10, 20, 200, 1000	Gilson Pipetman
Pipetus Akku	-	Hirschmann
Neubauer chamber	-	Celeromics
Nunc^{TM} dishes	6/10/15 cm Petri, 6-well plate	ThermoScientific
Lab-Tek^{TM} chambered coverslips	II, VIII	ThermoScientific
Centrifuge		Heraeus
Plastic tube	15 ml	Greiner bio-one
Falcon tube	50 ml	Corning Science
Pipette	2, 10, 25 ml	Corning Science
Pipette tips	P10, 20, 200, 1000	Molecular BioProducts
Cryo tube	1.8 ml	Thermo Scientific
Gloves		Kimtech

Table 5.2: Cell culture chemicals

Name	Unit	Provider	Purpose
High glucose Dulbecco's modified Eagle's medium (DMEM)	ml	Life Technology	cell growth culture
Fetal bovine serum (FBS)	ml	Life Technology	cell culture
Penicillin	10000 U/ml	Sigma	cell culture
Streptomycin	10 mg/ml	Sigma	cell culture
Glutamine	200 mM	Life Technology	cell culture
Sodium pyruvate	100 mM	Life Technology	cell culture
Geneticin^{TM}	50 mg/ml	Life Technology	cell selection culture
Hygromycin	100 mg/ml	Invitrogen	cell selection culture
Puromycin	g	Invivogen	cell selection culture
CO_2 independent DMEM	customized	Life Technology	cell imaging culture
Dextran	500000 MW	Life Technology	cell imaging culture
Dy481XL-NHS-ester	g	Dyomics	cell imaging culture
DMSO	ml	Sigma	cell storage
Phosphate buffered saline (PBS)	1x	EMBL Media kitchen	cell seeding
Trypsin-EDTA	1x	Gibco	cell seeding
Paraformaldehyde (PFA)	16%	EMS	cell fixation
Triton X-100	ml	Sigma	cell permeabilization
Thymidine	ml	Sigma	cell synchronization
BSA	g	Sigma	immuno staining

5.1.2 Cell modification

Table 5.3: HeLa Kyoto stable cell lines

Construct	Method	Background	Provider	Details
H2B-mCherry	cDNA	HeLa K.	Neumann et al. (2010)	single clone
pmEGFP	cDNA	HeLa K.	Mahen, EMBL HD	pool
mKIF11-GFP	BAC	H2B-mCherry	Maliga et al. (2013)	pool, no. 2354
mMIS12-LAP	BAC	H2B-mCherry	Hutchins et al. (2010)	pool, no. 2341
mTUBB2C-LAP	BAC	H2B-mCherry	Hutchins et al. (2010)	pool, no. 2637
LAB-mRACGAP1	BAC	H2B-mCherry	Hutchins et al. (2010)	pool, no. 2362
mCDCA8-LAP	BAC	H2B-mCherry	Poser, MPI Dresden	pool, no. 2607
mNEDD1-LAP	BAC	H2B-mCherry	Poser, MPI Dresden	pool, no. 311
mAURKB-LAP	BAC	H2B-mCherry	Poser, MPI Dresden	pool, no. 73
AURKB-mEGFP	ZFN	HeLa K.	Mahen et al. (2014)	clone no. H24
H2B-mCherry	cDNA	AURKB H24	Mahen et al. (2014)	pool
mNUP214-LAP	BAC	H2B-mCherry	Poser, MPI Dresden	pool, no. 2518
PLK1-mEGFP	ZFN	HeLa K.	Koch, EMBL HD	clone no. 24
H2B-mCheery	cDNA	PLK1 24	Mergenthaler, EMBL	pool
H2B-mCherry	cDNA	CENPA-EGFP	Isokane, EMBL HD	pool
BUB1-mEGFP	CrispR	HeLa K.	Koch, EMBL HD	clone no. 63
H2B-mCherry	cDNA	BUB1(ZFN)	Koch, EMBL HD	pool
NES-mEGFP	cDNA	H2B-mCherry	Koch, EMBL HD	pool
H2B-mCer3	cDNA	HeLa K.	Nijmeijer, EMBL	clone no. B7
NES-mCer3	cDNA	H2B-mCer3	Nijmeijer, EMBL	clone no. 29
Myrpalm-mCer3	cDNA	HeLa K.	Nijmeijer, EMBL	clone no. E11
H2B-mEGFP	cDNA	HeLa K.	Neumann et al. (2010)	single clone

Notes: The HeLa Kyoto cells were originally generated by Dr. S. Narumiya in the Department of Pharmacology, Kyoto University. The HeLa Kyoto cell line expressing EGFP-CENPA was provided by Toru Hirota (Cancer Institute, Tokyo, Japan).

Table 5.4: Recognition sequences and guide RNAs

Gene	Technique	Sequences
PLK1	ZFN	5' TCGGCCAGCAACCGTCTC 3'; 5' TCCTAATAGCTGCCC 3'
AURKB	ZFN	5' CGCCTGATGGTCCCT 3'; 5' CACTCGGGTGCGTGTGTT 3'
BUB1	ZFN	3' ACCAGACGGACACTTACTGAAGG 5'; 5'GGGCGCCTGGGGTTCGGGCCCGG 3'

5.1.3 Microscopy

Table 5.5: Instruments and materials for microscopy

Name	Provider
Zeiss LSM 780	Carl Zeiss, Jena
ScanR epifluorescence microscope	Olympus
Incubation chamber	EMBL Workshop
Water pump	Bartel
Control of water pump	EMBL Workshop
Water objective cup	EMBL Workshop
Silicon gel	Bayer
Alexa 488	Life Technology
Alexa 561	Life Technology
Rhodamin-Phaloidin	Invitrogen
Menzel-Gläser	Thermo Scientific

5.1.4 Softwares

Table 5.6: Computer softwares

Name	Provider	Purpose
ZEN 2011-2013	Carl Zeiss, Jena	Image acquisition
Imaging macro 2012	A. Politi	Image acquisition
Micronaut 2013	R. Hoefer, C. Sommer	Online image processing
Fiji	NIH, A. Politi	On/offline image processing
CellCognition	cellcognition.org	Image processing
Fluctuation Analyzer	M. Wachsmuth	FCS data processing
MATLAB®	The MathWorks	Pipeline development
MS Office	Microsoft	Data presentation
Illustrator CS2	Adobe	Graphics
Amira	FEI software	3D visualization
Adobe Reader	Adobe	Text processing
Texmaker	Xm1 Math	Text editing
jabRef	jabref.sourceforge.net	Reference management

5.2 Methods

5.2.1 Cell culture

Cells were cultured in high glucose Dulbecco's modified Eagle's medium (DMEM; Life Technology) with 10% (v/v) FBS, 1% (v/v) penicillin-streptomycin, 1% (v/v) Glutamine and 1% (v/v) Sodium pyruvate at 37 °C and 5% CO_2 in petri dishes. Depending on the genetic modification, one or more of the following antibiotics were supplied to the culture at the stated final concentration: Geneticin 500 µg/ml, Hygromycin 200 µg/ml or Puromycin 0.5 µg/ml. Once the cells reached 80-90% confluence, they were split and only a fraction of the cells were cultured in a fresh dish.

For splitting, cells were washed twice with PBS buffer before incubation with trypsin at 37 °C for 2 minutes. Afterwards, cells were detached from the petri dish mechanically and suspended in DMEM medium. A fraction of the suspension was transferred into a fresh dish.

5.2.2 Cell storage

HeLa cells were grown without antibiotics for mammalian cells for 3 days. After treatment with trypsin as described in cell splitting, the detached cells were suspended in 90% FBS, 10% DMSO solution and frozen in plastic boxes at -80 °C. Cells with 90% confluence on 15 cm dishes were distributed into 4 frozen aliquots, or from 10 cm dishes into 2 aliquots. Cells to be stored for a longer duration were deposited in liquid nitrogen.

Frozen cells could be re-cultured. The frozen aliquot was thawed at 37 °C in a water bath and then transferred into a 10 cm petri dish with 10 ml pre-warmed medium. After incubation at 37 °C with 5% CO_2 for 4 to 10 hours, the culture medium was refreshed.

5.2.3 Stable cell line production

All cell lines generated using the BAC system were taken from the publicly available resources. Genome edited cell lines were generated by B. Koch in the Ellenberg lab.

Stable cell lines expressing cDNA for POIs were produced by colleagues in the Ellenberg lab. HeLa cells were transfected with DNA encoding the gene of interest

and treated with selection medium with antibiotics. In order to obtain cells that strongly expressed the introduced gene, single cells or a cell population were harvested for imaging by fluorescence activated cell sorting (FACS, performed by the EMBL Heidelberg facility).

Both Zinc finger nuclease (ZFN) and Clustered regularly interspaced short palindromic repeats (CrispR) based technologies were used. In brief: Zinc finger nucleases were purchased from Sigma-Aldrich with different DNA-binding sequences (Table 5.4). Based on ENSEMBL release 75, donor plasmids were designed to contain the mEGFP cDNA sequence flanked by a left and a right homology arm consisting of the POI's genomic sequence. HeLa Kyoto cells were transfected with both ZFN and donor plasmid. With the CrispR technique, two gRNAs (Table 5.4) were transfected into the same cell in order to produce a double strand break close to the start codon of the gene of interest. During repair, the mEGFP coding sequence from the donor plasmid was transferred to the target locus via homologous recombination. Single clones expressing the POI were selected using the in-house developed validation pipeline (Mahen et al., 2014).

5.2.4 Wide field live cell microscopy

Wide field imaging was performed on a Olympus Scan R microscope using the 10x, NA 0.4 air objective controlled via the software developed by Olympus. An incubation chamber was keeping a constant temperature of 37 °C. Cells cultured in CO_2 independent medium were mounted one hour before the start of imaging. One position was selected as reference on the stage, and the software automatically generated a list of imaging positions in x-y based on the layout defined prior imaging. The z-positions were defined by the autofocus mode during the first round of imaging. A series of images were acquired at different depth within the specimen for each of the positions, and the z-position producing the image with the highest total intensity was used for the entire acquisition.

5.2.5 Confocal fluorescence microscopy

Confocal microscopy was performed on a Zeiss LSM780 laser scanning microscopes using, as indicated for each experimental pipeline, the 10x air NA 0.45, 63x, NA 1.4 oil or 40x, NA 1.2 water DIC Plan-Apochromat objectives and the GaASP detectors

(Zeiss). Cells were cultured in CO_2 independent medium. Within the incubation chamber temperature was kept at 37 °C. Time-lapse imaging was performed using the ZEN 2012-2013 image acquisition software as well as in-house software applications (developd by A. Politi). Imaging sessions taking longer than half an hour and using the water objective were done with an in-house developed cup and a water pump, that was controlled by software developed by M. Wachsmuth, such that water drops were regularly supplied to the objective-sample interphase.

A standard live cell imaging pipeline was supported by a macro (A. Politi). Before starting the imaging, a number of imaging positions were selected manually and saved in the software. Setups such as laser configuration, imaging field size and resolution, detection filters and pinhole at 1 AU were defined and saved. During live cell imaging, the microscope finds the focus automatically by performing line-scan imaging of the reflection signal around 633 nm. The vertical position of the microscopy glass coverslip was determined as the position with the maximum reflection intensity, and used as reference for acquiring a volume of the specimen at a particular deepness.

5.2.6 Automation of mitotic cell imaging

In collaboration with the Gerlich lab (IMBA Vienna) a software called Micronaut that worked within the CellCognition framework was developed for automatically finding mitotic cells in a sample. A training set was acquired from a HeLa Kyoto H2B-mCherry cell line by live cell confocal microscopy using an excitation laser at 561 nm every 5 min for about 16 hours. The resolution in x-y was 0.32 µm and 2.5 µm in z for 3 planes. Images were projected in z by taking the maximum intensity value. These images were used for building a SVM classifier for distinguishing between cells in interphase, prophase, mitosis (prometaphase till telophase) and artefacts (apoptosis, out of imaging field, out of focus, too low expression).

On the microscope, Micronaut uses the imaging and processing setups defined during the training of the SVM classifier and classified images on the fly. The classification score for the prophase, often interpreted as the probability of a cell being in the class of interest, was output, and a pre-defined threshold was used to make a decision on whether imaging setups for mitotic cell acquisition should be activated. According to how different the sample's H2B-mCherry expression level were to the training set, the threshold was set between 0.85 and 0.96. Once a

prophase cell was found, it was then imaged using a different imaging setup. For our purpose, mitotic cells were imaged live every 90 seconds for 31 z-planes with a spatial resolution of 0.25 µm in x-y and 0.75 µm in z with 488 nm laser.

5.2.7 Fluorophore concentration determination using FCS

Every day before FCS measurements, the pinhole size was calibrated using a 20 nM solution of Alexa488, and single EGFP brightness was calibrated by performing FCS measurements on the HeLa Kyoto EGFP cell line. For measurements of other fluorescently labeled molecules, a solution of the corresponding fluorescence molecule was used for the brightness calibration.

All FCS measurements were processed using Fluctuation Analyzer. Measurements of dye solutions were fitted using a two-component anomalous diffusion with triplet-like blinding, and measurements of fluorophore-fused proteins were fitted using a two rounds two-component anomalous diffusion with fluorescent protein-like blinking. For detailed parameter settings see 6.2.

The effective confocal volume was calculated using the form:

$$V_{eff} = (4 \cdot \pi \cdot D \cdot \tau \cdot 10^{-6})^{\frac{3}{2}} \cdot \kappa)$$

where D was the diffusion coefficient of the Alexa488 which is 441 µm^2/s at 37 °C and κ is 5.5 µm^3 for the ZEISS 780 confocal system. The τ was one of the fitting outputs for Alexa488. The number of fluorescent molecules within a confocal volume was calculated by multiplying the fitted number of the molecule (N) with the fitted correction factor.

As proteins might exist in complex with multiple molecules, a count per molecule (CPM) value was used to correct the number of molecules. As reference, I used the CPM value of EGFP as measured in the HeLa EGFP cell line. During analysis, I realized that some of the EGFP measurements had unusual high CPM values, indicative of multimeric EGPFs. Thus, I took the median value of all EGFP measurements with a CPM value below 4.25 (highest limit for monomeric EGPF from experience) as the reference CPM for each experiment. Dividing the CPM value of the fusion protein measurements by the reference CPM, I got the multimeric factor. If the multimeric factor was larger than 1, the fitted number of molecule value was corrected by multiplication with the multimeric factor. Finally, the local

concentration of the measured protein was determined as the corrected number of molecules divided by the effective confocal volume.

5.2.8 Imaging HeLa cells on micropatterns

Fibronectin mixed with Alexa647 was printed in different forms on a coverslip by A. Christ (Théry lab). The coverslips were attached to a 6-well plate by one water droplet. All coverslips were washed twice with PBS and supplied with 0.5 ml culturing medium on the top. Stable HeLa Kyoto H2B-mCer3 cells were detached from a 10 cm petri dish by treatment with 0.75 ml trypsin for 5 minutes. The processes ended by adding 6 ml culturing medium. The suspension was centrifuged at 800 rpm for 3 minutes and the supernatant discarded. The cells were re-suspended in culturing medium to a final concentration of 100,000 cells/ml by using a double pipette tip (200 µm tip on the top of a 1,000 µm tip) to ensure that all cells detach from each other. 1 ml of the cell suspension was carefully applied to the coverslip in such a way that a droplet forms above it that covers the entire coverslip. The coverslips were then incubated at 37 °C with 5% CO_2 for 1.5 hours. Afterwards, coverslips were washed twice with pre-warmed culturing medium and incubated for another 6.5 hours.

After incubation, the coverslips were exposed to 3% PFA for 10 minutes at room temperature before being washed twice with PBS. They were then incubated with 0.5% triton x-100 for 5 minutes before being washed twice with 2% BSA/PBS solutions. Cells were stained with 1:40 rhodamin-phaloidin diluted with 2% BSA/PBS for 25 minutes. After washing with PBS, the coverslips were mounted on a glass which could be installed on the ZEISS 780 microscope stage. The coverslips were finally imaged by confocal microscopy using 10x magnification and lasers at 458 nm, 561 nm and 633 nm. The signals were detected in three channels: 473 - 544 nm (H2B), 570 - 633 nm (Actin), 633 - 695 nm (pattern) at the x-y resolution of 0.25 µm.

5.2.9 Label Dextran with fluorescence

The cross linking of Dextran to fluorescent dyes was performed by Dr. Julia Roberti. Just briefly: 80 mg Dextran with an average size of 500,000 Dalton and labeled with multiple amino residues was solved in 4 ml 0.2 M $NAHCO_3$. 6.4 mg Dy481XL with ester residues was solved in 320 µl DMSO (Click Chemistry) and mixed with the Dextran solution at 500 rpm and room temperature for 60 minutes. The labeled

Dextran molecules were separated from unreacted Dy481XL molecules by size exclusion chromatography using PD-10 Sephadex desalting columns (GE Healthcare). The elutate was concentrated to about 3.5 ml using a centrifugal filter concentrator at NMWL 10,000 (Milipore), and dialyzed for two days against water in an 8,000 Dalton membrane. The solution was then diluted, filtered, and concentrated again using centrifugal filter concentrators first at NMWL 30,000 followed by another at NMWL 10,000.

5.2.10 Determination of Dextran concentration

A 1:250 dilution of the labeled Dextran stock was measured using the FCS mode on a ZEISS Confocal 780 system. For calibrating the brightness of Dy481XL and the confocal volume, a 100 nM Dy481XL solution was used. The solutions were excited by 488 nm argon laser, and photons in the range of 505 nm to 540 nm were counted by the APD detector. The biophysical parameters such as the concentration, diffusion time and molecular brightness were fitted with a two-component diffusion model using the Fluctuation Analyzer. The concentration of Dextran was converted to that of the culture solution I used (13.5 µl Dextran stock in 360 µl culture medium) which was estimated at 3.29 µM, and based on the brightness per molecule of both labelled Dextran and Dy481XL only solutions, the efficiency of the labeling was estimated at 1.89/molecule.

5.2.11 Quantification of Dextran toxicity

Mitotic time and cell cycle time were measured in order to test the toxicity of adding Dextran to the cell culture.

Mitotic time of cells cultured in labeled Dextran

HeLa Kyoto H2B-eGFP cells were seeded with 9,000 cells per Lab Tek VIII well at 25 hours before imaging. One hour before starting the imaging, DMEM medium was replaced by CO_2 independent medium supplied with Dy481XL labeled Dextran solution at different concentration. After a one hour pre-incubation, wide-field live cell imaging at 10x magnification was performed at 25 independent positions per well every 5 minutes for 48 hours. A SVM classifier of 10 classes was trained based on

the manual cell stage annotation of 925 nuclei with an overall accuracy of 91.9%. All cells were annotated with this classifier and tracked based on nearest neighbour in CellCognition version 1.4.1. Mitotic events were selected by detection of transitions from interphase to prophase or prometaphase and traced for two-and-half hours. Early mitosis duration, defined as the time between the start of prophase and the detected segregation of chromosomes, was used to characterize the influence of a Dextran solution on mitosis.

Cell cycle of cells in Dextran culture

78.5 hours prior imaging HeLa Kyoto H2B-mCherry cells were seeded at 6,000 cells/well in DMEM. Cells were supplied with 20 µM thymidine at 54.5 hours and released in DMEM 30.5 hours before the imaging. 8 hours later, cells were cultured again in DMEM supplied with 20 µM thymidine for another 16 hours. 6.5 hours after cells were released for a second time in DMEM, cells were imaged in CO_2 independent medium at different Dextran concentrations. Images were taken every 8 minutes in 4 z-planes for 40 to 60 hours using confocal live microscopy with 63x magnification. The resolution was 0.44 µm in x-y and 3 µm in z. 533 nuclei were manually annotated into 10 cell stages (7 mitotic stages, apoptosis, polylobed and artefact) for building a SVM classifier. Using CellCognition, all cells were annotated and tracked for 26 hours after the first mitotic event defined by the transition from interphase to prophase or prometaphase. The duration between the start of prometaphase of two sequential mitoses was defined as the cell cycle time for comparison.

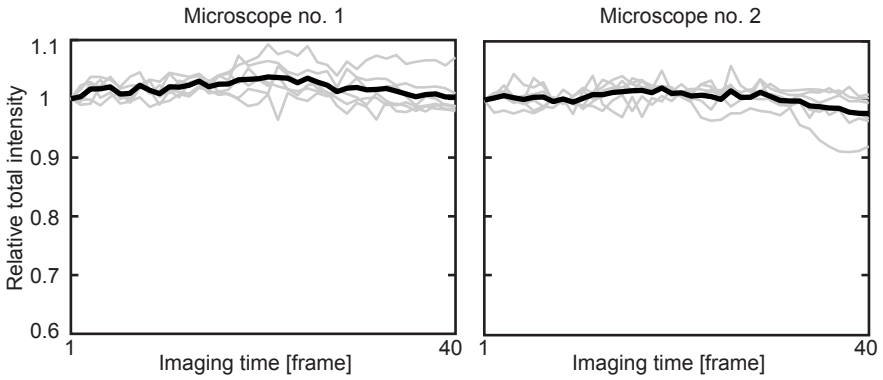
5.2.12 Automated calibrated 3D live mitotic cell imaging

HeLa Kyoto cells expressing both H2B-mCherry and EGFP fused POI cultured in medium with labeled Dextran were imaged live using the pipeline for automated mitotic cell imaging. According to the expression level of the EGFP (and thus the POI), the GaASP EGFP detector range was set such that no detection saturation happened during the imaging. However, laser intensity was set the same for all experiments and sufficiently weak to prevent EGFP bleaching (Figure 5.1). The strongly overexpressed H2B-mCherry and Dy481XL labeled Dextran could also be excited at 488 nm and detected by two band-pass sections of the GaASP detectors in

mCherry (587 - 621 nm, Figure 5.2a) and far-red range (621 - 692 nm, Figure 5.2b). High resolution mitotic cell imaging took one hour for 40 imaging rounds.

After the high resolution live imaging, an additional image at 0.25 µm in x-y was taken and the nucleus closest to the center of the imaging field (mostly from the just-divided cell) was segmented from the H2B-mCherry image channel on the fly using Fiji macro developed by A. Politi. Six FCS measurements (two in and four around the segmented nucleus with a distance of 1 µm) were performed using the same objective and an avalanche photodiode (APD) in the spectrally distinct region 505-540 nm (Figure 3.18). Each measurement was taken for 30 seconds. Afterwards, the microscope continued with the scan mode for searching further mitotic cells.

Figure 5.1: Bleaching of mEGFP over high-resolution live imaging.



Two confocal microscope systems were used for imaging acquisition in this thesis and were tested for bleaching. 6 H2B-mEGFP mitotic cells were imaged on each system using the same setup as for high-resolution live cell imaging. The total intensity in the GFP channel was measured for a constant volume including the cell of interest over time and normalized by the measurement of the first frame (grey lines). The mean through the 6 cells (black line) was calculated frame by frame. The bleaching of mEGFP was controlled at a negligible level. Images were acquired by myself and analysis was performed together with J. Hossain.

5.2.13 Manual quality control

Before the imaging data were processed, all images were manually controlled and images with unsatisfactory qualities were not processed further. Following criteria were used for quality control: cells that did not reach anaphase within the high resolution imaging time were sorted out. Cells without any obvious POI expression were deleted as well as cells with too high expression where the saturation in the EGFP channel was reached at some point during imaging. And finally, if the FCS

measurement was not performed in the cell of interest, the imaged cell was not further processed either.

5.2.14 Calibration of the GaASP image

All FCS measurements were analyzed using Fluctuation Analyzer for the determination of the fluorophore concentration at the measured position as described before. Measurements were discarded if: the R square was smaller than 0.65, or the Chi square was larger than 0.15, or the bleaching was more than 10% or the signal was lower than 2x of the background FCS signal that was determined for each of the microscopes in an separate experiment. Also measurements which showed an outlying (mean ± 2 standard deviation) count per molecule (CPM) value compared to all measurements performed on the same day on the same microscope were deleted. For the remaining measurements, the GaASP intensity at the corresponding position was calculated on the image taken just before the FCS measurements. In order to avoid the effect of noise, averaged filtering with a 9 by 9 sized window was performed on the image, and the background GaASP value calculated based on the cell-free area image subtracted from the averaged local intensity value. Measurement positions with a negative intensity were not considered for the calibration.

For each cell, the median CPM of all satisfactory measurements was used to determine how many molecules were present within a single complex. The corrected concentration was then divided by the GaASP intensity for each of the remaining positions after all quality controls. The mean among all calculated ratios within a cell was taken as the calibration factor for the transformation from the image to a density map.

5.2.15 Segmentation of landmarks

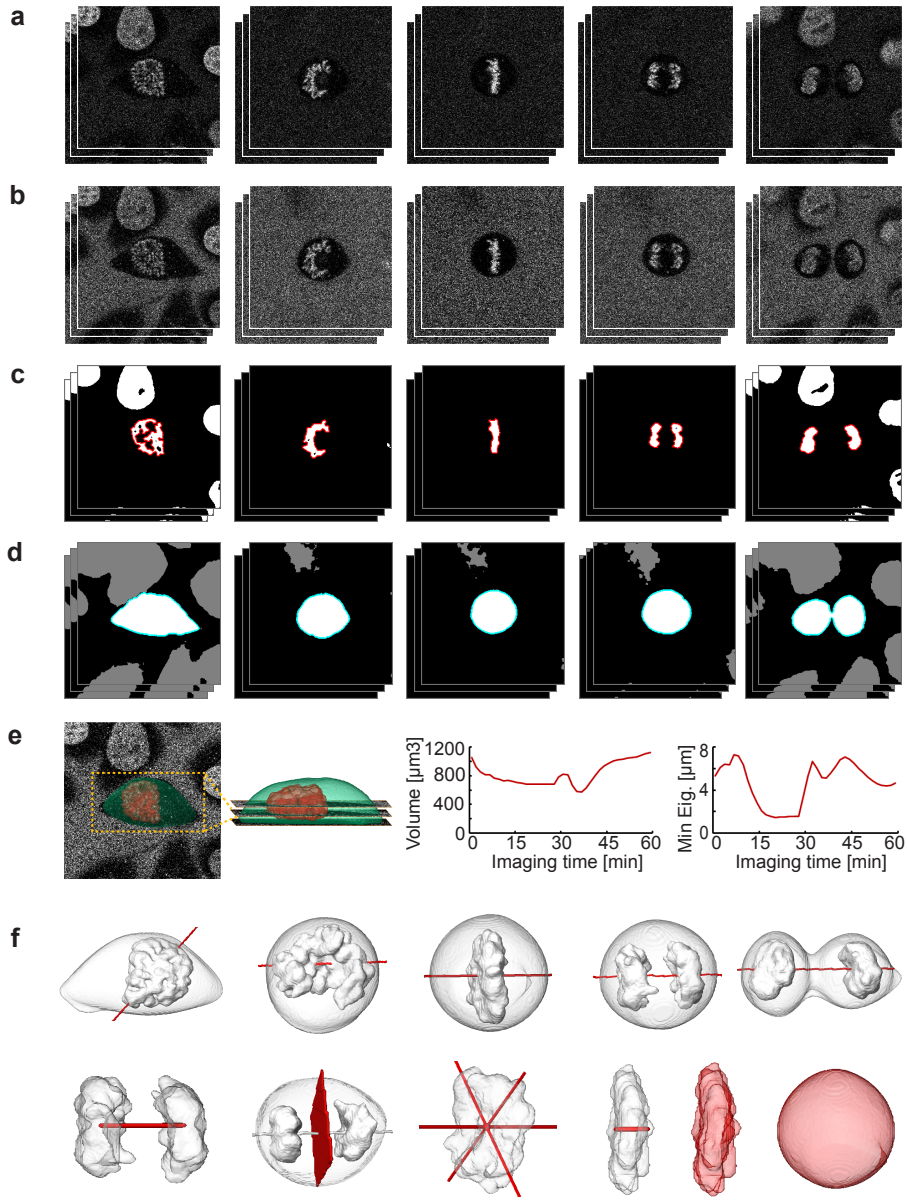
A fully automated computational pipeline was developed by Julius Hossain in MATLAB to segment chromosomes from the mCherry wavelength channel and the cell boundary from the far-red wavelength channel, and reconstruct the 3D cell and nuclear shape. Original stacks were interpolated along z axis to obtain isotropic resolution for better 3D image analysis. A 3D Gaussian filter was applied to reduce the effects of noise. To detect chromosome regions, mCherry channel (587 - 621 nm) was binarized first using a multi-level thresholding method as described in Hériché

et al. (2014) (Figure 5.2 a, c). First, a global threshold was estimated based on the entire 3D image Otsu (1975). The threshold was then adapted for each 2D slice by validation of the connectivity of binary components in 3D. The chromosome region of interest was detected in the first frame of the sequence by making use of the size and location information of the connected objects in the binary image. The detected chromosome was then tracked over the subsequent frames of the image sequence. To detect the cell boundary, a ratio image was obtained by first dividing the far-red wavelength channel by the mCherry channel. In this ratio image, the intensity of the chromosome signal was minimized as the mCherry labelling the chromatin gave emissions in both channel. The ratio image was then binarized using the multi-level thresholding in a similar way as for the mCherry channel. To identify the border between cells, a marker based watershed algorithm (Meyer, 1994) on the distance transformed ratio image was applied. The segmented chromosomes were used as markers (Figure 5.2 b, d).

5.2.16 Prediction of division axis

Julius Hossain developed computational pipelines to extract and predict parameters of the cellular geometry. The following text was formulated together with Julius Hossain. Chromosome volume in each frame was represented by three orthogonal eigenvectors and three associated eigenvalues, where the eigenvector with the largest eigenvalue represented the longest elongated axis of the chromosome volume and so on. Metaphase frames were automatically detected based on the low value of the smallest eigenvalue of the chromosomal volume, which forms a plate during metaphase. Division axis, the direction of spindle pole-to-pole, for metaphase cells was predicted first by simply taking the eigenvector having the minimum eigenvalue. This vector was always orthogonal to the metaphase plate. Using the predicted axis in the first and last metaphase frame, axis for the remaining frames were propagated backwards and forwards for stages before and after metaphase respectively. There the eigenvector with the smallest discrepancy to the axis predicted for the adjacent frame was used. For further analysis, the orthogonal plate to the division axis localized in the middle of the centers of both daughter nuclei was predicted as the midplane (Figure 5.2 e, f).

Figure 5.2: Overview of the segmentation and parameters prediction pipeline



(a) - (b) Raw image stacks. Single z plane image of H2B-mCherry and Dextran-DY481XL at 587 nm-621 nm (a) and 622 nm–695 nm (b) emission, respectively. (c) Detected chromosome markers where boundaries of the chromosomes of interest are marked with red. (d) Output of watershed transform on ratio image where boundary of the detected cell of interest is marked with green. (e) Left: reconstruction of cell and chromosome surfaces in 3D, right: chromosome volume and minimum Eigen value of chromosome mass over mitosis. (f) Some of the extracted parameters: upper row - the predicted cell axis, marked with red along with the landmarks in gray; lower row from left to right – distance of the two detected chromosome masses, predicted mid zone, three major axes of the chromosome mass, minimum Eigen value of chromosome mass, chromosome volume and cell volume. Figure and legend were provided by Julius Hossain.

Chapter 6

Appendix

6.1 Imaging macros

6.1.1 Configuration of the classifier for automated imaging

namingscheme: LSM_SplitCh version: 1.4.1

Object detection: channelid: 1; zslice_selection_slice: 3; zslice_projection: True; zslice_projection method: maximum; zslice_projection_begin: 1; zslice_projection_end: 3; zslice_projection_step: 1; intensity_normalize_max: 255; intensity_normalize_min: 0; segmentation_method: local adaptive threshold w/split_merge; medianradius: 3; window_size: 80; min_contrast: 4; local_adaptive_threshold2: False; holefilling: True; removeborderobjects: False; Split_and_merge: False; object_filter: True; intensity_max: inf; intensity_min: 10; size_max: 3800; size_min: 300.

Feature extraction: Statistical geometric features: False; Granulometry features: False; Basic shape features: True; Convex hull features: False; Haralick features: True; Basic intensity features: False; Distance map features: False; Moments: True.

Classification: Classifier: Mitosys1_40x_320nm_Automation and Mitosys2_40x_320nm_Automation; Class_1: prophase; Class_2: interphase; Class_3: mitotic and apoptotic; Class_4: polyplois and border nuclei.

Micronaut settings: class_of_interest: 1; threshold_probability: 0.85 - 0.96, adjusted for different cell lines.

6.1.2 Macro of automatic imaging pipeline

Version: AutofocusMacro for ZEN 2012 v3.0.24.

Global repetition settings: MultipleLocationToggle: True; Global repetition: 5 min; Global repetition interval: True; Global repetition number: 300 (stopped earlier).

Water pump settings: Interval: 25 min; Distance: 9 mm; Pumping_duration: 2 sec; Pumping_waiting: 2 sec.

Autofocus settings: Period: 1; TrackZ: True; TrackXY: False; Method: Center of Mass (thr); ScanMode: ZScan; SamplesPerLine: 512; LineStepNumber: 1; FramesPerStack: 25; FrameSpacing round1: 1 µm; FrameSpacing round2: 0.2 µm; Zoom: 1.3; SamplingNumber: 1; pixelDwell: 1.27 µsec; BitDepth: 8; ScanDirection: 0; CorrX: 0; CorrY: 0; TimeSeries: True; StacksPerRecord: 1.

Monitoring of H2B-mCherry: Channel_Track: H2BScan; ZOffset: 2.5 µm; Period: 1; TrackZ: False; TrackXY: False; Focus_Method: None; Analysis: True; Analysis_Sequential: True; TimeOut: False; ScanMode: Stack; SamplesPerLine: 512; LinesPerFrame: 512; LineStepNumber: 1; FramesPerStack: 3; FrameSpacing: 2.5 µm; Zoom: 1.3; SamplingNumber: 1; pixelDwell: 1.27 µsec; BitDepth: 8; ScanDirection: 0; CorrX: 0; CorrY: 0; TimeSeries: True; StacksPerRecord: 1.

High-resolution live 4D imaging: Channel: GFP_Cherry; ZOffset: 11 µm; Period: 1; TrackZ: False; TrackXY: False; Focus_Method: None; Analysis: Flase; Analysis_Sequential: none; TimeOut: False; ScanMode: Stack; SamplesPerLine: 256; LinesPerFrame: 256; LineStepNumber: 1; FramesPerStack: 31; FrameSpacing: 0.75 µm; Zoom: 3.3; SamplingNumber: 4; pixelDwell: 1.27 µsec; BitDepth: 16; ScanDirection: 0; CorrX: 0; CorrY: 0; TimeSeries: True; StacksPerRecord: 1; Repetition: 90 sec; Repetition_interval: True; Repetition_number: 40.

FCS calibration imaging: Channel: GFP_Cherry; ZOffset: 2.5 µm; Period: 1; TrackZ: False; TrackXY: False; Focus_Method: None; Analysis: True; Analysis_Sequential: True; TimeOut: False; ScanMode: Stack; SamplesPerLine: 256; LinesPerFrame: 256; LineStepNumber: 1; FramesPerStack: 1; FrameSpacing: none; Zoom: 3.3; SamplingNumber: 4; pixelDwell: 1.27 µsec; BitDepth: 16; ScanDirection: 0; CorrX: 0; CorrY: 0; TimeSeries: True; StacksPerRecord: 1.

FCS calibration position selection: Threshod method: Li; Smoothing filter radius: 2 pix; Exclude particles $\rangle$ 6000 µm^2; Watershed if $\rangle$ 1000 µm^2; Measurements in nucleus: 2; Measurements in cytoplasm: 4.

FCS calibration measurement: ZOffset: 0; Configuration: FCSAuto_eGFP; MeasurementTime: 30.

6.2 Parameters of the FCS fitting

Modify and correlate: Base freq. [Hz]: 10^6 (dye)/ 10^5 (POI); Eval freq. [Hz]: 2000; Detrend freq. [Hz]: 2; Correlate mode: Scale; Evaluation steps: 9; Evaluation depth: 4; Table offset: 0.

Intensity correlations: Offset GFP: 3.3 (System 1)/3.5 (System 2); Ampl. KHz: 0; Rate: 0.01.

Table 6.1: Fit correlations

Parameters	N	θ_T	τ_T	f_1	τ_{D1}	α_1	τ_{D2}	α_2	κ	offset
Starting value dye	10	0.2	4	1	500	1	5000	1	5.5	0
First round	free	fix	fix	fix	free	fix	fix	fix	fix	fix
Second round	free	link	link	free	free	free	free	free	free	free
Starting value POI	10	0.2	100	0.5	500	1	5000	1	5.5	0
First round	free	fix	fix	free	free	fix	free	fix	fix	fix
Second round	free	link	free	link	free	link	free	link	free	free
Upper limit	10^4	1	10^3	1	$5{\cdot}10^4$	1.2	$5{\cdot}10^6$	2	20	10
Lower limit	10^{-4}	0	0	0	10/100	0.5	500	0.5	1	-10

6.3 Code of the computational pipeline

The entire pipeline was tested with MATLAB 2013b. Toolbox dependency: Computer Vision System Toolbox, Curve Fitting Toolbox, Image Processing Toolbox and Statistics Toolbox.

6.3.1 Summarizing the data using a database file

Cells were first segmented by Julius Hossain. The segmentation outputs were examed by Julius and me manually. Errors in segmentation were either solved by re-processing with adjusted parameters or forced the deletion of the cell. Cells without complete divisions were also deleted For each segmentable cell, a result folder was generated and the chromatin and cellular volume segmentation results as binary volumes with basic parameters describing the chromosomal morphology were saved in the ‘‘.mat’’ format. First, a database file for all segmentable cells with the information of the result folder path, general information about the cell such as the imaged protein, the localization etc. was generated. The database file not only made the inputting of data during processing easier, but also kept tracking of processing and QC status through the entire analysis.

```
function exp_dir = extract_full_database(exp_dir)

% Interactive selection of experiment directory
if ¬nargin
    exp_dir = uigetdir('','Select the directory with your data ...
       series');
end
```

```
% Load the poi label and localization database
fID = fopen('POI_annotation_database.txt'); % Reference proteins ...
    can be assigned if needed
annotation = textscan(fID,'%s %s %s');

% Get all experiments
list_of_date = dir(fullfile(exp_dir,'*_*'));
num_of_data = length(list_of_date);

% Define the information for summary
filepath = {};
system = [];
poi = {};
label = {};
localization = {};
vec_idx = [];

% Go through the list of all experiments and summarizing
for date_idx = 1:num_of_data;
% Check whether the experiment is a usual high content experiment which
% has the naming pattern date_poi
splitname = ...
    strsplit(list_of_date(date_idx).name,'_','CollapseDelimiters',true);
if length(splitname) == 2;
if ¬isnan(str2double(splitname(1)));
% Find all microscope systems ran the same experiment
list_of_system = ...
    dir(fullfile(exp_dir,list_of_date(date_idx).name,'Mito*'));
num_of_system = length(list_of_system);
for sys_idx = 1:num_of_system;
if exist(fullfile(exp_dir,list_of_date(date_idx).name, ...
    list_of_system(sys_idx).name,'Result'), 'dir');
% Collect all segmentable cells on that day with that particular ...
    microscope system
list_of_cell = dir(fullfile(exp_dir,list_of_date(date_idx).name, ...
    list_of_system(sys_idx).name, 'Result', 'cell*'));
for cell_idx = 1:length(list_of_cell);
% Summarize all important information
```

```
filepath = cat(1,filepath, ...
    fullfile(exp_dir,list_of_date(date_idx).name, ...
    list_of_system(sys_idx).name, ...
    'Result',list_of_cell(cell_idx).name));
splitsys = strsplit(list_of_system(sys_idx).name, ...
    'ys','CollapseDelimiters',true);
system = cat(1,system,str2double(splitsys(2)));
poi = cat(1,poi,splitname(2));
annotation_idx = find(strcmp(annotation{1,1},splitname{2})>0);
label = cat(1,label,annotation{1,2}{annotation_idx,1});
localization = cat(1,localization,annotation{1,3}{annotation_idx,1});
% During the segmentation, the positions of the FCS measurements ...
    were checked and if more than 4 out of 6 measurements were ...
    performed outside of the cell, the cell failed the QC
segqc = importdata( fullfile(exp_dir,list_of_date(date_idx).name, ...
    list_of_system(sys_idx).name,'Result', ...
    list_of_cell(cell_idx).name,'Preprocessing', ...
    'Segmentation','dirProcStat.txt'));
if ismember(segqc,[1 2]);
vec_idx = cat(1,vec_idx,1);
else
vec_idx = cat(1,vec_idx,0);
end
end
end
end
end
end
end

% Write the database file in .txt format
num_of_cell = length(filepath);
summary_database = dataset(filepath,system,poi,label, localization, ...
    {vec_idx,'segmentation'}, {-ones(num_of_cell,7), ...
    'fcs_calibration', 'lm_features', 'time_alignment', ...
    'registration', 'feature_extract', 'classification', ...
    'post_processing'});
output_txt = fullfile(exp_dir,'full_database.txt');
export(summary_database,'file',output_txt);
```

```
end
```

Afterwards, cells with irregular (too high or too low) protein expression were manually identified and deleted for further processing by updating the database file manually.

6.3.2 Calculation of the calibration factor

The next step is to calculate the factor for the calibration of imaging detector intensity to the absolute molecule concentration based on the FCS measurements. All FCS measurements were processed using Fluctuation Analyzer beforehand using the parameters in 6.2.

```
function vec_QC = get_calibration(exp_dir,calibrated)

%% Interactive selection of experiment directory
if isempty(exp_dir)
    exp_dir = uigetdir('','Select the directory with your data ...
       series');
end

%% Link the APD to GaASP and calculate the calibration
if calibrated == 0; % Perform calibration calculation
% Find all date_poi folders
list_of_date = dir(fullfile(exp_dir,'*_*'));
num_of_data = length(list_of_date);
for date_idx = 1:num_of_data;
splitname = strsplit(list_of_date(date_idx).name,'_', ...
   'CollapseDelimiters',true);
if length(splitname) == 2;
if ¬isnan(str2double(splitname(1)));
% Find all microscope systems in the data_poi folder
list_of_system = ...
   dir(fullfile(exp_dir,list_of_date(date_idx).name,'Mito*'));
num_of_system = length(list_of_system);
for sys_idx = 1:num_of_system;
% If unprocessed, calculate the calibration factor for all cells ...
   using the function fcs_calibration
if exist(fullfile(exp_dir,list_of_date(date_idx).name, ...
   list_of_system(sys_idx).name, 'Result','fcs_fittings.res'),'file');
if ¬exist(fullfile(exp_dir,list_of_date(date_idx).name, ...
   list_of_system(sys_idx).name, ...
```

```
    'Result','FCS_calibration.mat'),'file');
disp(['Process fcs calibration for' ...
    fullfile(exp_dir,list_of_date(date_idx).name, ...
    list_of_system(sys_idx).name)])
fcs_calibration(fullfile(exp_dir,list_of_date(date_idx).name, ...
    list_of_system(sys_idx).name));
end
end
end
end
end
end
end

%% Summarize the calibration
% Load the database file
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
summary_database.fcs_calibration = -ones(size( ...
    summary_database.fcs_calibration));
cell_dirlist = summary_database.filepath;
cell_total = length(cell_dirlist);
vec_QC = zeros(cell_total,1);
% For each cell, save the calibration file
for cdx = 1:cell_total;
% load the corresponding FCS_calibration file
fcs_file = [cell_dirlist{cdx}(1:end-14) 'FCS_calibration.mat'];
if exist(fcs_file,'file');
savefile = ...
    fullfile(cell_dirlist{cdx},'Preprocessing','calibration.mat');
if ¬exist(savefile,'file');
disp(['Process' fullfile(cell_dirlist{cdx},'Calibration')])
load(fcs_file);
cellidx = str2double(cell_dirlist{cdx}(end-9:end-6)); % identify ...
    the index of the cell in that particular experiment
ratio_N_voxel = ratio_N_voxel(ratio_N_voxel(:,1)==cellidx,:);
if ¬isempty(ratio_N_voxel);
calibration_factor = ratio_N_voxel(2); % Calibration factor
stochiometry = ratio_N_voxel(3); % multimeric or monomeric
measured_position = LSM_Intensity(idxpath(:,1)==cellidx,1:2);
```

```
GaASP_intensity = LSM_Intensity(idxpath(:,1)==cellidx,5);
N_fitted = N_corr(idxpath(:,1)==cellidx);
subpos_QC = idxpath(idxpath(:,1)==cellidx,3);
save(savefile,'calibration_factor','stochiometry', ...
    'measured_position','GaASP_intensity','N_fitted', 'subpos_QC');
% Update the QC for the cell. Discard cells if less than 2 valid ...
    FCS measurements were performed
if sum(subpos_QC) ≥ 2;
vec_QC(cdx) = 1;
else
vec_QC(cdx) = 0;
end
else
vec_QC(cdx) = 0;
end
else
load(savefile);
if sum(subpos_QC) ≥ 2;
vec_QC(cdx) = 1;
else
vec_QC(cdx) = 0;
end
end
end
end

summary_database.fcs_calibration = vec_QC;
writetable(summary_database,exp_database,'Delimiter','\t');
end
```

The following function calculate the calibration factor for all cells with the same protein imaged on the same day with the same microscope.

```
function fcs_calibration(data_dir)

% Load the Bioformat Toolbox
addpath('Y:\Yin\Code\pipeline\bfmatlab');

% Fixed parameters
```

```
D_alexa488 = 441; % diffusion coefficient Alexa 488 in ...
    micrometer^2/s in water at 37 degree
kappa_lsm = 5.5; % ZEISS 780, based on Malte Wachsmuth
R_th = 0.65; % Fitting R square QC threshold
C_th = 0.15; % Fitting Chi square QC threshold
B_th = 0.9; % Bleaching correction QC threshold
SN_th = 2; % Signal to noise QC threshold
Diameter_filter = 9; % Mean filter size of the GaASP detector image
voxel_Z_size = 0.75; % resolution in z, in micrometer
measure_min = 4; % min number of measurements within a cell for CPM QC
GFP_CPM_QC = 4.25; % max value of the CPM value for mEGFP

% Read the res file and extract useful part
% Full path (3),Intensity (Interval1 Ch1, 29), Correction (Total ...
    Ch1, 44), Rsq (R sq adj Ch1, 65), N_fit (P01 value,
% 71), tauD (P05 value, 87), fraction of fast moving molecules (P04 ...
    value, 83), Background APD(offset Ch1,
% 17), Bleached (Bleaching Ch1, 35), Chi_sq (R sq adj Ch1, 64)
fitting_filename = fullfile(data_dir,'Result','fcs_fittings.res');
fittings_data = importdata(fitting_filename);

fullpath = fittings_data.textdata(3:end,3);
Offset = str2double(fittings_data.textdata(3:end,17));
Intensity = str2double(fittings_data.textdata(3:end,29));
Bleached = str2double(fittings_data.textdata(3:end,35));
Correction = str2double(fittings_data.textdata(3:end,44));
Rsq = str2double(fittings_data.textdata(3:end,65));
Csq = str2double(fittings_data.textdata(3:end,64));
N_fit = str2double(fittings_data.textdata(3:end,71));
tauD = str2double(fittings_data.textdata(3:end,87));
f1 = str2double(fittings_data.textdata(3:end,83));
clear fittings_data

% Generate a database file based on fullpath including the information:
total_sequences = length(fullpath);
idxpath = zeros(total_sequences,3);
for sidx = 1:total_sequences;
    splitted = strsplit(fullpath{sidx},{'Alexa\','_0'}, ...
        'CollapseDelimiters',true);
    if length(splitted) ≥ 3; % it is a dye
```

```
        splitted = splitted{2};
        if splitted(1) == '4';
            idxpath(sidx,1:2) = [488 0]; % Alexa 488
        elseif splitted(1) == '5';
            idxpath(sidx,1:2) = [561 0]; % Alexa 561
        else
            idxpath(sidx,1:2) = [0 0]; % Dy481XL on Dextran 500
        end
    else
       splitted = strsplit(fullpath{sidx},{'GFP\','_0'}, ...
          'CollapseDelimiters',true);
       if length(splitted) ≥ 3; % it is a GFP calibration measurement
           p_splitted = splitted{2};
           sub_splitted = splitted{end};
           idxpath(sidx,1) = -str2double(p_splitted(2:end)); % cell ...
              index
           idxpath(sidx,2) = str2double(sub_splitted(6)); % ...
              position index
       else
           splitted = strsplit(fullpath{sidx},{'AQ_','_P0001'}, ...
              'CollapseDelimiters',true); % it is a measurement in ...
              the cell with the POI
           p_splitted = splitted{2};
           sub_splitted = splitted{end};
           idxpath(sidx,1) = str2double(p_splitted(2:end)); % cell ...
              index
           idxpath(sidx,2) = str2double(sub_splitted(end-11)); % ...
              position index
       end
    end
end

% First relaxed QC based on good R square, Chi square values and ...
   signal to
% noise ratio.
idxpath(:,3) = double((Rsq ≥ R_th).*(Csq ≤ C_th).*(Bleached ≥ ...
   B_th).* ((Intensity.*Bleached./Offset)≥SN_th));

% Calculate the confocal volume
isalexa488 = (idxpath(:,1)+idxpath(:,3)==489);
```

```
Veff = ...
    mean((4*D_alexa488*pi*tauD(isalexa488)*10^(-6)).^(3/2)*kappa_lsm); ...
    % tauD unit: us. Veff unit: um^3

% Correct the bleaching effect in Intensity
% Assuming that the bleaching at P1 doesn't affect the ground ...
    intensity of P6
Intensity = Intensity.*Bleached;

% Calculate the averaged Intencity/molecule for mEGFP
N_corr = N_fit.*Correction;
CPM = (Intensity-Offset)./N_corr; % count per molecule value, ...
    indicating the brightness
isgfp = ((idxpath(:,1)<0).*idxpath(:,3).*(CPM<GFP_CPM_QC))>0; % QC ...
    mEGFP: no dimer measurements
CPM_singleGFP = median(CPM(isgfp));
idxpath(((idxpath(:,1)<0).*idxpath(:,3).*(CPM≥GFP_CPM_QC))>0,3) = 0;

% Calculate the background for the GaASP detector;
dir_cellfree = fullfile(data_dir,'Calibration','Alexa');
idx_dexfile = find(idxpath(:,1) == 0);
sum_background = zeros(length(idx_dexfile),6); % Values: Position ...
    start/end, Detector range start/end, BG, Bit
for didx = 1:length(idx_dexfile);
    dex_filepath = fullpath{idx_dexfile(didx)}; % Using the image ...
        in cell free dextran-Dy481XL medium for background calculation
    splitted = strsplit(dex_filepath,{'Alexa\','_R1'}, ...
        'CollapseDelimiters',true);
    dex_root = splitted{2};
    % Summarize in which well the dextran medium image was acquired ...
        using
    % the index of positions on which the cells were imaged
    P_position = strfind(dex_root,'P');
    if isempty(P_position);
        sum_background(didx,1) = 0; % just small enough
        sum_background(didx,2) = 100; % just large enough
    else
        P_to_P = dex_root((P_position(1)+1):(end-2));
        splitP = strsplit(P_to_P,{'P','to'},'CollapseDelimiters',true);
        if length(splitP) == 1;
```

```
            sum_background(didx,1) = str2double(splitP{1});
            sum_background(didx,2) = str2double(splitP{1});
        else
            sum_background(didx,1) = str2double(splitP{1});
            sum_background(didx,2) = str2double(splitP{2});
        end
    end
    % Load the image and calculate the background as the mean of ...
       the image
    % in the mEGFP channel
    dex_file = fullfile(dir_cellfree,[dex_root,'_preFCS.lsm']);
    dex_data = bfopen(dex_file);
    dex_image = dex_data{1,1};
    dex_image = dex_image{1,1};
    dex_metadata = dex_data{1,2};
    sum_background(didx,3) = dex_metadata.get( 'DetectionChannel ...
       SPI Wavelength Start #1');
    sum_background(didx,4) = dex_metadata.get( 'DetectionChannel ...
       SPI Wavelength End #1');
    sum_background(didx,5) = mean(dex_image(:));
    dex_bit = class(dex_image);
    sum_background(didx,6) = str2double(dex_bit(5:end));
    voxel_XY_size = dex_metadata.get('Recording Line Spacing #1');
    clear dex_data
    clear dex_image
    clear dex_metadata
end

% Calculate the GaASP intensity for all FCS measured positions
dir_cell = fullfile(data_dir,'LSM');
idx_cellfileP1 = find(double(idxpath(:,1) > 0).*double( ...
   idxpath(:,2) == 1) > 0);
LSM_Intensity = zeros(size(idxpath,1),6);
for cidx = 1:length(idx_cellfileP1);
    idx_subpos = find(idxpath(:,1)==idxpath(idx_cellfileP1(cidx),1));
    tr2_leveldir = ...
       strsplit(fullpath{idx_cellfileP1(cidx)},{'LSM\','FCS'}, ...
       'CollapseDelimiters',true);
    tr2_leveldir = strsplit(tr2_leveldir{2},{'\'}, ...
       'CollapseDelimiters',true);
```

```
dir_cell_tr2 = fullfile(dir_cell,tr2_leveldir{1}, ...
    tr2_leveldir{2},tr2_leveldir{3});
lsm_cell_tr2 = dir(fullfile(dir_cell_tr2,'*.lsm'));
lsm_cell_tr2 = fullfile(dir_cell_tr2,lsm_cell_tr2(1).name);
txt_cell_tr2 = dir(fullfile(dir_cell_tr2,'*.txt'));
txt_cell_tr2 = fullfile(dir_cell_tr2,txt_cell_tr2(1).name);
% Get the positions from the txt
txt_positions = importdata(txt_cell_tr2);
txt_positions = txt_positions.data;
txt_positions = txt_positions(7:12,1:2);
% Get the lsm image and metadata
lsm_data = bfopen(lsm_cell_tr2);
tr2_image = lsm_data{1,1};
tr2_image = tr2_image{1,1};
lsm_bit = class(tr2_image);
lsm_bit = str2double(lsm_bit(5:end));
% Mean-filter the image with the pre-defined diameter
tr2_image_filtered = imfilter(tr2_image,fspecial('average', ...
    [Diameter_filter,Diameter_filter]));
lsm_metadata = lsm_data{1,2};
detect_start = lsm_metadata.get( 'DetectionChannel SPI ...
    Wavelength Start #1');
detect_end = lsm_metadata.get( 'DetectionChannel SPI Wavelength ...
    End #1');
% Summary for corresponding positions, Value: position x, y, ...
    detector
% range start, end, intensity filtered, depth of the bit
for subidx = 1:length(idx_subpos);
    LSM_Intensity(idx_subpos(subidx),1) = ...
        max(1,txt_positions(idxpath( idx_subpos(subidx),2),1));
    LSM_Intensity(idx_subpos(subidx),2) = ...
        max(1,txt_positions(idxpath( idx_subpos(subidx),2),2));
    LSM_Intensity(idx_subpos(subidx),3:6) = [detect_start ...
        detect_end ...
        tr2_image_filtered(LSM_Intensity(idx_subpos(subidx),2), ...
        LSM_Intensity(idx_subpos(subidx),1)) lsm_bit];
end
clear lsm_data
clear tr2_image
clear tr2_image_filtered
```

```
    clear lsm_metadata
end

% Calculate the calibration factor for each cell
iscell = (idxpath(:,1).*idxpath(:,2))>0;
% Set the QC criteria based on the CPM value. Outlier measurements ...
   will be
% excluded later.
mean_cpm = mean(CPM(iscell));
std_cpm = std(CPM(iscell));
qc_cpm_min = max(1,mean_cpm-2*std_cpm);
qc_cpm_max = mean_cpm+2*std_cpm;
% Define output matrices
Vvoxel = voxel_XY_size^2*voxel_Z_size; %um^3
ratio_N_voxel = zeros(length(idx_cellfileP1),3);
compare_cpm = zeros(length(idx_cellfileP1),2);
compare_ratio = zeros(length(idx_cellfileP1),2);
% For each cell
for cidx = 1:length(idx_cellfileP1);
    idx_subpos = find(idxpath(:,1)==idxpath(idx_cellfileP1(cidx),1));
    qc_intracell = zeros(length(idx_subpos),5);
    for subidx = 1:length(idx_subpos);
        qc_intracell(subidx,1) = idx_subpos(subidx); % index
        qc_intracell(subidx,2) = idxpath(idx_subpos(subidx),3); % ...
           Basic QC
        qc_intracell(subidx,3) = CPM(idx_subpos(subidx)); % CPM
        qc_intracell(subidx,4) = ...
           LSM_Intensity(idx_subpos(subidx),5); % LSM intensity ...
           with background
        qc_intracell(subidx,5) = N_corr(idx_subpos(subidx)); % N ...
           from the fitting
    end
    % If there were enough QC+ measurements for this cell, QC the
    % measurements based on the CPM value.
    if size(qc_intracell,1) >  measure_min
        qc_intracell(qc_intracell(:,3) > qc_cpm_max,2) = 0;
        qc_intracell(qc_intracell(:,3) < qc_cpm_min,2) = 0; % not ...
           passed the CPM QC
    end
    % identify the background for this position
```

```
for bidx = 1:size(sum_background,1);
    if ismember(idxpath(idx_subpos(1),1), ...
        sum_background(bidx,1):sum_background(bidx,2));
         bg_index = bidx;
    end
end
% substract the background
qc_intracell(:,4) = qc_intracell(:,4)-sum_background(bg_index,5);
qc_intracell(qc_intracell(:,4) ≤ 0,2) = 0; % negative intensity ...
   not passed the intensity
qc_intracell(qc_intracell(:,3) ≤ 0,2) = 0; % netative CPM not ...
   passed the CPM QC %
% Give warnings if the metadata doesn't match
if ¬isequal(round([sum_background(bg_index,3:4) ...
    sum_background(bg_index,6)]),round([LSM_Intensity(idx_subpos(1),3:4) ...
    LSM_Intensity(idx_subpos(1),6)]));
    disp([data_dir 'calibration metadata missmatch aquisition']);
    [sum_background(bg_index,3:4) sum_background(bg_index,6) ...
        LSM_Intensity(idx_subpos(1),3:4) ...
        LSM_Intensity(idx_subpos(1),6)]
end
% Get the calibration ratio
ratio_N_voxel(cidx,1) = idxpath(idx_subpos(1),1);
if sum(qc_intracell(:,2)>0)>0;
    % Normalize the CPM value of measurements in the cytoplasma ...
       agains mEGFP CPM
    cpm_cyt = qc_intracell(3:6,3);
    median_cpm = median(cpm_cyt(qc_intracell(3:6,2)>0));
    ratio_cpm = median_cpm/CPM_singleGFP;
    % Correct the N value if multimeric molecules are existing
    if ratio_cpm > 1;
        qc_intracell(:,5) = qc_intracell(:,5)*ratio_cpm;
        ratio_N_voxel(cidx,3) = ratio_cpm;
    else
        ratio_N_voxel(cidx,3) = 1;
    end
    % Calculate the calibration factor
    all_ratio = (qc_intracell(qc_intracell(:,2)>0,5)./ ...
       qc_intracell(qc_intracell(:,2)>0,4))*Vvoxel/Veff;
```

```
        ratio_N_voxel(cidx,2) = mean(all_ratio); % Average of all ...
            QC+ measurements in that cell: I*ratio_N_voxel = N in a ...
            voxel;
    else
        ratio_N_voxel(cidx,2) = 0;
    end
    for qidx = 1:size(qc_intracell,1);
        idxpath(qc_intracell(qidx,1),3) = qc_intracell(qidx,2);
        LSM_Intensity(qc_intracell(qidx,1),5) = qc_intracell(qidx,4);
    end
end
ratio_N_voxel(ratio_N_voxel(:,2)==0,:) = [];

% Write the output file (for now! after the segmentation is done, ...
   it can be
% saved into the generated cell individual result space
outputfile = fullfile(data_dir,'Result','FCS_calibration.mat');
save(outputfile,'fullpath','idxpath','sum_background','ratio_N_voxel', ...
   'N_corr','CPM','tauD','Rsq','LSM_Intensity','Veff', ...
   'D_alexa488','kappa_lsm','R_th','Diameter_filter', ...
   'voxel_Z_size','measure_min','f1', 'CPM_singleGFP','compare_ratio');

end
```

6.3.3 Extraction of the geometrical features of the landmarks

Afterwards, features describing the landmarks geometry were extracted.

```
function LM_shape_feature(exp_dir)

%% Parameter settings
feature_name = ...
   {'nuc_dist';'nuc_dist2';'nuc_vol';'nuc_3eigenval';'divaxis_cs_l'; ...
   'divaxis_cs_h';'time'};

% Interactive selection of experiment directory
if ~nargin
exp_dir = uigetdir('','Select the directory with your data series');
end
```

```
%% Load the database file
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
summary_database.lm_features = ...
    -ones(size(summary_database.lm_features));
cell_dirlist = ...
    summary_database.filepath(summary_database.segmentation>0);
cell_total = length(cell_dirlist);
vec_QC = zeros(cell_total,1);
% extract features cell by cell
for cdx = 1:cell_total;
cellseq_list = ...
    dir(fullfile(cell_dirlist{cdx},'Preprocessing','Segmentation', ...
    '*T0*.mat'));
% Make the output directiry
temporal_dir = fullfile(cell_dirlist{cdx},'Temporal_Align');
if ¬exist(temporal_dir,'dir');
mkdir(temporal_dir);
end
LM_feature_seq = fullfile(temporal_dir,'LM_feature_seq.mat');
% If the output already exists and extracts the same features, then ...
    set the QC to 1 and skip processing
if exist(LM_feature_seq,'file');
M = load(LM_feature_seq);
if isequal(feature_name,M.feature_name);
vec_QC(cdx) = 1;
end
end
% Process un-processed image sequences
if vec_QC(cdx) ≠ 1;
disp(['Process' fullfile(cell_dirlist{cdx},'Landmarks feature ...
    extraction')])
LM_feats = zeros(length(feature_name),length(cellseq_list));
for tdx = 1:length(cellseq_list);
load(fullfile(cell_dirlist{cdx},'Preprocessing','Segmentation', ...
    cellseq_list(tdx).name)); % load the result from the ...
    segmentation pipeline
LM_feats(1:2,tdx) = chrDistPix; % extracted by Julius Hossain, ...
    inter-nuclei distance from the technical complete disconnection.
```

```
if isequal(mitoTime{1},'Ana');
if numChr == 1;
    LM_feats(2,tdx) = chrSptDistPix; % distance between two ...
        chromosomes from anaphase onwards even both nuclei still ...
        have connections
end
end

LM_feats(3,tdx) = chrVolMic; % volume of the chromosome(s), ...
    extracted by Julius Hossain

LM_feats(4,tdx) = eigValuesMic(1,1); % The 3rd (smallest) ...
    eigenvalue of the chromatin volume, extracted by Julius Hossain

% Cell length and height on the section of the predicted division ...
    axis Calculate the 2D section through the division axis
dx = isectPointsPix(1,1,1)-isectPointsPix(2,1,1);
dy = isectPointsPix(1,2,1)-isectPointsPix(2,2,1);
if dx == 0;
div_axis_sec = ...
    cat(2,isectPointsPix(1,1,1)*ones(size(cellVolume,2),1), ...
    (1:size(cellVolume,2))');
elseif dy == 0;
div_axis_sec = cat(2,(1:size(cellVolume,1))',isectPointsPix(1,2,1)* ...
    ones(size(cellVolume,1),1));
elseif abs(dx) > abs(dy);
d = dy/dx;
div_axis_y = isectPointsPix(1,2,1):-abs(d):1;
if d > 0;
    div_axis_x = isectPointsPix(1,1,1):-1:1;
else
    div_axis_x = isectPointsPix(1,1,1):1:size(cellVolume,1);
end
l = min(length(div_axis_y),length(div_axis_x));
div_axis_y = ...
    cat(2,div_axis_y(1:l),isectPointsPix(1,2,1):abs(d):size(cellVolume,2));
if d > 0;
    div_axis_x = ...
        cat(2,div_axis_x(1:l),isectPointsPix(1,1,1):1:size(cellVolume,1));
else
```

```
    div_axis_x = cat(2,div_axis_x(1:l),isectPointsPix(1,1,1):-1:1);
end
l = min(length(div_axis_y),length(div_axis_x));
div_axis_sec = cat(2,div_axis_x(2:l)',div_axis_y(2:l)');
div_axis_sec = round(sortrows(div_axis_sec));
elseif abs(dx)≤ abs(dy);
d = dx/dy;
div_axis_x = isectPointsPix(1,1,1):-abs(d):1;
if d > 0;
    div_axis_y = isectPointsPix(1,2,1):-1:1;
else
    div_axis_y = isectPointsPix(1,2,1):1:size(cellVolume,2);
end
l = min(length(div_axis_y),length(div_axis_x));
div_axis_x = ...
   cat(2,div_axis_x(1:l),isectPointsPix(1,1,1):abs(d):size(cellVolume,1));
if d > 0;
    div_axis_y = ...
       cat(2,div_axis_y(1:l),isectPointsPix(1,2,1):1:size(cellVolume,2));
else
    div_axis_y = cat(2,div_axis_y(1:l),isectPointsPix(1,2,1):-1:1);
end
l = min(length(div_axis_y),length(div_axis_x));
div_axis_sec = cat(2,div_axis_x(2:l)',div_axis_y(2:l)');
div_axis_sec = round(sortrows(div_axis_sec));
end
l = size(div_axis_sec,1);
sec_mat = zeros(l,size(cellVolume,3));
for ldx = 1:l;
sec_mat(ldx,:) = ...
   reshape(cellVolume(div_axis_sec(ldx,1),div_axis_sec(ldx,2),:), ...
   1,size(cellVolume,3));
end
% Calculate the length and height by finding the first and last ...
   non-zero pixel. The 3 frames closest to the bottom were not ...
   included due to irregular adhesions.
length_vec = double(sum(sec_mat(:,4:end),2)>0);
height_vec = double(sum(sec_mat(:,4:end),1)>0);
length_vec = [find(length_vec,1,'first') find(length_vec,1,'last')];
height_vec = [find(height_vec,1,'first') find(height_vec,1,'last')];
```

```
if isempty(length_vec);
LM_feats(5,tdx) = 0;
LM_feats(6,tdx) = 0;
else
LM_feats(5,tdx) = sqrt(sum((div_axis_sec(length_vec(end),:) - ...
    div_axis_sec(length_vec(1),:)).^2));
LM_feats(6,tdx) = height_vec(end) - height_vec(1) + 1;
end

LM_feats(8,tdx) = str2double(splitted{end-1});

end
% Sort the matrix by time
LM_feats = (sortrows(LM_feats',[7,1]))';
% QC+ if 40 frames were processed
if size(LM_feats,2)==40;
if LM_feats(2,end) > 0;
vec_QC(cdx) = 1;
save(LM_feature_seq,'LM_feats','feature_name')
end
end
end
end
summary_database.lm_features(summary_database.segmentation>0) = vec_QC;
writetable(summary_database,exp_database,'Delimiter','\t');

end
```

6.3.4 Generation of the data set with H2B as POI

Since all data has H2B imaged as landmarks, images were randomly selected and the H2B signal from the landmarks channel was transferred into the POI channel in order to generate a data set with H2B as POI.

```
nuc_dir = fullfile(exp_dir,'150506_H2B');

function Synthetic_nuc(nuc_dir,exp_dir)
% Load the database file
exp_database = fullfile(exp_dir,'full_database.txt');
```

```
summary_database = readtable(exp_database,'Delimiter','\t');

% Number of H2B cells generated
select_number = 35;

% Find cells with reference proteins imaged (here, only cells with ...
    reference proteins were used for the H2B image synthesis. ...
    However, any images can be used for this purpose.)
ref_idx = strcmp(summary_database.label,'ref');
% Only cells with positive QC were considered
ref_celldir_list = ...
    summary_database.filepath((ref_idx.*(summary_database.fcs_calibration>0) ...
    .*(summary_database.segmentation>0).*(summary_database.lm_features>0))>0);
ref_number = length(ref_celldir_list);
% Generate their QC and processing status
basetab = cell2table({'n',1,'H2B','ref','Chr',1,1,1,1,0,0,0,0});
basetab.Properties.VariableNames = ...
    summary_database.Properties.VariableNames;

% Select cells
selected = 0;
for idx = 1:ref_number;
if selected < select_number;
disp(['Generate H2B' num2str(selected)])
% Generate the output directory
celldir = fullfile(nuc_dir,ref_celldir_list{idx}(end-29:end));
% If there is no directory with the same name
if ¬exist(celldir,'dir');
selected = selected + 1;
mkdir(fullfile(celldir,'Preprocessing','Segmentation'));
mkdir(fullfile(celldir,'Temporal_Align'));
seg_filelist = dir(fullfile(ref_celldir_list{idx}, ...
    'Preprocessing','Segmentation','TR*.mat'));
% Generate frame by frame
for f_idx = 1:length(seg_filelist);
    % Load the segmentation results
    load(fullfile(ref_celldir_list{idx},'Preprocessing','Segmentation', ...
        seg_filelist(f_idx).name));
    % Generate the poi channel by using the mCherry channel image ...
        croped by the cell volume from the segmentation
```

```
    poi = nuc.*cellVolume;
    % Save the new segmentation file to the H2B directory
    nuc_savefile = fullfile(celldir,'Preprocessing','Segmentation', ...
        seg_filelist(f_idx).name);
    if isequal(mitoTime{1},'Ana');
        if numChr == 1;
            save(nuc_savefile,'bgMask','cellVolume','neg', ...
                'nucVolume','nuc','poi','chrCentMic', ...
                'isectPointsPix', ...
                'midPlane','regPointsMic','eigVectors','eigValuesMic', ...
                'cellAxis','chrVolMic','chrDistPix','numChr','mitoTime', ...
                'avgBgInt','chrSptDistPix');
        else
            save(nuc_savefile,'bgMask','cellVolume','neg', ...
                'nucVolume','nuc','poi','chrCentMic', ...
                'isectPointsPix', ...
                'midPlane','regPointsMic','eigVectors','eigValuesMic', ...
                'cellAxis','chrVolMic','chrDistPix','numChr','mitoTime', ...
                'avgBgInt');
        end
    else
        save(nuc_savefile,'bgMask','cellVolume','neg', ...
            'nucVolume','nuc','poi','chrCentMic', 'isectPointsPix', ...
            'midPlane','regPointsMic','eigVectors','eigValuesMic', ...
            'cellAxis','chrVolMic','chrDistPix','numChr','mitoTime', ...
            'avgBgInt');
    end
end
% Save the fcs calibration file to the H2B directory by using the ...
   pre-defined calibration rate of 0.002.
fcs_savefile = fullfile(celldir,'Preprocessing','calibration.mat');
load(fullfile(ref_celldir_list{idx},'Preprocessing','calibration.mat'));
calibration_factor = 0.002;
stochiometry = 1;
save(fcs_savefile,'calibration_factor','stochiometry');
% Save the landmarks feature file to the H2B directory
lmfeats_savefile = ...
   fullfile(celldir,'Temporal_Align','LM_feature_seq.mat');
load(fullfile(ref_celldir_list{idx},'Temporal_Align','LM_feature_seq.mat'));
save(lmfeats_savefile,'LM_feats','feature_name','var_cell');
```

```
basetab.filepath{1} = celldir;
basetab.system = str2double(ref_celldir_list{idx}(end-22));
summary_database = cat(1,summary_database,basetab);
end
end
end
% Upate the database file
writetable(summary_database,exp_database,'Delimiter','\t');

end
```

6.3.5 Generate the Mitotic Standard Time model

Using a subset of the data, a model of the mitotic standard time was generated using multi sequence dynamic time warping.

```
% model_dir = '\temporal_model';
% mkdir(model_dir)

function model_dir = temporal_alignment(exp_dir,model_dir)

% Interactive selection of experiment directory
if isempty(exp_dir)
    exp_dir = uigetdir('','Select the directory with your data ...
       series');
end

% Generate a subset of the data for the modelling. In this work, ...
   all cells
% with reference proteins, i.e. proteins localized in a single ...
   subcellular
% structure except for H2B, were selected. The total number was 132.
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
ref_idx = (summary_database.segmentation>0).* ...
   (summary_database.lm_features>0);
ref_idx = ref_idx .*(strcmp(summary_database.label,'ref')>0).* ...
   (strcmp(summary_database.poi,'H2B')==0);
filepath = summary_database.filepath(ref_idx>0);
```

```
addpath = cellstr(repmat([filesep 'Temporal_Align' filesep ...
    'LM_feature_seq.mat'], [length(filepath) 1]));
feat_filelist = strcat(filepath',addpath')';
savefile = fullfile(exp_dir,'LM_align_feat.mat');
save(savefile,'featseq_list');

% Set up the parameters
% Select landmarks geometrical features for the modelling. In this ...
    work,
% only three features were used: inter-nuclei distance, nuclear ...
    volume and
% third eigenvalue of the nuclear volume
takefeat = [0;1;1;1;0;0;0];
seq_length = 40;
% Parameters for the smoothing for first derivative calculation
smooth_degree = 3; % filtering parameters for calculating the ...
    derivatives
spline_Δ = 0.05;
% Dynamic time warping parameters
penat_fuse = [0;0];
penat_chop = [5;54];
penat_gap = [12;54];
unit_t = 0.3;
max_rounds = 4; % number of rounds for multi sequence alignment

%% Generate the model
feature_number = sum(takefeat);
cell_total = length(feat_filelist);
savefile = fullfile(model_dir,'temporal_alignment.mat');

%% Get the feature matrix for all selected sequences and normalization
feat_original = zeros(feature_number,seq_length,cell_total);
for cdx = 1:cell_total;
    if exist(feat_filelist{cdx},'file')
        load(feat_filelist{cdx});
        if size(LM_feats,2) == seq_length;
            feat_original(:,:,cdx) = LM_feats(takefeat>0,:);
        end
    end
end
```

```
feat_original(:,:,max(max(feat_original,[],1),[],2)==0) = []; % ...
    delete sequences without any non-zero data
cell_total = size(feat_original,3);

% Normalize all features using mean and std
feat_mean = mean(reshape(feat_original,[feature_number, ...
    seq_length*cell_total]),2);
feat_std = std(reshape(feat_original,[feature_number, ...
    seq_length*cell_total]),[],2);
feat_normalized=(feat_original-repmat(feat_mean,[1,seq_length,cell_total])) ...
    ./repmat(feat_std,[1,seq_length,cell_total]);

%% Calculate the first derivative based on filtered data and ...
    normalize piecewise smoothing fit and derivative from the fit
t_aq = 1:seq_length;
feat_derivative = zeros(size(feat_normalized));
for fdx = 1:feature_number;
    for cdx = 1:cell_total;
        [smooth_seq,stat] = ...
            fit(t_aq',feat_normalized(fdx,:,cdx)','smoothingspline');
        feat_derivative(fdx,:,cdx) = ...
            (differentiate(smooth_seq,t_aq'))';
    end
end

% Normalize all derivative features using mean and std
dev_mean = mean(reshape(feat_derivative,[feature_number, ...
    seq_length*cell_total]),2);
dev_std = std(reshape(feat_derivative,[feature_number, ...
    seq_length*cell_total]),[],2);
feat_derivative=(feat_derivative-repmat(dev_mean,[1,seq_length,cell_total])) ...
    ./repmat(dev_std,[1,seq_length,cell_total]);

feature_parameters = struct('select',takefeat,'plotf',plotfeat_p, ...
    'mean_0_deriv',feat_mean,'std_0_deriv',feat_std, ...
    'mean_1_deriv',dev_mean,'std_1_deriv',dev_std, ...
    'smoothing',smooth_degree,'spline',spline_Δ, ...
    'penatration',[penat_fuse penat_chop penat_gap], ...
    'unit_t',unit_t,'frame_num',seq_length,'cellnorm',cellnorm);
```

```
% Calculate the total standard deviation
std_ground = std([feat_normalized;feat_derivative],[],3);
dev_track = [sum(std_ground(¬isnan(std_ground))) 0];

%% Alignment first round
% Calculate the Euclidic distance between all sequences
dist_matrix = zeros(cell_total,cell_total);
for cell1_idx = 1:cell_total;
    for cell2_idx = (cell1_idx+1):cell_total;
        calc_f = ...
            [feat_normalized(:,:,cell1_idx)-feat_normalized(:,:,cell2_idx) ...
            feat_derivative(:,:,cell1_idx)-feat_derivative(:,:,cell2_idx)];
        calc_f(isnan(calc_f)) = 0;
        dist_matrix(cell1_idx,cell2_idx) = sum(sum(calc_f.^2));
        dist_matrix(cell2_idx,cell1_idx) = ...
            dist_matrix(cell1_idx,cell2_idx);
    end
end
align_ranking = zeros(cell_total,1);
% find the cell having the smallest total distance to all other cells
[val,align_ranking(1)] = min(sum(dist_matrix,2));
dist_matrix(:,align_ranking(1)) = inf;
[val,align_ranking(2)] = min(dist_matrix(align_ranking(1),:));
dist_matrix(:,align_ranking(2)) = inf;

% Align the first two sequences against each other using the function
% multidim_DDTW
align_TD = zeros(cell_total,1);
align_model = [feat_normalized(:,:,align_ranking(1)); ...
    feat_derivative(:,:,align_ranking(1))];
align_seq = [feat_normalized(:,:,align_ranking(2)); ...
    feat_derivative(:,:,align_ranking(2))];
[align_TD(align_ranking(2)),matching_vector]=multidim_DDTW( ...
    align_model,t_aq,align_seq,t_aq,penat_fuse, ...
    penat_chop,penat_gap,unit_t);

% Assign the alignment and calculate the average
aligned_full = zeros(cell_total,size(matching_vector,1));
aligned_full(align_ranking(1),:) = matching_vector(:,1)';
aligned_full(align_ranking(2),:) = matching_vector(:,2)';
```

```
master_seq = zeros(size(align_model,1),size(aligned_full,2));
std_master = zeros(size(master_seq));
for atidx = 1:size(aligned_full,2);
    aligned_frames = zeros(size(align_model,1), ...
       sum(aligned_full(:,atidx)>0));
    seq_idx = 0;
    for cidx = 1:cell_total;
        if aligned_full(cidx,atidx) > 0;
            seq_idx = seq_idx + 1;
            aligned_frames(:,seq_idx) = [feat_normalized(:, ...
               aligned_full(cidx,atidx),cidx);feat_derivative(:, ...
               aligned_full(cidx,atidx),cidx)];
        end
    end
    actual_feat = sum((1-isnan(aligned_frames)),2);
    calc_f = aligned_frames;
    calc_f(isnan(calc_f)) = 0;
    master_seq(:,atidx) = sum(calc_f,2)./actual_feat;
    calc_f = aligned_frames-repmat(master_seq(:,atidx), [1 ...
       size(aligned_frames,2)]);
    calc_f(isnan(calc_f)) = 0;
    actual_feat(actual_feat<2) = 2;
    std_master(:,atidx) = sqrt(sum((calc_f).^2,2)./(actual_feat-1));
end

% Calculate the timeline for the master_seq using the progression time
timeline = [aligned_full(align_ranking(1),:); ...
   aligned_full(align_ranking(2),:)];
for cellidx = 1:2;
    for atidx = 2:size(aligned_full,2);
        if timeline(cellidx,atidx) < timeline(cellidx,atidx-1);
            timeline(cellidx,atidx) = timeline(cellidx,atidx-1);
        end
    end
end
timeline = diff(cat(1,zeros(1,2),timeline'));
timeline = mean(timeline,2);
for atidx = 2:length(timeline);
    timeline(atidx) = timeline(atidx-1) + timeline(atidx);
end
```

```
%% Align the remaining sequences to the averaged sequence and ...
   refine the model after each alignment
for align_cdx = 3:cell_total;
    % find the next processing sequence
    [val,align_ranking(align_cdx)] = min( ...
       dist_matrix(align_ranking(align_cdx-1),:));
    dist_matrix(:,align_ranking(align_cdx)) = inf;
    % align the sequence to the model
    align_seq = [feat_normalized(:,:,align_ranking(align_cdx)); ...
       feat_derivative(:,:,align_ranking(align_cdx))];
    [align_TD(align_ranking(align_cdx)),matching_vector]=multidim_DDTW( ...
       master_seq,timeline,align_seq,t_aq,penat_fuse, ...
       penat_chop,penat_gap,unit_t);
    % Assign the alignment using the function AssAligned
    aligned_full = ...
       AssAligned(aligned_full,matching_vector,cell_total, ...
       align_ranking(align_cdx));
    % Update the averaged sequence and the timeline using the ...
       function GetMaster
    [master_seq,timeline,StD] = GetMaster([feat_normalized; ...
       feat_derivative],aligned_full);
end

% Quality control of the alignment using the function qc_masterseq ...
   and update the averaged sequence after
% delting sequences with bad alignment
[aligned_full,qc_idx] = qc_masterseq(align_TD,aligned_full);
[master_seq,timeline,StD] = ...
   GetMaster([feat_normalized;feat_derivative],aligned_full);

% Document the total standard deviation
norm_StD = StD.*repmat((diff([0;timeline]))',[size(StD,1) 1]);
dev_track = cat(1,dev_track,[sum(norm_StD(¬isnan(norm_StD))) ...
   sum(align_TD)/cell_total]);

% Refine the alignment for pre-defined number of rounds
al_round = 1;
while al_round < max_rounds;
    align_ranking = [align_TD (1:cell_total)'];
```

```
    align_ranking = sortrows(align_ranking);
    align_ranking = align_ranking(:,2);
    for cidx = 1:cell_total;
        align_seq = [feat_normalized(:,:,align_ranking(cidx)); ...
            feat_derivative(:,:,align_ranking(cidx))];
        aligned_full(align_ranking(align_cdx),:) = 0;
        [master_seq,timeline,StD] = ...
            GetMaster([feat_normalized;feat_derivative], aligned_full);
        [align_TD(align_ranking(cidx)),matching_vector]=multidim_DDTW( ...
            master_seq,timeline,align_seq,t_aq,penat_fuse, ...
            penat_chop,penat_gap,unit_t);
        aligned_full = ...
            AssAligned(aligned_full,matching_vector,cell_total, ...
            align_ranking(align_cdx));
        [master_seq,timeline,StD] = GetMaster([feat_normalized; ...
            feat_derivative],aligned_full);
    end
    [aligned_full,qc_idx] = qc_masterseq(align_TD,aligned_full);
    [master_seq,timeline,StD] = ...
       GetMaster([feat_normalized;feat_derivative], aligned_full);
    norm_StD = StD.*repmat((diff([0;timeline]))',[size(StD,1) 1]);
    dev_track = cat(1,dev_track,[sum(norm_StD(¬isnan(norm_StD))) ...
       sum(align_TD)/cell_total]);
    al_round = al_round + 1;

end
save(savefile,'feature_parameters','feat_normalized','feat_derivative', ...
    'feat_filelist','master_seq','aligned_full','timeline', ...
    'dev_track','StD','align_TD','qc_idx');
end
```

Functions that was used for generating the model:

Dynamic time warping based pairwise sequence alignment

```
function ...
    [TD,w]=multidim_DDTW(sm,tm,ss,ts,pen_fuse,pen_chop,pen_gap,unit_t)

% sm and ss are the sequence of the model and the sequence
% tm and ts are the timeline of the model and the sequence
```

```
% pen_fuse is the penalization parameter that is used when multiple ...
    frames of one sequence are aligned to a single frame of the ...
    other sequence
% pen_chop is the penalization parameter for chopping the starting ...
    or ending frame away for alignment
% pen_gap is the penalization parameter for jumps in alignment.
% unit_t is the unit time for solving the end duration in 'time' mode

%% Calculate the euroclidian distance matrix from all points of sm ...
    to ss
num_feat = size(sm,1);
lm = size(sm,2);
ls = size(ss,2);
d = zeros(lm,ls);
% calculate the distance without considering the nan, however ...
    normalize to full dimensions
for midx = 1:lm;
    for sidx = 1:ls;
        nan_idx = 1-isnan(sm(:,midx)+ss(:,sidx));
        d(midx,sidx) = ...
            sum((sm(nan_idx>0,midx)-ss(nan_idx>0,sidx)).^2)/ ...
            sum(nan_idx)*num_feat;
    end
end

%% Calculate the lowest cummulative cost of a path from (1,1) to ...
    (lm,ls)
% At the start, missing alignment is allowed and the cost for each ...
    missing frame is denpendent on the pen_chop and the frame position
% At the end, missing alignment is also allowed. The cost is ...
    calculated in the same way as the start
% Within the sequence, alignment gap up to 1 frame is allowed and ...
    will have a cost of pen_gap
% Non diagonal alignment will have a cost weighted in a ...
    time-dependent way (mode time)

D = zeros(size(d));
ID = D;
D(1,1) = d(1,1);
ID(1,1) = 0;
```

```
% First row and column
for midx = 2:lm;
    [D(midx,1),m] = ...
       min([d(midx-1,1)*(tm(midx)-tm(midx-1))+D(midx-1,1), ...
       tm(midx)*pen_chop(1)]);
    if m == 2;
        ID(midx,1) = 0;
    else
        ID(midx,1) = 2;
    end
end
for sidx = 2:ls;
    [D(1,sidx),m] = ...
       min([d(1,sidx-1)*(ts(sidx)-ts(sidx-1))+D(1,sidx-1), ...
       ts(sidx)*pen_chop(2)]);
    if m == 2;
        ID(1,sidx) = 0;
    else
        ID(1,sidx) = 3;
    end
end
% Second row and column
for midx = 2:lm;
    [D(midx,2),ID(midx,2)] = min([d(midx-1,1)*(tm(midx)-tm(midx-1) ...
       +ts(2)-ts(1))+D(midx-1,1), ...
       (d(midx-1,2)+pen_fuse(1))*(tm(midx)-tm(midx-1))+D(midx-1,2), ...
       (d(midx,1)+pen_fuse(2))*(ts(2)-ts(1))+D(midx,1)]);
end
for sidx = 2:ls;
    [D(2,sidx),ID(2,sidx)] = min([d(1,sidx-1)*(tm(2)-tm(1)+ts(sidx) ...
       -ts(sidx-1))+D(1,sidx-1), ...
       (d(1,sidx)+pen_fuse(1))*(tm(2)-tm(1))+D(1,sidx), ...
       (d(2,sidx-1)+pen_fuse(2))*(ts(sidx)-ts(sidx-1))+D(2,sidx-1)]);
end
% The remaining part of the matrix
for midx = 3:lm;
    for sidx = 3:ls;
        [D(midx,sidx),ID(midx,sidx)] = min([d(midx-1,sidx-1)* ...
           (tm(midx)-tm(midx-1)+ts(sidx)-ts(sidx-1))+D(midx-1,sidx-1), ...
```

```
            (d(midx-1,sidx)+pen_fuse(1))*(tm(midx)-tm(midx-1))+D(midx-1,sidx), ...
            (d(midx,sidx-1)+pen_fuse(2))*(ts(sidx)-ts(sidx-1))+D(midx,sidx-1), ...
            d(midx-2,sidx-1)*(tm(midx)-tm(midx-2)+ts(sidx)-ts(sidx-1)) ...
            +D(midx-2,sidx-1)+pen_gap(1)*(tm(midx)-tm(midx-2)), ...
            d(midx-1,sidx-2)*(tm(midx)-tm(midx-1)+ts(sidx)-ts(sidx-2)) ...
            +D(midx-1,sidx-2)+pen_gap(2)*(ts(sidx)-ts(sidx-2))]);
    end
end
% Last row and column
for midx = 2:lm;
D(midx,end) = ...
   D(midx,end)+d(midx,end)*unit_t+pen_chop(1)*(tm(end)-tm(midx));
end
for sidx = 2:ls;
D(end,sidx) = ...
   D(end,sidx)+d(end,sidx)*unit_t+pen_chop(2)*(ts(end)-ts(sidx));
end

%% Find the best solution where the last frame of at least one ...
   sequence is aligned
[v_t,end_t] = min(D(:,end));
[v_r,end_r] = min(D(end,:));

if v_t < v_r
    seq_idx = [zeros(lm-end_t,1);ls];
    mod_idx = (lm:-1:end_t)';
else
    seq_idx = (ls:-1:end_r)';
    mod_idx = [zeros(ls-end_r,1);lm];
end
TD = min(v_t,v_r);

%% Assign the alignment vector from the last to the first frame
w = [mod_idx,seq_idx];
seq_idx = w(end,2);
mod_idx = w(end,1);
idx = ID(mod_idx,seq_idx);
while idx > 0;
    switch idx
        case 1
```

```
            seq_idx = seq_idx-1;
            mod_idx = mod_idx-1;
        case 2
            mod_idx = mod_idx-1;
        case 3
            seq_idx = seq_idx-1;
        case 4
            w = cat(1,w,[mod_idx-1,0]);
            mod_idx = mod_idx-2;
            seq_idx = seq_idx-1;
        case 5
            w = cat(1,w,[0,seq_idx-1]);
            mod_idx = mod_idx-1;
            seq_idx = seq_idx-2;
    end
    w=cat(1,w,[mod_idx,seq_idx]);
    idx = ID(mod_idx,seq_idx);
end
if w(end,1) > 1;
    w = cat(1,w,[((w(end,1)-1):-1:1)' zeros(w(end,1)-1,1)]);
elseif w(end,2) > 1;
    w = cat(1,w,[zeros(w(end,2)-1,1) ((w(end,2)-1):-1:1)']);
end

% Arange the alignment list from start to end
w = w(end:-1:1,:);
end
```

Assign the alignment to a cell x frame matrix from the matching vector

```
function A = AssAligned(A,vecM,numcell,c2)

APast = A;
A = zeros(numcell,size(vecM,1));
for midx = 1:size(vecM,1);
    if vecM(midx,1) > 0;
        A(:,midx) = APast(:,vecM(midx,1));
    end
end
```

```
A(c2,:) = vecM(:,2)';
A(:,sum(A,1)==0) = [];
AS = A(:,1);
for atidx = 2:size(A,2);
    if ¬isequal(A(A(:,atidx)>0,atidx),A(A(:,atidx)>0,atidx-1));
        AS = cat(2,AS,A(:,atidx));
    end
end
A = AS;
end
```

Quality control: delete sequences with large (mean + std) distance to the model after alignment

```
function [AS,qc_seq] = qc_masterseq(Dist,A)

mean_TD = mean(Dist);
std_TD = std(Dist,[],1);
qc_seq = double(Dist≤(mean_TD+std_TD));
A(qc_seq==0,:) = 0;
AS = A(:,1);
for atidx = 2:size(A,2);
    if ¬isequal(A(A(:,atidx)>0,atidx),A(A(:,atidx)>0,atidx-1));
        AS = cat(2,AS,A(:,atidx));
    end
end

end
```

Calculate the averaged sequence and its timeline

```
function [F,T,SD] = GetMaster(featmat,alignmat)

F = zeros(size(featmat,1),size(alignmat,2));
S = zeros(size(F));
SD = S;
for atidx = 1:size(alignmat,2);
    af = zeros(size(featmat,1),sum(alignmat(:,atidx)>0));
    seq_idx = 0;
    for cidx = 1:size(alignmat,1);
        if alignmat(cidx,atidx) > 0;
```

```
            seq_idx = seq_idx + 1;
            af(:,seq_idx) = featmat(:,alignmat(cidx,atidx),cidx);
        end
    end
    actual_feat = sum((1-isnan(af)),2);
    calc_f = af;
    calc_f(isnan(calc_f)) = 0;
    F(:,atidx) = sum(calc_f,2)./actual_feat;
    calc_f = af-repmat(F(:,atidx),[1 size(af,2)]);
    calc_f(isnan(calc_f)) = 0;
    actual_feat(actual_feat<2) = 2;
    SD(:,atidx) = sqrt(sum((calc_f).^2,2)./(actual_feat-1));
end

T = alignmat(sum(alignmat,2)>0,:);
for cellidx = 1:size(T,1);
    for atidx = 2:size(alignmat,2);
        if T(cellidx,atidx) < T(cellidx,atidx-1);
            T(cellidx,atidx) = T(cellidx,atidx-1);
        end
    end
end
T = diff(cat(1,zeros(1,size(T,1)),T'));
T = mean(T,2);
for atidx = 2:length(T);
    T(atidx) = T(atidx-1) + T(atidx);
end

end
```

6.3.6 Objective determination of Mitotic Standard Transitions

Characteristic transitions were then identified in the mitotic standard time model using the function cluster_temporal_model. The function reduces the model to a temporal resolution of 15 seconds where the feature value for each time was linearly interpolated based on the neighboring two measurements. Then, transitions were found for the temporal model based on the derivative.

```
function time_cluster = cluster_temporal_model(model_dir)

%% Parameters
```

```
red_dt = (15/60/1.5); % resolution at 15 seconds
dt = 3; % 4.5 minutes, window size for derivative calculation
nosplit = 8; % 12 minutes, longest duration for one phase between ...
    two transitions
toptr = 0.7; % Threshold for selecting significant peaks as transitions
minsplit = 1; % 1.5 minutes, the minimum temporal duration for one ...
    phase between two transitions

%% Reduce/equalize the model's temporal resolution
tt = timeline(1):red_dt:feature_parameters.frame_num;
ms2 = zeros(size(master_seq,1),length(tt));
st2 = ms2;
for i = 1:length(tt);
    br = sum(timeline<=tt(i));
    dt1 = tt(i)-timeline(br);
    if br < length(timeline);
        dt2 = timeline(br+1)-tt(i);
        ms2(:,i) = master_seq(:,br) + ...
            (master_seq(:,br+1)-master_seq(:,br))*(dt1/(dt1+dt2));
        st2(:,i) = StD(:,br) + (StD(:,br+1)-StD(:,br))*(dt1/(dt1+dt2));
    else
        ms2(:,i) = master_seq(:,br);
        st2(:,i) = StD(:,br);
    end
end
align_original_model = struct('master_seq',master_seq,'StD',StD, ...
    'timeline',timeline);
master_seq = ms2;
StD = st2;
timeline = tt;

%% Calculate the approximation of the second derivative
frame_start = sum(timeline<=(dt+timeline(1)))+1;
frame_end = sum(timeline<=(timeline(end)-dt));
add_start = frame_start-1;
add_end = size(master_seq,2)-frame_end;
frame_end = length(timeline)+add_start;
master_seq = cat(2,repmat(master_seq(:,1),[1 add_start]),master_seq);
master_seq = cat(2,master_seq,repmat(master_seq(:,end),[1 add_end]));
StD = cat(2,repmat(StD(:,1),[1 add_start]),StD);
```

```
StD = cat(2,StD,repmat(StD(:,end),[1 add_end]));
timeline = [-red_dt*(add_start:-1:1) timeline 40+red_dt*(1:add_end)];
tangentseq = zeros(size(master_seq,2),2+size(master_seq,1));
tangentseq(:,1) = (1:size(master_seq,2))';
for i = frame_start:frame_end;
    i_minus = sum(timeline<=(timeline(i)-dt));
    i_plus = sum(timeline<(timeline(i)+dt))+1;
    if i_minus == i;
        i_minus = i-1;
    end
    if i_plus == i;
        i_plus = i+1;
    end
    tangentseq(i,3:end) = abs(mean(master_seq(:,i_minus:i-1)- ...
       repmat(master_seq(:,i),[1 ...
       i-i_minus]),2)+mean(master_seq(:,i+1:i_plus) ...
       -repmat(master_seq(:,i),[1 i_plus-i]),2));
    tangentseq(i,2) = sum(((master_seq(:,i-1)+StD(:,i-1)) ...
       <(master_seq(:,i)-StD(:,i)))+((master_seq(:,i-1)-StD(:,i-1)) ...
       >(master_seq(:,i)+StD(:,i))));
end

%% Find transitions by selecting the peaks beyond the threshold
breakpoints = double(tangentseq(:,2) > 0);
tangentseq_original = tangentseq;
break_max_single = zeros(length(breakpoints),size(tangentseq,2)-2);
for i = 3:size(tangentseq,2);
    val = 10;
    while val > toptr;
        [val,idx] = max(tangentseq(:,i));
        if val > toptr;
            breakpoints(idx) = 1;
            break_max_single(idx,i-2) = 1;
            i_minus = sum(timeline<=(timeline(idx)-dt));
            i_plus = sum(timeline<(timeline(idx)+dt))+1;
            if i_minus == idx;
                i_minus = idx-1;
            end
            if i_plus == idx;
                i_plus = idx+1;
```

```
            end
            [¬,idx_min] = min(tangentseq((idx-1):-1:i_minus,i));
            tangentseq((idx-idx_min):idx,i) = 0;
            [¬,idx_min] = min(tangentseq((idx+1):1:i_plus,i));
            tangentseq(idx:(idx+idx_min),i) = 0;
        end
    end
end
tangentseq = tangentseq_original;
th_tan = toptr*ones(1,size(tangentseq,2)-2);
th_tan = (th_tan-mean(tangentseq(frame_start:frame_end,3:end),1)) ...
    ./std(tangentseq(frame_start:frame_end,3:end),[],1);
tangentseq(frame_start:frame_end,3:end) = ...
    (tangentseq(frame_start:frame_end,3:end)-repmat(mean( ...
    tangentseq(frame_start:frame_end,3:end),1), ...
    [frame_end-frame_start+1,1]))./  ...
    repmat(std(tangentseq(frame_start:frame_end,3:end), [],1), ...
    [frame_end-frame_start+1,1]);
frame_end0 = frame_end;

%% If the time between two transitions is too long, find additional ...
    transitions which have the maximum second derivative in one ...
    feature dimension through all points within the corresponding ...
    duration
breakpoints(frame_start) = 1;
for i = 1:sum(timeline≤timeline(end)-nosplit);
     frame_end = sum(timeline≤(timeline(i)+nosplit));
     if sum(breakpoints(i:frame_end)) == 0;
         [¬,idx] = max(max(tangentseq(i:frame_end,3:end),[],2),[],1);
         breakpoints(idx+i-1) = 1;
     end
end

%% If the time between two transitions is too short, delete the ...
    transitions with smaller second derivatives
fnum = (size(tangentseq,2)-2)/2 + 2;
i = 1;
while i ≤ sum(timeline≤timeline(end)-minsplit);
     frame_end = sum(timeline≤(timeline(i)+minsplit));
     if sum(breakpoints(i:frame_end)) > 1;
```

```
            s = sum(tangentseq(i:frame_end,3:fnum),2);
            s(breakpoints(i:frame_end)==0) = -inf;
            [¬,idx] = max(s,[],1);
            breakpoints(i:frame_end) = 0;
            breakpoints(idx+i-1) = 1;
            i = idx+i;
        else
            i = i+1;
        end
end

%% Update all sequences
breakpoints(frame_start) = 1;
master_seq = master_seq(:,frame_start:frame_end0);
StD = StD(:,frame_start:frame_end0);
timeline = timeline(frame_start:frame_end0);
tangentseq = tangentseq(frame_start:frame_end0,:);
breakpoints = breakpoints(frame_start:frame_end0);


%% Write output
class_start = find(breakpoints==1);
class_label = (1:length(class_start))';
class_end = cat(1,class_start(2:end)-1,length(breakpoints));
class_mean = zeros(length(class_start),size(master_seq,1));
class_creteria = zeros(length(class_start),2);
for i = 1:length(class_start);
    class_mean(i,:) = ...
        (mean(master_seq(:,class_start(i):class_end(i)),2))';
    if tangentseq(class_start(i),2) == 0;
        [class_creteria(i,2),class_creteria(i,1)] = max(tangentseq( ...
            class_start(i),3:end),[],2);
    end
end

cluster_parameter = struct('model_t_resolution',red_dt,'dt',dt, ...
    'longest_cluster',nosplit,'taketoptransition',toptr, ...
    'mingap_bettransition',minsplit);
time_cluster = ...
    table(class_label,class_start,class_end,class_mean,class_creteria);
```

```
save(model_file,'feature_parameters','feat_normalized','feat_derivative', ...
    'feat_filelist','master_seq','aligned_full','timeline','dev_track', ...
    'StD','align_TD','qc_idx','time_cluster','tangentseq', ...
    'breakpoints','cluster_parameter','align_original_model');

end
```

In order to visualize the standard mitotic sequence, representative cells were selected for each stage between two transitions.

```
function select_master_mitosis
F = [feat_normalized;feat_derivative];
class_number = size(time_cluster,1);

cellname_cl = {};
celltime_cl = [];
for i = 1:class_number;
allcells = [];
allidx = [];
subalign = aligned_full(:,(sum(align_original_model.timeline < ...
    timeline(time_cluster.class_start(i)))+1): ...
    sum(align_original_model.timeline ≤ ...
    timeline(time_cluster.class_end(i))));
if isempty(subalign)
subalign = aligned_full(:,(sum(align_original_model.timeline < ...
    timeline(time_cluster.class_start(i-1)))+1): ...
    sum(align_original_model.timeline ≤ ...
    timeline(time_cluster.class_end(i+1))));
end
for j = 1:size(subalign,1);
     subframe = unique(subalign(j,:));
     subframe(subframe==0) = [];
     if ¬isempty(subframe)
         allidx = cat(2,allidx,cat(1,repmat(j,[1 ...
             length(subframe)]),subframe));
         for k = 1:length(subframe);
              allcells = cat(2,allcells,F(:,subframe(k),j));
         end
     end
end
```

```
dist_classmean = ...
    sum((allcells-repmat((time_cluster.class_mean(i,:))', [1 ...
    size(allcells,2)])).^2,1);
[¬,clidx] = min(dist_classmean);
cellpath_cl = strsplit(feat_filelist{allidx(1,clidx)},{'Temporal'}, ...
    'CollapseDelimiters',true);
cellpath_cl = cellpath_cl{1}(1:end-1);
addname = ['cluster' num2str(i,'%02.f')];
cellname_cl = cat(1,cellname_cl,cellpath_cl);
celltime_cl = cat(1,celltime_cl,allidx(2,clidx));
end
savefile_cl = fullfile(model_dir,'selected_cell.mat');
save(savefile_cl,'cellname_cl','celltime_cl');
end
```

6.3.7 Temporal registration of all image sequences in the data

All cells were then registered temporally to the mitotic standard time model.

```
function temporal_annotation(exp_dir, model_dir)
modelpath = fullfile(model_dir,'temporal_alignment.mat');
load(modelpath);
pp = feature_parameters;

% Set the parameters such that all frames in the cell sequence ...
    should be practically aligned
penat_fuse = [0;360];
penat_chop = [15;540];
penat_gap = [36;540];
unit_t = 0.3;
th_qc = 3*dev_track(end,2); % threshold for the quality control ...
    whether a cell has a successful alignment
feature_number = sum(pp.select);

% Load the database file
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
summary_database.time_alignment = -ones(size( ...
    summary_database.time_alignment));
```

```
feat_filelist = ...
    summary_database.filepath(summary_database.lm_features>0);
cell_total = length(feat_filelist);

%% Align cell by cell
vecQC = zeros(cell_total,1);
for align_idx = 1:cell_total;
% Load the feature matrix for selected sequence
matfile = fullfile(feat_filelist{align_idx},'Temporal_Align', ...
    'LM_feature_seq.mat');
savefile = fullfile(feat_filelist{align_idx},'Temporal_Align', ...
    'Align_annotation.mat');
disp(feat_filelist{align_idx})
% If the result file exists, check whether the sequence was aligned ...
    to the same model
if exist(savefile,'file');
M = load(savefile);
if isequal(M.modelpath,modelpath);
vecQC(align_idx) = 1;
end
end
if vecQC(align_idx) ~= 1;
disp(['Process' fullfile(feat_filelist{align_idx}, 'Temporal align ...
    to the model')])
if exist(matfile,'file');
load(matfile);
if size(LM_feats,2) == pp.frame_num && LM_feats(1,end) > 0;
feat_original = LM_feats(pp.select>0,:);

% Normalize all features using mean and std saved during modeling
feat_normalized = (feat_original-repmat(pp.mean_0_deriv, ...
    [1,pp.frame_num])) ./repmat(pp.std_0_deriv,[1,pp.frame_num]);

%% Calculate the first derivative based on filtered data and normalize
% piecewise smoothing fit and derivative from the fit
t_aq = 1:pp.frame_num;
feat_derivative = zeros(size(feat_normalized));
for fdx = 1:feature_number;
    [smooth_seq,stat] = fit(t_aq',feat_normalized(fdx,:)', ...
       'smoothingspline');
```

```
    feat_derivative(fdx,:) = (differentiate(smooth_seq,t_aq'))';
end

% Normalize all derivative features using mean and std saved during ...
   the modeling
feat_derivative = (feat_derivative-repmat(pp.mean_1_deriv, ...
   [1,pp.frame_num]))./ repmat(pp.std_1_deriv,[1,pp.frame_num]);

%% Align the sequence using the function multidim_DDTW
align_seq = [feat_normalized;feat_derivative];
[align_TD,matching_vector]=multidim_DDTW(master_seq,timeline, ...
   align_seq,t_aq,pp.mode,penat_fuse,penat_chop, penat_gap,pp.unit_t);
% Quality control of alignment
if align_TD > th_qc;
    vecQC(align_idx) = 0;
else
    % Assign the alignment. If a frame in the sequence is
    % aligned to multiple frames of the model, the model
    % frame with the shortest distance to it will be used
    % for the assignment
    mitotime = zeros(pp.frame_num,2);
    for aidx = 1:pp.frame_num;
        alignframe = find(matching_vector(:,2)==aidx);
        modelframe = master_seq(:,matching_vector(alignframe(1),1));
        if length(alignframe) > 1;
            for fidx = 2:length(alignframe);
                modelframe = cat(2,modelframe,master_seq(:, ...
                   matching_vector(alignframe(fidx),1)));
            end
        end
        framedist = (modelframe-repmat(align_seq(:,aidx),[1, ...
           size(modelframe,2)])).^2;
        framedist(isnan(framedist)) = 0;
        framedist = sum(framedist,1);
        [¬,frame] = min(framedist);
        mitotime(aidx,:) = [matching_vector(alignframe(frame),1) ...
           timeline(matching_vector(alignframe(frame),1))];
    end
    vecQC(align_idx) = 1;
```

```
    save(savefile,'align_seq','mitotime','pen_fact','th_qc', ...
        'modelpath','align_TD')
end
end
end
end
end

% Update the database file
summary_database.time_alignment(summary_database.lm_features>0) = ...
    vec_QC;
writetable(summary_database,exp_database,'Delimiter','\t');

end
```

6.3.8 Calibration and image processing of the POI channel

Then, the POI channel image was processed into a 3D density map by calibration, smoothing and rotating for particular processing steps later.

```
function vec_QC = preprocess(exp_dir)

%% Interactive selection of experiment directory
if ¬nargin
exp_dir = uigetdir('','Select the directory with your data series');
end

%% Set up parameters
filter_size = [3 3 1]; % 3D gaussian filter size
xy_resolution = 0.25; % in um
z_resolution = 0.75; % in um
z_resolution = z_resolution/xy_resolution;
tmax = 40; % number of frames

%% Load the database file and only process files with positive QC ...
    for segmentation and FCS calibration
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
```

```
cell_dirlist = ...
    summary_database.filepath(((summary_database.segmentation>0) ...
    .*(summary_database.fcs_calibration>0))>0);
cell_total = length(cell_dirlist);
vec_QC = ...
    summary_database.registration(((summary_database.segmentation>0) ...
    .*(summary_database.fcs_calibration>0))>0);
summary_database.registration = ...
    -ones(size(summary_database.registration));

%% Load the mat file from the segmentation
for cdx = 1:cell_total;
cellseq_list = ...
    dir(fullfile(cell_dirlist{cdx},'Preprocessing','Segmentation', ...
    '*T0*.mat'));
if length(cellseq_list) == tmax;
% Make the direcory for saving processed data
registration_dir = ...
    fullfile(cell_dirlist{cdx},'Preprocessing','Registration');
if ¬exist(registration_dir,'dir')
mkdir(registration_dir);
end
% If the image sequence has been processed, then skip
if exist(fullfile(registration_dir,cellseq_list(tmax).name),'file');
vec_QC(cdx) = 1;
end
% If not, then processing
if vec_QC(cdx) ≠ 1;
% Load the calibration file
calibration_file = ...
    fullfile(cell_dirlist{cdx},'Preprocessing','calibration.mat');
load(calibration_file);
% Process frame by frame
for tdx = 1:length(cellseq_list);
savefile = fullfile(registration_dir,cellseq_list(tdx).name);
if ¬exist(savefile,'file')
    disp(['Process' cell_dirlist{cdx} 'at ' num2str(tdx) ' ...
       Registration'])
    % load the result from the segmentation pipeline
```

```
load(fullfile(cell_dirlist{cdx},'Preprocessing','Segmentation', ...
    cellseq_list(tdx).name));
%% Pre-process the protein of interest channel
% Calibrate the POI image
proc_poi = poi*calibration_factor;
% Filter the POI image and the landmarks image
proc_poi = smooth3(proc_poi,'gaussian',filter_size);
proc_nuc = smooth3(nuc,'gaussian',filter_size);
% Substract the background which is the mean intensity of the ...
    cell-free volume.
background_488 = sum(proc_poi(:).*bgMask(:))/sum(bgMask(:)>0);
proc_poi = proc_poi - background_488;
proc_poi(proc_poi<0) = 0;
% Crop POI within the cell based on the segmentation result
proc_poi = proc_poi.*cellVolume;
% Crop the landmarks image using the segmented nuclear volume
proc_nuc = proc_nuc.*(nucVolume>0);
%% Calculating the rotational angle based on intersetion ...
    axis-cell locations
rotation_xy = ...
    180*atan2(isectPointsPix(1,2)-isectPointsPix(2,2), ...
    isectPointsPix(1,1) -isectPointsPix(2,1))/pi;
% Caculating the center of rotation, i.e. the center of ...
    chromatin volume
nucdata = bwconncomp(nucVolume>0);
size_nuc = cellfun(@numel,nucdata.PixelIdxList);
nucnumber = length(size_nuc);
centronuc = regionprops(nucdata,'Centroid');
if nucnumber == 1;
     nuccent = centronuc(1).Centroid;
else % only consider the biggest two parts of the nucleus, i.e. ...
    two divided doughters
     [val1,nucidx1] = max(size_nuc);
     size_nuc(nucidx1) = 0;
     [val2,nucidx2] = max(size_nuc);
     nuccent = (centronuc(nucidx1).Centroid*val1 + ...
         centronuc(nucidx2).Centroid*val2)/(val1+val2);
end
if length(nuccent) < 3;
     nuccent = cat(2,nuccent,1);
```

```
end
nuccent = round([nuccent(2) nuccent(1) nuccent(3)]);
% Rotational registration of the protein of interest channel ...
   using the function yc_imrotate3d
[reg_poi,¬,¬] = yc_imrotate3d(proc_poi,-rotation_xy,nuccent,[]);
%% Calculate for each POI pixel the distance to the nuclear and ...
   cellular boundary with further geometrical informations
imgsize = size(poi);
celldistmap2d = zeros(size(cellVolume));
for i = 1:imgsize(3);
    celldistmap2d(:,:,i) = bwdist(bwperim(cellVolume(:,:,i),4));
end
chr_full = imfill(nucVolume,'holes');
nucdistmap2d = zeros(size(chr_full));
for i = 1:imgsize(3);
    nucdistmap2d(:,:,i) = bwdist(bwperim(chr_full(:,:,i),4));
end
celldistmap3d = zeros(size(celldistmap2d));
nucdistmap3d = zeros(size(nucdistmap2d));
if imgsize(3)>1;
    for i = 1:imgsize(3);
        distvec = (z_resolution*abs((1:imgsize(3))-i)).^2;
        distvec = permute(repmat(distvec,[imgsize(2) 1 ...
            imgsize(1)]),[3 1 2]);
        celldistmap3d(:,:,i) = ...
            sqrt(min(celldistmap2d.^2+distvec,[],3));
        nucdistmap3d(:,:,i) = ...
            sqrt(min(nucdistmap2d.^2+distvec,[],3));
    end
else
    celldistmap3d = celldistmap2d;
    nucdistmap3d = nucdistmap2d;
end
nucdistmap3d(chr_full>0) = -nucdistmap3d(chr_full>0);
clear celldistmap2d;
clear nucdistmap2d;
clear distvec;
lm = prod(imgsize)-1;
trans_matrix = [(0:lm)' cellVolume(:) celldistmap3d(:) ...
   nucdistmap3d(:) proc_poi(:)];
```

```
    clear chr_full;
    clear celldistmap3d;
    clear nucdistmap3d;
    trans_matrix(trans_matrix(:,2)==0,:) = []; % only keep the data ...
        inside of the cell
    trans_matrix(:,2) = sum(abs(trans_matrix(:,3:4)),2);
    trans_matrix(trans_matrix(:,2)==0,:) = []; % delete totdist=0
    coords = zeros(size(trans_matrix,1),3);
    coords(:,3) = floor(trans_matrix(:,1)/imgsize(1)/imgsize(2))+1;
    trans_matrix(:,1) = mod(trans_matrix(:,1),imgsize(1)*imgsize(2));
    coords(:,2) = floor(trans_matrix(:,1)/imgsize(1))+1;
    coords(:,1) = mod(trans_matrix(:,1),imgsize(1))+1;
    normdists = trans_matrix(:,4)./trans_matrix(:,2); % dist to ...
        nuc/(abs(dist to nuc)+abs(dist to cell bound))

    angles = zeros(size(coords,1),2);
    angles(:,1) = ...
        180*(atan2(coords(:,2)-nuccent(2),coords(:,1)-nuccent(1)) - ...
        atan2(isectPointsPix(1,2) - nuccent(2),isectPointsPix(1,1) - ...
        nuccent(1)))/pi;
    angles(:,2) = ...
        180*(atan2(coords(:,3)-nuccent(3),sqrt((coords(:,2)-nuccent(2)).^2 ...
        + (coords(:,1)-nuccent(1)).^2)) - ...
        atan2(isectPointsPix(1,3)-nuccent(3),sqrt((isectPointsPix(1,2) ...
        - nuccent(2)).^2 + (isectPointsPix(1,1)-nuccent(1)).^2)))/pi;
    dist_to_nuc = trans_matrix(:,4);
    dist_to_pm = trans_matrix(:,3);
    intensities = trans_matrix(:,5);
    %% Saving and updating
    save(savefile,'proc_nuc','proc_poi','reg_poi','background_488', ...
        'normdists','dist_to_nuc', 'dist_to_pm','intensities', ...
        'coords','angles');
end
end
% QC criteria: if all 40 frames exist and can be processed
if exist(fullfile(registration_dir,cellseq_list(tmax).name),'file');
vec_QC(cdx) = 1;
else
vec_QC(cdx) = 0;
end
```

```
end
end
end
summary_database.registration(((summary_database.segmentation>0).* ...
    (summary_database.fcs_calibration>0))>0) = vec_QC;
writetable(summary_database,exp_database,'Delimiter','\t');
end
```

Using the following function, images can be rotated counterclockwise for "angle" degree with "center_input" as fixed point in xy plane. In the output image, "center_input" will located on the position "center_output". The output image will have the same size as the input image. If the "center_output" is not given the function will try to find the best location where the signal is minimum cut out and image as close to the center as possible. The image will be rotated in x-y but not in z.

```
function [Rotated newcenter center_input] = ...
    yc_imrotate3d(A,angle,center_input,center_output)

imagesize = size(A);
if length(imagesize) < 3;
    imagesize = [imagesize 1];
end

% Extend the image so that the center_input is located at the image ...
    center
if center_input(1) < (imagesize(1)+1)/2;
    num_addrow = imagesize(1)-2*center_input(1)+1;
    A = [zeros(num_addrow,imagesize(2),imagesize(3));A];
    center_input(1) = center_input(1)+num_addrow;
else
    num_addrow = -imagesize(1)+2*center_input(1)-1;
    A = [A;zeros(num_addrow,imagesize(2),imagesize(3))];
end

temp_imagesize = size(A);
if length(temp_imagesize)<3;
    temp_imagesize = [temp_imagesize 1];
end
if center_input(2) < (imagesize(2)+1)/2;
```

```
    num_addcol = imagesize(2)-2*center_input(2)+1;
    A = [zeros(temp_imagesize(1),num_addcol,temp_imagesize(3)) A];
    center_input(2) = center_input(2)+num_addcol;
else
    num_addcol = -imagesize(2)+2*center_input(2)-1;
    A = [A zeros(temp_imagesize(1),num_addcol,temp_imagesize(3))];
end

% Rotate the image using imrotate

Rot_A = imrotate(A,angle,'bilinear','loose');
temp_imagesize = size(Rot_A);
if length(temp_imagesize)<3;
    temp_imagesize = [temp_imagesize 1];
end
temp_cent = round((temp_imagesize+1)/2);

% In the case that center_output is given, move the center of the ...
   image to the center_output and crop the image into the size of ...
   the original image

if length(center_output) == 3;
    Rot_A_org = sum(sum(sum(Rot_A)));
    topdelete = temp_cent(1)-center_output(1);
    bottemdelete = ...
       temp_imagesize(1)-temp_cent(1)-imagesize(1)+center_output(1);
    leftdelete = temp_cent(2)-center_output(2);
    rightdelete = ...
       temp_imagesize(2)-temp_cent(2)-imagesize(2)+center_output(2);
    if topdelete ≥ 0;
        Rot_A(1:topdelete,:,:) = [];
    else
        Rot_A = ...
           [zeros(abs(topdelete),temp_imagesize(2),temp_imagesize(3));Rot_A];
    end
    center_input(1) = center_input(1)-topdelete;
    if bottemdelete ≥ 0;
        Rot_A(end-bottemdelete+1:end,:,:) = [];
    else
```

```
        Rot_A = [Rot_A;zeros(abs(bottemdelete),temp_imagesize(2), ...
            temp_imagesize(3))];
    end
    if leftdelete ≥ 0;
        Rot_A(:,1:leftdelete,:) = [];
    else
        Rot_A = [zeros(imagesize(1),abs(leftdelete),imagesize(3)) ...
            Rot_A];
    end
    center_input(2) = center_input(2)-leftdelete;
    if rightdelete ≥ 0;
        Rot_A(:,end-rightdelete+1:end,:) = [];
    else
        Rot_A = [Rot_A ...
            zeros(imagesize(1),abs(rightdelete),imagesize(3))];
    end
    if sum(sum(sum(Rot_A)))≠Rot_A_org;
        warning('rotated image extended the size of the original ...
            image')
    end
    if center_output(3)≠center_input(3);
        error('the center of input and output has to be the same in ...
            z-direction')
    end
elseif isempty(center_output); % Automatic searching of best center
    center_output = [0 0 center_input(3)];
    Rot_A_mid = Rot_A(:,:,center_input(3));
    signalrows = find(sum(Rot_A_mid,2)>0);
    signalrows = [signalrows(1) signalrows(end)];
    num_rows = signalrows(2)-signalrows(1)+1;
    signalcols = find(sum(Rot_A_mid,1)>0);
    signalcols = [signalcols(1) signalcols(end)];
    num_cols = signalcols(2) - signalcols(1)+1;
    if num_rows ≤ imagesize(1);
        addrows_top = round((imagesize(1)-num_rows)/2);
        addrows_buttom = imagesize(1)-num_rows-addrows_top;
        row1 = signalrows(1)-addrows_top;
        rowend = signalrows(2)+addrows_buttom;
        if row1≥1 && rowend ≤ temp_imagesize(1);
            Rot_A = Rot_A(row1:rowend,:,:);
```

```
            center_input(1) = center_input(1)-row1+1;
        elseif row1 < 1;
            Rot_A = Rot_A(1:imagesize(1),:,:);
        elseif rowend > temp_imagesize(1);
            Rot_A = Rot_A(end-imagesize(1)+1:end,:,:);
            center_input(1) = ...
                center_input(1)-size(Rot_A,1)+imagesize(1);
        end
        center_output(1) = temp_cent(1)-signalrows(1)+addrows_top+1;
    else
        warning('rotated image extended the size of the original ...
            image')
        delrows_top = round((num_rows-imagesize(1))/2);
        delrows_bottum = num_rows-delrows_top-imagesize(1);
        Rot_A = Rot_A(signalrows(1)+delrows_top:signalrows(2)- ...
            delrows_bottum,:,:);
        center_output(1) = temp_cent(1)-signalrows(1)-delrows_top+1;
    end
    if num_cols ≤ imagesize(2);
        addcols_top = round((imagesize(2)-num_cols)/2);
        addcols_buttom = imagesize(2)-num_cols-addcols_top;
        col1 = signalcols(1)-addcols_top;
        colend = signalcols(2)+addcols_buttom;
        if col1≥1 && colend ≤ temp_imagesize(2);
            Rot_A = Rot_A(:,col1:colend,:);
            center_input(2) = center_input(2)-col1+1;
        elseif col1<1
            Rot_A = Rot_A(:,1:imagesize(2),:);
        elseif colend > temp_imagesize(2);
            Rot_A = Rot_A(:,end-imagesize(2)+1:end,:);
            center_input(2) = ...
                center_input(2)-size(Rot_A,2)+imagesize(2);
        end
        center_output(2) = temp_cent(2)-signalcols(1)+addcols_top+1;
    else
        warning('rotated image extended the size of the original ...
            image')
        delcols_top = round((num_cols-imagesize(2))/2);
        delcols_bottum = num_cols-delcols_top-imagesize(2);
```

```
        Rot_A = Rot_A(:,signalcols(1)+delcols_top:signalcols(2)- ...
            delcols_bottum,:);
        center_output(2) = temp_cent(2)-signalcols(1)-delcols_top+1;
    end

elseif ¬isempty(center_output);
    error('center_output must be empty or a 3-dimension coordinate')
end

Rotated = Rot_A;
newcenter = center_output;

end
```

6.3.9 Dissect the POI image into a series of interest points

The next step is to extract features from each image. In this step, the features can be sorted into two categories. The first are the global features describing the global distribution of the protein. The second are local features in the form as SURF interest points. Each interest point was then described using a feature vector. The features need to be further processed into different interest point clusters. Together with global features are saved in the feature folder.

```
function poi_feature_extraction_surf(exp_dir)

%% Interactive selection of experiment directory

if ¬nargin
    exp_dir = uigetdir('','Select the directory with your data ...
       series');
end

%% Set up parameters
tmax = 40;
parameter_thbw = 0.48; % Parameter for thresholding the BW image: ...
   factor multiplied on the intensity level for foreground found ...
   using otsu method with the default set up
surf_th = 100; % Surf IP extraction: metric selection threshold
factor = 4; % Surf IP extraction: resize image factor such that the ...
   Surf IP size covers both kinetochores (small) and chromatin (large)
```

```
bitnum = 16; % number of image bit
ip_overlap = 0.4; % threshold on intensity fraction that two IP can ...
    share
sI_a = 5; % 2D soft spin_image parameter, weighing the distance to ...
    measuremnet, the bigger the more bloring
sI_b = 0.05;  % 2D soft spin_image parameter, keep the weight ...
    between distance and intensity
d_level = [0 9 18 27 36]; % spin_image, selected levels for ...
    distance counting, as pixels after normalization
i_level = [0 0.1 0.2 0.3 0.4 0.5]; % spin_image, selected levels ...
    for intensity, normalized to the saturation intensity
randref = ...
    [0.111,0.032,0.016,0.013,0.015,0.031,0.110,0.112,0.111,0.449, ...
    0.124,0.126,0.122,0.124,0.126,0.129,0.126,0.123]; % uLBP value ...
    for random distributed intensity, estimated from multiple ...
    simulations
maxvalue = 1.156; % simulated threshold for defining random texture ...
    for the summarized LBP
min_sig = 0.05; % minimum intensity for being a LBP center
% Define the weight between different feature categories such that ...
    the clustering later does not prefer one categrory of features.
weight_spinImage = 0.2;
weight_LBP = 0.5;
weight_corr = 1;
weight_pos = 1;
feature_parameters = struct('bwth',parameter_thbw,'surfth',surf_th, ...
    'factor',factor,'bit',bitnum,'ip_overlap',ip_overlap,'spinImage', ...
    struct('dI_weight',[sI_a sI_b],'ds',d_level,'is',i_level), ...
    'LBP',struct('randref',randref,'maxdist',maxvalue,'minsig',min_sig));
weight_parameters = struct('global',1,'spinImage',weight_spinImage, ...
    'LBP',weight_LBP,'corr',weight_corr,'pos',weight_pos);

%% Load the database file
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
summary_database.feature_extract = ...
    -ones(size(summary_database.feature_extract));
cell_dirlist = ...
    summary_database.filepath(((summary_database.registration>0).* ...
    (summary_database.time_alignment>0))>0);
```

```
cell_total = length(cell_dirlist);
vec_QC = zeros(cell_total,1);

%% Process cell by cell
for cdx = 1:cell_total;
cellseq_list = dir(fullfile(cell_dirlist{cdx},'Preprocessing', ...
    'Registration','*T0*.mat'));
if length(cellseq_list) == 40;
% Generate the folder for saving the output
feature_dir = fullfile(cell_dirlist{cdx},'Features_words');
if ¬exist(feature_dir,'dir')
mkdir(feature_dir);
end
if exist(fullfile(feature_dir,cellseq_list(tmax).name),'file');
vec_QC(cdx) = 1;
end
if vec_QC(cdx) ≠ 1;
% load the calibration file
calibration_file = fullfile(cell_dirlist{cdx},'Preprocessing', ...
    'calibration.mat');
load(calibration_file);
for tdx = 1:length(cellseq_list);
savefile = fullfile(feature_dir,cellseq_list(tdx).name);
if ¬exist(savefile,'file')
disp(['Process' cell_dirlist{cdx} ' at ' num2str(tdx) ' Surf ...
    feature extraction'])
load(fullfile(cell_dirlist{cdx},'Preprocessing','Segmentation', ...
    cellseq_list(tdx).name)); % load the result from the ...
    segmentation pipeline
load(fullfile(cell_dirlist{cdx},'Preprocessing','Registration', ...
    cellseq_list(tdx).name)); % load the result from the ...
    registration pipeline
nucVolume = double(nucVolume>0);
POI_Feats = [];
feature_name = table({},{},[],'VariableNames',{'name' 'category' ...
    'level'});
registered = [];
% Calculate the total intensity
tot_intensity = sum(intensities);
if tot_intensity > 0;
```

```
%% Determine foreground and background
th_bw = parameter_thbw*max(proc_poi(:))*graythresh(proc_poi);
bw_proc_poi = proc_poi>th_bw;
poi_foreground = proc_poi.*bw_proc_poi;
% Calculate the fraction of protein in the foreground for the 3D ...
    image and the 2D maximum projection
frac_foreground = sum(poi_foreground(:))/sum(proc_poi(:));
[maxProj_low,mPidx] = max(proc_poi,[],3);
poi_foreground = max(poi_foreground,[],3);
frac_foreground2d = sum(poi_foreground(:))/sum(maxProj_low(:));
%% Global feature extraction
% 1. fraction of protein over threshold in the nuclear volume
POI_Feats = cat(1,POI_Feats,sum(intensities(((normdists≤0).*( ...
    intensities>th_bw))>0))/sum(intensities(intensities>th_bw)));
registered = cat(1,registered,0);
feature_name = cat(1,feature_name,table({'%poi in ...
    nuc'},{'geometry'},0,'VariableNames',{'name' 'category' 'level'}));
% 2. fraction of protein in the predicted spindle volume which is ...
    the convex hull volume of the nuclear volume and the two ...
    intersession points of the predicted division axis to the ...
    cellular boundary
spindle_volume = nucVolume;
spindle_volume(isectPointsPix(1,1),isectPointsPix(1,2), ...
    sum(sum(nucVolume))>0) = 1;
spindle_volume(isectPointsPix(2,1),isectPointsPix(2,2), ...
    sum(sum(nucVolume))>0) = 1;
for sp_idx = 1:size(spindle_volume,3);
    spindle_volume(:,:,sp_idx) = ...
        bwconvhull(spindle_volume(:,:,sp_idx),'union',4);
end
spindle_volume = spindle_volume.*cellVolume;
outOFsp = cellVolume - spindle_volume;
spindle_volume = spindle_volume - imfill(nucVolume,'holes');
sp_acc = ...
    mean(proc_poi(spindle_volume(:)>0))/(mean(proc_poi(outOFsp(:)>0)) ...
    +mean(proc_poi(spindle_volume(:)>0)));
if isnan(sp_acc);
    sp_acc = 0;
end
POI_Feats = cat(1,POI_Feats,sp_acc);
```

```
registered = cat(1,registered,0);
feature_name = cat(1,feature_name,table({'%poi in ...
    spindle'},{'geometry'},0,'VariableNames',{'name' 'category' ...
    'level'}));

%% Extract Interest Points in 2D maximum projected image
satInt = (2^bitnum-1)*calibration_factor - background_488; % The ...
    calibrated value at the detector's saturation
maxProj_low = maxProj_low/satInt; % normalize the projected image
% Project the mCherry channel, croped with the nuclear volume, ...
    using the pixels taken in the maximum projection of the protein ...
    channel
nuc = nuc.*nucVolume;
nucProj = zeros(size(maxProj_low));
for i = 1:size(nucProj,1);
    for j = 1:size(nucProj,2);
        nucProj(i,j) = nuc(i,j,mPidx(i,j))/satInt;
    end
end
% Dilate the prediced midplane to get a volume
for zdx = 1:size(midPlane,3);
    midPlane(:,:,zdx) = imdilate(midPlane(:,:,zdx),ones(8));
end
% Resize the project poi channel such that the default SURF scales ...
    match the biological purpose
maxProj = imresize(maxProj_low,factor);
% Extract SURF interest points beyond the metric threshold set before
InterestPoints = detectSURFFeatures(maxProj,'MetricThreshold',surf_th);
%% Interest Points selection: each pixel in the image is included ...
    in multiple IPs. The selection tries to find a subset of the IPs ...
    which cover most area of the image only once.
ipidx_mask = cell(size(maxProj));
% Generate matrices
mean_int = zeros(length(InterestPoints),1); % for averaged ...
    intensity 17x17 pixels around the IP center
loc = repmat(mean_int,[1 2]); % localization of the IP center
tot_int = mean_int; % for selected IP, the total intensity in the ...
    IPs (using the IP scale and circular crop)
add_int = mean_int; % during the selection: the intensity in the ...
    area which only covered by the IP under examination
```

```
flag = mean_int; % index whether an IP is selected
for i = 1:length(InterestPoints);
    loc(i,:) = round(InterestPoints(i).Location);
    mean_int(i) = sum(sum((maxProj(loc(i,2)+ ...
        (-8:8),loc(i,1)+(-8:8)))))/(17^2);
    % Select the IP if the center of the IP is not covered
    % by any other IP
    % or, in case that other IPs cover the center of the examing ...
       IP, select if the IP has the biggest
    % averaged intensity
    % in case of equal intensity, the smaller scaled IP is selected)
    if isempty(ipidx_mask{loc(i,2),loc(i,1)}); % a new interest point
        flag(i) = 1;
    else
        for j = 1:length(ipidx_mask{loc(i,2),loc(i,1)});
            if flag(ipidx_mask{loc(i,2),loc(i,1)}(j)) > 0;
                if mean_int(i) > mean_int(ipidx_mask{loc(i,2), ...
                   loc(i,1)}(j));
                    flag(ipidx_mask{loc(i,2),loc(i,1)}(j)) = 0;
                    flag(i) = 1;
                elseif mean_int(i) == mean_int(ipidx_mask{loc(i,2), ...
                   loc(i,1)}(j));
                    if InterestPoints(i).Scale < ...
                       InterestPoints(ipidx_mask{loc(i,2),loc(i,1)}(j)).Scale;
                        flag(ipidx_mask{loc(i,2),loc(i,1)}(j)) = 0;
                        flag(i) = 1;
                    end
                end
            end
        end
    end
    % Further selection: if the IP is selected using the criteria ...
       above, it will still be deleted if more that 40% of its ...
       intensity has been covered by already selected IPs.
    if flag(i) == 1;
        s = 6*InterestPoints(i).Scale; % 6 is defined by matlab as ...
           SURF IP scaled surrounding
        rs = round(s);
        [x,y] = meshgrid(-rs:rs,-rs:rs);
        pos = [x(:) y(:)]; % circular scaled IP surrounding
```

```
        pos(sum(pos.^2,2)>s^2,:) = [];
        pos = pos + repmat(loc(i,:),[size(pos,1) 1]);
        pos(((pos(:,1)<1) + (pos(:,2)<1))>0,:) = [];
        pos(((pos(:,1)>size(maxProj,2))+ ...
            (pos(:,2)>size(maxProj,1)))>0,:) = [];
        for iparea = 1:size(pos,1);
            tot_int(i) = tot_int(i) + maxProj(pos(iparea,2), ...
               pos(iparea,1));
            existint = 0;
            for j = 1:length(ipidx_mask{pos(iparea,2),pos(iparea,1)});
                existint = existint+flag(ipidx_mask{pos(iparea,2), ...
                   pos(iparea,1)}(j));
            end
            add_int(i) = add_int(i) + double(existint==0)* ...
               maxProj(pos(iparea,2),pos(iparea,1));
        end
        if add_int(i) ≤ tot_int(i)*(1-ip_overlap);
            flag(i) = 0;
        else
            for iparea = 1:size(pos,1);
                ipidx_mask{pos(iparea,2),pos(iparea,1)} = ...
                   cat(1,i,ipidx_mask{pos(iparea,2),pos(iparea,1)});
            end
        end
    end
end
%% Feature Extraction for each selected IP
feature_spinImage = zeros(length(InterestPoints),length(d_level)* ...
   length(i_level)); % 6 intensity level and 5 distance level
feature_LBP = zeros(length(InterestPoints),4);
feature_corr = zeros(length(InterestPoints),1); % correlation to ...
   the H2B signal
feature_pos = zeros(length(InterestPoints),5); % localization features
feature_detail = zeros(length(InterestPoints),2); % detailed ...
   information about the IP localization
feature_mid = zeros(length(InterestPoints),1); % correlation to the ...
   midplane volume
for i = 1:length(InterestPoints);
    if flag(i)>0;
    % crop the IP surrounding as a squared image
```

```
s = 6*InterestPoints(i).Scale;
rs = round(s);
if min(loc(i,:)) ≤ rs;
    rs = min(loc(i,:)-1);
end
if max(loc(i,:))+rs > min(size(maxProj));
    rs = min(size(maxProj))-max(loc(i,:));
end
I = (maxProj(loc(i,2)+(-rs:rs),loc(i,1)+(-rs:rs)));
I(I<0) = 0;
%% 2D soft spin_Image features this is similar as a 2D ...
   histogram on distance-to-the-center and intensity with ...
   the modification that intensities larger than the max ...
   i-level will be counted in the max i-level bin.
% resize the IP to the maximum distance level
rs2 = ceil(d_level(end));
I2 = imresize(I,length(-rs2:rs2)/size(I,1));
I2 = I2(1:length(-rs2:rs2),1:length(-rs2:rs2));
[x,y] = meshgrid(-rs2:rs2,-rs2:rs2);
distmap = sqrt(x.^2+y.^2);
distmap = distmap(:);
I3 = I2(:);
% saturate the intensity at the highest i-level
I3(I3>i_level(end)) = i_level(end);
% calculate the 2D soft histogram
for d_i = 1:length(d_level);
    for i_i = 1:length(i_level)-1;
        feature_spinImage(i,((d_i-1)*6+i_i)) = ...
            sum(exp(-(((distmap-d_level(d_i)).^2/2/sI_a^2)+ ...
            ((I2(:)-i_level(i_i)).^2/2/sI_b^2))));
    end
    feature_spinImage(i,((d_i-1)*6+length(i_level))) = ...
       sum(exp(-(((distmap-d_level(d_i)).^2/2/sI_a^2) ...
       +((I3-i_level(end)).^2/2/sI_b^2))));
    feature_spinImage(i,((d_i-1)*6+(1:6))) = ...
       feature_spinImage(i,((d_i-1)*6+(1:6)))/ ...
       sum(exp(-(((distmap-d_level(d_i)).^2/2/sI_a^2))));
end
```

```
%% LBP features: this is a modified version of LBP feature. ...
   First, uLBP was calculated (count of multitransitions ...
   and count of the pixel number greater than the center ...
   with a single transition or none transition, 10 ...
   dimensions) and then, the major orientation of the ...
   5-by-5 field are calculated for each pixel and ...
   summarized for the IP. Afterwards, the 18 dimensional ...
   vector is summarized into 4 indicating the degree of ...
   randomness, structured brightness, portion of ...
   homegeneous field and orientation dominance (rotational ...
   symmetric).
full_lbp = zeros(1,18);
lbp_vec = -ones(size(I,1)*size(I,2),10);
% binarize the surrounding circle based on intensity ...
   comparison to the center pixel
for x_i = 3:(size(I,1)-2);
    for y_i = 3:(size(I,2)-2);
        v = I((x_i-2):2:(x_i+2),(y_i-2):2:(y_i+2));
        v = v(:);
        if v(5) < min_sig; % threshold for signal
            v = 0.1*ones(9,1);
        end
        ref = v(5);
        v(5) = [];
        if ref == 0;
            if max(v) > 0;
                ref = min(v(v>0))/5; % avoid artefect of NaN
            else
                ref = 0.01;
            end
        end
        v = [v(1:3);v(5);v(8:-1:6);v(4)]; % arrange into a ...
           circle
        v = v';
        v(v≤ref) = 0;
        lbp_vec(((y_i-1)*(size(I,2)-1)+x_i),:) = [v/ref 0 0];
        v = double(v>0); % binarization
        for s_i = 1:7;
            v(s_i) = v(s_i)-v(s_i+1);
        end
```

```
            v(8) = 0;
            lbp_vec(((y_i-1)*(size(I,2)-1)+x_i),9) = ...
                sum(v==-1); % count the transitions
            lbp_vec(((y_i-1)*(size(I,2)-1)+x_i),10) = sum(v==1);
        end
    end
    lbp_vec(sum(lbp_vec,2)<0,:) = []; % delete the border
    totbox = size(lbp_vec,1);
    % Summrize to the uLBP values
    full_lbp(10) = sum(max(lbp_vec(:,9:10),[],2)>1); % count of ...
        multitransitions
    lbp_vec(max(lbp_vec(:,9:10),[],2)>1,:) = [];
    full_lbp(8) = sum(sum(lbp_vec(:,1:8)>1,2)==8); % count of ...
        complete bright surround
    lbp_vec(sum(lbp_vec(:,1:8)>1,2)==8,:) = [];
    full_lbp(9) = sum(sum(lbp_vec(:,1:8),2)==0); % count of ...
        complete dark/homogeneous surround
    lbp_vec(sum(lbp_vec(:,1:8),2)==0,:) = [];
    sumvec_value = sum(lbp_vec(:,1:8),2);
    sumvec_count = sum(lbp_vec(:,1:8)>1,2);
    for p_i = 1:7;
        full_lbp(p_i) = sum((sumvec_count==p_i));
    end
    % Calculate the major orientation
    lbp_vec(:,9:10) = []; % orientation only calculated for ...
        sinlge transitions
    orientation = zeros(size(lbp_vec,1),1);
    for idx = 1:size(lbp_vec,1); % orientation is the intensity ...
        weighted balance center
        if lbp_vec(idx,1) == 0;
            for vdx = 1:8;
                v(vdx) = sum(lbp_vec(idx,1:vdx));
            end
            orientation(idx) = find(v ≥ (sumvec_value(idx)/2),1);
        else
            start_idx = find(lbp_vec(idx,:)==0,1);
            for vdx = 1:(8-start_idx+1);
                v(vdx) = ...
                    sum(lbp_vec(idx,start_idx:(vdx+start_idx-1)));
            end
```

```
        for vdx = (8-start_idx+2):8;
            v(vdx) = v(9-start_idx) + sum(lbp_vec(idx,1: ...
                (vdx-(8-start_idx+1))));
        end
        orientation(idx) = find(v ≥ ...
            (sumvec_value(idx)/2),1)+start_idx-1;
        if orientation(idx) > 8;
            orientation(idx) = orientation(idx)-8;
        end
    end
end
% Summrize the orientation
for o_i = 1:8;
    full_lbp(o_i+10) = sum(orientation==o_i);
end
full_lbp(1:10) = full_lbp(1:10)/totbox; % Normalization to 1
if size(orientation,1) == 0; % Avoid artefect of NaN
    orientation = 1;
end
full_lbp(11:end) = full_lbp(11:end)/size(orientation,1); % ...
   normalize to 1
% Summarize the uLBP and orientation to 4-dimensional features
feature_LBP(i,1) = sum((full_lbp-randref).^2)/maxvalue;% ...
   distance to complete random
l = sum(full_lbp(1:7)); % single transition proportion, ...
   inhomogeneity
if l > 0;
    feature_LBP(i,2) = sum(full_lbp(3:4))/l; %indicating ...
       stripe and structured bright area
else
    feature_LBP(i,2) = 0;
end
if full_lbp(10) > 0;
    feature_LBP(i,3) = full_lbp(9)./sum(full_lbp(9:10)); % ...
       homogeneous area proportion
else
    feature_LBP(i,3) = 1;
end
feature_LBP(i,4) = ...
   max(full_lbp(11:14)+full_lbp(15:18),[],2); % dominante ...
```

```
    direction for stripe

%% Correlation feature: Correlation between POI and max ...
    projected H2B in the squal IP field
rs_low = floor(rs/factor);
loc_low = round(loc(i,:)/factor);
if min(loc_low) ≤ rs_low;
    rs_low = min(loc_low-1);
end
if max(loc_low)+rs_low > min(size(maxProj_low));
    rs_low = min(size(maxProj_low))-max(loc_low);
end
I_low = maxProj_low(loc_low(2)+(-rs_low:rs_low),loc_low(1)+ ...
    (-rs_low:rs_low));
h2bsig = ...
    nucProj(loc_low(2)+(-rs_low:rs_low),loc_low(1)+(-rs_low:rs_low));
corrsig = corrcoef([I_low(:) h2bsig(:)]);
feature_corr(i) = corrsig(2,1);

%% Location features location features categorize the IP ...
    into complete nuc, half-nuc-half-cyt, complete cyt and ...
    half-outside. The position definition is calculated ...
    based on the 3D position of each pixel in the max ...
    projected POIn image.
%% Also the correlation to the midplane volume is ...
    calculated for an additional feature
cell_low = ...
    cellVolume(loc_low(2)+(-rs_low:rs_low),loc_low(1)+ ...
    (-rs_low:rs_low),:);
nuc_low = nucVolume(loc_low(2)+(-rs_low:rs_low),loc_low(1)+ ...
    (-rs_low:rs_low),:);
mid_low = midPlane(loc_low(2)+(-rs_low:rs_low),loc_low(1)+ ...
    (-rs_low:rs_low),:);
mPidx_low = mPidx(loc_low(2)+(-rs_low:rs_low),loc_low(1)+ ...
    (-rs_low:rs_low));
I_distnup = zeros(size(I_low));
I_distcell = zeros(size(I_low));
I_distmid = zeros(size(I_low));
for xidx = 1:size(I_low,1);
    for yidx = 1:size(I_low,2);
```

```
                I_distnup(xidx,yidx) = nuc_low(xidx,yidx, ...
                    mPidx_low(xidx,yidx));
                I_distcell(xidx,yidx) = cell_low(xidx,yidx, ...
                    mPidx_low(xidx,yidx));
                I_distmid(xidx,yidx) = mid_low(xidx,yidx, ...
                    mPidx_low(xidx,yidx));
            end
        end
        I_distnup = I_distnup(:);
        I_distcell = I_distcell(:);
        I_distmid = I_distmid(:);
        pixnum = length(I_distnup);
        if sum(I_distcell > 0) < (pixnum*0.8);
            feature_pos(i,1) = 0.25; % partially outside of the cell
        else
            nuppart = sum(I_distnup.*I_low(:))/sum(I_low(:));
            if nuppart ≤ 0.25; % almost complete in cell
                feature_pos(i,2) = 0.25;
            else
                if nuppart > 0.85; % complete in nuc
                    feature_pos(i,4) = 0.25;
                else % nup border
                    feature_pos(i,3) = 0.25;
                end
            end
            feature_detail(i,1) = nuppart;
            feature_detail(i,2) = sum(I_distnup>0)/pixnum;
        end
        %% Metric value and fraction of protein in the midplane volume
        feature_pos(i,5) = ...
            log(InterestPoints(i).Metric)/(log(100))-1; % Metric as ...
            a feature in addition as well
        feature_mid(i) = sum(I_distmid.*I_low(:))/sum(I_low(:));
    end
end
feature_spinImage = feature_spinImage * weight_spinImage;
feature_LBP = feature_LBP * weight_LBP;
feature_corr = feature_corr * weight_corr;
feature_pos = feature_pos * weight_pos;
feature_mid = feature_mid *  weight_corr;
```

```
else
POI_Feats = [];
SurfFeats = [];
InterestPoints = [];
addFeats = [];
end
%% Saving and updating
maxProj = imresize(maxProj,1/factor);
save(savefile,'frac_foreground','frac_foreground2d','poi_foreground', ...
    'POI_Feats','registered','feature_name','maxProj_low', ...
    'InterestPoints','tot_int','flag','loc','feature_spinImage', ...
    'feature_LBP','feature_corr','feature_pos','feature_mid', ...
    'feature_detail','feature_parameters','weight_parameters');
end
end
% QC: if all 40 frames can be processed
if exist(fullfile(feature_dir,cellseq_list(tmax).name),'file');
vec_QC(cdx) = 1;
else
vec_QC(cdx) = 0;
end
end
end
end
% Update the database file
summary_database.feature_extract(((summary_database.registration>0).* ...
    (summary_database.time_alignment>0))>0) = vec_QC;
writetable(summary_database,exp_database,'Delimiter','\t');
end
```

6.3.10 Generation of the SURF interest point dictionary by clustering

For the Bag-of-words approach, the next step was to built a dictionary by clustering similar interest points. For building the dictionary, a subset (5 %) of all interest points was used. The dictionary was generated by multi-step clustering. First, interest points were clustered based on their localization feature, metric value and correlation features with a k-d tree method. Furthermore, for each sub-class, clusters were found based on spin-image, and LBP features using the dbscan algorithm. Interest points with particular high metric value were further clustered by processing the spin-image features. The dictionary can then be used for assigning all interest points extracted from all images.

```
function generate_dict(exp_dir)

%% Interactive selection of experiment directory
if isempty(exp_dir)
    exp_dir = uigetdir('','Select the directory with your data ...
       series');
end

%% Set parameters
metric_perclass = 0.03; % A threshold of the metric value for ...
   performing dbscan clustering
dict_method = 'dbscan';
pca_th = 85; % PCA feature dimension reduction parameter: coverage ...
   of the variance
density_fac = 6; % dbscan parameter: for calculating the distance ...
   defining neighborhood
min_cluster = 3; % dbscan parameter: minimum number of data points ...
   in a cluster
% Threshold parameters for clustering data points by correlation ...
   features.
NE_corr = 0.65;
NE_anticorr = 0.05;
NUC_corr = 0.7;
NUC_anticorr = 0.1;
MID_corr = 0.5;
MID_none = 0.2;
% Parameter structure for saving
dict_parameter = ...
   struct('metric_cl',metric_perclass,'method',dict_method, ...
   'NE',[NE_corr;NE_anticorr],'NUC',[NUC_corr;NUC_anticorr], ...
   'MID',[MID_corr;MID_none]);

%% Define the saving directories and load the database file
features_dirname = 'Features_words';
current_dir = what;
centroid_dir = fullfile(current_dir.path,['Dictionary_' ...
   features_dirname]);
if ¬exist(centroid_dir,'dir');
    mkdir(centroid_dir);
```

```
end
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
celldir_list = ...
    summary_database.filepath(summary_database.feature_extract>0);

%% Generate the subset of interest points for dictionary building
cellnamefile = fullfile(centroid_dir,'cells_in_dictionary.mat');
if ~exist(cellnamefile,'file');
cluster_filepaths = {};
for cellidx = 1:length(celldir_list);
frame_list = dir(fullfile(celldir_list{cellidx},features_dirname, ...
    '*T0*.mat'));
for fridx = 1:length(frame_list)
    cluster_filepaths = cat(1,cluster_filepaths,fullfile( ...
        celldir_list{cellidx},features_dirname, ...
        frame_list(fridx).name));
end
end

%% Generate the 5% cell list at random base
cell_number = length(cluster_filepaths);
selcellidx = cat(2,rand(cell_number,1),(1:cell_number)');
selcellidx = sortrows(selcellidx,1);
selcellidx = selcellidx(1:round(cell_number*0.05),2); % save
disp('cell list generated')

%% Make the list of feature vectors of all interest points in the ...
    selected 5% cells
cell_number = length(selcellidx);
selcellidx = cat(2,selcellidx,zeros(cell_number,1));
FeatMat = []; % save
IP_idx_list = {};
for cellidx = 1:cell_number;
idx1 = size(FeatMat,1);
M = load(cluster_filepaths{selcellidx(cellidx,1)});
M.feature_corr(isnan(M.feature_corr)) = 0;
M.feature_mid(isnan(M.feature_mid)) = 0;
merged_corr = 4*(M.feature_corr.*sum(M.feature_pos(:,3:4),2)+ ...
    M.feature_mid.*sum(M.feature_pos(:,1:2),2));
```

```
SurfFeats = [M.feature_LBP M.feature_spinImage merged_corr ...
    M.feature_pos];
SurfFeats = SurfFeats(M.flag>0,:);
FeatMat = cat(1,FeatMat,SurfFeats);
selcellidx(cellidx,2) = size(SurfFeats,1);
idx2 = size(FeatMat,1);
IP_idx_list = cat(1,IP_idx_list,[cluster_filepaths{ ...
    selcellidx(cellidx,1)}(1:(end-4)) '_' num2str(idx1+1) '_' ...
    num2str(idx2)]);
end
save(cellnamefile,'cluster_filepaths','selcellidx','FeatMat', ...
    'IP_idx_list');
else
% If the feature vector list had been generated, just load
load(cellnamefile);
M = load(cluster_filepaths{selcellidx(1,1)});
disp('list of cells for training loaded')
end

%% Clustering based on the metric value, separated by median, or ...
    the localization features
ipnumber = size(FeatMat,1);
featnumber = size(FeatMat,2);
FeatMat = cat(2,(1:ipnumber)',FeatMat); % add index column
FeatMat = cat(2,zeros(ipnumber,1),FeatMat); % output clustering ...
    level 1
FeatMat = sortrows(FeatMat,featnumber+2); % sort based on metric value
splitidx = floor(ipnumber/2); % determine index of the median data ...
    point
FeatMat(1:splitidx,1) = 1; % smaller metric
metric_level1 = FeatMat(splitidx,end); % the thredhold for separation
FeatMat = cat(2,4*sum(FeatMat(:,(end-2):(end-1)),2),FeatMat); % ...
    nuclear or half nuclear localization assigned to 1
devcl = 2*FeatMat(:,1)+FeatMat(:,2); % the data was separated into ...
    4 clusters. small metric = cluster 1 and 3; nuclear localization ...
    in clusters 2 and 3
FeatMat = cat(2,devcl,FeatMat);
FeatMat = sortrows(FeatMat,[1 featnumber+4]); % In each of the 4 ...
    clusters, sort the data by metric value.
FeatMat(:,1) = 0;
```

```
splitidx = 0;
% Split each cluster into two subclusters with lower and higher ...
   metric value
metric_level2 = zeros(4,1);
for i = 0:3;
    level1_number = sum(devcl==i);
    FeatMat(splitidx+(1:floor(level1_number/2)),1) = 1;
    metric_level2(i+1) = FeatMat(splitidx+floor(level1_number/2),end);
    splitidx = level1_number+splitidx;
end
% Finalize the clusters by separating cell boundary from ...
   cytoplasmic localization and nuclear boundary from chromosomal ...
   localization. In total, 16 clusters were formed
FeatMat = cat(2,4*(FeatMat(:,(end-1))+FeatMat(:,(end-3))),FeatMat);
devcl = 8*FeatMat(:,1)+4*FeatMat(:,2)+2*FeatMat(:,3)+FeatMat(:,4);
disp('First level clustering done')

%% Further clustering for each of the 16 interest point clusters
subfeats_number = ...
   size(M.feature_LBP,2)+size(M.feature_spinImage,2)+ ...
   size(M.feature_corr,2);
level1_number = max(devcl);
subclass_meta = zeros(level1_number+1,2);
devcl2 = zeros(size(devcl));

FeatMat = cat(2,devcl,FeatMat);
FeatMat = sortrows(FeatMat,6);
devcl = FeatMat(:,1);
FeatMat(:,1:6) = [];

clustering_meta = cell(level1_number,1); % document clustering ...
   procedure
for i= 0:level1_number;
% feature vector list of IPs in one cluster
SubMat = FeatMat(devcl==i,:);
ipnumber = size(SubMat,1);
maxmetric = max(SubMat(:,end));
SubMat = SubMat(:,1:subfeats_number);
% for clusters with more than 15 IPs, further clustering
if ipnumber > 15;
```

```
% decide how further clustering should be performed by the maximum ...
   metric value
class_number = round(maxmetric/metric_perclass);
if class_number > 2;
% cluster the interest points by correlation features
subcl = zeros(size(SubMat,1),1);
if ismember(i,[7 6 3 2]); % nuc boundary clusters, use the H2B ...
   correlation
    corr_cl = ones(size(SubMat,1),1) + ...
       double(SubMat(:,end)>NE_corr) + ...
       double(SubMat(:,end)>NE_anticorr);
    corr_cl_number = 3;
end
if ismember(i,[15 14 11 10]); % chromosomal clusters, use the H2B ...
   correlation with different threshold
    corr_cl = ones(size(SubMat,1),1) + ...
       double(SubMat(:,end)>NUC_corr) + ...
       double(SubMat(:,end)>NUC_anticorr);
    corr_cl_number = 3;
end
if ismember(i,[13 12 9 8 5 4 1 0]); % cytoplasm and cell boundary, ...
   use the midbody correlation
    corr_cl = ones(size(SubMat,1),1) + ...
       double(SubMat(:,end)>MID_corr) + double(SubMat(:,end)>MID_none);
    corr_cl_number = 3;
end
clustering_meta{i+1} = cell(corr_cl_number,1);
class_number = 0;
sub2_cl = zeros(size(corr_cl));
% for each cluster of IPs with similar metric and H2B/midplane ...
   correlation and same localization, dbscan was used for further ...
   clustering
for j = 1:corr_cl_number;
    sub2_ipnumber = sum(corr_cl==j);
    % if the size of the cluster if large enough
    if sub2_ipnumber > 15;
        Sub2Mat = SubMat(corr_cl==j,1:(end-1));
        sub2_cl(corr_cl==j) = 1;
            % Perform the PCA of the feature matrix for dimension ...
               reduction
```

```
            [coeff,¬,latent,¬,v_percent] = pca(Sub2Mat);
            for kk = length(v_percent):-1:2;
                v_percent(kk) = sum(v_percent(1:kk));
            end
            num_feat = sum(v_percent<pca_th)+1;
            if num_feat < 2;
                num_feat = 2;
            end
            S2 = Sub2Mat*coeff(:,1:num_feat);
            % Calculate the distance defining neighborhood ...
               based on the density of the data set in the ...
               feature space
            dbscan_env = ...
               1/(sqrt(size(S2,1)/(max(S2(:,1))-min(S2(:,1)))/ ...
               (max(S2(:,2))-min(S2(:,2))))) *density_fac;
            [sub2_test,score] = dbscan(S2,dbscan_env,min_cluster);
            [¬,¬,sub2_test] = unique(sub2_test);
            kupdate = max(sub2_test);
            sub2_cl(corr_cl==j) = sub2_test;
            clustering_meta{i+1}{j} = ...
               struct('algorithm','dbscan', ...
               'pca_coeff',coeff,'dim',num_feat,'parameter', ...
               [density_fac;dbscan_env;min_cluster;kupdate],'refine',0);
        if ismember(i,[0 2 8 10]); % very high metric classes, ...
           further cluster based on spinImage I-center distance
                clustering_meta{i+1}{j} = ...
                   struct('algorithm','dbscan', ...
                   'pca_coeff',coeff,'dim',num_feat,'parameter', ...
                   [density_fac;dbscan_env;min_cluster;kupdate*3], ...
                   'refine',1,'assigned',sub2_test);
            l = zeros(size(Sub2Mat,1),1);
            for alpha = 1:length(l);
                rsh = ...
                   (reshape(Sub2Mat(alpha,(size(M.feature_LBP,2)+1):end), ...
                   [6 5]))';
                for beta = 6:-1:2;
                    rsh(:,beta) = sum(rsh(:,1:beta),2);
                end
                lsh = zeros(5,1);
                for beta2 = 1:5;
```

```
                    lsh(beta2) = find(rsh(beta2,:)> ...
                        (rsh(beta2,end)/2),1);
                end
                l(alpha) = mean(lsh)/std(lsh)*(max(lsh)>3);
                if isnan(l(alpha));
                    l(alpha) = -1;
                end
            end
            %  Three classes: bright homogeneous, dim  homogeneous, ...
               bright structured (dot, ring etc.). In the ...
               dictionary, the threshod for ``bright'' is defined by ...
               setting a value. During the assignment, the threshold ...
               is adjusted cell specifically
            sub2_test = sub2_test + kupdate*double(l>0) + ...
                kupdate*double(l>4);
            kupdate = kupdate*3;
            sub2_cl(corr_cl==j) = sub2_test;
        end
    else % if the size of the cluster is small, no clustering
        kupdate = 1;
        sub2_cl(corr_cl==j) = 1;
        clustering_meta{i+1}{j} = struct('algorithm','none', ...
            'parameter',1);
    end % assign the clustering result
    subcl = subcl + class_number*double(corr_cl==j) + sub2_cl.* ...
        double(corr_cl==j);
    class_number = class_number + kupdate;
end
else % very low metric, only a dbscan clustering
clustering_meta{i+1} = cell(1);
        [coeff,¬,latent,¬,v_percent] = pca(SubMat(:,1:(end-1)));
        for j = length(v_percent):-1:2;
            v_percent(j) = sum(v_percent(1:j));
        end
        num_feat = sum(v_percent<pca_th)+1;
        if num_feat < 2;
            num_feat = 2;
        end
        S2 = SubMat(:,1:(end-1))*coeff(:,1:num_feat);
```

```
        dbscan_env = 1/(sqrt(size(S2,1)/(max(S2(:,1))-min(S2(:,1))) ...
            /(max(S2(:,2))-min(S2(:,2)))))*density_fac;
        [subcl,score] = dbscan(S2,dbscan_env,min_cluster);
        [¬,¬,subcl] = unique(subcl);
        class_number = max(subcl);
        clustering_meta{i+1}{1} = struct('algorithm','dbscan', ...
            'pca_coeff',coeff,'dim',num_feat,'parameter', ...
            [density_fac;dbscan_env;min_cluster]);
end
else % very small cluster, no further clustering
class_number = 1;
subcl = ones(ipnumber,1);
clustering_meta{i} = cell(1,1);
clustering_meta{i}{1} = struct('algorithm','none','parameter',1);
end
devcl2(devcl==i) = subcl;
subclass_meta(i+1,:) = [ipnumber class_number];
disp(['clustering for cluster level 1 number ' num2str(i)])
end

%% Calculate the centroids for all classes
feats_centroids = [];
index_centroids = [];
for i = 0:level1_number;
for j = 1:subclass_meta(i+1,2);
if sum((devcl==i).*(devcl2==j))>0;
    index_centroids = cat(1,index_centroids,[i j]);
    feats_centroids = cat(1,feats_centroids, ...
       mean(FeatMat(((devcl==i).* ...
       (devcl2==j))>0,1:subfeats_number),1));
end
end
end

%% save the results
savefile = fullfile(centroid_dir,['dictionary_' features_dirname ...
   '_' dict_method '.mat']);
save(savefile,'cluster_filepaths','selcellidx','FeatMat','metric_level1', ...
   'metric_level2','devcl','devcl2','subclass_meta','feats_centroids', ...
   'index_centroids','clustering_meta','dict_parameter')
```

```
end
```

The function dbscan was downloaded from Thanh N. Tran, Klaudia Drab, Michal Daszykowski, "Revised DBSCAN algorithm to cluster data with dense adjacent clusters", Chemometrics and Intelligent Laboratory Systems, 120:92–96.

6.3.11 Assign SURF interest points into IP clusters

Once the dictionary was built, all interet points extracted from the POI images were assigned to the clusters accordingly.

```
function assign_surf_ips(exp_dir,time_dir,dictionary_file, ...
    alignmatname)

%% Loading the temporal model, IP dictionary and database files
if isempty(exp_dir)
     exp_dir = uigetdir('','Select the directory with your data ...
        series');
end
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database,'Delimiter','\t');
cell_dirlist = summary_database.filepath( ...
    summary_database.feature_extract>0);
cell_total = length(cell_dirlist);
vec_QC = ones(cell_total,1);

if isempty(time_dir)
     time_dir = uigetdir('','Select the directory with your time ...
        alignment data');
end
load(fullfile(time_dir,'temporal_alignment.mat'));

load(dictionary_file);
features_dirname = 'Features_words';
add_updatename = 'version';

tmax = 40;

%% Assign IPs cell by cell
```

```
for cdx = 1:cell_total;
% make the output directory
savedir = fullfile(cell_dirlist{cdx},[features_dirname  '_' ...
    dict_parameter.method add_updatename]);
cellseq_list = dir(fullfile(cell_dirlist{cdx}, ...
    features_dirname,'*T0*.mat'));
if ¬exist(savedir, 'dir');
mkdir(savedir);
end
if exist(fullfile(savedir,cellseq_list(tmax).name),'file');
vec_QC(cdx) = 2;
end
if vec_QC(cdx) ≠ 2;
% load the temporal registration result
timefile = fullfile(cell_dirlist{cdx},'Temporal_Align', alignmatname);
load(timefile);
% assign the temporal registration into one of the 20 standard stages
for tdx = 1:length(mitotime);
mitotime(tdx,1) = sum(time_cluster.class_start ≤ mitotime(tdx,1));
end
if length(cellseq_list) == tmax;
if length(mitotime) == tmax;
for tdx = 1:length(cellseq_list);
savefile = fullfile(savedir,cellseq_list(tdx).name);
if ¬exist(savefile,'file')
% load the interest point file of each image
featfile = fullfile(cell_dirlist{cdx},features_dirname, ...
    cellseq_list(tdx).name);
disp(['Process' cell_dirlist{cdx} ' at ' num2str(tdx) ' Assign the ...
    surf interest points'])
M = load(featfile);
if sum(M.flag)==0;
vec_QC(cdx) = 0;
end
if vec_QC(cdx) == 1;
M.feature_corr(isnan(M.feature_corr)) = 0;
M.feature_mid(isnan(M.feature_mid)) = 0;
merged_corr = 4*(M.feature_corr.*sum(M.feature_pos(:,3:4),2) + ...
    M.feature_mid.*sum(M.feature_pos(:,1:2),2));
```

```
SurfFeats = [M.feature_LBP M.feature_spinImage merged_corr ...
    M.feature_pos];
SurfFeats = SurfFeats(M.flag>0,:);
IP_idxvec = (1:length(M.flag))';
IP_idxvec = IP_idxvec(M.flag>0);
clustering_featnum = size(M.feature_LBP,2) + ...
    size(M.feature_spinImage,2);

% determine the background intensity using the interest point with ...
    the lowerst metric value in the cell for thresholding ``bright'' ...
    from ``dim''
lastpos = find(M.feature_pos(:,2)>0);
if isempty(lastpos);
    lastpos = find(M.flag>0);
end
lastpos = lastpos(end);
lastsize = 6*M.InterestPoints(lastpos).Scale; % 6 is defined by ...
    matlab as SURF IP scaled surrounding
lastsize = (round(lastsize)*2+1)^2;
background_int = M.tot_int(lastpos)/lastsize;

% First level clustering by localization and metric value
kd_clustering = zeros(size(SurfFeats,1),4);
kd_clustering(:,4) = double(SurfFeats(:,end) ≤ metric_level1);
kd_clustering(:,3) = 4*sum(SurfFeats(:,(end-2):(end-1)),2);
metric_level2_idx = 2*kd_clustering(:,3)+kd_clustering(:,4)+1;
for midx = 1:4;
    kd_clustering(metric_level2_idx==midx,2) = double(SurfFeats( ...
        metric_level2_idx==midx,end)≤metric_level2(midx));
end
kd_clustering(:,1) = 4*(SurfFeats(:,(end-1))+SurfFeats(:,(end-3)));
asscl = 8*kd_clustering(:,1)+4*kd_clustering(:,2) + ...
    2*kd_clustering(:,3)+kd_clustering(:,4);
% Find the closest cluster if the cluster to be assigned does not ...
    exist in the dictionary
for l1_idx = 1:length(asscl);
    if ¬ismember(asscl(l1_idx),index_centroids(:,1));
        ant1 = 8*kd_clustering(l1_idx,1)+ ...
            4*mod(kd_clustering(l1_idx,2)+1,2) + ...
            2*kd_clustering(l1_idx,3)+ kd_clustering(l1_idx,4);
```

```
        if ismember(ant1,index_centroids(:,1));
            asscl(l1_idx) = ant1;
        else
            ant2 = 8*kd_clustering(l1_idx,1)+ ...
               4*kd_clustering(l1_idx,4)+ ...
               2*kd_clustering(l1_idx,3)+ ...
               mod(kd_clustering(l1_idx,4)+1,2);
            if ismember(ant2,index_centroids(:,1));
                asscl(l1_idx) = ant2;
            else
                asscl(l1_idx) = 8*kd_clustering(l1_idx,1)+ ...
                   4*mod(kd_clustering(l1_idx,4)+1,2)+ ...
                   2*kd_clustering(l1_idx,3)+ ...
                   mod(kd_clustering(l1_idx,4)+1,2);
            end
        end
    end
end

% Next, assign each ip into the subclass
asscl2 = zeros(size(asscl));
lib_idx = zeros(size(asscl));
for l1_idx = 1:length(asscl);
    % in case that the cluster was only sub-divided using dbscan
    if size(clustering_meta{asscl(l1_idx)+1},1) == 1;
            correspond_centers = FeatMat(devcl==asscl(l1_idx), ...
               1:clustering_featnum);
            correspond_index = devcl2(devcl==asscl(l1_idx));
            correspond_centers = correspond_centers* ...
               clustering_meta{asscl(l1_idx)+1}{1}.pca_coeff(:, ...
               1:clustering_meta{asscl(l1_idx)+1}{1}.dim);
            dbscan_dist = sum((correspond_centers - ...
               repmat(SurfFeats(l1_idx,1:clustering_featnum) * ...
               clustering_meta{asscl(l1_idx)+1}{1}.pca_coeff(:, ...
               1:clustering_meta{asscl(l1_idx)+1}{1}.dim), ...
               [size(correspond_centers,1) 1])).^2,2);
            [¬,asscl2(l1_idx)] = min(dbscan_dist,[],1);
            asscl2(l1_idx) = correspond_index(asscl2(l1_idx));
            lib_idx(l1_idx) = ...
               find(index_centroids(:,1)==asscl(l1_idx),1) + ...
```

```
            find(index_centroids(index_centroids(:,1) ...
            ==asscl(l1_idx),2)==asscl2(l1_idx),1) - 1;
elseif size(clustering_meta{asscl(l1_idx)+1},1) > 1; % in case ...
   that hte cluster was multiple divided into subclusters
    % get the correlation class
    correspond_centers = FeatMat(devcl==asscl(l1_idx), ...
       1:(clustering_featnum+1));
    [¬,corr_idx] = min(abs(correspond_centers(:,end) - ...
       SurfFeats(l1_idx,clustering_featnum+1)),[],1);
    corr_val = correspond_centers(corr_idx,end);
    if ismember(asscl(l1_idx),[7 6 3 2]); % nuc boundary
        corr_cl = 1 + double(corr_val>dict_parameter.NE(1)) + ...
           double(corr_val>dict_parameter.NE(2));
    elseif ismember(asscl(l1_idx),[15 14 11 10]); % pure nuc
        corr_cl = 1 + double(corr_val>dict_parameter.NUC(1)) + ...
           double(corr_val>dict_parameter.NUC(2));
    elseif ismember(asscl(l1_idx),[13 12 9 8 5 4 1 0]); % cyto ...
       and cell boundary
        corr_cl = 1 + double(corr_val>dict_parameter.MID(1)) + ...
           double(corr_val>dict_parameter.MID(2));
    end
        % get the index range
        cl2min = 0;
        jidx = 1;
        while jidx < corr_cl;
            cl2min = ...
               clustering_meta{asscl(l1_idx)+1}{jidx}.parameter(end) ...
               + cl2min;
            jidx = jidx + 1;
        end
        % assign into the dbscan cluster which has the closest ...
           data point to the IP being processed
        cl2max = cl2min + ...
           clustering_meta{asscl(l1_idx)+1}{corr_cl}.parameter(end);
        correspond_centers = FeatMat(((devcl==asscl(l1_idx)) ...
           .*(devcl2>cl2min).*(devcl2≤cl2max))>0, ...
           1:clustering_featnum);
        correspond_centers = correspond_centers* ...
           clustering_meta{asscl(l1_idx)+1}{corr_cl}. ...
           pca_coeff(:,1:clustering_meta{asscl(l1_idx)+1}{corr_cl}.dim);
```

```
dbscan_dist = sum((correspond_centers- ...
    repmat(SurfFeats(l1_idx,1:clustering_featnum)* ...
    clustering_meta{asscl(l1_idx)+1}{corr_cl}. ...
    pca_coeff(:,1:clustering_meta{asscl(l1_idx)+1}{corr_cl}.dim), ...
    [size(correspond_centers,1) 1])).^2,2);
[¬,ass2_test] = min(dbscan_dist,[],1);
% in case of low metric cluster without further clustering
if clustering_meta{asscl(l1_idx)+1}{corr_cl}.refine == 0;
    correspond_index = devcl2(((devcl==asscl(l1_idx)).* ...
        (devcl2>cl2min).*(devcl2≤cl2max))>0);
    asscl2(l1_idx) = correspond_index(ass2_test);
else % high metric clusters with further clustering ...
    based on spin_image features
    ass2_test = ...
        clustering_meta{asscl(l1_idx)+1}{corr_cl}.assigned( ...
        ass2_test);
    rsh = ...
        (reshape(SurfFeats(l1_idx,(size(M.feature_LBP,2)+1): ...
        clustering_featnum),[6 5]))';
    for beta = 6:-1:2;
        rsh(:,beta) = sum(rsh(:,1:beta),2);
    end
    lsh = zeros(5,1);
    for beta2 = 1:5;
        lsh(beta2) = ...
            find(rsh(beta2,:)>(rsh(beta2,end)/2),1);
    end
    ipsize = 6*M.InterestPoints(IP_idxvec(l1_idx)).Scale;
    ipsize = (round(ipsize)*2+1)^2;
    meanint = M.tot_int(IP_idxvec(l1_idx))/ipsize;
    lsh = ...
        mean(lsh)/std(lsh)*(meanint>(2*background_int)); ...
        % "bright" IP: averaged intensity twice larger ...
        than the mean intensity of the IP with the ...
        lowest metric
    if isnan(lsh);
        lsh = -1;
    end
    ass2_test = ass2_test + ...
        (clustering_meta{asscl(l1_idx)+1}{corr_cl}.parameter(end) ...
```

```
    /3)*double(lsh>0)+(clustering_meta{asscl( ...
    l1_idx)+1}{corr_cl}.parameter(end)/3) ...
    *double(lsh>4)+cl2min;
% assign to a close cluster if the correct cluster ...
    was not included in the dictionary
if ¬...
    ismember(ass2_test,index_centroids(index_centroids(:,1) ...
    ==asscl(l1_idx),2));
     if lsh≤0;
         ass2_test = ass2_test + ...
            2*(clustering_meta{asscl(l1_idx)+1}{ ...
            corr_cl}.parameter(end)/3);
         if ¬ismember(ass2_test,index_centroids( ...
            index_centroids(:,1) == asscl(l1_idx),2));
             ass2_test = ass2_test - ...
                (clustering_meta{asscl(l1_idx)+1}{ ...
                corr_cl}.parameter(end)/3);
         end
     elseif lsh≤4;
         ass2_test = ass2_test + ...
            (clustering_meta{asscl(l1_idx)+1}{ ...
            corr_cl}.parameter(end)/3);
         if ¬ismember(ass2_test,index_centroids( ...
            index_centroids(:,1) == asscl(l1_idx),2));
             ass2_test = ass2_test - ...
                2*(clustering_meta{asscl(l1_idx)+1}{ ...
                corr_cl}.parameter(end)/3);
         end
     elseif lsh > 4;
         ass2_test = ass2_test - ...
            2*(clustering_meta{asscl(l1_idx)+1}{ ...
            corr_cl}.parameter(end)/3);
         if ¬ismember(ass2_test,index_centroids( ...
            index_centroids(:,1) == asscl(l1_idx),2));
             ass2_test = ass2_test + ...
                (clustering_meta{asscl(l1_idx)+1}{ ...
                corr_cl}.parameter(end)/3);
         end
     end
end
```

```
                asscl2(l1_idx) = ass2_test;
            end
        lib_idx(l1_idx) = find(((index_centroids(:,1) == ...
            asscl(l1_idx)).*(index_centroids(:,2) == ...
            asscl2(l1_idx)))>0);
    end
end

% Calculate the feature vector for the image as fraction of ...
   intensities in each interest point cluster
POI_Feats_int = zeros(size(index_centroids,1),1);
vector_int = M.tot_int(M.flag>0);
for fh_idx = 1:length(POI_Feats_int);
    POI_Feats_int(fh_idx) = sum(vector_int(lib_idx==fh_idx));
end
% Save the file
red_time = mitotime(tdx,1);
save(savefile,'red_time','POI_Feats_int','asscl','asscl2','lib_idx', ...
   'dictionary_file','background_int');
end
end
end
end
end
end
end

% Update the database file
vec_QC(vec_QC==2) = 1;
summary_database.feature_extract(summary_database.feature_extract>0) ...
   = vec_QC;
writetable(summary_database,exp_database,'Delimiter','\t');

end
```

6.3.12 Automatic annotation of the protein distribution using supervised modeling

Based on the feature vectors, the protein distribution in each image could be automatically assigned. Here, a linear model was built based on a subset of the data for six characteristic mitotic subcellular structures and was then used to assign all images in the data set.

First, the feature vectors of each image sequence were smoothed over time based on the image intensity correlation between adjacent frames. A table summarize all feature vectors was built.

```
function smooth_feature_vector
% select data directories
exp_dir = uigetdir('', 'Select the directory with your data series');
time_dir = uigetdir('', 'Select the directory with your time ...
    alignment data');
features_dirname = 'Features_words';
add_name = '_dbscan';
% load the database file
exp_database = fullfile(exp_dir,'full_database.txt');
summary_database = readtable(exp_database, 'Delimiter','\t');
cell_dirlist = ...
    summary_database.filepath(summary_database.feature_extract>0);
poinamelist = summary_database.poi(summary_database.feature_extract>0);
cell_total = length(cell_dirlist);
% load the temporal model
load(fullfile(time_dir,'temporal_alignment.mat'));
alignmatname = 'Align_annotation.mat';
% Information in the summary table
dir_list = {};
poi_name = {};
poi_namefull = {};
time_vec = [];
feature_matall = [];
BG = [];
% Process cell by cell
for cdx = 1:cell_total;
timefile = fullfile(cell_dirlist{cdx}, 'Temporal_Align',alignmatname);
load(timefile)
for tdx = 1:length(mitotime);
% assign temporal registration into standard mitotic stages
mitotime(tdx,2) = sum(time_cluster.class_start ≤ mitotime(tdx,1));
end

featlist = dir(fullfile(cell_dirlist{cdx}, [features_dirname ...
    add_name],'*T0*.mat'));
% Calculate the correlation coefficient between two adjacent frames ...
```

```
    and assign other information to the summary table
if length(featlist)==40;
corrtrace = zeros(40,1);
load(fullfile(cell_dirlist{cdx},features_dirname, featlist(1).name));
frame_t0 = maxProj_low(:);
f_ip = [];
f_glob = [];
for tdx = 1:length(featlist);
load(fullfile(cell_dirlist{cdx},features_dirname, featlist(tdx).name));
load(fullfile(cell_dirlist{cdx},[features_dirname ...
    add_name],featlist(tdx).name));
frame_t1 = maxProj_low(:);
frame_common = frame_t0+frame_t1;
m = corrcoef(frame_t0(frame_common>0), frame_t1(frame_common>0));
corrtrace(tdx) = m(2,1);
frame_t0 = frame_t1;
POI_Feats_int = POI_Feats_int/(sum(POI_Feats_int)+ ...
    1*double(sum(POI_Feats_int)==0));
f_ip = cat(1,f_ip,POI_Feats_int');
f_glob = cat(1,f_glob,POI_Feats');
poi_name = cat(1,poi_name,cell_dirlist{cdx}(33:35));
poi_namefull = cat(1,poi_namefull,poinamelist{cdx});
dir_list = cat(1,dir_list,cell_dirlist{cdx});
BG = cat(1,BG,[frac_foreground frac_foreground2d]);
end
smooth_fcell = zeros(size(f_ip));
% translate the correlation coefficient to the correlation factor
tr = mean(corrtrace);
fcorr = 1./(1+exp(-(corrtrace-tr)*20));
% calculate the smoothing function over time and smooth the feature ...
    sequence
for k = 1:40;
fcorr_k = zeros(40,1);
for p = 1:(k-1);
    fcorr_k(p) = prod(fcorr((p+1):k));
end
for p = (k+1):40;
    fcorr_k(p) = prod(fcorr((k+1):p));
end
fcorr_k(k) = 1;
```

```
fcorr_k = fcorr_k/sum(fcorr_k);
smooth_fcell(k,:) = sum(f_ip.*repmat(fcorr_k, [1 size(f_ip,2)]),1);
end
time_vec = cat(1,time_vec,mitotime);
% record in the summary table
feature_matall = cat(1,feature_matall, [smooth_fcell f_glob]);
end
end
% Output and save
summary_database = dataset({dir_list,'path'},{poi_name,'poi'}, ...
    {poi_namefull,'fullname'},{time_vec,'time'}, ...
    {BG,'foreground'},{feature_matall,'f'});
featuretxt = fullfile(exp_dir,['Summary_feature' add_name ...
    '_smooth.txt']);
export(summary_database,'file',featuretxt);
end
```

Then, a regression model was trained on a subset of the data and used for assigning all images.

```
function distribution_analysis(featuretxt)
% Define the training set: structure, protein and time classes
pattern_name = {'cyt';'chr';'kt';'ct';'sp';'mid'};
pattern_def = {'NES';'H2B';'CEN';'NED';'TUB';'RAC'};
pattern_time = {(1:20);(1:20);(1:20);(1:16); (4:20);(13:20)};
tmax = 40;
poi_all = {'KIF';'MIS';'TUB';'RAC';'CDC';'NED';'AUR'; ...
    'NUP';'PLK';'CEN';'BUB';'NES';'H2B'};

% load the data
Data = readtable(featuretxt,'Delimiter','\t');
feature_fullmat = table2array(Data(:,8:end));

% feature QC: delete features where too little cells have them and ...
    too low fraction of proteins were assigned to
del_feat = (max(feature_fullmat,[],1)<0.05)+ ...
    (sum(feature_fullmat>0,1)<(size(feature_fullmat,1)/length(poi_all)/5));
feature_fullmat(:,del_feat>0) = [];

% define the objective function based on the foreground fraction
```

```
fg = Data.foreground_1.^(2/3); % 3D converto to 2D foreground

% Define the parameters for the training of the model
cv_fac = 10; % cross validation factor
lr_alpha = 0.3; % balance the lasso and l2 weight in elastic net

% Construct the training matrix: For each localization, select up ...
   to 400 cells with the right poi at the right stage
labelmatrix = [];
featmat = [];
trainingcells = [];
for pdx = 1:length(pattern_name);
    selvec = strcmp(Data.poi,pattern_def{pdx});% scan based on the ...
       poi name
    selvec = selvec.*ismember(Data.time_2, pattern_time{pdx});% ...
       scan based on the time
    if sum(selvec) > 400;
        selvec = [(1:length(selvec))' rand(length(selvec),1) selvec];
        selvec = sortrows(selvec,[3 2]);
        selvec(1:(end-400),3) = 0;
        selvec = sortrows(selvec,1);
        selvec = selvec(:,3);
    end
    lm = zeros(sum(selvec),length(pattern_name));
    lm(:,pdx) = fg(selvec>0);
    labelmatrix = cat(1,labelmatrix,lm);
    featmat = cat(1,featmat,feature_fullmat(selvec>0,:));
    trainingcells = cat(2,trainingcells,selvec);
end
labelmatrix(:,1) = labelmatrix(:,1) + 1 - sum(labelmatrix,2); % the ...
   fraction in cytoplasm is the background fraction

% normalize the feature matrix such that for each feature, the ...
   maximum is 1 in the training set
featmat_max = max(featmat,[],1);
featmat_max(featmat_max==0) = 1;
featmat = featmat./repmat(featmat_max,[size(featmat,1),1]);

% training set qc: delete empty localization
del_idx = (sum(labelmatrix,1)==0);
```

```
labelmatrix(:,del_idx>0) = [];
pattern_name(del_idx'>0) = [];
pattern_def(del_idx'>0) = [];
pattern_time(del_idx'>0) = [];

% Define the output structures
classnum = size(labelmatrix,2);
ncell = size(labelmatrix,1);
Beta_tot = cell(1,classnum);
Fit_tot = cell(1,classnum);
labelpredict_cv = zeros(ncell,classnum);
fg_train = sum(labelmatrix(:,2:6),2); % foreground original
labelmatrix_original = labelmatrix; % labelmatrix original

% Log the objective function if the distribution is far from normal ...
    (very concentrated in the low part, e.g. centrosomes)
mm = zeros(2,classnum);
for i = 1:classnum;
pos = labelmatrix(labelmatrix(:,i)>0,i);
mm(1,i) = median(pos);
mm(2,i) = (max(pos)+min(pos))/2;
end
logfeat = (mm(2,:)./mm(1,:))>2;
labelmatrix(:,logfeat) = log(labelmatrix(:,logfeat) + 0.0001) - ...
    log(0.0001);

% Normalize the objective function for each localization to max=1 ...
    such that the weight between classes get balanced
max_label = max(labelmatrix,[],1);
labelmatrix = labelmatrix./repmat(max_label, [size(labelmatrix,1) 1]);

% Train the linear model for each localization pattern
for cl_round = 1:classnum;
% randomly select the equal number of negative cells to train with ...
    the positive one such that the training weigt get balanced
posnum = sum(labelmatrix(:,cl_round)>0);
negnum = sum(labelmatrix(:,cl_round)==0);
if negnum ≥ posnum;
     featmat_unbias = cat(2,labelmatrix(labelmatrix(:,cl_round)==0, ...
        cl_round),featmat(labelmatrix(:,cl_round)==0,:));
```

```
    featmat_unbias = cat(2,rand(negnum,1),featmat_unbias);
    featmat_unbias = sortrows(featmat_unbias,[1 2]);
    label_unbias = featmat_unbias(1:posnum,2);
    featmat_unbias = featmat_unbias(1:posnum,3:end);
    label_unbias = cat(1,label_unbias,labelmatrix(labelmatrix(:, ...
       cl_round)>0,cl_round));
    featmat_unbias = cat(1,featmat_unbias,featmat(labelmatrix(:, ...
       cl_round)>0,:));
else
    featmat_unbias = featmat;
    label_unbias = labelmatrix(:,cl_round);
end
% Train the model
[B,FitInfo] = lasso(featmat_unbias,label_unbias,'CV',cv_fac, ...
   'Alpha',lr_alpha);
Beta_tot{cl_round} = B;
Fit_tot{cl_round} = FitInfo;
labelpredict_cv(:,cl_round) = FitInfo.Intercept(FitInfo.Index1SE) ...
   +featmat*B(:,FitInfo.Index1SE);
end
predict_error = sqrt(sum((labelmatrix-labelpredict_cv).^2,2));

%% Use the model to predict the distribution for all cells
% normalize the feature matrix as for the training
feature_fullmat = feature_fullmat./repmat(featmat_max, ...
   [size(feature_fullmat,1),1]);
% Predict the distribution using the linear model
predictresult = zeros(size(feature_fullmat,1),classnum);
for cl_round = 1:classnum;
predictresult(:,cl_round) = ...
   Fit_tot{cl_round}.Intercept(Fit_tot{cl_round}.Index1SE) ...
   +feature_fullmat*Beta_tot{cl_round}(:,Fit_tot{cl_round}.Index1SE);
end
% Time dependent smoothing for each image sequence
smoothresult = predictresult;
for k = 1:length(poi_all);
poiname = poi_all{k};
idxlist = strcmp(Data.poi,poiname);
framelist = Data.path(idxlist>0);
celllist = unique(framelist);
```

```
analysis_result = predictresult(idxlist>0,:);
cellnum = length(celllist);
for i = 1:cellnum;
cellframe = strcmp(framelist,celllist{i});
if sum(cellframe) == tmax;
subresult = analysis_result(cellframe>0,:);
for j = 1:size(analysis_result,2);
    diffvec = diff(subresult(:,j));
    dv = diffvec;
    meandiff = mean(abs(diffvec));
    stddiff = std(abs(diffvec),[],1);
    peakdiff = (diffvec(1:end-1).*diffvec(2:end))<0;
    diffvec = abs(diffvec(1:end-1)-diffvec(2:end));
    diffvec = cat(2,(2:tmax-1)',diffvec);
    diffvec = sortrows(diffvec,2);
    % Smooth if one data point differs to the adjacent data point ...
       on both sides strongly beyond the average difference by ...
       taking the mean of the adjacent neighborhood.
    for p = (tmax-2):-1:1;
        if peakdiff(diffvec(p,1)-1) == 1;
            if max(abs([dv(diffvec(p,1)),dv(diffvec(p,1)-1)])) ...
               >(meandiff+stddiff);
                if min(abs([dv(diffvec(p,1)),dv(diffvec(p,1)-1)])) ...
                   >meandiff;
                    subresult(diffvec(p,1),j) = ...
                       mean([subresult(diffvec(p,1)-1,j),  ...
                       subresult(diffvec(p,1)+1,j)]);
                    dv = diff(subresult(:,j));
                end
            end
        end
    end

end
analysis_result(cellframe>0,:) = subresult;
end
end
smoothresult(idxlist>0,:) = analysis_result;
end
```

```
% Back-normalize the prediction output
smoothresult(smoothresult<0) = 0;
smoothresult = smoothresult.* ...
    repmat(max_label,[size(smoothresult,1) 1]);
smoothresult(:,logfeat) = exp(smoothresult(:,logfeat) + ...
    log(0.0001))-0.0001;
smoothresult = smoothresult./repmat(sum(smoothresult,2) + ...
    double(sum(smoothresult,2)==0), [1 size(smoothresult,2)]);
smoothresult(:,2:end) = smoothresult(:,2:end).^(3/2);
smoothresult(:,1) = 1-sum(smoothresult(:,2:end),2);

% Save the file
savefile = 'prediction_3Dconvert_lassol2_norm.mat';
save(savefile,'featuretxt','poi_all','featmat_max','cv_fac', ...
    'lr_alpha','Beta_tot','Fit_tot','labelmatrix','featmat', ...
    'labelpredict_cv','predict_error','pattern_name','pattern_def', ...
    'pattern_time','predictresult','fg','fg_train','trainingcells', ...
    'labelmatrix_original','max_label','logfeat','smoothresult');

end
```

6.3.13 Protein kinetics based on the average of multiple cells

The prediction of the protein distribution was then registered into the mitotic standard time. A distribution trace was calculated for each cell at a temporal resolution of 15 seconds where the missing data points were interpolated linearly from the adjacent existing data points.

By multiplication with the total protein number within the cell, the number of protein molecules in each mitotic structures was then determined.

```
function P = distribution_kinetics(featuretxt,analysisfile)

tmax = 40;
tot_t = 237;
Data = readtable(featuretxt,'Delimiter','\t');
A = load(analysisfile);

% each cell distance to the mean query
meanassign = cell(length(A.poi_all),1);
assignmat_concat = meanassign;
```

```
assignmat_full = meanassign;
P = zeros(tot_t,6,length(A.poi_all));
for k = 1:length(A.poi_all);
poiname = A.poi_all{k};
idxlist = strcmp(Data.poi,poiname);
framelist = Data.path(idxlist>0);
celllist = unique(framelist);
analysis_result = A.smoothresult(idxlist>0,:);
cellnum = length(celllist);
matcomplete = nan(tot_t,size(analysis_result,2),cellnum);
matconcat = nan(tmax,1+size(analysis_result,2),cellnum);
for i = 1:cellnum;
cellframe = strcmp(framelist,celllist{i});
if sum(cellframe) == tmax;
% load the timeline of the cell and the calibrated protein image
load(fullfile(celllist{i},'Temporal_Align', 'mitotime.mat'));
TRlist = dir(fullfile(celllist{i},'Preprocessing', ...
    'Registration','*_T0*.mat'));
M = load(fullfile(celllist{i},'Preprocessing', ...
    'Registration',TRlist(1).name));
subresult = analysis_result(cellframe>0,:);
matconcat(1,1,i) = mitotime(1,1);
matconcat(1,2:end,i) = subresult(1,:)*sum(M.proc_poi(:));
matcomplete(mitotime(1,1),:,i) = matconcat(1,2:end,i);
counter = 1;
for j = 2:tmax;
M = load(fullfile(celllist{i},'Preprocessing', ...
    'Registration',TRlist(j).name));
matconcat(j,2:end,i) = subresult(j,:)*sum(M.proc_poi(:));
matconcat(j,1,i) = mitotime(j,1);
if isnan(matcomplete(mitotime(j,1),1,i));
    matcomplete(mitotime(j-1,1),:,i) = ...
        matcomplete(mitotime(j-1,1),:,i)/counter;
    matcomplete(mitotime(j,1),:,i) = matconcat(j,2:end,i);
    counter = 1;
else
    matcomplete(mitotime(j,1),:,i) = matcomplete(mitotime(j,1),:,i) ...
        + matconcat(j,2:end,i);
    counter = counter + 1;
end
```

```
end
matcomplete(mitotime(tmax,1),:,i) = ...
   matcomplete(mitotime(tmax,1),:,i)/counter;
isnnan = double(¬isnan(matcomplete(:,1,i)));
if isnan(matcomplete(1,1,i));
nextv = find(isnnan,1,'first');
matcomplete(1,:,i) = matcomplete(nextv,:,i);
end
% Linearly interpolate the data points between two measured data points
for j = 2:tot_t;
if isnan(matcomplete(j,1,i));
    nextv = find(isnnan(j+1:end),1,'first');
    if isempty(nextv);
        matcomplete(j,:,i) = matcomplete(j-1,:,i);
    else
        matcomplete(j,:,i) = (matcomplete(j-1,:,i)*nextv + ...
            matcomplete(j+nextv,:,i))/(nextv+1);
    end
end
end
end
end
% Average through all cells with the same protein
meanresult = nanmean(matcomplete,3);
meanassign{k} = meanresult;
assignmat_concat{k} = matconcat;
assignmat_full{k} = matcomplete;
P(:,:,k) = meanresult;
end
end
```

6.4 Abbreviation

H2B	histone protein 2B
Myrpalm	Myristoylation and Palmitoylation sequence
KIF11	kinesin family 11
MIS12	Protein MIS12 homolog
TUBB2C	Tubulin beta-4B chain
RACGAP1	Rac GTPase-activating protein 1

NEDD1	Protein NEDD1
AURKB	Aurora kinase B
NUP214	Nuclear pore complex protein Nup214
PLK1	Serine/threonine-protein kinase PLK1
CDCA8	Borealin
CENPA	Histone H3-like centromeric protein A
BUB1/3	budding uninhibited by benzimidazoles 1/3
NES	nuclear export signal
MAD1/2	Mitotic spindle assembly checkpoint protein MAD1/MAD2A
BUBR1	Mitotic checkpoint serine/threonine-protein kinase BUB1 beta
CDC20	Cell division cycle protein 20 homolog
CDK1	Cyclin-dependent kinase 1
KNL1	Cancer susceptibility candidate gene 5 protein
NDC80	Kinetochore protein NDC80 homolog
DNA	Deoxyribonucleic acid
RNA	Ribonucleic acid
siRNA	Small interfering RNA
gRNA	Guide RNA
RNAi	RNA interference
cDNA	complementary DNA
BAC	Bacterial artificial chromosomes
ZFN	Zinc finger nuclease
TALEN	Transcription activator-like effector nuclease
CrispR	Clustered regularly interspaced short palindromic repeats
LAP	localization and affinity purification tag
DMEM	Dulbecco's modified Eagle's medium
PBS	Phosphate buffered saline
FBS	Fetal bovine serum
DMSO	Dimethyl sulfoxide
PFA	Paraformaldehyde
BSA	Bovine serum albumin
NEBD	nuclear envelop break down
APC/C	anaphase-promoting complexes/cyclosome
MCC	mitotic checkpoint complex
CPC	chromosomal passenger complex
SAC	spindle assembly checkpoint
mEGFP	monomeric enhanced green fluorescence protein
mCherry	monomeric cherry fluorescence protein
mCer3	monomeric cerulean3 fluorescence protein
GaASP	Gallium arsenide phosphide detector

APD	Avalanche photodiode
NA	Numerical aperture
DIC	Differential interference contrast
CPM	count per molecule
FCS	fluorescence correlation spectroscopy
FCCS	fluorescence cross correlation spectroscopy
FRET	fluorescence resonance energy transfer
POI	protein of interest
ROI	region of interest
2D	two-dimensional
3D	three-dimensional
PCA	principle component analysis
SURF	speeded up robust features
IP	interest point
SVM	support vector machine
NTF	non-negative tensor factorization
NMF	non-negative matrix factorization
TCC	Tucker's congruence coefficient
uLBP	uniform local binary pattern

Bibliography

Aebersold, R. and M. Mann (2003): ''Mass spectrometry-based proteomics,'' *Nature*, 422, 198--207.

Alberts, B., A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter (1997): *Molecular Biology of the Cell*, Garland Science, New York, 2002.

Ando, J., K. Sugimoto, K. Tamayose, M. Sasaki, M. Ando, and K. Oshimi (2008): ''Changes in cell morphology and cytoskeletal organization are induced by human mitotic checkpoint gene, Bub1,'' *Biochemical and biophysical research communications*, 365, 691--697.

Bacia, K., S. A. Kim, and P. Schwille (2006): ''Fluorescence cross-correlation spectroscopy in living cells,'' *Nature methods*, 3, 83--89.

Barton, G. J. and M. J. Sternberg (1987): ''A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons,'' *Journal of molecular biology*, 198, 327--337.

Bay, H., T. Tuytelaars, and L. Van Gool (2006): ''Surf: Speeded up robust features,'' in *Computer vision--ECCV 2006*, Springer, 404--417.

Beck, M., A. Schmidt, J. Malmstroem, M. Claassen, A. Ori, A. Szymborska, F. Herzog, O. Rinner, J. Ellenberg, and R. Aebersold (2011): ''The quantitative proteome of a human cell line,'' *Molecular systems biology*, 7, 549.

Boeke, D., S. Trautmann, M. Meurer, M. Wachsmuth, C. Godlee, M. Knop, and M. Kaksonen (2014): ''Quantification of cytosolic interactions identifies Ede1 oligomers as key organizers of endocytosis,'' *Molecular systems biology*, 10, 756.

Boland, M. V. and R. F. Murphy (1999): ''Automated analysis of patterns in fluorescence-microscope images,'' *Trends in cell biology*, 9, 201--202.

--------- (2001): ''A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells,'' *Bioinformatics*, 17, 1213--1223.

Bootsma, D., L. Budke, and O. Vos (1964): ''Studies on synchronous division of tissue culture cells initiated by excess thymidine,'' *Experimental cell research*, 33, 301--309.

Boyarchuk, Y., A. Salic, M. Dasso, and A. Arnaoutov (2007): ''Bub1 is essential for assembly of the functional inner centromere,'' *The Journal of cell biology*, 176, 919--928.

Brown, R. G. and A. M. Gleason (2004): *Advanced mathematics: precalculus with discrete mathematics and data analysis*, Recording for the Blind & Dyslexic.

Cadart, C., E. Zlotek-Zlotkiewicz, M. Le Berre, M. Piel, and H. K. Matthews (2014): ''Exploring the function of cell shape and size during mitosis,'' *Developmental cell*, 29, 159--169.

Campbell, N. A. and J. Reese (2003): *Biologie*, Spektrum Verlag Heidelberg.

Carmena, M., M. Wheelock, H. Funabiki, and W. C. Earnshaw (2012): ''The chromosomal passenger complex (CPC): from easy rider to the godfather of mitosis,'' *Nature reviews Molecular cell biology*, 13, 789--803.

Carpenter, A. E., T. R. Jones, M. R. Lamprecht, C. Clarke, I. H. Kang, O. Friman, D. A. Guertin, J. H. Chang, R. A. Lindquist, J. Moffat, et al. (2006): ''CellProfiler: image analysis software for identifying and quantifying cell phenotypes,'' *Genome biology*, 7, R100.

Chang, C.-C. and C.-J. Lin (2011): ''LIBSVM: A library for support vector machines,'' *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2, 27.

Charnley, M., F. Anderegg, R. Holtackers, M. Textor, and P. Meraldi (2013): ''Effect of cell shape and dimensionality on spindle orientation and mitotic timing,'' *PloS one*, 8, e66918.

Chebira, A., Y. Barbotin, C. Jackson, T. Merryman, G. Srinivasa, R. F. Murphy, and J. Kovačević (2007): ''A multiresolution approach to automated classification of protein subcellular location images,'' *BMC bioinformatics*, 8, 210.

Chen, X. and R. F. Murphy (2004): ''Robust classification of subcellular location patterns in high resolution 3D fluorescence microscope images,'' in *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, IEEE, vol. 1, 1632--1635.

Chong, Y. T., J. L. Koh, H. Friesen, K. Duffy, M. J. Cox, A. Moses, J. Moffat, C. Boone, and B. J. Andrews (2015): ''Yeast proteome dynamics from single cell imaging and automated analysis,'' *Cell*, 161, 1413--1424.

Cichocki, A., R. Zdunek, A. H. Phan, and S.-i. Amari (2009): *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons.

Cleveland, D. W., Y. Mao, and K. F. Sullivan (2003): ''Centromeres and kinetochores: from epigenetics to mitotic checkpoint signaling,'' *Cell*, 112, 407--421.

Coelho, L. P., J. D. Kangas, A. W. Naik, E. Osuna-Highley, E. Glory-Afshar, M. Fuhrman, R. Simha, P. B. Berget, J. W. Jarvik, and R. F. Murphy (2013): ''Determining the subcellular location of new proteins from microscope images using local features,'' *Bioinformatics*, 29, 2343--2349.

Coelho, L. P., T. Peng, and R. F. Murphy (2010): ''Quantifying the distribution of probes between subcellular locations using unsupervised pattern unmixing,'' *Bioinformatics*, 26, 7--12.

Conrad, C., H. Erfle, P. Warnat, N. Daigle, T. Lörch, J. Ellenberg, R. Pepperkok, and R. Eils (2004): ''Automatic identification of subcellular phenotypes on human cell arrays,'' *Genome research*, 14, 1130--1136.

Conrad, C., A. Wünsche, T. H. Tan, J. Bulkescher, F. Sieckmann, F. Verissimo, A. Edelstein, T. Walter, U. Liebel, R. Pepperkok, et al. (2011): ''Micropilot: automation of fluorescence microscopy-based imaging for systems biology,'' *Nature methods*, 8, 246--249.

Cooper Geoffrey, M. (2000): *The cell a molecular approach*, ASM Press.

Csurka, G., C. Dance, L. Fan, J. Willamowski, and C. Bray (2004): ''Visual categorization with bags of keypoints,'' in *Workshop on statistical learning in computer vision, ECCV*, Prague, vol. 1, 1--2.

Danuser, G. (2011): ''Computer vision in cell biology,'' *Cell*, 147, 973--978.

De Souza, C. P. and S. A. Osmani (2007): ''Mitosis, not just open or closed,'' *Eukaryotic Cell*, 6, 1521--1527.

Dephoure, N., C. Zhou, J. Villén, S. A. Beausoleil, C. E. Bakalarski, S. J. Elledge, and S. P. Gygi (2008): ''A quantitative atlas of mitotic phosphorylation,'' *Proceedings of the National Academy of Sciences*, 105, 10762--10767.

Deysson, G. (1968): ''Antimitotic substances,'' *Int. Rev. Cytol*, 24, 99--148.

Doxsey, S., D. McCollum, and W. Theurkauf (2005): ''Centrosomes in cellular regulation,'' *Annu. Rev. Cell Dev. Biol.*, 21, 411--434.

Dulla, K. and A. Santamaria (2011): ''Large-Scale Mitotic Cell Synchronization,'' in *Cell Cycle Synchronization*, Springer, 65--74.

Dultz, E., E. Zanin, C. Wurzenberger, M. Braun, G. Rabut, L. Sironi, and J. Ellenberg (2008): ''Systematic kinetic analysis of mitotic dis-and reassembly of the nuclear pore in living cells,'' *The Journal of cell biology*, 180, 857--865.

Dumont, J., K. Oegema, and A. Desai (2010): ''A kinetochore-independent mechanism drives anaphase chromosome separation during acentrosomal meiosis,'' *Nature cell biology*, 12, 894--901.

Dunn, K. W., M. M. Kamocka, and J. H. McDonald (2011): ''A practical guide to evaluating colocalization in biological microscopy,'' *American Journal of Physiology-Cell Physiology*, 300, C723--C742.

El-Labban, A., A. Zisserman, Y. Toyoda, A. Bird, and A. Hyman (2011): ''Dynamic time warping for automated cell cycle labelling,'' *Microscopic Image Analysis with Applications in Biology*.

Elvin, P. and C. Evans (1984): ''Cell adhesiveness and the cell cycle: correlation in synchronized Balb/c 3T3 cells,'' *Biology of the Cell*, 48, 1--9.

Eot-Houllier, G., M. Venoux, S. Vidal-Eychenié, M.-T. Hoang, D. Giorgi, and S. Rouquier (2010): ''Plk1 regulates both ASAP localization and its role in spindle pole integrity,'' *Journal of Biological Chemistry*, 285, 29556--29568.

Ester, M., H.-P. Kriegel, J. Sander, and X. Xu (1996): ''A density-based algorithm for discovering clusters in large spatial databases with noise.'' in *Kdd*, vol. 96, 226--231.

Fay, F. S., K. L. Taneja, S. Shenoy, L. Lifshitz, and R. H. Singer (1997): ''Quantitative digital analysis of diffuse and concentrated nuclear distributions of nascent transcripts, SC35 and poly (A),'' *Experimental cell research*, 231, 27--37.

Fink, J., N. Carpi, T. Betz, A. Bétard, M. Chebah, A. Azioune, M. Bornens, C. Sykes, L. Fetler, D. Cuvelier, et al. (2011): ''External forces control mitotic spindle positioning,'' *Nature cell biology*, 13, 771--778.

Fink, J., M. Théry, A. Azioune, R. Dupont, F. Chatelain, M. Bornens, and M. Piel (2007): ''Comparative study and improvement of current cell micro-patterning techniques,'' *Lab on a Chip*, 7, 672--680.

Gascoigne, K. E. and I. M. Cheeseman (2013): ''CDK-dependent phosphorylation and nuclear exclusion coordinately control kinetochore assembly state,'' *The Journal of cell biology*, 201, 23--32.

Gassmann, R., A. Carvalho, A. J. Henzing, S. Ruchaud, D. F. Hudson, R. Honda, E. A. Nigg, D. L. Gerloff, and W. C. Earnshaw (2004): ''Borealin a novel chromosomal passenger required for stability of the bipolar mitotic spindle,'' *The Journal of cell biology*, 166, 179--191.

Glotzer, M., N. Kraut, J. Peters, et al. (2007): ''Polo-like kinase 1 triggers the initiation of cytokinesis in human cells by promoting recruitment of the RhoGEF Ect2 to the central spindle,'' *Dev. Cell*, 12, 713--725.

Goranov, A. I. and A. Amon (2010): ''Growth and division: not a one-way road,'' *Current opinion in cell biology*, 22, 795--800.

Grabsch, H. I., J. M. Askham, E. E. Morrison, N. Pomjanski, K. Lickvers, W. J. Parsons, A. Boecking, H. E. Gabbert, and W. Mueller (2004): ''Expression of BUB1 protein in gastric cancer correlates with the histological subtype, but not with DNA ploidy or microsatellite instability,'' *The Journal of pathology*, 202, 208--214.

Hamilton, N. A., R. S. Pantelic, K. Hanson, and R. D. Teasdale (2007): ''Fast automated cell phenotype image classification,'' *BMC bioinformatics*, 8, 110.

Handfield, L.-F., Y. T. Chong, J. Simmons, B. J. Andrews, and A. M. Moses (2013): ''Unsupervised clustering of subcellular protein expression patterns in high-throughput microscopy images reveals protein complexes and functional relationships between proteins,'' *PLOS Computational Biology*.

Hansen, D. V., A. V. Loktev, K. H. Ban, and P. K. Jackson (2004): ''Plk1 regulates activation of the anaphase promoting complex by phosphorylating and triggering SCFβTrCP-dependent destruction of the APC inhibitor Emi1,'' *Molecular biology of the cell*, 15, 5623--5634.

Haralick, R. M., K. Shanmugam, and I. H. Dinstein (1973): ''Textural features for image classification,'' *Systems, Man and Cybernetics, IEEE Transactions on*, 610--621.

Harder, N., F. Mora-Bermúdez, W. J. Godinez, A. Wünsche, R. Eils, J. Ellenberg, and K. Rohr (2009): ''Automatic analysis of dividing cells in live cell movies to detect mitotic delays and correlate phenotypes in time,'' *Genome research*, 19, 2113--2124.

Hasson, D., T. Panchenko, K. J. Salimian, M. U. Salman, N. Sekulic, A. Alonso, P. E. Warburton, and B. E. Black (2013): ''The octamer is the major form of CENP-A nucleosomes at human centromeres,'' *Nature structural & molecular biology*, 20, 687--695.

Hauf, S. and Y. Watanabe (2004): ''Kinetochore orientation in mitosis and meiosis,'' *Cell*, 119, 317--327.

Heikkilä, M. and M. Pietikäinen (2006): "A texture-based method for modeling the background and detecting moving objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28, 657--662.

Held, M., M. H. Schmitz, B. Fischer, T. Walter, B. Neumann, M. H. Olma, M. Peter, J. Ellenberg, and D. W. Gerlich (2010): "CellCognition: time-resolved phenotype annotation in high-throughput live cell imaging," *Nature methods*, 7, 747--754.

Herbert, M., J. Wolstenholme, A. Murdoch, and T. Butler (1995): "Mitotic activity during preimplantation development of human embryos," *Journal of reproduction and fertility*, 103, 209--214.

Hériché, J.-K., J. G. Lees, I. Morilla, T. Walter, B. Petrova, M. J. Roberti, M. J. Hossain, P. Adler, J. M. Fernández, M. Krallinger, et al. (2014): "Integration of biological data by kernels on graph nodes allows prediction of new genes involved in mitotic chromosome condensation," *Molecular biology of the cell*, 25, 2522--2536.

Ho, Y., A. Gruhler, A. Heilbut, G. D. Bader, L. Moore, S.-L. Adams, A. Millar, P. Taylor, K. Bennett, K. Boutilier, et al. (2002): "Systematic identification of protein complexes in Saccharomyces cerevisiae by mass spectrometry," *Nature*, 415, 180--183.

Horton, J. S., C. T. Wakano, M. Speck, and A. J. Stokes (2015): "Two-pore channel 1 interacts with citron kinase, regulating completion of cytokinesis," *Channels*, 9, 21--29.

Hu, Y., E. Osuna-Highley, J. Hua, T. S. Nowicki, R. Stolz, C. McKayle, and R. F. Murphy (2010): "Automated analysis of protein subcellular location in time series images," *Bioinformatics*, 26, 1630--1636.

Huang, K., M. Velliste, and R. F. Murphy (2003): "Feature reduction for improved recognition of subcellular location patterns in fluorescence microscope images," in *Biomedical Optics 2003*, International Society for Optics and Photonics, 307--318.

Hutchins, J. R., Y. Toyoda, B. Hegemann, I. Poser, J.-K. Hériché, M. M. Sykora, M. Augsburg, O. Hudecz, B. A. Buschhorn, J. Bulkescher, et al. (2010): "Systematic analysis of human protein complexes identifies chromosome segregation proteins," *Science*, 328, 593--599.

Ikegami, S., T. TAGUCHI, M. OHASHI, M. OGURO, H. NAGANO, and Y. MANO (1978): "Aphidicolin prevents mitotic cell division by interfering with the activity of DNA polymerase-α," *Nature*, 275, 458--460.

Ji, L., J. Lim, and G. Danuser (2008): "Fluctuations of intracellular forces during cell protrusion," *Nature cell biology*, 10, 1393--1400.

Jia, L., S. Kim, and H. Yu (2013): ''Tracking spindle checkpoint signals from kinetochores to APC/C,'' *Trends in biochemical sciences*, 38, 302--311.

Johnson, V. L., M. I. Scott, S. V. Holt, D. Hussein, and S. S. Taylor (2004): ''Bub1 is required for kinetochore localization of BubR1, Cenp-E, Cenp-F and Mad2, and chromosome congression,'' *Journal of cell science*, 117, 1577--1589.

Johnston, O., A. Nagaraja, H. Hochheiser, and I. Goldberg (2006): ''A flexible framework for web interfaces to image databases: supporting user-defined ontologies and links to external databases,'' in *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, IEEE, 1380--1383.

Jolliffe, I. (2002): *Principal component analysis*, Wiley Online Library.

Kafri, R., J. Levy, M. B. Ginzberg, S. Oh, G. Lahav, and M. W. Kirschner (2013): ''Dynamics extracted from fixed cells reveal feedback linking cell growth to cell cycle,'' *Nature*, 494, 480--483.

Karsenti, E. and I. Vernos (2001): ''The mitotic spindle: a self-made machine,'' *Science*, 294, 543--547.

Kelly, A. E. and H. Funabiki (2009): ''Correcting aberrant kinetochore microtubule attachments: an Aurora B-centric view,'' *Current opinion in cell biology*, 21, 51--58.

Keys, R. G. (1981): ''Cubic convolution interpolation for digital image processing,'' *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 29, 1153--1160.

Kim, M. and G. D. Kao (2005): ''Newly identified roles for an old guardian: profound deficiency of the mitotic spindle checkpoint protein BubR1 leads to early aging and infertility,'' *Cancer biology & therapy*, 4, 172--173.

Kitajima, T. S., M. Ohsugi, and J. Ellenberg (2011): ''Complete kinetochore tracking reveals error-prone homologous chromosome biorientation in mammalian oocytes,'' *Cell*, 146, 568--581.

Kitano, H. (2007): ''Scientific challenges in systems biology,'' in *Introduction to Systems Biology*, Springer, 3--13.

Kito, K. and T. Ito (2007): ''Methods for Protein-Protein Interaction Analysis,'' in *Introduction to Systems Biology*, Springer, 160--182.

Kittler, R., L. Pelletier, C. Ma, I. Poser, S. Fischer, A. A. Hyman, and F. Buchholz (2005): ''RNA interference rescue by bacterial artificial chromosome transgenesis in mammalian tissue culture cells,'' *Proceedings of the National academy of Sciences of the United States of America*, 102, 2396--2401.

Kiyomitsu, T. and I. M. Cheeseman (2013): ''Cortical dynein and asymmetric membrane elongation coordinately position the spindle in anaphase,'' *Cell*, 154, 391--402.

Kukulski, W., M. Schorb, S. Welsch, A. Picco, M. Kaksonen, and J. Briggs (2012): ''Precise, correlated fluorescence microscopy and electron tomography of lowicryl sections using fluorescent fiducial markers,'' *Methods Cell Biol*, 111, 235--257.

Lan, W. and D. W. Cleveland (2010): ''A chemical tool box defines mitotic and interphase roles for Mps1 kinase,'' *The Journal of cell biology*, 190, 21--24.

Lara-Gonzalez, P., F. G. Westhorpe, and S. S. Taylor (2012): ''The spindle assembly checkpoint,'' *Current Biology*, 22, R966--R980.

Laufer, C., B. Fischer, M. Billmann, W. Huber, and M. Boutros (2013): ''Mapping genetic interactions in human cancer cells with RNAi and multiparametric phenotyping,'' *Nature methods*, 10, 427--431.

Lázaro-Diéguez, F., I. Ispolatov, and A. Müsch (2015): ''Cell shape impacts on the positioning of the mitotic spindle with respect to the substratum,'' *Molecular biology of the cell*, 26, 1286--1295.

Lazebnik, S., C. Schmid, and J. Ponce (2005): ''A sparse texture representation using local affine regions,'' *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27, 1265--1278.

Lee, D. D. and H. S. Seung (1999): ''Learning the parts of objects by non-negative matrix factorization,'' *Nature*, 401, 788--791.

Lens, S. M., R. M. Wolthuis, R. Klompmaker, J. Kauw, R. Agami, T. Brummelkamp, G. Kops, and R. H. Medema (2003): ''Survivin is required for a sustained spindle checkpoint arrest in response to lack of tension,'' *The EMBO journal*, 22, 2934--2947.

Li, J., J. Y. Newberg, M. Uhlén, E. Lundberg, and R. F. Murphy (2012): ''Automated analysis and reannotation of subcellular locations in confocal images from the human protein atlas,'' *PLOS One*.

Lo, Y.-C., S. Senese, C.-M. Li, Q. Hu, Y. Huang, R. Damoiseaux, and J. Z. Torres (2015): ''Large-Scale Chemical Similarity Networks for Target Profiling of Compounds Identified in Cell-Based Chemical Screens,'' *PLoS computational biology*, 11, e1004153--e1004153.

Loo, L.-H., H.-J. Lin, R. J. Steininger, Y. Wang, L. F. Wu, and S. J. Altschuler (2009): ''An approach for extensibly profiling the molecular states of cellular subpopulations,'' *Nature methods*, 6, 759--765.

Ma, H. T. and R. Y. Poon (2011): ''Synchronization of HeLa cells,'' in *Cell Cycle Synchronization*, Springer, 151--161.

Maciejowski, J., K. A. George, M.-E. Terret, C. Zhang, K. M. Shokat, and P. V. Jallepalli (2010): ''Mps1 directs the assembly of Cdc20 inhibitory complexes during interphase and mitosis to control M phase timing and spindle checkpoint signaling,'' *The Journal of cell biology*, 190, 89--100.

Mahen, R., B. Koch, M. Wachsmuth, A. Z. Politi, A. Perez-Gonzalez, J. Mergenthaler, Y. Cai, and J. Ellenberg (2014): ''Comparative assessment of fluorescent transgene methods for quantitative imaging in human cells,'' *Molecular biology of the cell*, 25, 3610--3618.

Maliga, Z., M. Junqueira, Y. Toyoda, A. Ettinger, F. Mora-Bermúdez, R. W. Klemm, A. Vasilj, E. Guhr, I. Ibarlucea-Benitez, I. Poser, et al. (2013): ''A genomic toolkit to investigate kinesin and myosin motor function in cells,'' *Nature cell biology*, 15, 325--334.

Manders, E., J. Stap, G. Brakenhoff, R. Van Driel, and J. Aten (1992): ''Dynamics of three-dimensional replication patterns during the S-phase, analysed by double labelling of DNA and confocal microscopy,'' *Journal of cell science*, 103, 857--862.

Meyer, F. (1994): ''Topographic distance and watershed lines,'' *Signal processing*, 38, 113--125.

Michelman-Ribeiro, A., D. Mazza, T. Rosales, T. J. Stasevich, H. Boukari, V. Rishi, C. Vinson, J. R. Knutson, and J. G. McNally (2010): ''Measurement of Protein Binding Rates in Live Cells with FCS,'' *Biophysical Journal*, 98, 234a.

Mickoleit, M., B. Schmid, M. Weber, F. O. Fahrbach, S. Hombach, S. Reischauer, and J. Huisken (2014): ''High-resolution reconstruction of the beating zebrafish heart,'' *nature methods*.

Morgan, D. O. (2007): *The cell cycle: principles of control*, New Science Press.

Müller, M. (2007): ''Dynamic time warping,'' *Information retrieval for music and motion*, 69--84.

Murata-Hori, M., M. Tatsuka, and Y.-L. Wang (2002): ''Probing the dynamics and functions of aurora B kinase in living cells during mitosis and cytokinesis,'' *Molecular biology of the cell*, 13, 1099--1108.

Murphy, R. F. (2010): ''Communicating subcellular distributions,'' *Cytometry Part A*, 77, 686--692.

Musacchio, A. and E. D. Salmon (2007): ''The spindle-assembly checkpoint in space and time,'' *Nature reviews Molecular cell biology*, 8, 379--393.

Myers, C. S. and L. R. Rabiner (1981): ''A Comparative Study of Several Dynamic Time-Warping Algorithms for Connected-Word Recognition,'' *Bell System Technical Journal*, 60, 1389--1409.

Myers, G. (2012): ''Why bioimage informatics matters,'' *Nature methods*, 9, 659--660.

Nanni, L., A. Lumini, Y.-S. Lin, C.-N. Hsu, and C.-C. Lin (2010): ''Fusion of systems for automated cell phenotype image classification,'' *Expert Systems with Applications*, 37, 1556--1562.

Neumann, B., T. Walter, J.-K. Heriche, J. Bulkescher, H. Erfle, C. Conrad, P. Rogers, I. Poser, M. Held, U. Liebel, et al. (2010): ''Phenotypic profiling of the human genome by time-lapse microscopy reveals cell division genes,'' *Nature*, 464, 721--727.

Nguyen, P. A., A. C. Groen, M. Loose, K. Ishihara, M. Wühr, C. M. Field, and T. J. Mitchison (2014): ''Spatial organization of cytokinesis signaling reconstituted in a cell-free system,'' *Science*, 346, 244--247.

Nigg, E. A. (2001): ''Mitotic kinases as regulators of cell division and its checkpoints,'' *Nature reviews Molecular cell biology*, 2, 21--32.

Noatynska, A., M. Gotta, and P. Meraldi (2012): ''Mitotic spindle (DIS) orientation and DISease: cause or consequence?'' *The Journal of cell biology*, 199, 1025--1035.

Nunez-Iglesias, J., R. Kennedy, S. M. Plaza, A. Chakraborty, and W. T. Katz (2014): ''Graph-based active learning of agglomeration (GALA): a Python library to segment 2D and 3D neuroimages,'' *Frontiers in neuroinformatics*, 8.

O'Connell, C. B. and Y.-l. Wang (2000): ''Mammalian spindle orientation and position respond to changes in cell shape in a dynein-dependent fashion,'' *Molecular Biology of the Cell*, 11, 1765--1774.

Otsu, N. (1975): ''A threshold selection method from gray-level histograms,'' *Automatica*, 11, 23--27.

Pandey, A. and M. Mann (2000): ''Proteomics to study genes and genomes,'' *Nature*, 405, 837--846.

Parthasarathy, R. (2012): ''Rapid, accurate particle tracking by calculation of radial symmetry centers,'' *Nature methods*, 9, 724--726.

Paweletz, N. (2001): ''Walther Flemming: pioneer of mitosis research,'' *Nature Reviews Molecular Cell Biology*, 2, 72--76.

Peng, H. (2008): ''Bioimage informatics: a new area of engineering biology,'' *Bioinformatics*, 24, 1827--1836.

Peng, T., G. M. Bonamy, E. Glory-Afshar, D. R. Rines, S. K. Chanda, and R. F. Murphy (2010): ''Determining the distribution of probes between different subcellular locations through automated unmixing of subcellular patterns,'' *Proceedings of the National Academy of Sciences*, 107, 2944--2949.

Peng, T. and R. F. Murphy (2011): ''Image-derived, three-dimensional generative models of cellular organization,'' *Cytometry Part A*, 79, 383--391.

Petrovic, A., S. Pasqualato, P. Dube, V. Krenn, S. Santaguida, D. Cittaro, S. Monzani, L. Massimiliano, J. Keller, A. Tarricone, et al. (2010): ''The MIS12 complex is a protein interaction hub for outer kinetochore assembly,'' *The Journal of cell biology*, 190, 835--852.

Poser, I., M. Sarov, J. R. Hutchins, J.-K. Hériché, Y. Toyoda, A. Pozniakovsky, D. Weigl, A. Nitzsche, B. Hegemann, A. W. Bird, et al. (2008): ''BAC TransgeneOmics: a high-throughput method for exploration of protein function in mammals,'' *Nature methods*, 5, 409--415.

Pramanik, A. (2004): ''Ligand-receptor interactions in live cells by fluorescence correlation spectroscopy,'' *Current pharmaceutical biotechnology*, 5, 205--212.

Qi, W., Z. Tang, and H. Yu (2006): ''Phosphorylation-and polo-box--dependent binding of Plk1 to Bub1 is required for the kinetochore localization of Plk1,'' *Molecular biology of the cell*, 17, 3705--3716.

Raaijmakers, J. A., R. G. van Heesbeen, J. L. Meaders, E. F. Geers, B. Fernandez-Garcia, R. H. Medema, and M. E. Tanenbaum (2012): ''Nuclear envelope-associated dynein drives prophase centrosome separation and enables Eg5-independent bipolar spindle formation,'' *The EMBO journal*, 31, 4179--4190.

Reece, J., L. A. Urry, N. Meyers, M. L. Cain, S. A. Wasserman, P. V. Minorsky, R. B. Jackson, and B. N. Cooke (2011): *Campbell biology*, Pearson Higher Education AU.

Rieder, C. L. and E. Salmon (1998): ''The vertebrate cell kinetochore and its roles during mitosis,'' *Trends in cell biology*, 8, 310--318.

Roberts, B. T., K. A. Farr, and M. A. Hoyt (1994): ''The Saccharomyces cerevisiae checkpoint gene BUB1 encodes a novel protein kinase.'' *Molecular and Cellular Biology*, 14, 8282--8291.

Ronneberger, O., K. Liu, M. Rath, D. Rueβ, T. Mueller, H. Skibbe, B. Drayer, T. Schmidt, A. Filippi, R. Nitschke, et al. (2012): ''ViBE-Z: a framework for 3D virtual colocalization analysis in zebrafish larval brains,'' *Nature Methods*, 9, 735--742.

Samuels, L., W. Kisieleski, and R. Baserga (1964): ''Tritiated thymidine toxicity in mammalian systems,'' *Atompraxis (West Germany) Incorporated in Kerntechnik*, 10.

Satisbury, J. L. (1995): ''Centrin, centrosomes, and mitotic spindle poles,'' *Current opinion in cell biology*, 7, 39--45.

Schauer, K., T. Duong, K. Bleakley, S. Bardin, M. Bornens, and B. Goud (2010): ''Probabilistic density maps to study global endomembrane organization,'' *nAture methods*, 7, 560--566.

Schauer, K., J.-P. Grossier, T. Duong, V. Chapuis, S. Degot, A. Lescure, E. Del Nery, and B. Goud (2013): ''A Novel Organelle Map Framework for High-Content Cell Morphology Analysis in High Throughput,'' *Journal of biomolecular screening*, 1087057113497399.

Schmitz, M. H., M. Held, V. Janssens, J. R. Hutchins, O. Hudecz, E. Ivanova, J. Goris, L. Trinkle-Mulcahy, A. I. Lamond, I. Poser, et al. (2010): ''Live-cell imaging RNAi screen identifies PP2A-B55 [alpha] and importin-[beta] 1 as key mitotic exit regulators in human cells,'' *Nature cell biology*, 12, 886--893.

Schwille, P. (2001): ''Fluorescence correlation spectroscopy and its potential for intracellular applications,'' *Cell biochemistry and biophysics*, 34, 383--408.

Scott, K. C. and K. S. Bloom (2014): ''Lessons learned from counting molecules: how to lure CENP-A into the kinetochore,'' *Open biology*, 4, 140191.

Sekar, R. B. and A. Periasamy (2003): ''Fluorescence resonance energy transfer (FRET) microscopy imaging of live cell protein localizations,'' *The Journal of cell biology*, 160, 629--633.

Shao, H., Y. Huang, L. Zhang, K. Yuan, Y. Chu, Z. Dou, C. Jin, M. Garcia-Barrio, X. Liu, and X. Yao (2015): ''Spatiotemporal dynamics of Aurora B-PLK1-MCAK signaling axis orchestrates kinetochore bi-orientation and faithful chromosome segregation,'' *Scientific reports*, 5.

Shariff, A., R. F. Murphy, and G. K. Rohde (2010): ''A generative model of microtubule distributions, and indirect estimation of its parameters from fluorescence microscopy images,'' *Cytometry Part A*, 77, 457--466.

Sharp-Baker, H. and R.-H. Chen (2001): ''Spindle checkpoint protein Bub1 is required for kinetochore localization of Mad1, Mad2, Bub3, and CENP-E, independently of its kinase activity,'' *The Journal of cell biology*, 153, 1239--1250.

Shima, D. T., N. Cabrera-Poch, R. Pepperkok, and G. Warren (1998): ''An ordered inheritance strategy for the Golgi apparatus: visualization of mitotic disassembly reveals a role for the mitotic spindle,'' *The Journal of cell biology*, 141, 955--966.

Sommer, C. and D. W. Gerlich (2013): ''Machine learning in cell biology--teaching computers to recognize phenotypes,'' *Journal of cell science*, 126, 5529--5539.

Sommer, C., M. Held, B. Fischer, W. Huber, and D. W. Gerlich (2013): ''CellH5: a format for data exchange in high-content screening,'' *Bioinformatics*, btt175.

Sprague, B. L., C. G. Pearson, P. S. Maddox, K. S. Bloom, E. Salmon, and D. J. Odde (2003): ''Mechanisms of microtubule-based kinetochore positioning in the yeast metaphase spindle,'' *Biophysical journal*, 84, 3529--3546.

Strambio-De-Castillia, C., M. Niepel, and M. P. Rout (2010): ''The nuclear pore complex: bridging nuclear transport and gene regulation,'' *Nature reviews Molecular cell biology*, 11, 490--501.

Stryer, L. and R. P. Haugland (1967): ''Energy transfer: a spectroscopic ruler.'' *Proceedings of the National Academy of Sciences of the United States of America*, 58, 719.

Szymanski, J. E. (1989): *Basic mathematics for electronic engineers: models and applications*, vol. 16, Taylor & Francis.

Tagwerker, C., K. Flick, M. Cui, C. Guerrero, Y. Dou, B. Auer, P. Baldi, L. Huang, and P. Kaiser (2006): ''A Tandem Affinity Tag for Two-step Purification under Fully Denaturing Conditions Application in Ubiquitin Profiling and Protein Complex Identification Combined with in vivoCross-Linking,'' *Molecular & Cellular Proteomics*, 5, 737--748.

Tahir, M., A. Khan, and A. Majid (2012): ''Protein subcellular localization of fluorescence imagery using spatial and transform domain features,'' *Bioinformatics*, 28, 91--97.

Tanaka, K., E. Kitamura, and T. U. Tanaka (2010): ''Live-cell analysis of kinetochore--microtubule interaction in budding yeast,'' *Methods*, 51, 206--213.

Ten Holt, G., M. Reinders, and E. Hendriks (2007): ''Multi-dimensional dynamic time warping for gesture recognition,'' .

Théry, M. (2010): ''Micropatterning as a tool to decipher cell morphogenesis and functions,'' *Journal of cell science*, 123, 4201--4213.

Théry, M. and M. Piel (2009): ''Adhesive micropatterns for cells: a microcontact printing protocol,'' *Cold Spring Harbor Protocols*, 2009, pdb--prot5255.

Théry, M., V. Racine, A. Pépin, M. Piel, Y. Chen, J.-B. Sibarita, and M. Bornens (2005): ''The extracellular matrix guides the orientation of the cell division axis,'' *Nature cell biology*, 7, 947--953.

Thummel, R., S. C. Kassen, J. E. Montgomery, J. M. Enright, and D. R. Hyde (2008): ''Inhibition of Müller glial cell division blocks regeneration of the light-damaged zebrafish retina,'' *Developmental neurobiology*, 68, 392.

Tran, T. N., K. Drab, and M. Daszykowski (2013): ''Revised DBSCAN algorithm to cluster data with dense adjacent clusters,'' *Chemometrics and Intelligent Laboratory Systems*, 120, 92--96.

Tucker, L. R. (1951): ''A method for synthesis of factor analysis studies,'' *Personnel Research Section Report No.984*.

Uzunova, K., B. T. Dye, H. Schutz, R. Ladurner, G. Petzold, Y. Toyoda, M. A. Jarvis, N. G. Brown, I. Poser, M. Novatchkova, et al. (2012): ''APC15 mediates CDC20 autoubiquitylation by APC/CMCC and disassembly of the mitotic checkpoint complex,'' *Nature structural & molecular biology*, 19, 1116--1123.

van der Waal, M. S., R. C. Hengeveld, A. van der Horst, and S. M. Lens (2012): ''Cell division control by the Chromosomal Passenger Complex,'' *Experimental cell research*, 318, 1407--1420.

van Dongen, S. F., P. Maiuri, E. Marie, C. Tribet, and M. Piel (2013): ''Triggering cell adhesion, migration or shape change with a dynamic surface coating,'' *Advanced Materials*, 25, 1687--1691.

van Steensel, B., E. P. van Binnendijk, C. D. Hornsby, H. Van der Voort, Z. S. Krozowski, E. R. de Kloet, and R. van Driel (1996): ''Partial colocalization of glucocorticoid and mineralocorticoid receptors in discrete compartments in nuclei of rat hippocampus neurons,'' *Journal of cell science*, 109, 787--792.

Vasilescu, J., X. Guo, and J. Kast (2004): ''Identification of protein-protein interactions using in vivo cross-linking and mass spectrometry,'' *Proteomics*, 4, 3845--3854.

Vorsanova, S. G., A. D. Kolotii, I. Y. Iourov, V. V. Monakhov, E. A. Kirillova, I. V. Soloviev, and Y. B. Yurov (2005): ''Evidence for high frequency of chromosomal mosaicism in spontaneous abortions revealed by interphase FISH analysis,'' *Journal of Histochemistry & Cytochemistry*, 53, 375--380.

Wachsmuth, M., C. Conrad, J. Bulkescher, B. Koch, R. Mahen, M. Isokane, R. Pepperkok, and J. Ellenberg (2015): ''High-throughput fluorescence correlation spectroscopy enables analysis of proteome dynamics in living cells,'' *Nature biotechnology*, 33, 384--389.

Walter, T., M. Held, B. Neumann, J.-K. Hériché, C. Conrad, R. Pepperkok, and J. Ellenberg (2010): ''Automatic identification and clustering of chromosome phenotypes in a genome wide RNAi screen by time-lapse imaging,'' *Journal of structural biology*, 170, 1--9.

Wang, D. and S. Bodovitz (2010): ''Single cell analysis: the new frontier in omics,'' *Trends in biotechnology*, 28, 281--290.

Wawer, M. J., K. Li, S. M. Gustafsdottir, V. Ljosa, N. E. Bodycombe, M. A. Marton, K. L. Sokolnicki, M.-A. Bray, M. M. Kemp, E. Winchester, et al. (2014): ''Toward performance-diverse small-molecule libraries for cell-based phenotypic screening using multiplexed high- dimensional profiling,'' *Proceedings of the National Academy of Sciences*, 111, 10911--10916.

Weaver, B. A. and D. W. Cleveland (2005): ''Decoding the links between mitosis, cancer, and chemotherapy: The mitotic checkpoint, adaptation, and cell death,'' *Cancer cell*, 8, 7--12.

Werner, A., A. Disanza, N. Reifenberger, G. Habeck, J. Becker, M. Calabrese, H. Urlaub, H. Lorenz, B. Schulman, G. Scita, et al. (2013): ''SCFFbxw5 mediates transient degradation of actin remodeller Eps8 to allow proper mitotic progression,'' *Nature cell biology*, 15, 179--188.

Whitesides, G. M., E. Ostuni, S. Takayama, X. Jiang, and D. E. Ingber (2001): ''Soft lithography in biology and biochemistry,'' *Annual review of biomedical engineering*, 3, 335--373.

Xu, F. L. and W. S. Saunders (2008): ''Actin and microtubules: working together to control spindle polarity,'' *Cancer cell*, 14, 197--199.

Xylourgidis, N., P. Roth, N. Sabri, V. Tsarouhas, and C. Samakovlis (2006): ''The nucleoporin Nup214 sequesters CRM1 at the nuclear rim and modulates NFκB activation in Drosophila,'' *Journal of cell science*, 119, 4409--4419.

Yamaguchi, M., S. Yu, R. Qiao, F. Weissmann, D. J. Miller, R. VanderLinden, N. G. Brown, J. J. Frye, J.-M. Peters, and B. A. Schulman (2015): ''Structure of an APC3--APC16 Complex: Insights into Assembly of the Anaphase-Promoting Complex/Cyclosome,'' *Journal of molecular biology*, 427, 1748--1764.

Yu, H. (2002): ''Regulation of APC--Cdc20 by the spindle checkpoint,'' *Current opinion in cell biology*, 14, 706--714.

Zaal, K. J., C. L. Smith, R. S. Polishchuk, N. Altan, N. B. Cole, J. Ellenberg, K. Hirschberg, J. F. Presley, T. H. Roberts, E. Siggia, et al. (1999): ''Golgi membranes are absorbed into and reemerge from the ER during mitosis,'' *Cell*, 99, 589--601.

Zeitlin, S. G., R. D. Shelby, and K. F. Sullivan (2001): ''CENP-A is phosphorylated by Aurora B kinase and plays an unexpected role in completion of cytokinesis,'' *The Journal of cell biology*, 155, 1147--1158.

Zhao, T. and R. F. Murphy (2007): ''Automated learning of generative models for subcellular location: building blocks for systems biology,'' *Cytometry Part A*, 71, 978--990.

Zhong, Q., A. G. Busetto, J. P. Fededa, J. M. Buhmann, and D. W. Gerlich (2012): ''Unsupervised modeling of cell morphology dynamics for time-lapse microscopy,'' *nature methods*, 9, 711--713.

Zhu, C., E. Bossy-Wetzel, and W. Jiang (2005): ''Recruitment of MKLP1 to the spindle midzone/midbody by INCENP is essential for midbody formation and completion of cytokinesis in human cells,'' *Biochem. J*, 389, 373--381.

Zieve, G. W. (1984): ''Nocodazole and cytochalasin D induce tetraploidy in mammalian cells,'' *American Journal of Physiology-Cell Physiology*, 246, C154--C156.

Zou, H. and T. Hastie (2005): ''Regularization and variable selection via the elastic net,'' *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67, 301--320.