TOWARDS LARGE SCALE RECONSTRUCTION OF NEURO-ANATOMY AT NANOMETER RESOLUTION

Dissertation

zur Erlangung des akademischen Grades Doctor of Philosophy (Ph.D.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Diplom-Medieninformatiker Stephan Saalfeld
geboren am 24. August 1975 in Magdeburg

Gutachter:

Prof. Dr. Michael Schroeder, BIOTEC, Technische Universität Dresden

Prof. Dr. Jan Modersitzki, Institut für Mathematische Methoden der Bildverarbeitung, Universität zu Lübeck

Tag der Verteidigung: 25. Februar 2013

Dresden, den 4. März 2013

To my family, Diana, Maja and Karl.

Acknowledgements

Many thanks to my thesis advisory committee Pavel Tomančák, Iva Tolic-Nørrelykke, and Michael Schroeder. Pavel was the perfect supervisor. He let me work independently and was available whenever I needed support. It is his merit pushing and writing with me during night time that we have published our papers. I cannot thank Albert Cardona enough for his drive to make things happen. Albert taught me biology and wet-lab work, he made it possible that I visited Zürich and Janelia Farm to work with him, he arranged support for TrakEM2 development, he is the major force behind the Fiji and CATMAID communities, he has initiated the reoccurring hackathons, and he generated exceptional data and made it available for my work. I thank Albert, Stephan Preibisch and Tobias Pietzsch for the great cooperation in software development—it's an honor to work with you. I thank Volker Hartenstein for teaching me neurobiology, for his hospitality, and for numerous fruitful discussions. I thank Rick Fetter for great data, teaching me properties of electron microscopes and testing my methods. Both Volker and Rick are great people to work with. I thank Cori Bargmann for making the *C. elegans* data available; Forrest Collman, Nick Weiler, Kristina Micheva and Stephen Smith for sharing the exemplary Array Tomography dataset; and Michaela Wilsch-Bräuninger and Yoon Jeung Chang for sharing examples from their EM comparison. Thanks to Marta Zlatič, Mitya Chklovskii, Kevin Eliceiri, Francesca Peri, Kota Miura, Davi Bock, and Rodney Douglas for their support for traveling, visits and hackathons. Thanks to all members of the Tomančák lab, in particular Maria Bogdanzaliew for help with flies and wet-lab work. Thanks to Peter Pitrone (LMF MPI-CBG) and Alma Arnold (LMF JFRC) for guidance on the microscope. Thanks to Aljoscha Nern at JFRC for sharing his phenomenal genetic constructs and protocols and thanks to Tomoko Oyama for guiding me in the Zlatič lab. Thanks to MPI-CBG, DIGGS-BB, HHMI JFRC, and INI at Uni and ETHZ, for funding. I thank Diana Daniel for taking the larger part of the burden to raise our two children, Maja and Karl. This thesis is for you.

Contents

List of Figures

List of Tables

Preface

When I started working on this thesis, my naïve idea was to reconstruct a relevant part of a fly brain at synapse resolution. At Janelia Farm, Albert Cardona and Rick Fetter had generated a promising Transmission Electron Microscopy (TEM) section series covering a full segment of the ventral nerve cord of an early *Drosophila* larva and it seemed that processing this and then generating more of such series would be a matter of months. I would then establish an atlas that would be continually enriched with light microscopy data generated by 'the community', computationally registered exploiting formally declared anatomical relations, almost automatically revealing more and more of the mystery of the brain.

Then I discovered the obstacles and they turned out to be much more fundamental in nature than the noble plan to understand brains. The TEM series required non-rigid alignment beyond what I had achieved during my previous work and beyond what available methods could deliver. The images had artifacts of various origin. There was no software available to handle the sheer amount of image data effectively other than Albert Cardona's TrakEM2. Ironically, even TrakEM2 had serious design issues that prevented it from handling large image data without failure, and it lacked support for non-affine geometric transformations. 'The community' is small. It turned out that generating this first TEM series had been an exceptional early effort and no further data sets at that quality were generated for years. This has changed only recently when again Albert and Rick started generating new series. Light microscopy images at the desired resolution and quality of the targeted specimen only exist since I generated them myself last summer at Janelia Farm.

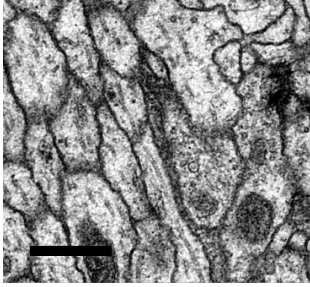
I wrote a software library for generic n-dimensional geometric transformations (including but not limited to affine), model fitting and optimization that we could use in TrakEM2. Together with Albert, we made TrakEM2 able to handle the enormous amount of image data efficiently and correctly. I helped to include newly developed methods for deformation compensation, registration and image enhancement into TrakEM2 and I developed a method to align large series of microscopy sections at unprecedented quality despite the images containing substantial artifacts. You will find two flipbook animations at the top corner of this thesis show-

ing a fragment of the aligned *Drosophila* section series at two scales separated by a factor of ten (scalebars 500 nm and 5 µm). Together with my colleague Stephan Preibisch, we developed an efficient method to register multi-view volumetric acquisitions from Selective Plane Illumination Microscopy (SPIM) for which substantial parts of my previously developed framework could be re-used. During the course of developing this software we got frustrated by the lack of a modern and flexible programming framework for the reoccurring n-dimensional image processing problems. Eventually, we decided to develop it ourselves. This work, together with Tobias Pietzsch who joined the lab last year, has accumulated into the by now well received open source library ImgLib2.

I spent half a year in the lab trying to generate sparsely labeled light-microscopy images of early larval *Drosohila* brains for later registration with the electron microscopy dataset. These experiments succeeded thanks to the protocols and constructs of Aljoscha Nern and Barret Pfeiffer during a summer visit at Janelia Farm. Then, four years were gone and a lot of data still waits to be processed.

That said, this thesis is a report on work in progress. The methods and software components that I have developed have brought us closer to the naïve ideas that I had in the beginning and I am looking forward to proceed on this road.

This book contains a complementary DVD with source code, documentations, and illustrating videos. In alternative to the DVD, the material is available on-line at http://fly.mpi-cbg.de/saalfeld/thesis.



Chapter 1

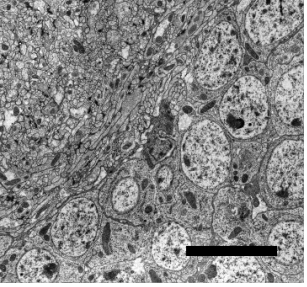
Introduction

I am more than my genome.
I am my connectome.

Sebastian Seung

Understanding how information is processed in animal *central nervous systems* (CNS) is the ultimate goal of both neurobiology and neuroinformatics. Since Ramón y Cajal (1891) and Sherrington (1906) we know that *neurons*, *synapses*, and *glial cells* are the building blocks of biological nervous systems. Neurons are *excitable* cells that, when excitation exceeds a specific threshold, generate *action potentials* (*spikes*), rapidly changing their polarity and triggering synaptic activity. Synapses connect neurons to each other and are today considered the main operators in the system. Depending on the neurotransmitters available in the pre- and post-synaptic neurons, a synapse may be either *excitatory* (increase the probability that the post-synaptic neuron fires), *inhibitory* (decrease the probability that the post-synaptic neuron fires) or it may serve a different purpose, changing the state of the post-synaptic neuron otherwise. Besides chemical synapses, there exist *gap junctions* (Revel and Karnovsky, 1967; Robertson, 1981), direct electric couplings between neurons, sometimes referred to as *electrical synapses*. Signaling from one cell to others is not exclusively performed through synapses. Neurotransmitters can be released into the extracellular space affecting all cells with corresponding receptors in a local vicinity. Glial cells nourish and insulate neurons and are involved in neurotransmitter clearance. Despite that they cannot generate action potentials, they contribute to signal transmission in other ways (Verkhratsky and Kettenmann, 1996). Neurons and glial cells communicate through neurotransmitter release into the extracellular space, electrical synapses and specialized neuron-glial synapses (Fields and Stevens-Graham, 2002).

In-silico simulations of basic concepts of biological neural networks were early understood to have the potential for general information pro-



cessing (McCulloch and Pitts, 1943). Today, *artificial neural networks* (ANN) are a widely accepted tool to solve machine learning problems (LeCun et al., 1989; Turaga et al., 2010) and available as ready-to-use software components for custom applications (e. g. in the *Waikato Environment for Knowledge Analysis*, WEKA Witten and Frank, 2005; Hall et al., 2009). It is important to understand that artificial neural networks that are used in todays machine learning applications do not aim at simulating the properties of biological tissue. Instead, the shared concepts remain superficial: simple units are connected by operators and process inputs in a highly parallelized multi-layer architecture.

The simulation of information processing in actual biological neural networks is target of active research in the neuroinformatics community (e. g. Lang et al., 2011) and has accumulated in large scale modelling attempts such as the *Blue Brain Project*[1] (Markram, 2006), one among six flagship candidates competing for a *Future and Emerging Technologies* (FET) research grant from the European Commission of €1 billion over ten years.

Sporns et al. (2005) introduced the term *connectome* to describe the connectivity of individual neurons and neuronal compartments in the central nervous system. While they considered the reconstruction of connectomes at the level of individual neurons and synapses unimportant, Lichtman and Sanes (2008) claimed the opposite and formed the *Connectome Project*[2] focusing on high-throughput high-resolution microscopy to reconstruct large microcircuits at the synaptic level in mammalian brains. Interesting tools have emerged in the context of this project, notably the genetic construct *Brainbow* (Livet et al., 2007) to visualize individual neurons in mice by light microscopy each in a unique color and the *Automatic Tape-Collecting Lathe Ultramicrotome* (ATLUM, Hayworth et al., 2006) to automatically section and collect large volumes of neuronal tissue.

The historically most important connectome of the nematode *C. elegans* (302 neurons) had been reconstructed long before the term was introduced by White et al. (1986) and Hall and Russell (1991) from serial electron micrographs. The endeavor took almost two decades and was later extended and corrected by Chen et al. (2006) and Varshney et al. (2011).

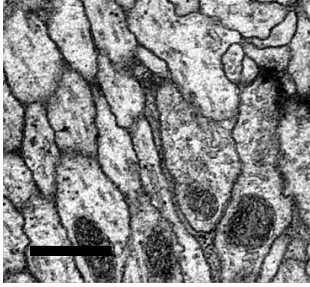
Recently, more partial connectomes have been reconstructed at synaptic resolution providing new insight in the structure of the visual system of rabbit (Anderson et al., 2009), mouse (Bock et al., 2011; Briggman et al., 2011) and *Drosophila* (Rivera-Alba et al., 2011). Bock et al. (2011) provide their data for free public access through the *Open Connectome Project*.[3]

In this thesis, I describe my contribution to a project driven by Albert Cardona to reconstruct the connectome of the early larva of the fruit fly

[1] http://bluebrain.epfl.ch/

[2] http://cbs.fas.harvard.edu/science/connectome-project

[3] http://openconnecto.me/



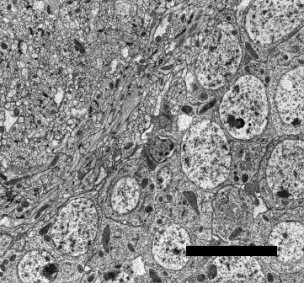
Drosophila melanogaster from *serial section Transmission Electron Microscopy* (ssTEM; Cardona et al., 2010). In Chapter 2, I describe a new method for volume reconstruction from large series of distorted microscopy sections (Saalfeld et al., 2010, 2012) and how the same principles could be transferred to aid the solution of a related problem, registration of multi-view recordings from *Selective Plane Illumination Microscopy* (SPIM; Huisken et al., 2004). The SPIM project was a collaboration with Stephan Preibisch (Preibisch et al., 2010a). During the course of this work, I have designed and implemented a substantial number of software components that I describe in Chapter 3. Particular focus lies on the generic image processing library *ImgLib2*, a collaboration with Tobias Pietzsch and Stephan Preibisch (Pietzsch et al., 2012). In Chapter 4, I discuss my results and draw directions for future work.

1.1 Electron Microscopy

Complete reconstruction of microcircuits in neuronal tissue is only possible if the relevant structures can be distinguished and separated unambiguously, i. e. *axons* and *dendrites* (~100 nm), *synapses* (~200 nm), *membranes*, *synaptic vesicles*, the *synaptic cleft* and *microtubules* (~20 nm). As densely packed as they occur in neuronal tissue, these structures are too small to be imaged by light microscopy which according to the Abbe diffraction limit (Abbe, 1873) cannot separate objects at a distance of less than ~200 nm. The resolution limit can be calculated using the empirically estimated *Rayleigh criterion*. Let λ be the wavelength of the light source, n the refractive index of the medium separating specimen and lenses, θ the half angle of the collected light cone, and $A_N = n \sin\theta$ the *numerical aperture* of the microscope (see Alberts et al., 2002), then the minimal resolvable distance Δr is

$$\Delta r = \frac{0.61\lambda}{A_N} = \frac{0.61\lambda}{n \sin\theta} \tag{1.1}$$

It is possible to work around this physical limitation by embedding additional information into the the captured signal that later enables to reconstruct an image at higher resolution (see review by Patterson, 2009). *Stimulated Emission Depletion* (STED; Hell and Wichmann, 1994) narrows the region in which the fluorescence signal is excited using two focused lasers, one for excitation and one for immediate depletion of everything but a very small region in the center of the excited region. The image is then scanned point by point. Other methods are based on accurate localization of individual fluorescent molecules which requires that the fluorophores are photo-switchable to get an independent image of each of them. Prominent examples for 3d imaging are *interferometric Photo Activated Localization Microscopy* (iPALM, Shtengel et al., 2009), *Stochastic Opti-*



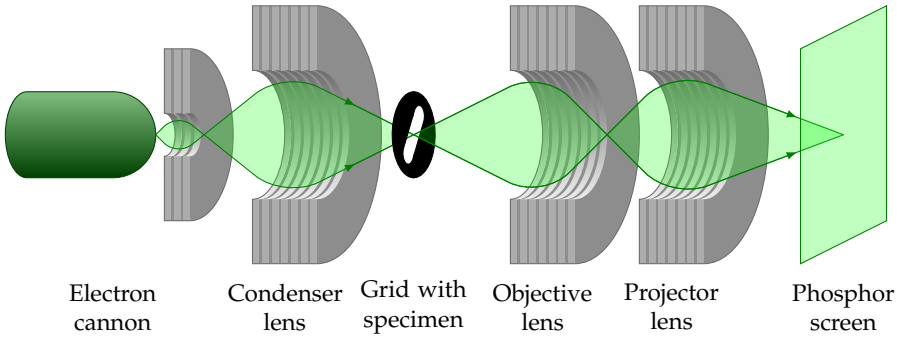


Figure 1.1: Schematic illustration of the main components of a Transmission Electron Microscope, inspired by Alberts et al. (2002, Figure 9-22).

cal Reconstruction Microscopy (3d-STORM, Huang et al., 2008), and *Double-Helix Point Spread Function* microscopy (DH-PSF; Pavani et al., 2009). More accessible for large scale imaging are methods based on interference patterns (I^nM; Gustafsson et al., 1999) or structured illumination (Lukosz and Marchand, 1963; Gustafsson, 2005; Gustafsson et al., 2008) which enable to image the specimen plane by plane and do not depend on specialized fluorophores. However, the resolution that can be achieved by these techniques is in practice still limited at approximately a quarter of the wavelength, i. e. 50–100 nm.

In an *Electron Microscope* (EM, Knoll and Ruska, 1932), a highly accelerated electron beam instead of electromagnetic radiation (e. g. light) is used to generate the image. The electron beam is focused by a series electromagnetic lenses (see Figure 1.1). According to the wave-particle duality that de Broglie (1924) described, the wavelength of a particle depends on its speed and mass

$$\lambda = \frac{h}{mv} \qquad \text{with} \qquad \begin{aligned} m_e &= 9.11 \times 10^{-31}\,\text{kg} \\ h &= 6.63 \times 10^{-34}\,\text{Js} \end{aligned} \tag{1.2}$$

with m_e being the mass of an electron and h being Planck's constant. In the microscope, electrons can be accelerated to $\sim\frac{2}{3}c$ (c = speed of light) which corresponds to a De Broglie wavelength of ~0.004 nm rendering the diffraction limit irrelevant for the resolution required in neuroanatomical imaging. Electrons are scattered by air molecules which makes it necessary that imaging is performed in a vacuum (or at least at very low air pressure). The consequence is that the sample needs to be dehydrated, i. e. electron microscopy cannot be used for in-vivo imaging. Electron microscopes work at low A_N and are therefore not suited for 'optical' sectioning.

The *Transmission Electron Microscope* (TEM) projects the shadow of a partially electron-transparent probe (i. e. an ultra-thin section) on a phosphor-screen, film, or a scintillator and is then imaged with a CCD camera (see Fig-

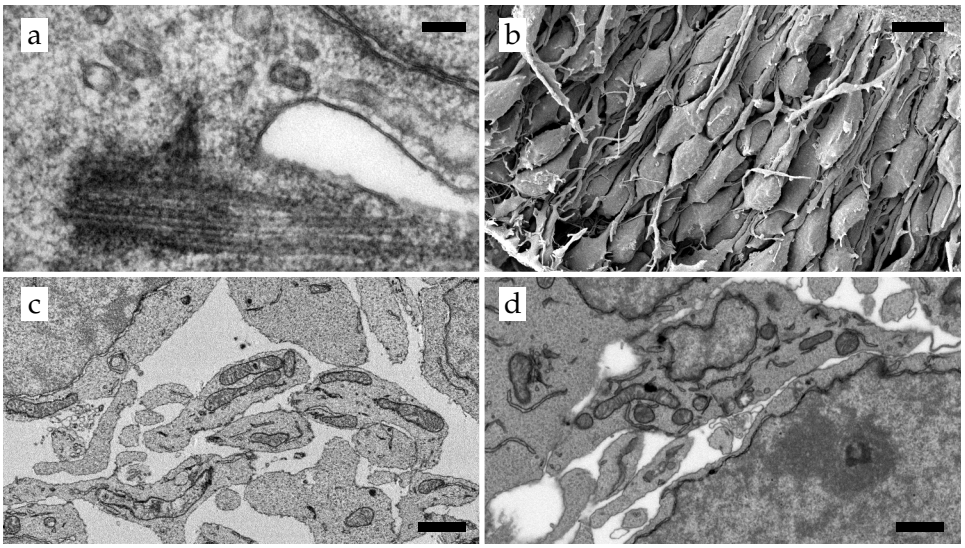
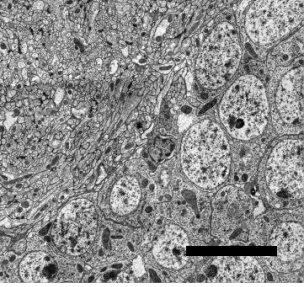


Figure 1.2: Examples demonstrating the typical imaging results of the various electron microscopy modes. Shown is a sample of mouse E11 neuroepithelium prepared for and imaged with TEM (a), SEM (b, gold-coated surface), BF-SEM (c), and FIB-SEM (d). Scalebars are 0.1 µm (a), 10 µm (b) and 1 µm (c,d). TEM (a) clearly outperforms SEM (b–d) in terms of resolution and display of intracellular structures due to on-section staining. Block-face SEM (c,d) using a low energy beam enables thinner sections to be imaged (~20 nm here) compared to TEM (a) which is always a projection of the entire section (~70 nm here). Surface SEM (b) is shown for demonstration purposes only because it cannot be used for volumetric imaging. Images courtesy of Michaela Wilsch-Bräuninger and Yoon Jeung Chang.

ures 1.1 and 1.2a). In order to generate a three-dimensional image of a block of tissue the sample has to be either sectioned and imaged as a series of ultra-thin sections or reconstructed from multiple 2d projections at various angles through the volume (Electron Tomography, Hoppe, 1974). The latter is limited to reconstruct volumes of ~400 nm thickness and is relatively time consuming. It has therefore not been used for large scale 3d imaging yet. Serial sections can be generated reliably at a thickness of 30–50 nm today which is suboptimal but enabling detailed neuroanatomical reconstruction given the high lateral resolution of the TEM (Mishchenko, 2009; Chklovskii et al., 2010; Cardona et al., 2010; Briggman and Bock, 2012). The major disadvantage is that serial sectioning interrupts the continuity of the volume and introduces deformations to individual sections. How to compensate for this in order to restore a continuous volume from distorted section series is the major target of this thesis and is discussed in Chapter 2.

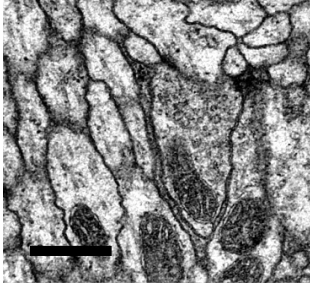


For the high energy electron beam used in a TEM (40–400 keV), biological samples are almost homogeneously transparent. To obtain sufficient contrast, they need to be stained with electron-dense material (e. g. heavy metal compounds) that selectively enriches in particular structures. Commonly used stains are *Osmium tetroxide* (OsO_4) which also serves as a secondary fixative and is applied to the sample as a whole; *uranyl acetate* ($UO_2(CH_3COO)_2 \cdot 2H_2O$) which can be applied *en bloc* and to individual sections; and lead (Pb) compounds such as *Reynold's lead* (Reynolds, 1963) or *Sato's lead* (Sato, 1968; Hanaichi et al., 1986) which is applied to individual sections. The quality of *en bloc* staining strongly depends on how permeable the tissue is to the stain which poses a limit on the size of a volume that can be stained successfully (Bozzola and Russell, 1999, chapter 2). For sections, limited penetration depth is obviously irrelevant.

The *Scanning Electron Microscope* (SEM) images the electron-reflective properties of a sample by collecting scattered electrons, scanning the sample point-by-point with a focused electron beam. It is often used to generate projections of 100 % electron-reflecting surfaces. The result is a picture that appears like a ray-cast rendering of a three-dimensional surface illuminated by a single light source because the number of scattered electrons depends on the angle between surface normal, electron beam and collector according to Lambert's cosine law (see Figure 1.2b). Similarly, a planar surface can be scanned to generate an image of its reflective variance (see Figure 1.2c,d). Denk and Horstmann (2004) have introduced *Serial Block-Face Scanning Electron Microscopy* (SBF-SEM) as a tool for automatic imaging of an entire volume of neuronal tissue. They have built and ultramicrotome that can be operated in the vacuum chamber of the microscope, subsequently generating an image of the block-face and shaving off an ultra-thin section (20–30 nm today, Briggman et al., 2011). In serial mode, the instrument automatically generates a large stack of block-face scans by which a three-dimensional picture of the specimen is created. Heymann et al. (2006) and Knott et al. (2008) have replaced the ultramicrotome by a focussed ion beam which today enables to shave off sections as thin as 5 nm. *Focused Ion Beam Scanning Electron Microscopy* (FIB-SEM) has thus the best axial resolution that exists today in both light and electron microscopy.

For block-face scanning, a low energy electron beam (1–5 keV) is used to reduce the penetration depth of electrons. This is necessary to limit damage of the block, charge accumulation, and the effective 'thickness' of the imaged section. Reducing the energy of the beam increases scanning time but has the potential of generating images with higher contrast with lower demands on electron dense staining (Denk and Horstmann, 2004). The lateral resolution that can be achieved with a low energy electron beam (in practice ~5 nm) is significantly lower than that of TEM but suffices for detailed neuroanatomical reconstruction.

Block-face scanning enables higher axial resolution than TEM because



the shaved off sections do not need to be conserved. FIB-SEM has the best axial resolution but is today limited to a total lateral field of view of less than 80 µm (Shan Xu, personal communication) compared to several millimeters for both TEM and SBF-SEM. In comparison with the imaging speed of TEM, block-face scanning is slow. For comparably small volumes this may be outweighed by TEM series requiring computational post-processing for volume reconstruction (see Chapter 2) and significantly more fragile sample preparation. The major advantage of serial section TEM over block-face SEM is the unsurpassed image quality. Staining of individual sections enables that intracellular structures are visualized at a great level of detail uncompromised by the apparent penetration problems and charging artifacts present in block-face scans.

1.2 The fruitfly *Drosophila melanogaster*

The fruit fly *Drosophila melanogaster* is one of the most popular model organisms used in biological research. Its genome was sequenced by Adams et al. (2000) and is today the the best annotated genome of multicellular organisms, publicly available at *FlyBase* (McQuilton et al., 2012).[4] The genetic mechanisms determining its embryonic development are target of active investigation in the *Drosophila* research community including our lab (Tomančák et al., 2002; Fowlkes et al., 2008; Kalinka et al., 2010). Morphology and development of the *Drosophila* nervous system have been described with increasing detail by Campos-Ortega and Hartenstein (1985); Truman and Bate (1988); Ito et al. (1995); Younossi-Hartenstein et al. (1996); Schmid et al. (1999); Landgraf et al. (1997, 2003); Truman et al. (2004, 2010); Technau and Urban (2005); Hartenstein et al. (2008); Zlatic et al. (2009); Cardona et al. (2010); and many others.

After the egg is laid, it undergoes ~22 h of embryonic development after which the fully functional *first instar larva* hatches. The development of the nervous system begins at ~3:40 h when neuronal stem cells (*neuroblasts*) delaminate from specialized regions of the *ectoderm*. In the course of embryonic development (see Figure 1.3), neuroblasts undergo several asymmetric divisions generating a number of neuronal and glial precursors (*ganglion mother cells*). Each of these precursor cells cleaves into two fully differentiated neurons or glial cells (see Technau and Urban, 2005). From stage 15 during the first larval stage (*first instar*), neuroblasts do not further proliferate constituting a phase of mitotic dormancy (Truman and Bate, 1988). In contrast to neurons in mammalian brains that form *axons* and *dendrites*, neurons in the *Drosophila* CNS typically have one single *neurite* that splits into both axonal and dentritic branches. Chemical synapses are generally *polyadic*, one pre-synaptic terminal communicates to two or

[4] http://flybase.org/

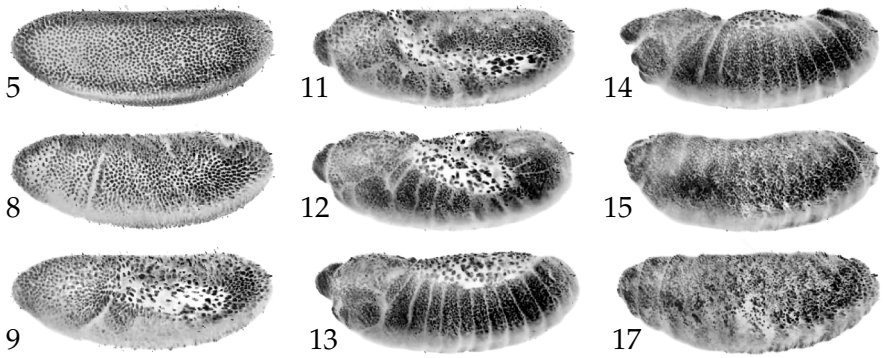
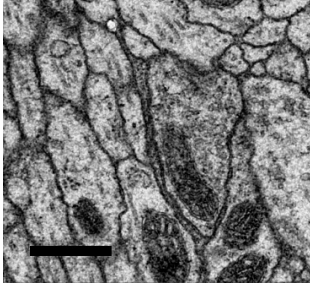


Figure 1.3: Selected stages of *Drosophila* embryogenesis recorded with SPIM, Histone YFP is expressed in all nuclei, colors are inverted and contrast is enhanced. Stage 5 (~2:30 h): cellular blastoderm. Stage 8 (~3:30 h): gastrulation. Stage 9 (~4 h): germ band elongation, beginning of organ primordia formation. Stage 11 (~6 h): parasegmental furrows subdivide germ band into metameric units. Stage 12 (~8 h): germband retraction. Stage 13 (~10 h): primordia of most organs formed. Stage 14, 15 (~11 h): dorsal closure and head involution. Stage 17 (~16 h): larval shape, CNS shortening. Figure after Hartenstein (1993).

more post-synaptic dendrites. Most pre-synaptic sites are found in thickenings of axonal branches, the *pre-synaptic varicosities* or *boutons* (Cardona et al., 2010). Neuronal cell bodies are arranged in a lateral enclosing layer, the *cortex*, projecting their neurites towards the center of the CNS (*neuropil*) where they branch and form synaptic connections. First instar larvae are collected after hatching, their CNS' dissected and prepared for imaging.

The first instar larval CNS comprises ~12,000 neurons, ~2,800 forming the two brain lobes, ~1,200 in the *subesophageal ganglion* (SOG) and ~600 per segment of the *ventral nerve cord* (VNC; Albert Cardona, personal communication). This is ~40× more than in the previously reconstructed *C. elegans* nervous system (302 neurons; White et al., 1986; Hall and Russell, 1991) and thus seems a challenging though technically feasible target for comprehensive neuroanatomical reconstruction. Its relatively small size, the comparably well described neuronal morphology and development at 'macroscopic' (light-microscopy) resolution, the availability of a rich genetic toolbox to visualize distinct groups of individual cells (e. g. recent work by Pfeiffer et al., 2008, 2012; Nern et al., 2011) and standardized functional assays (e. g. Hughes and Thomas, 2007, and the Janelia Farm Olympiads[5]) render the *Drosophila* larva an ideal model to decipher the formation and function of a complex biological nervous system at synaptic resolution.

[5]Fly Olympiad: http://www.janelia.org/team-project/fly-olympiad
Larval Olympiad: http://www.janelia.org/team-project/larval-olympiad



Chapter 2

Registration and Alignment

> Slicing is awful! I do not wanna process slices, unless I really have to, ok?! I wanna do block-face, always do block-face.
>
> Gene Myers

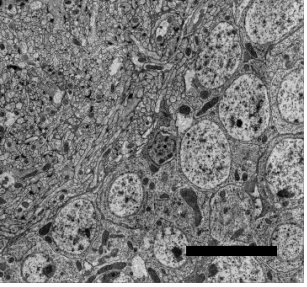
The major subject of this thesis is an image-based method to restore a continuous volume from distorted serial microscopy sections. This Chapter begins with a brief discussion of the benefits and disadvantages of serial section microscopy (Sections 2.1 and 2.2). I argue that, although alternative methods have been proposed, serial section microscopy is still an indispensable tool for large scale high-resolution neuroanatomy.

Image-based volume restoration from section series as well as montaging overlapping images are *image registration* problems. Section 2.3 gives a brief introduction to relevant subjects of the field and comments related work including previous attempts dealing with serially sectioned volumes.

Sections 2.4 to 2.6 comprise a detailed description of the method followed by a demonstration of its applicability to three large section series from electron and light microscopy in Section 2.7. While the results are visually convincing, evaluation based on this perception alone is subjective and prevents comparison with future improvements. I have therefore developed an extensive evaluation framework based on artificially generated ground truth that I describe in Section 2.8.

In Section 2.10, I summarize a method for matching and registering partially overlapping n-dimensional point clouds, a collaboration with Stephan Preibisch to combine 3d images that were recorded from multiple viewpoints using *Selective Plane Illumination Microscopy* (SPIM). The project is comprehensively described in Preibisch's PhD thesis (2010) and the accompanying publication (Preibisch et al., 2010a).

The Chapter closes with a summary of our results, discussion of consequences, and directions for further research in Section 2.11.



2.1 Serial Section Microscopy

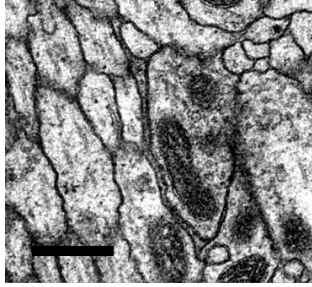
Serial section microscopy is a classic technique for detailed three-dimensional reconstruction of large biological specimens (Born, 1883; Sjöstrand, 1958). Typically, the fixed specimen is embedded in a block of solid medium such as paraffin or synthetic resin and then cut into a series of thin sections. Sections are collected, mounted, stained individually, and eventually imaged with a microscope. In theory, a fully transparent 3d image of the original volume can later be reconstructed from individual section images. The axial resolution of that 3d image is specified by the thickness of individual sections and depends on the application. It may vary from several millimeters down to some tens of nanometers. Connectome reconstruction at the level of single neurites and synapses requires operating at the lower end of that spectrum. Today, sections of 40 nm thickness can be generated routinely using an ultra-microtome which beats the axial resolution that can be achieved by standard optical sectioning techniques by a factor of ~10×. Combined with EM, the resulting 3d resolution is appropriate for neuroanatomy reconstruction at synaptic resolution.

As discussed in Chapter 1, 3d images of a specimen may be generated using alternative techniques such as optical sectioning, tomography, and imaging block faces instead of sections. However, using ultra-thin sections has one key advantage over these techniques: it effectively eliminates the penetration problem for both staining and imaging. In addition, Micheva and Smith (2007) demonstrated that the same section may be stained, eluted and re-stained multiple times with different antibodies by which a large number of independent channels of information can be extracted to study co-localization of potentially many proteins in the same tissue. A similar method has yet to be developed for alternative methods to serially sectioning the specimen.

As a consequence of using a high energy electron beam, TEM still excels over block-face SEM in the combination of resolution, imaging speed and signal to noise ratio. Virtually infinite fields of view can be imaged as mosaics of overlapping image tiles deploying a motorized stage that is preferably operated using a comfortable automatic acquisition system such as Leginon (Suloway et al., 2005).

In combination, these advantages render serial section microscopy in combination with electron microscopy particularly useful for large scale high resolution reconstruction of dense neuronal tissue where the method recently experienced a renaissance (see Hayworth et al., 2006; Anderson et al., 2009; Cardona et al., 2010; Chklovskii et al., 2010; Bock et al., 2011; Briggman and Bock, 2012).

The major downside of physically cutting a block into sections is that continuity between sections is lost. Ultra-thin sections of some tens of nanometers thickness are an extremely fragile matter requiring extraor-



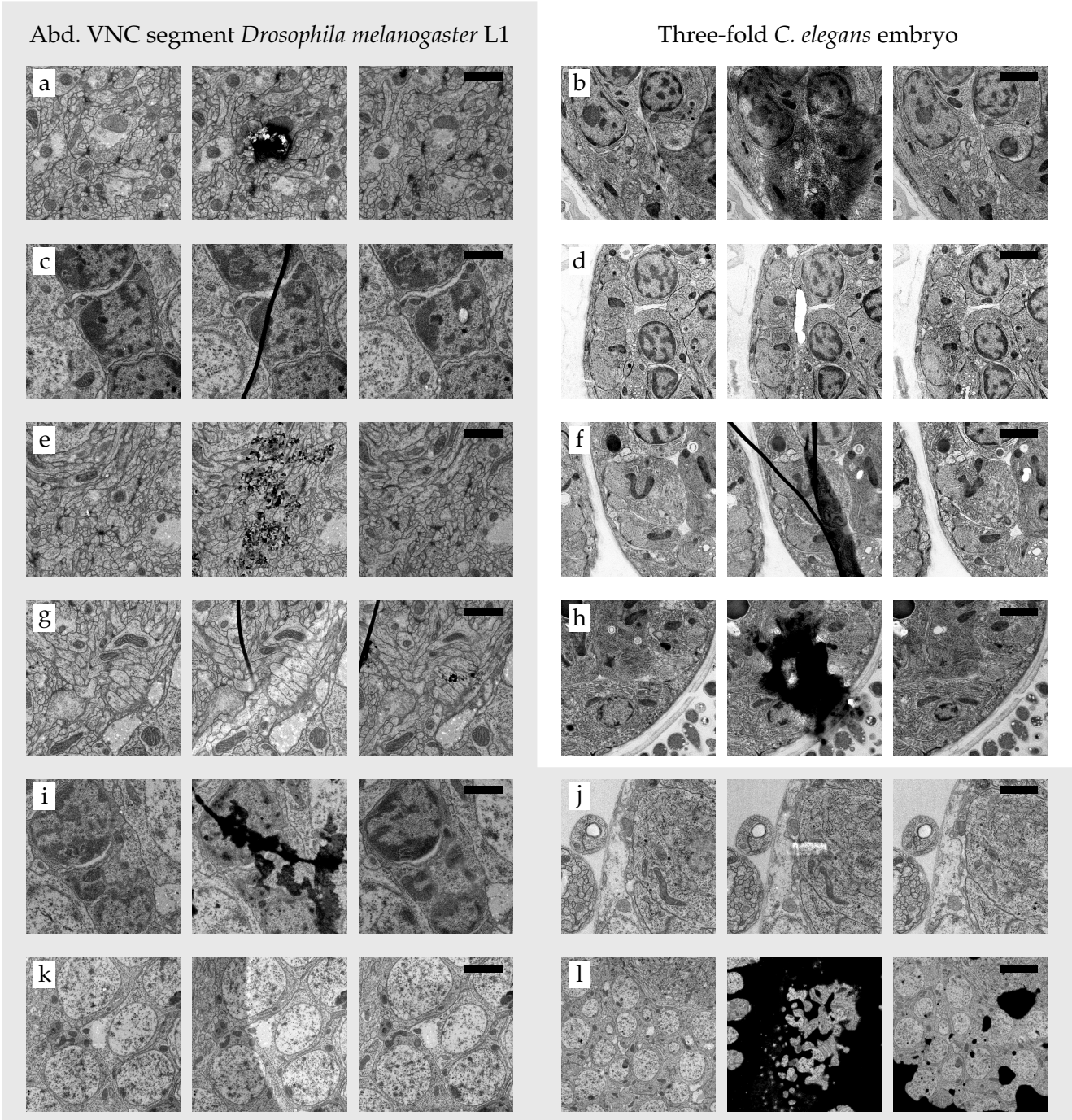
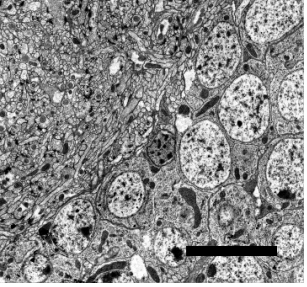


Figure 2.1: Typical artifacts that inevitably occur in large TEM section series. Snapshots of aligned section series show a section compromised by an artifact in context with adjacent sections. Some artifacts impact image similarity but have no effect on geometry (dirt (a,h), staining precipitates (a,b,e,i,l), support film folds (f,g,i), illumination variance (g,k). Others distort local geometry (section folds (c), stretches (g,j), cracks (d,j)). Scale bars are 1 µm (a–j), 2 µm (k), and 4 µm (l); 1 µm = 250 px.



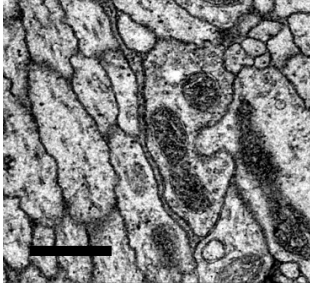
dinary caution during preparation. In order to image section series with TEM, no functional alternative is yet available to manually collecting the floating sections from the water-surface behind the diamond knife of the ultra-microtome. Both cutting and collecting introduce non-linear distortions to individual sections and even the most careful operator will loose a number of sections by accident (currently the lower bound is at 1–5 per 1,000, Richard Fetter, personal communication). Automatic systems such as the ATLUM (Hayworth et al., 2006) are capable of cutting and collecting sections more reliably with lower amounts of distortion. However, the system collects sections on tape which requires them to be imaged with SEM. Furthermore, staining sections individually comes with an inevitable amount of artifacts that are independent from section to section and complicate inter-section comparison (Figure 2.1). In order to recover the imaged volume and extract biologically interesting information such as neuronal circuits (Anderson et al., 2009; Cardona et al., 2010; Bock et al., 2011), sections need to be aligned and distortion must be removed.

2.2 Mosaics

Before high quality digital cameras became available, micrographs were recorded on analog film. For the reconstruction of the *C. elegans* nervous system, White et al. (1986) and Hall and Russell (1991) recorded and developed all images on film, neurites and synapses were annotated directly on film slides. Film still has superior resolution compared to off the shelf digital cameras such that a single image can cover a larger field of view. However, to transfer the images to a digital storage medium requires them to be scanned using a film scanner which is laborious and may result in degraded quality regarding both resolution and dynamic range. Therefore, current microscopes typically use a digital camera and a motorized stage for capturing images and record large areas as mosaics of overlapping tiles.

Generating a seamless montage from partially overlapping image tiles seems trivial. In fact it should be sufficient to map all tiles into a common target space using the coordinates at which the tile was acquired. Unfortunately, the coordinates stored by motorized stages include significant odometry errors that become especially relevant at the high magnification that is used in electron microscopy. With electron microscopes, further distortions have to be taken into account: The electron beam is formed by a series of electromagnetic lenses. The accuracy of those lenses is, despite the option to calibrate them in the running system, significantly lower than what can be achieved with an optical system (see Alberts et al., 2002). Hence, each image tile is distorted by imperfect lenses.

In TEM, another effect has to be taken into account. Heat introduced by the high energy electron beam induces increasing deformation of the



support film carrying the sections. Since tiles are imaged sequentially, each tile is the image of a differently distorted section.

In summary, to generate seamless montages from partially overlapping TEM image tiles, it is necessary to compensate for odometry errors of the motorized stage, lens distortion, and heat induced section distortion.

2.3 Related Work

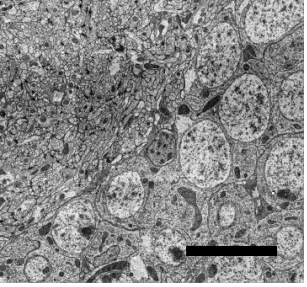
Both montaging and serial section alignment can be treated as an *image matching* or *registration* problem. The field is well studied, mostly in the context of medical image processing. Image registration is the task to bring two images into alignment such that corresponding content is superimposed. Typically, a *template* image is to be transformed such that it matches a *reference* image. The task comprises (1) identifying corresponding content and (2) the transformation required to align it. Solving this task requires an application driven decision on (1) what image features to use for matching, (2) what would be the expected class of transformations, (3) how to explore the search space, and (4) what similarity measure to use in order to evaluate (intermediate) results (see Brown, 1992; Modersitzki, 2009).

Features and the associated similarity metrics to identify corresponding image content can loosely be ordered by their level of abstraction. The most basic are absolute intensities, typically compared using the *sum of square differences* (SSD). Linear dependency of intensities can be estimated using *normalized cross-correlation* (NCC), general dependency by *mutual information* (MI; Collignon et al., 1995; Viola and Wells III, 1997).

Instead of using intensities directly, all above methods can be applied to pre-processed data, e.g. the gradients of an image. Depending on various conditions such as presence of artifacts, variance in imaging conditions, or for registration of multiple modalities showing disjunct structures, preprocessing can aid the comparison using an intensity-based metric by removing variance that is irrelevant for the registration task. Wein et al. (2008) demonstrated that registration of *computer tomography* (CT) and ultrasonic images can be achieved by first simulating the ultrasonic modality from CT and then using a low-level intensity based similarity metric for registration. An even more elegant though computationally demanding approach is the combination of *segmentation* and registration as shown by Cremers and Schnörr (2002); Flach et al. (2002); Flach and Sara (2004).

It is clear that considering local properties alone is not sufficient to register two images in a meaningful way. Rohlfing (2012) nicely demonstrated that a registration scheme that independently assigns all template pixels to their best matching reference pixel[1] generates outstanding results in terms

[1] *Completely Useless Registration Tool* (CURT), available as part of Rohlfing's *Computational Morphometry Toolkit* (CMTK, http://nitrc.org/projects/cmtk)



of local similarity metrics but breaks the topology of the template image. It is thus necessary to take the expected class of transformation into account.

Brown (1992) broadly classifies transformations as *global* or *local* with respect to the image. A global transformation refers to a single parametric model with infinite support in the coordinate domain that is used to transform the entire image. The most common examples are (1) *translation*, (2) *rigid body transformation* (translation + rotation), (3) *similarity transformation* (rigid body + isotropic scale), (4) *affine transformation* (rigid body + non-isotropic scale + shear), and (5) *polynomial transformations* of arbitrary degree. Local transformations consist of independent transformations with usually limited support in the coordinate domain. Popular examples include all kinds of *tessellations* (e. g. triangular meshes or transformation grids), *splines* (Schoenberg, 1946), or *moving least-squares* interpolation (Levin, 1998; Schaefer et al., 2006). The choice of transformation has strong impact on the strategy, reliability, and computational cost for finding a solution. Typically, a transformation with less degrees of freedom can be identified quicker and more reliably than one with more.[2] A usual practice is therefore to estimate an approximate global transformation first that is then used to initialize the search for a more complex local transformation. That can be generalized to a coarse-to-fine approach with more than two steps. Search space and computational cost are thus greatly reduced, and it becomes more likely that a satisfying solution will be found.

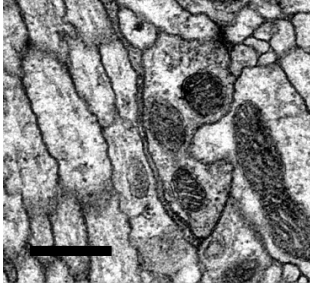
Even in case that a reasonable hierarchy of transformations is chosen, it remains a common problem that correspondence cannot be established reliably and unambiguously for each particular location in an image. Consider two images of a circle. They obviously do not provide the necessary features to disclose their relative rotation, their translational displacement, however, can be found easily. Such problems are considered *ill-posed*, i.e. there is no unique optimal solution to the problem but (often infinitely) many of equal goodness. In order to find a unique solution to such problems they need to be *regularized*. Regularization can be achieved by penalizing the distance to a simpler transformation class than the desired or to require local transformations to be smooth (e.g. Arganda-Carreras et al., 2006; Schaefer et al., 2006). A physically inspired option is to minimize the *bending energy* as in *Thin-Plate Splines* (TPS; Duchon, 1976; Bookstein, 1989) or by using an *elastic* transformation model (Broit, 1981; Gee et al., 1997).

While Broit (1981) was not the first to use elastic constraints in image matching (e. g. Fischler and Elschlager, 1973), he introduced the function

$$cost = deformation - similarity \tag{2.1}$$

as a general minimization target to solve image registration. Let $\mathbf{x}$ denote a vector in n-dimensional real space $\mathbb{R}^n$ and $\mathbf{y}$ denote a vector in m-

[2] The perfect counter example is again Rohlfing's CURT, however, for practically useful transformations, the rule generally holds.

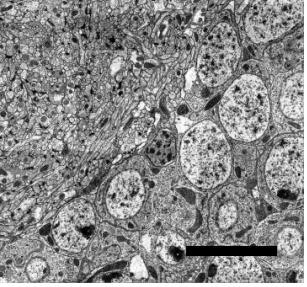


dimensional real space $\mathbb{R}^m$. Let further $T : \mathbb{R}^n \to \mathbb{R}^m$ be a transformation $T(\mathbf{x}) = \mathbf{y}$ and $\mathcal{I}[T] : \mathbb{R}^m \to \mathbb{I}$ and $\mathcal{J} : \mathbb{R}^m \to \mathbb{J}$ be functionals that represent the transformed template and the reference image mapping from an m-dimensional coordinate domain into a pixel value domain. Let now $\mathcal{D}[\mathcal{I}, \mathcal{J}]$ be a dissimilarity metric defined over both pixel value domains and $\mathcal{S}[T]$ denote a regularizer, typically the cost for deformation. Then, generalizing the formulation by Modersitzki (2009), Equation (2.1) constitutes the optimization problem

$$\arg\min_T \mathcal{D}[\mathcal{I}[T], \mathcal{J}] + \mathcal{S}[T]. \tag{2.2}$$

In practice, this very general framework is often simplified to solve a particular task more efficiently. Simple transformations such as global translation or orientation may be derived directly from properties of global statistics over all intensities (Goshtasby, 1985; Hibbard and Hawkins, 1988; Alpert et al., 1990). Alternatively, instead of considering an image in its entirety, it is often beneficial to focus on sparse discriminative samples. Popular examples are corners (Harris and Stephens, 1988), blobs (Mikolajczyk and Schmid, 2004; Lindeberg, 1998; Lowe, 2004), extremal regions (Matas et al., 2002), or salient regions (Kadir et al., 2004). Such samples are then usually matched using invariant local descriptors that are either extracted from underlying intensities (Lowe, 2004; Brown et al., 2005; Bay et al., 2008), their spatial configuration (Belongie et al., 2002; Frome et al., 2004; Preibisch et al., 2010a), or both (Brown and Lowe, 2002). A global transformation is then calculated from those matches using an appropriate estimator. False matches may be filtered based on their consensus with that transformation. The most popular method serving this purpose is the *Random Sample Consensus* (RANSAC) by Fischler and Bolles (1981) and its variants. Focusing on a few representative samples instead of entire images has the potential to greatly reduce the computational complexity of a registration problem while delivering satisfying results. Applications range from *stitching panoramas* (Brown and Lowe, 2007) to *object* and *scene recognition* (Lowe, 1999) and *visual Simultaneous Localization and Mapping* (SLAM; Davison, 2003; Pietzsch, 2011). Careful evaluation of local feature descriptors (Mikolajczyk and Schmid, 2003) has shown that those based on the *Scale Invariant Feature Transform* (SIFT; Lowe, 2004) outperform others in terms of their discriminative power under a variety of perturbations.

Three-dimensional reconstruction from serial sections is commonly accepted to be solvable by image registration. Until today, alignment of consecutive sections is often performed manually, either by selecting corresponding landmarks or through interactive adjustment (see e. g. Kremer et al., 1996; Fiala, 2005; Anderson et al., 2009; Cardona et al., 2012, and Section 3.1.7 of this thesis). Manual alignment tools may complement automatic alignment methods as a backup for situations where the automatic



method fails to deliver an appropriate result.

Rigid body alignment has historically been achieved using the *Principal Axes Transformation* (PAT; Hibbard and Hawkins, 1988; Alpert et al., 1990), often serving as initialization for further improvement. Intensity based similarity metrics in combination with multi-scale search enable better estimates of rigid body alignment (Randall et al., 1998) and more complex transformations such as affine (Thévenaz et al., 1998), cubic B-Splines (Arganda-Carreras et al., 2006, 2010) or elastic transformation (Bajcsy and Kovačič, 1989; Gee, 1999; Wirtz et al., 2004).

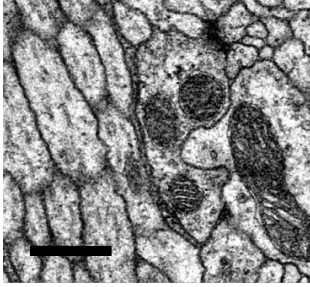
Ourselin et al. (2001) proposed to estimate a displacement field between two adjacent sections using NCC-based block matching, successively refined from coarse to fine scales. A rigid body transformation is then calculated from all matches using a robust L_1-estimator. They evaluated their approach using an aligned series of block-face images. The images were artificially transformed rigidly and then aligned using their method. In this evaluation framework, the authors report to achieve sub-pixel accuracy.

Dealing with more prominent non-rigid deformation, Anderson et al. (2009) and Tasdizen et al. (2010) proposed a multi-step protocol for montaging and alignment of ultra-thin serial sections imaged with TEM. Montages are generated by sequentially mapping image tiles into the current montage according to their relative location to overlapping tiles that is estimated using the *phase-correlation* method (Kuglin and Hines, 1975). Remaining tile to tile deformation from uncalibrated lenses and support film deformation is then compensated by tessellating each tile into a mesh of triangles, estimating an individual offset for each vertex by phase-correlation in a local vicinity, and warping the image accordingly. It is important to understand that their approach is strictly forward, a single template image is registered and mapped into the current montage constituting the reference for the subsequent image. I. e., no globally optimal solution is achieved. Serial section alignment of these montages is performed similarly, registering each section to its predecessor. First, a rigid body alignment is estimated combining exhaustive search over all orientations with phase-correlation for the translation at multiple-scales. Then, the template montage is tessellated into a mesh of triangles which is warped according to local displacements of its vertices, again estimated by phase-correlation in a local vicinity of each vertex. Their approach is publicly available as the *NCR toolset*.[3] In the toolset, automatic alignment is accompanied by a graphical user interface for interactive non-rigid alignment.

In earlier work (Koshevoy et al., 2006), the authors have investigated the applicability of SIFT to solve section-to-section registration but rejected it as "unreliable" (Ross Whitaker, personal communication).

The so far discussed non-linear methods focus mainly on a robust so-

[3] http://www.sci.utah.edu/software/129-ncrtoolset.html



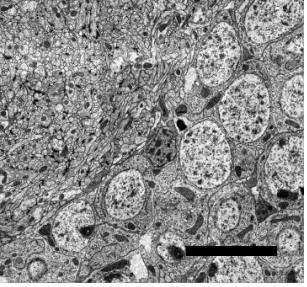
lution to pairwise registration of two adjacent sections, assuming that sequential section alignment will result in an appropriate reconstruction of the original volume. In large series, however, this is not the case because:

Theorem 1. *Sequential concatenation of non-rigid pairwise registration accumulates registration errors and will result in increasing artificial deformation along the series.*

Proof. For simplification, I consider the one-dimensional case where positions and uncertainties can be specified as scalars instead of vectors and covariance matrices. Be x_i and y_i two points at some distance $a_i = |x_i - y_i|$ in an image I_i. Let $\epsilon(x_i) \geq 0$ specify the uncertainty at which the position of a point x_i has been estimated, it may be either the variance or an error margin. If x_i and y_i are uncorrelated then the distance $a_i = |x_i - y_i|$ has an uncertainty $\epsilon(a_i) = \epsilon(x_i) + \epsilon(y_1)$ according to the addition rule for uncertainties. The displacement $d(x)_i$ of a point x_i relative to an adjacent image I_{i-1} is estimated by pairwise matching. Matching has an uncertainty $\epsilon(d(x)_i)$. If a series of images is registered sequentially then the absolute position of a point x_n integrates all previously measured displacements $x_n = x_0 + \sum_{i \in \{1 \ldots n\}} d(x)_i$. The respective uncertainty $\epsilon(x_n)$ accumulates accordingly $\epsilon(x_n) = \epsilon(x_0) + \sum_{i \in \{1 \ldots n\}} \epsilon(d(x)_i)$. I. e., the uncertainty $\epsilon(x_i)$ as a function of index i is *monotonically increasing*. In case that $\epsilon(d(x)_i) > 0$ for all i, it is *strictly increasing*. That means that the expected registration error in sequential non-rigid series alignment is strictly increasing with increasing distance from the chosen template. In practice, this will lead to artificial deformation and renders sequential non-rigid series alignment inappropriate for the reconstruction of large section series. □

Anderson et al. (2009) noticed and described this effect in their earlier work (Koshevoy et al., 2007) but later declared that "no error emerges from transforming all sections into the same volume space. [...] slice-to-slice distortions [...] do not accumulate and section-to-volume distortions are statistically indistinguishable from [...] those of any slice pair" (Anderson et al., 2009). I believe that this observation can be explained only by insufficient statistics due to comparably short section series.

Guest and Baldock (2001) first described registration of histological section series as a global elastic problem, acknowledging that error accumulation as explained in Theorem 1 can be prevented explicitly by minimizing section deformation globally. The elastic constraint is modeled by spring connected finite elements. Optimization is done iteratively, all vertices moving towards the point between their matches in adjacent sections above and below. At each iteration, the matching locations in both adjacent sections for each vertex are estimated anew, considering both similarity of local pixel intensities and a penalty for elastic deformation. Accordingly, both the first and the last section of the series remain undeformed and by that



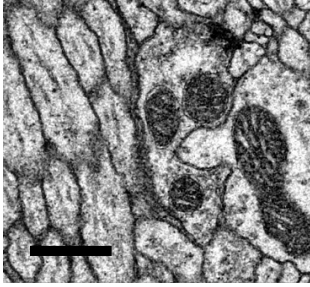
serve as global templates. The authors state that "At least one section must be fixed to provide boundary conditions for the warping process.", however, the approach penalizes section deformation globally and therefore effectively prevents accumulation of erroneous warping. Sections need to be pre-registered using a rigid body transformation. This is either achieved manually or using the PAT method (Hibbard and Hawkins, 1988; Alpert et al., 1990).

Wirtz et al. (2004); Schmitt et al. (2007) developed an alternative method for elastic alignment of histological section series. They propose an efficient iterative solution for global elastic registration using a variational formulation of a joint cost function as in Equation (2.2). The cost comprises a pixel-based similarity between adjacent sections (using SSD) and the force for elastic deformation. Initial alignment is achieved using a variation of the PAT method (Hibbard and Hawkins, 1988; Alpert et al., 1990) that, beyond a rigid body transformation, enables the estimation of shear.

Ju et al. (2006) proposed another alternative method for non-rigid alignment of histological section series. They constrained the deformation model, enabling non-linear deformation in only one dimension. This constraint was motivated by the observation that deformation present in histological section series seems to be dominant in the slicing direction. Pairwise warps are identified using an efficient one-dimensional dynamic programming scheme and then smoothed across the series. That way, deformation is not minimized in a global sense but remains limited for all sections. Sections need to be oriented such that the slicing direction is aligned with the y-axis which requires a preparation protocol that enables to identify the slicing direction reliably.

Arganda-Carreras et al. (2010) have extended their earlier approach for sequential pairwise registration (Arganda-Carreras et al., 2006) to an iterative scheme where each section in the series is aligned not only to its predecessor but to both adjacent sections simultaneously. This 'tri-wise' registration is repeated for all section triples in the series until convergence is reached. Iteration order through the series is switched at each iteration. While the approach does not formulate a global cost for deformation, the cost for local deformation 'diffuses' across the series during iteration.

In summary, a number of valuable approaches for volume reconstruction from serial sections were proposed and it was shown that reconstruction can be achieved by image registration in case that no further information about the shape of the original volume is available. It means that the shape is recovered from differentials, i. e. relative orientation and deformation of all sections. These relative transformations are then integrated which has the potential to reconstruct the volume up to an unknown (and irrelevant) global orientation. The base-line for most registration methods is to discover the relative orientation of all sections for which basically two methods are generally used: (1) principal component analysis, or (2) ex-



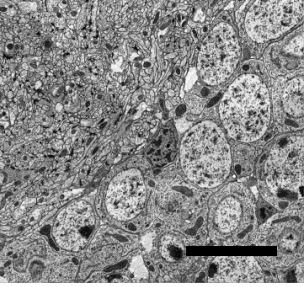
haustive search. Higher order transformations ranging from affine transformations to elastic deformation were found to improve registration of adjacent sections. Some approaches simply chain these pairwise transformations (Thévenaz et al., 1998; Arganda-Carreras et al., 2006; Anderson et al., 2009; Tasdizen et al., 2010) which renders them problematic for large section series where increasing artificial deformation would be introduced (see Theorem 1). Other approaches address this problem, either by smoothing (Ju et al., 2006; Arganda-Carreras et al., 2010) or through a global elastic constraint (Guest and Baldock, 2001; Wirtz et al., 2004; Schmitt et al., 2007). The field is dominated by applications focusing on semi-thick histological sections (0.2–2 µm) imaged by light microscopy. Only Anderson et al. (2009); Tasdizen et al. (2010) and Arganda-Carreras et al. (2010) have demonstrated their work on ultra-thin TEM section series where deformations and artifacts are different than in semi-thick histological sections. Only Anderson et al. (2009); Tasdizen et al. (2010) have developed their approach for large datasets of several hundred Gigabytes. All approaches rely on pairwise registration of adjacent sections and do not consider wider context in the series.

The method described in this thesis chapter implements volume reconstruction from serial sections with a global elastic constraint and can deal with very large data sets. It incorporates two complementary registration methods: (1) robust initialization with landmark correspondences from invariant local features and (2) global elastic alignment based on pairwise deformation fields estimated by NCC-based block matching. The approach incorporates several methods to prevent erroneous deformation due to mismatches from image artifacts. A unique feature of my approach is that it considers a wider context for each section than direct adjacency and I will show that this greatly improves the global quality of the reconstruction.

2.4 The Transformation Model

In order to align section series and montage mosaics from ssTEM, we have to gain sufficient understanding of the contributing transformations that map pixel coordinates from a digital image into a 3d world space. As in the related work discussed above, I assume that sections are of constant thickness such that the index s of a section scaled by the section thickness d represents the z-coordinate in 3d world space. The remaining registration problem is thus two-dimensional. As depicted in Section 2.1, cutting and collecting sections means that their relative translation and orientation becomes unknown which leads to an unknown rigid body transformation

$$\begin{aligned} R_s : \mathbb{R}^2 &\to \mathbb{R}^2 \\ \mathbf{x} &\mapsto \mathbf{R}\mathbf{x} + \mathbf{t} \end{aligned} \quad \text{with} \quad \mathbf{R} = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}, \quad \mathbf{t} = \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.3}$$



$\mathbf{R}$ being a 2×2 orthogonal rotation matrix with $\det \mathbf{R} = 1$ and $\mathbf{t}$ a translation vector. In addition, each section is deformed by an independent unknown non-rigid deformation $D_s : \mathbb{R}^2 \to \mathbb{R}^2$. Imaging sections with the microscope introduces lens-distortion $L : \mathbb{R}^2 \to \mathbb{R}^2$ which can be assumed identical for all images.

If sections were imaged as mosaics of overlapping tiles, two further components are to be considered, firstly the position of the stage, which is an independent translation vector $\mathbf{t}_t$ for each tile, and the heat-induced deformation of the section during imaging which is an unknown non-rigid deformation $D_t : \mathbb{R}^2 \to \mathbb{R}^2$. The combined geometric transformation $T_t : \mathbb{R}^2 \to \mathbb{R}^2$ mapping an image coordinate $\mathbf{p}$ into its two-dimensional world-coordinate $\mathbf{q}$ is thus

$$\mathbf{q} = T_t(\mathbf{p}) = D_s(R_s(D_t(L(\mathbf{p}) + \mathbf{t}_t))). \tag{2.4}$$

If D_t is considered independent of orientation R_s, then their order becomes irrelevant and Equation (2.4) can be written

$$T_t(\mathbf{p}) = D_s(D_{t'}(R_s(L(\mathbf{p}) + \mathbf{t}_t))) \tag{2.5}$$

which enables to join the translation of section and tile

$$T_t(\mathbf{p}) = D_s(D_{t'}(\mathbf{R}_s L(\mathbf{p}) + \mathbf{t}_{st})). \tag{2.6}$$

Knowing little about the deformations D_s and $D_{t'}$, they may be subsumed to a more general unkown deformation $D_{st'}$

$$T_t(\mathbf{p}) = D_{st'}(\mathbf{R}_s L(\mathbf{p}) + \mathbf{t}_{st}). \tag{2.7}$$

In case that only a single section is to be montaged the model is

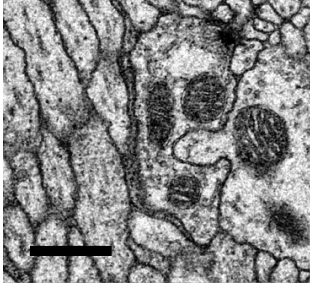
$$M_t(\mathbf{p}) = D_{t'}(L(\mathbf{p}) + \mathbf{t}_t) \tag{2.8}$$

and in case that each section was recorded as a single image, it is

$$S_s(\mathbf{p}) = D_s(\mathbf{R}_s L(\mathbf{p}) + \mathbf{t}_s). \tag{2.9}$$

2.5 Landmark-Based Alignment and Montaging

As proposed by Guest and Baldock (2001); Schmitt et al. (2007); Anderson et al. (2009) and others, it is usually a good idea to estimate an approximate simpler transformation first because it constrains the search space for the computationally more expensive transformation later. However, global statistics such as PAT (Hibbard and Hawkins, 1988; Alpert et al., 1990) do not perform well on serial TEM images of dense textured structures. This



is due to the lack of a distinctive global shape and the regular occurrence of staining artifacts (see Figure 2.1) that dominate the statistics.

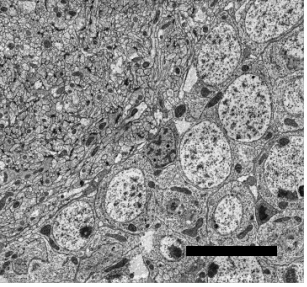
In my Diploma thesis (Saalfeld, 2008), I have proposed a SIFT-based approach to align TEM section series consisting of mosaics of overlapping tiles. I have shown that a good approximation of an elastic alignment can be achieved by minimizing the *sum of square residuals* (SSR) of all corresponding landmarks within and across sections enabling an independent rigid body transformation for each tile. Contrary to Anderson et al. (2009), I found SIFT to perform with outstanding reliability for both montaging and registering consecutive sections despite the presence of significant non-linear distortion and artifacts. While matching features across overlapping tiles within a montage required no adjustment, achieving reliable results across sections became only possible by modifying SIFT to work with larger descriptors than proposed by Lowe (2004). Partially reiterating earlier work (Saalfeld, 2008), I will describe the SIFT-based method with particular focus on concepts that will later be adopted to other contexts. New in this PhD thesis are three extensions of the proposed approach:

1. an (almost) parameter-free robust estimator that can be used for both pairwise and global models (Section 2.5.2),
2. a generic regularization scheme that prevents higher order transformation models (e. g. affine transformations) from collapsing when used in the proposed optimization framework (Section 2.5.5),
3. lens-distortion correction with an improved version of the method developed by Kaynig et al. (2010a) (Section 2.5.6), and
4. a method to warp image tiles instead of directly applying the local transformation model by which the otherwise compromised local continuity within section montages is preserved (Section 2.5.7).

2.5.1 Scale Invariant Feature Transform (SIFT)

SIFT was originally proposed for robust object recognition under partial occlusion (Lowe, 1999, 2004) and became most popular in the context of automatic panorama recognition and stitching (Brown and Lowe, 2007). It performs surprisingly well under a wide range of non-linear distortion and image artifacts (Mikolajczyk and Schmid, 2003) and therefore appears well suited for large TEM section series where such effects are inevitable (Figure 2.1). SIFT comprises a scale and rotation invariant interest point detector and an intensity based local descriptor for those interest points. Interest points are blob-like structures detectable as local extrema in a *Difference of Gaussian* (DoG) pyramid (see Figure 2.2).

Let $\mathbf{x} \in \mathbb{Z}^n$ be an n-dimensional discrete coordinate vector and $\mathcal{I} : \mathbb{Z}^n \to \mathbb{R}$ be a gray-scale image. Let further $\mathcal{G} : \mathbb{Z}^{n+1} \to \mathbb{R}$ be a series



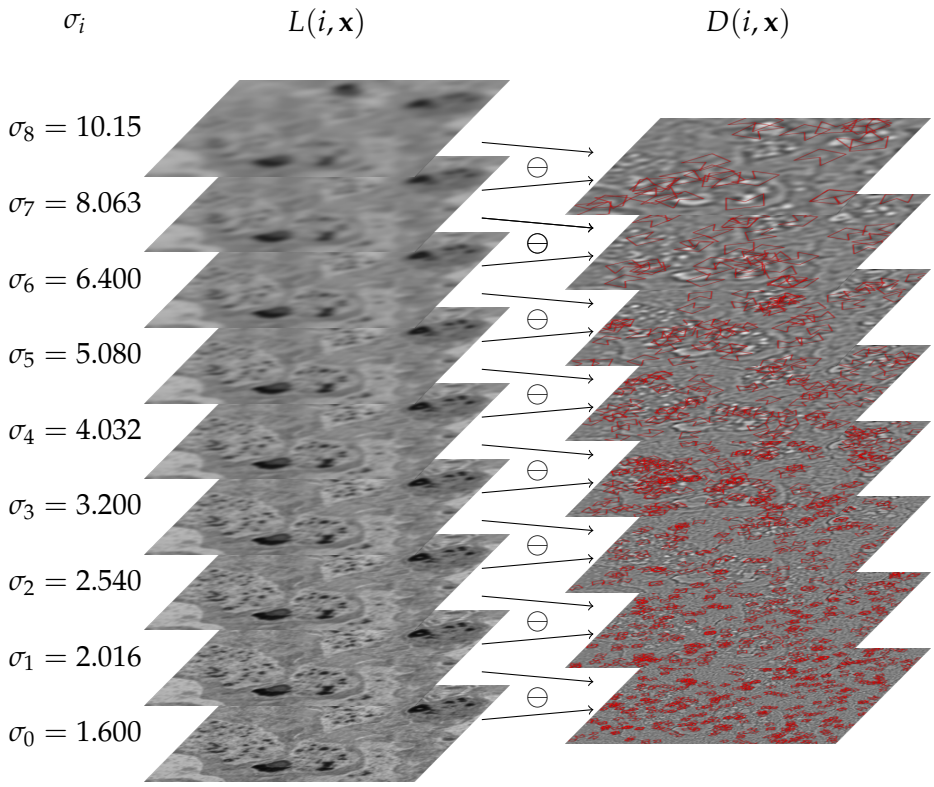
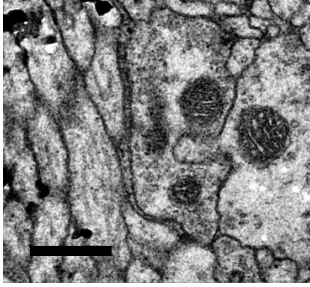


Figure 2.2: The Difference of Gaussian interest point detector. A discrete scale pyramid is built smoothing the image with Gaussian kernels of increasing size $\mathcal{L}(i, \mathbf{x}) = \mathcal{G}(i, \mathbf{x}) * \mathcal{I}(\mathbf{x})$. Two adjacent Gaussians $\mathcal{G}(i, \mathbf{x})$ and $\mathcal{G}(i + 1, \mathbf{x})$ are separated by a constant factor k such that $\sigma_{i+1} = k\sigma_i$. $\mathcal{D}(i, \mathbf{x}) = \mathcal{L}(i + 1, \mathbf{x}) - \mathcal{L}(i, \mathbf{x})$ is a constantly scaled approximation of $\sigma^2 \nabla^2 \mathcal{G}(i, \mathbf{x})$. Extrema of $\mathcal{D}(i, \mathbf{x})$ are interest point candidates that are afterwards filtered to eliminate edge responses and low contrast detections. Main orientation is sampled from the gradients of an interest point's local neighborhood. The resulting interest points are shown as oriented squares, their size corresponding to the detected scale. Figure from Saalfeld (2008).



of normalized Gaussian kernels with $i \in \mathbb{Z}_+$ enumerating scale such that $\sigma_i = k^i \sigma_0$ and

$$\mathcal{G}(i, \mathbf{x}) = \frac{1}{2\pi\sigma_i^2} e^{\frac{|\mathbf{x}|^2}{2\sigma_i^2}}. \tag{2.10}$$

Then a scale pyramid $\mathcal{L} : \mathbb{Z}^{n+1} \to \mathbb{R}$ is generated by convolution of the image with the series of Gaussian kernels $\mathcal{L}(i, \mathbf{x}) = \mathcal{G}(i, \mathbf{x}) * \mathcal{I}(\mathbf{x})$. The DoG pyramid $\mathcal{D} : \mathbb{Z}^{n+1} \to \mathbb{R}$ is built by subtracting adjacent entries of the scale pyramid $\mathcal{D}(i, \mathbf{x}) = \mathcal{L}(i+1, \mathbf{x}) - \mathcal{L}(i, \mathbf{x})$.[4] Local extrema are found by comparison of each value in $\mathcal{D}(i, \mathbf{x})$ to its direct neighborhood in both space and scale.

An approximate sub-pixel location of each detection can be calculated by fitting a quadric to its local neighborhood. Brown and Lowe (2002) use the Taylor expansion and finite first and second order derivatives at the discrete extremum-detection $\mathcal{D}$ to parameterize the quadric

$$\mathcal{D}(\mathbf{x}) = \mathcal{D} + \frac{\partial \mathcal{D}}{\partial \mathbf{x}}^\mathsf{T} \mathbf{x} + \frac{1}{2} \mathbf{x}^\mathsf{T} \frac{\partial^2 \mathcal{D}}{\partial \mathbf{x}^2} \mathbf{x} \tag{2.11}$$

with $\mathbf{x} \in \mathbb{Z}^{n+1}$ comprising both spatial dimensions and scale. Derivation of this quadric with respect to $\mathbf{x}$ and setting it to zero leads to the offset of the extremum relative to its discrete location

$$\hat{\mathbf{x}} = -\frac{\partial^2 \mathcal{D}}{\partial \mathbf{x}^2}^{-1} \frac{\partial \mathcal{D}}{\partial \mathbf{x}}. \tag{2.12}$$

DoG integrates differentials in all directions to a single scalar value and therefore has strong responses not only at the desired blob-like structures but also on hyper-planes, planes, edges, or lines, where minor fluctuations in the signal lead to spurious extrema detections. Such spurious detections lack distinct localization in at least one direction and are therefore not suited as interest points. They can be identified through analysis of their principal curvatures that are proportional to the eigenvalues of the Hessian matrix $\mathbf{H}(\mathcal{D})$ in spatial domain $\mathbb{Z}^n$. Homogeneous areas are indicated by low principal curvatures, hyper-planes by a single high principal curvatures, hyper-edges by two high principal curvatures, *etc*. Since DoG extrema exclude homogeneous areas, the absolute value of a principal curvature becomes irrelevant and it is sufficient to analyze whether there is a single low principal curvature. This can be achieved by testing the ratio of the largest eigenvalue to all other eigenvalues. In the two-dimensional case, there are only two eigenvalues α and β such that a single ratio for

[4] In practice, either of the dimensions of the coordinate domain in both space and scale will be limited. However, this complicates notation (e. g. convolution needs additional explanation when dealing with limited supports, see Chapter 3) and is not necessary to understand the general principle.

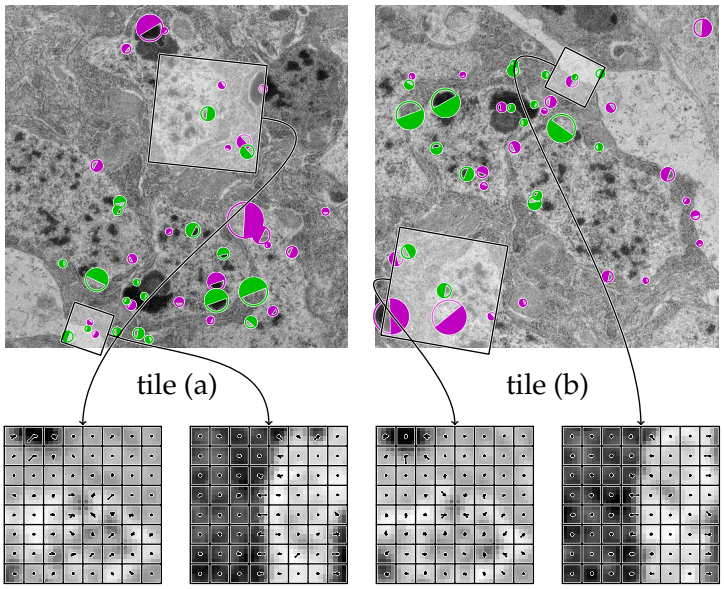


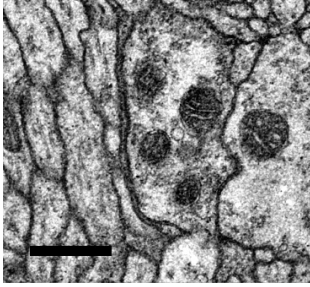
Figure 2.3: SIFT-feature correspondences in two overlapping tiles from adjacent serial TEM sections. 1003 feature candidates were extracted in tile (a), 971 in tile (b). 41 correspondence pairs were identified by local feature descriptor matching using an $8 \times 8 \times 8$ SIFT descriptor, 19 of them are true matches consistent to common transformation model. False matches are displayed in magenta, true matches in green. The size of the circles is proportional to the features scale, the filled part visualizes a feature's orientation. Two true correspondence pairs are selected as an example for the local SIFT-descriptor. Local histogram bin values are shown as combs on top of the corresponding image patch. Figure from Saalfeld (2008).

$\alpha = r\beta$ has to be tested. Lowe (2004) employed the approach by Harris and Stephens (1988) to avoid explicit calculation of the eigenvalues using the trace $\mathrm{tr}(\mathbf{H})$ and the determinant $\det(\mathbf{H})$ of the Hessian:

$$\frac{\mathrm{tr}(\mathbf{H})^2}{\det(\mathbf{H})} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{2.13}$$

DoG extrema with a curvature ratio larger than some threshold (e. g. 10) are rejected. Gradient orientations in a local region around each detection are then collected in a histogram to identify its dominant orientation (details in Lowe, 2004; Saalfeld, 2008).

For each detection, a local descriptor is extracted from a square patch around the detection. Size and orientation of the patch are defined by the scale and dominant orientation of the detection. Lowe (2004) proposed to split that patch into 4×4 sub-patches. For each sub-patch, the rotated gradient orientations are collected in a histogram with 8 bins each, resulting



in a $4 \times 4 \times 8 = 128$-dimensional description vector for each feature (details in Lowe, 2004; Saalfeld, 2008). This descriptor is the main contribution of SIFT and can be combined with detectors other than DoG (Mikolajczyk and Schmid, 2003). As mentioned earlier, I found that larger regions comprising 8×8 sub-patches were required to reliably identify matches across serial TEM sections, resulting in 512-dimensional feature descriptors (see Figure 2.3).

Both orientation detection and feature extraction were described for two-dimensional images where equally spaced orientation histograms are straight-forward to implement. This is sufficient for the application described here, however, with some minor adjustments, it is possible to transfer the concepts to higher-dimensional data (Cheung and Hamarneh, 2009).

Candidate matches in two sets of feature descriptors (e. g. from two images) are then identified by nearest neighbor search in feature descriptor space. Lowe (2004) proposed to use the ratio between the Euclidean distances to the nearest and second nearest neighbor as a criterion for distinct matches that are promising candidates. The result is a set of interest point correspondence candidates $\boldsymbol{C} = \{(\mathbf{p}_1, \mathbf{q}_1), \ldots, (\mathbf{p}_m, \mathbf{q}_m)\}$ with $(\mathbf{p}, \mathbf{q})$ denoting a pair of corresponding interest point coordinates $\mathbf{p}, \mathbf{q} \in \mathbb{R}^n$ associated with the matching descriptors in the first and second set.

Using these landmark correspondences, approximate alignment of pairs or groups of overlapping images can be established. However, since matching local descriptors does not consider global context, it is prone to deliver a substantial number of false positives. Therefore, a robust method for determining the transformation is required.

2.5.2 A Robust Estimator

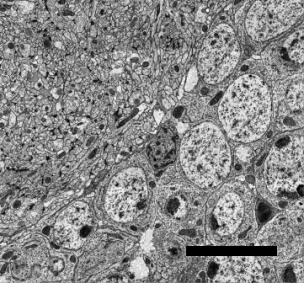
M-estimators (maximum likelihood-type estimators) were introduced by Huber (1964, 1981) as a generalization of *ML-estimators* (maximum likelihood estimators). An M-estimator of a model $T : \mathbb{R}^n \to \mathbb{R}^m$ minimizes the sum of a function $\rho : \mathbb{R} \to \mathbb{R}$ over the residuals of all samples

$$\arg\min_T \sum_{i=1}^{n} \rho(|T(\mathbf{p}_i) - \mathbf{q}_i|). \tag{2.14}$$

The most popular M-estimator is the least-squares estimator

$$\arg\min_T \sum_{i=1}^{n} |T(\mathbf{p}_i) - \mathbf{q}_i|^2 \tag{2.15}$$

which for many classes of T can be solved closed-form and thus very efficiently. Its major disadvantage is that 'outliers', e. g. samples that were falsely included in the set and therefore have large residuals, have strong deteriorating impact on the estimated model.



Huber (1981) suggested to avoid this by using a threshold c beyond which residuals are weighted using a non-quadratic function, effectively reducing their influence. The simplest implementation of this idea is the trimming function

$$\arg\min_T \sum_{i=1}^{n} \rho(|T(\mathbf{p}_i) - \mathbf{q}_i|) \quad \text{with} \quad \rho(x) = \begin{cases} x^2 & \text{if } x \leq c, \\ 0 & \text{if } x > c \end{cases} \tag{2.16}$$

which declares all samples with a residual $> c$ as 'outliers' that do not contribute to the solution.

Instead of making c a random (or observation driven) choice, it can be estimated automatically employing statistics over the residuals. The idea is to iteratively cut off the tail of the error distribution. If the residuals are approximately normal distributed then all 'outliers' will be rejected effectively. Initialize $c = \infty$ by which the estimator is equivalent with the least-squares estimator. Estimate T and calculate the mean residual with respect to T and c

$$\bar{\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \omega(|T(\mathbf{p}_i) - \mathbf{q}_i|) \quad \text{with} \quad \omega(x) = \begin{cases} x & \text{if } x \leq c, \\ 0 & \text{if } x > c. \end{cases} \tag{2.17}$$

Set $c = k\bar{\epsilon}$ and repeat until c remains unchanged. Note that k remains an adjustable parameter that constrains the trimming threshold relative to the mean residual. However, I found that the estimator performs almost equivalently for all k between 2 and 4 and therefore chose 3 as a default.

While the number of iterations is typically very small (< 5), it should be avoided for repeated estimation of the model (see Section 2.5.4). Instead of adjusting the threshold, the estimator can be used to explicitly remove 'outliers' from the sample set. The remaining set of 'inliers' enables to estimate the model directly with a least-squares estimator (see Algorithm 1).

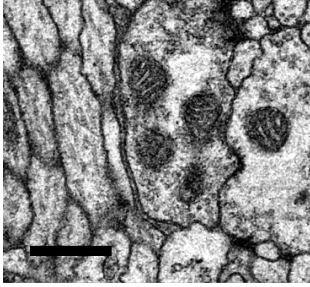
Since automatic adjustment of the trimming threshold c depends on statistics derived from models that are expected to be close to the desired solution, the following conditions must be met:

1. the model T is sufficiently overdetermined such that a meaningful mean residual $\bar{\epsilon}$ can be estimated;

2. the residuals of 'outliers' are not excessively large;

3. the number of 'outliers' is small compared to the number of 'inliers'.

For ssTEM reconstruction, Item 1 can be met by choosing an appropriately simple class of transformation (e. g. a rigid body or affine transformation) and assuring in advance that enough samples are available. Item 2 is met



Algorithm 1 Robust Trimming Estimator

Input: Set of data points $\boldsymbol{C} = \{(\mathbf{p}_1, \mathbf{q}_1), \ldots, (\mathbf{p}_n, \mathbf{q}_n)\}$
Maximal multiple k of mean residual $\bar{\epsilon}$
Output: Set of inlier data points $\boldsymbol{I} \subseteq \boldsymbol{C}$
Model T best fitting to $\boldsymbol{I}$

```
I ← C                                   ▷ Initialize best set of inliers.
ε̄ ← ∞                                   ▷ Initialize threshold.
while |I| is changing do
    T ← least-squares estimate for I    ▷ Update the model.
    calculate ε̃ for I and T
    for all (p, q) ∈ C do
        if |Tp − q| > kε̄ then
            I ← I \ {(p, q)}
        end if
    end for
end while
```

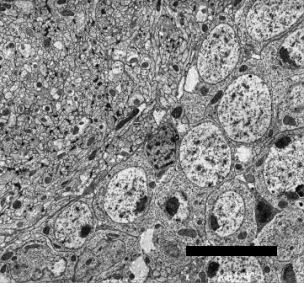
by design when comparing images of limited size and no significant scaling is involved. Item 3, however, is problematic because artifacts and distortions can lead to sets of matches where only a small fraction is correct, sometimes less than 10 %.

The tool to solve this problem is the *Random Sample Consensus* (RANSAC; Fischler and Bolles, 1981). RANSAC makes use of the assumption that all 'correct' samples are in *consensus* with the hidden transformation whereas 'wrong' samples do not support a common transformation. The hidden transformation is identified by testing many hypotheses generated from random minimal sample sets. For each hypothesis, all samples are tested whether or not they are in consensus with the hypothesis. The criterion is that the residual of the sample with respect to the hypothesis is below some threshold. If the minimal sample set contains only 'correct' samples then the corresponding hypothesis will be supported by all 'correct' matches. The best hypothesis is that with the largest consensus set (see Algorithm 2).

RANSAC in its originally proposed form is probabilistic which means that it may fail to identify the hidden transformation by non-zero chance. The algorithm has failed if each randomly selected minimal sample set contained at least one 'outlier' such that no hypothesis was correct. The probability of this event can be calculated depending on the number of iterations k, the expected ratio of 'inliers' $r = |I| \,/\, |C|$ and the minimal number of samples $m = |C_{min}|$ to estimate a hypothesis

$$p = (1 - r^m)^k. \tag{2.18}$$

In practice, it is interesting to calculate the number of iterations k required to reach a sufficiently low probability p that the algorithm fails given some



Algorithm 2 Random Sample Consensus

Input: Set of point matches $\boldsymbol{C} = \{(\mathbf{p}_1, \mathbf{q}_1), \ldots, (\mathbf{p}_n, \mathbf{q}_n)\}$
Maximal number of iterations k
Maximal accepted error ϵ_{max}
Minimal required inlier ratio r
Output: Set of inlier point matches $\boldsymbol{I} \subseteq \boldsymbol{C}$
Model T best fitting to $\boldsymbol{I}$

$\boldsymbol{I} \leftarrow \emptyset$ ▷ Initialize best set of inliers.
for k iterations **do**
  $\boldsymbol{C}_{min} \leftarrow$ choose random minimal subset of $\in \boldsymbol{C}$ to solve T
  $T \leftarrow$ estimate for $\boldsymbol{C}_{min}$ ▷ Generate hypothesis.
  $\boldsymbol{C}_+ \leftarrow \emptyset$ ▷ Initialize consensus set.
  for all $(\mathbf{p}, \mathbf{q}) \in \boldsymbol{C}$ **do**
    if $|T\mathbf{p} - \mathbf{q}| \leq \epsilon_{max}$ **then**
      $\boldsymbol{C}_+ \leftarrow \boldsymbol{C}_+ \cap \{(\mathbf{p}, \mathbf{q})\}$
    end if
  end for
  if $|\boldsymbol{C}_+| > |\boldsymbol{I}|$ **then**
    $\boldsymbol{I} \leftarrow \boldsymbol{C}_+$
  end if
end for
if $|\boldsymbol{I}|/|\boldsymbol{C}| \geq r$ **then**
  $T \leftarrow$ fit to $\boldsymbol{I}$ ▷ Refine the model with respect to all inliers.
end if

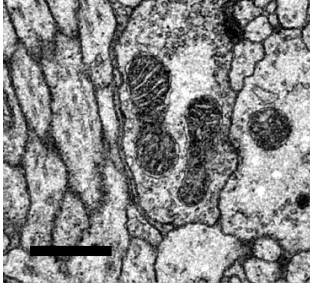
m and an expected lower bound for the inlier ratio r

$$k = \frac{\log(p)}{\log(1 - r^n)}. \tag{2.19}$$

In a large dataset, the desired lower bound for pairwise failure can be extremely small by which k becomes excessively large. In case that $k \geq |\boldsymbol{C}|^m$ it is better to test all possible minimal sample sets deterministically which has zero probability of failure.

RANSAC is very effective in separating a small fraction of 'correct' samples from a large set of false positives but leaves open where to put the threshold for accepting a match as a supporter. That gap is closed by combining it with the robust M-estimator described above. First, an approximate hidden model is identified using RANSAC with a large residual threshold. That way, all 'outliers' with excessive residuals are removed. The remaining number of 'outliers' is small and has moderate residuals such that the final model can be estimated with the robust M-estimator.

As demonstrated for robust object recognition (Lowe, 2004), and panorama recognition and stitching (Brown and Lowe, 2007), local image fea-



ture matches can be used not only to align overlapping images but to recognize whether two images overlap or not. Applications are automatic reconstruction of unordered montages and automatic identification of those sections in a series that look similar enough to be registered. To that end, the features for each candidate pair of images need to be matched and 'outliers' rejected using RANSAC and the robust M-estimator as described above. The remaining 'inliers' are in consensus with the 'true' transformation model that brings both images into alignment. Where no 'inliers' for two images could be identified, the conclusion can be drawn that they do not contain similar enough image content.

2.5.3 Affine Transformation Models

In the previous Section, I have described robust estimators for general models from sets of correspondence samples and their application for model estimation and recognition of image overlap. While this framework is generic with respect to the class of transformation, I have mainly used the family of affine transformations

$$\begin{aligned} A : \mathbb{R}^n &\to \mathbb{R}^n \\ \mathbf{x} &\mapsto \mathbf{A}\mathbf{x} + \mathbf{t}, \end{aligned} \tag{2.20}$$

with $\mathbf{A}$ being an $n \times n$ matrix and $\mathbf{t}$ being a translation vector. In addition to general affine transformations, three important subgroups were used:

Translations where $\mathbf{A} = \mathbf{I}$;

Rigid body transformations where $\mathbf{A}$ is a rotation matrix, i. e. orthogonal $\mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{A}\mathbf{A}^\mathsf{T} = \mathbf{I}$ and $\det \mathbf{A} = 1$;

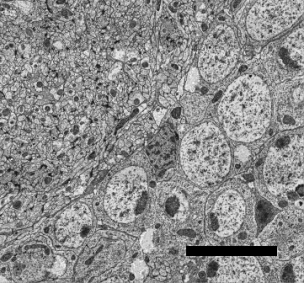
Similarity transformations where $\mathbf{A}$ complies with $\mathbf{A}^\mathsf{T}\mathbf{A} = \mathbf{A}\mathbf{A}^\mathsf{T} = v^2\mathbf{I}$, with v being a scale factor such that $\mathbf{R} = \frac{1}{v}\mathbf{A}$ is a rotation matrix.

In their work on moving least-squares interpolation, Levin (1998) and Schaefer et al. (2006) described closed-form least-squares solutions for sets of weighted correspondence samples $\boldsymbol{C} = \{(w_1, \mathbf{p}_1, \mathbf{q}_1), \ldots, (w_n, \mathbf{p}_n, \mathbf{q}_n)\}$

$$\arg\min_{\mathbf{A},\mathbf{t}} \sum_{i=1}^{n} w_i \left|\mathbf{A}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i\right|^2. \tag{2.21}$$

They showed that $\mathbf{t}$ is the translation between the weighted centroids of the sample correspondences $\bar{\mathbf{p}}$ and $\bar{\mathbf{q}}$ with respect to $\mathbf{A}$

$$\mathbf{t} = \bar{\mathbf{q}} - \mathbf{A}\bar{\mathbf{p}} \qquad \text{with} \qquad \begin{aligned} \bar{\mathbf{p}} &= \frac{\sum_{i=1}^{n} w_i \mathbf{p}_i}{\sum_{i=1}^{n} w_i}, \\ \bar{\mathbf{q}} &= \frac{\sum_{i=1}^{n} w_i \mathbf{q}_i}{\sum_{i=1}^{n} w_i}. \end{aligned} \tag{2.22}$$



which is already the solution for the translation model and enables to rewrite Equation (2.21) to

$$\arg\min_{\mathbf{A}} \sum_{i=1}^{n} w_i \left|\mathbf{A}\hat{\mathbf{p}}_i - \hat{\mathbf{q}}_i\right|^2 \qquad \text{with} \qquad \begin{aligned} \hat{\mathbf{p}}_i &= \mathbf{p}_i - \bar{\mathbf{p}}, \\ \hat{\mathbf{q}}_i &= \mathbf{q}_i - \bar{\mathbf{q}}. \end{aligned} \tag{2.23}$$

The least-squares solution for $\mathbf{A}$ in general affine transformations is accordingly

$$\mathbf{A} = \sum_{i=1}^{n} w_i \hat{\mathbf{q}}_i \hat{\mathbf{p}}_i^{\mathsf{T}} \left(\sum_{j \in \{1 \ldots n\}} w_j \hat{\mathbf{p}}_j \hat{\mathbf{p}}_j^{\mathsf{T}} \right)^{-1}. \tag{2.24}$$

For similarity transformations, Schaefer et al. (2006) have embedded the constraint $\mathbf{A}^{\mathsf{T}}\mathbf{A} = \mathbf{A}\mathbf{A}^{\mathsf{T}} = \sigma^2\mathbf{I}$ in Equation (2.24) yielding a compact solution for the two-dimensional case

$$\mathbf{A} = \frac{1}{\mu} \sum_{i=1}^{n} \begin{pmatrix} \hat{\mathbf{q}}_i^{\mathsf{T}} \\ -\hat{\mathbf{q}}_i^{\perp\mathsf{T}} \end{pmatrix} \begin{pmatrix} \hat{\mathbf{p}}_i & -\hat{\mathbf{p}}_i^{\perp} \end{pmatrix} \tag{2.25}$$

where $\perp$ denotes a 90° rotation such that $(x, y)^{\perp} = (-y, x)$ and μ is the variance of all weighted $\hat{\mathbf{p}}_i$

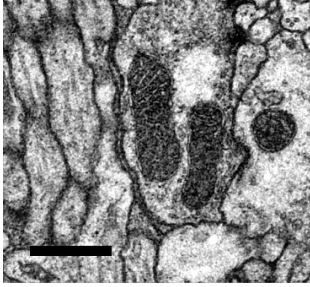
$$\mu = \sum_{i=1}^{n} w_i \hat{\mathbf{p}}_i^{\mathsf{T}} \mathbf{p}_i. \tag{2.26}$$

The authors then show that the least-squares estimate of a rigid body transformation as described by Horn (1987) is related to the similarity estimate above by only a scale factor. It is hence sufficient to calculate μ such that $\mathbf{M}$ is normalized.

2.5.4 Global Solution

Using SIFT matches (Section 2.5.1) in combination with robust estimators (Section 2.5.2) based on efficient least-squares solutions for affine transformations or one of its subgroups (Section 2.5.3), I can recognize whether or not two images contain overlapping content and identify a pairwise transformation between them.

In a large set of images such as a section series, a montage, or a series of montages, pairwise transformations need to be combined to map all images into a common registration space. As demonstrated in Theorem 1, all transformations that enable two arbitrary points to change their relative distance will lead to increasing artificial deformation when concatenated sequentially. Furthermore, if there exists more than one single path through the the graph formed by images (nodes) and pairwise transformations (edges), there are many possible sequences and it remains unclear which is best.



Algorithm 3 Iterative Optimizer

Input: Number of images t
Set of sets of pairwise point matches $\boldsymbol{C} = \{\boldsymbol{C}_{11}, \boldsymbol{C}_{12}, \ldots, \boldsymbol{C}_{tt}\}$
Output: Optimal set of transformations $\boldsymbol{T} = \{T_1, \ldots, T_t\}$

$\epsilon \leftarrow \infty$ ▷ Initialize the average residual.
$\boldsymbol{T} \leftarrow \{I, \ldots, I\}$ ▷ Initialize all transformations with the identity.
repeat
  for all $a \leq t$ **do**
    $T \leftarrow$ estimate for $\boldsymbol{C}_a = \boldsymbol{C}_{a1} \cup \cdots \cup \boldsymbol{C}_{at}$
    for all $(\mathbf{p}, \mathbf{q}) \in \boldsymbol{C}_a$ **do**
      $\mathbf{p} \leftarrow T(\mathbf{p})$ ▷ Update landmarks in $\boldsymbol{C}_a$.
    end for
    $T_a \leftarrow T \cdot T_a$ ▷ Concatenate transformations.
  end for
  $\epsilon \leftarrow$ calculate global cost
until ϵ converges

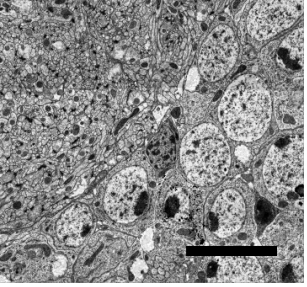
Both issues can be solved by treating registration of sets of images that are connected by landmark correspondences as a global optimization problem. Although this Section is about two-dimensional images and transformations (see Section 2.4), I will keep the formulation generic in terms of dimensionality which will simplify its transfer to other projects later. Let $\boldsymbol{C}_{ab} = \{(\mathbf{p}_1, \mathbf{q}_1), \ldots, (\mathbf{p}_c, \mathbf{q}_c)\}$ be the set of landmark correspondences between two images indexed by a and b. Let now $T_a : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be the transformation that maps an image indexed by a into registration space $\mathbb{R}^m$. The desired set of transformations $\boldsymbol{T} = \{T_1, \ldots, T_t\}$ is that minimizing the cost

$$\arg\min_{\boldsymbol{T}} \sum_{a=1}^{t} \left(\sum_{b=1}^{t} \left(\sum_{(\mathbf{p},\mathbf{q}) \in \boldsymbol{C}_{ab}} \rho \left(|T_a(\mathbf{p}) - T_b(\mathbf{q})| \right) \right) \right). \tag{2.27}$$

If the sets of pairwise landmark correspondences contain only 'inliers', the task can be formulated directly by means of least-square residuals

$$\arg\min_{\boldsymbol{T}} \sum_{a=1}^{t} \left(\sum_{b=1}^{t} \left(\sum_{(\mathbf{p},\mathbf{q}) \in \boldsymbol{C}_{ab}} |T_a(\mathbf{p}) - T_b(\mathbf{q})|^2 \right) \right). \tag{2.28}$$

I minimize this cost function using an iterative optimization scheme (see Algorithm 3). In case that prior knowledge about the gross configuration of all tiles is available, the minimization is initialized with this configuration. Otherwise all transformations initially become the identity transformation. In each iteration i and for each image indexed by a, the optimal transformation model T_a relative to the current global configuration is estimated and applied to all landmark coordinates in this tile. The scheme



terminates on convergence of the global cost ϵ as a function of i. In the least-squares formulation, this is the sum of all square residuals. As convergence criteria, I require ϵ to have a low absolute slope $|\nabla_h \epsilon|$ and an absolute value below some user-defined threshold. It is crucial to prevent being trapped in local minima or at long plateaus. This is addressed effectively by requiring $|\nabla_h \epsilon|$ to be very low for all h from a set of decreasing lengths. The maximal length h of a plateau or local valley, the maximal total number of iterations and the maximal accepted remaining error are user-defined parameters. It is possible to use individual images as global anchors by skipping them during the iteration and keeping their transformation unchanged.

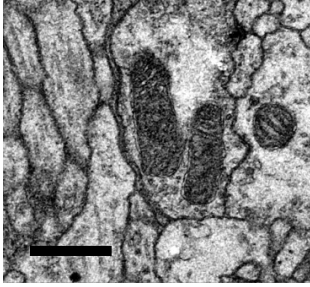
I have originally formulated and implemented this solution in Saalfeld (2008). An improved version using the robust estimator described in Section 2.5.2 has been published in Saalfeld et al. (2010) and applied to related problems together with Stephan Preibisch et al. (2009a,b, 2010a).

2.5.5 Regularization

Using a transformation model that enables scaling (e. g. an affine transformation or even a similarity transformation) with a least-squares cost as in Equation (2.28) has its optimum at a scale factor of zero. That can be prevented by including a regularization term into Equation (2.27) that penalizes non-rigid transformation (such as scaling or shear) while otherwise enabling the full degrees of freedom of the transformation model. I have realized regularization through a combination of two transformations $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $R : \mathbb{R}^n \rightarrow \mathbb{R}^m$ that transfers a coordinate $\mathbf{x}$ to the linearly interpolated location $\mathbf{q} = T(\mathbf{p}) = (1 - \lambda)A(\mathbf{p}) + \lambda R(\mathbf{p})$. This can be directly used in the global least-squares minimizer in Equation (2.28). In the iterative solution, the pseudo-least-squares solution of T is estimated by independently calculating the least-squares solution for A and R.

Note that for regularization to work as proposed here, the solution described in Algorithm 3 needs to be adjusted. Instead of accumulating transformations during the iterative process, the local coordinates of landmarks per each image are stored redundantly to their actual location in registration space. Then, at each iteration, the transformation is estimated for a set of landmark correspondences associating these local coordinates to their correspondences in the current state of the registration space. I. e., the local to world transformation is calculated directly at each iteration instead of accumulating it. I have implemented the optimizer in this way, but decided to omit the redundant local coordinates in Algorithm 3 and instead show the accumulating version because it is shorter in notation and easier to understand.

For the family of affine transformations, another observation becomes interesting. We know that in *homogeneous coordinates* (Möbius, 1827) an



affine transformation constitutes a square matrix of the form

$$\mathbf{A}_{\mathrm{H}} = \begin{pmatrix} \mathbf{A} & \mathbf{t} \\ 0 & 1 \end{pmatrix}. \tag{2.29}$$

For $\mathbf{A}$ and $\mathbf{R}$ being square matrices of the same size, an interpolated matrix $\mathbf{T}$ can be calculated instead of applying both matrices independently to a source vector $\mathbf{p}$ and interpolating the target vectors $\mathbf{Ax}$ and $\mathbf{Rx}$:

$$\begin{aligned} \mathbf{q} = \mathbf{Tp} &= (1-\lambda)\mathbf{Ap} + \lambda\mathbf{Rp} \\ &= \mathbf{Ap} - \lambda\mathbf{Ap} + \lambda\mathbf{Rp} \\ &= (\mathbf{A} - \lambda\mathbf{A} + \lambda\mathbf{R})\mathbf{p} \\ &= ((1-\lambda)\mathbf{A} + \lambda\mathbf{R})\mathbf{p} \end{aligned} \tag{2.30}$$

$$\mathbf{T} = (1-\lambda)\mathbf{A} + \lambda\mathbf{R}. \tag{2.31}$$

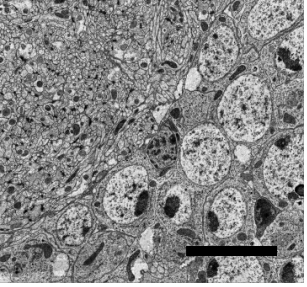
Interpolating two matrices once and then applying the interpolated transformation to a point is effectively twice as efficient as interpolating two individually transferred point locations. This is particularly important for an iterative solution that requires repeated transfer of landmarks as described in Section 2.5.4.

2.5.6 Correct lens distortion

As discussed in Sections 2.2 and 2.4, the beam-path in electron microscopes is formed by a series of electromagnetic lenses (see Figure 1.1). Currently available instruments allow the operator to adjust the objective lens system to minimize distortion. However, Kaynig et al. (2010a) found that a substantial amount of distortions originate from the the projective lens system which usually cannot be calibrated. That is, even carefully operated and calibrated electron microscopes suffer from visible image distortion induced by beam path disturbance.

Kaynig et al. (2010a) have developed a method for automatic lens-distortion correction from montages of redundantly overlapping textured images, i. e. the L function in Equation (2.8). Since their approach is not expected to compensate for tile-to-tile section distortion, they excluded this component. Instead of a translation, they enable tiles to be transformed by an affine transformation, using one tile as a global anchor. Similarly to our method described above, the residuals of automatically extracted landmark correspondences constitute the cost to be minimized. The respective optimization problem with respect to some cost function ρ on the landmark residuals is

$$\arg\min_{A,L} \sum_{a=1}^{t} \left(\sum_{b=1}^{t} \left(\sum_{(\mathbf{p},\mathbf{q})\in C_{ab}} \rho\left(|A_a(L(\mathbf{p})) - A_b(L(\mathbf{q}))|\right) \right) \right) \tag{2.32}$$



simultaneously estimating the affine transformations $\boldsymbol{A} = \{A_1, \ldots, A_t\}$ for each image and a shared lens-correction model L that is modeled by an n-th order polynomial transformation.

Comparable to the approach described in Section 2.5.4, the authors solve this minimization problem with an iterative solution. In each iteration, they first estimate the set of affine transformations for a given lens-correction model and then estimate the lens-correction model for a given set of affine transformations.

Originally, they proposed to use the Huber-loss function (Huber, 1964)

$$\rho(x) = \begin{cases} \frac{x^2}{2} & \text{if } x \leq c, \\ cx - \frac{c^2}{2} & \text{if } x > c \end{cases} \tag{2.33}$$

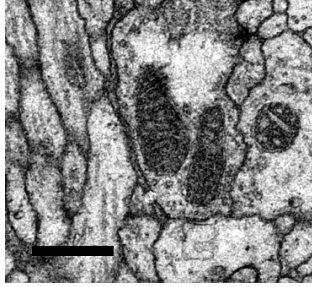
instead of the sum of square residuals as a more robust estimator with respect to 'outliers'. They described and implemented the method for 3×3 montages of images with ~50 % overlap. All tiles were mapped into the center tile using an affine transformation. I have generalized this by replacing their estimator for the set of affine transformations by the above described optimizer (see Section 2.5.4). This enables three improvements over the original solution:

1. it can be used on montages of arbitrary size;
2. it does not require one tile to serve as a global anchor;
3. it enables to use other transformations than general affine transformations.

I consider the last point the most important contribution because it helps constraining the model better. While lens-distortion correction leads to visual improvement over non-corrected montages, it cannot compensate heat-induced deformation that is individual to each single tile. In Section 2.6, I will show that this can be accomplished using an elastic model.

2.5.7 Warping with Moving Least-Squares

The global approach described in Section 2.5.4 can be used to montage individual sections, section series or series of montages. The only requirement is that images are connected by landmark correspondences. When applied to a series of montages as a global solution, the result is a piecewise approximation of the non-linear elastic deformation of each individual section montage. Pieces are individual tiles that are transformed by an affine or even more constrained transformation (e. g. rigid body or similarity). The consequence is that the registration quality within each montage is worse than when montaged individually. Instead of transforming individual tiles using the estimated affine transformation, one can calculate a



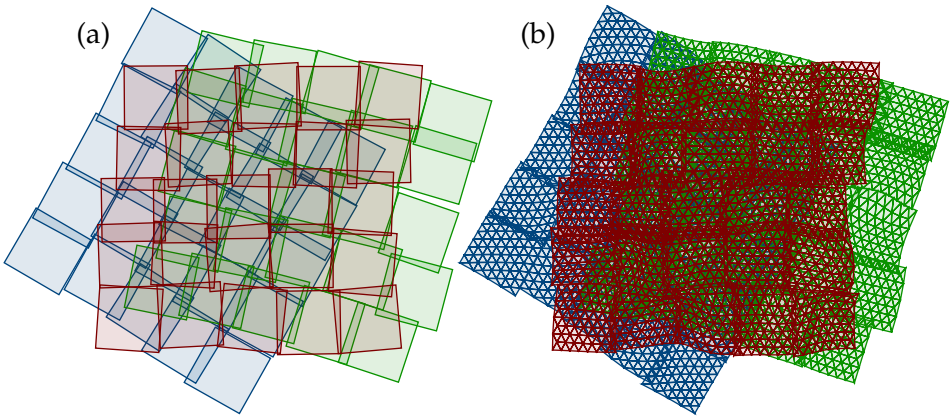
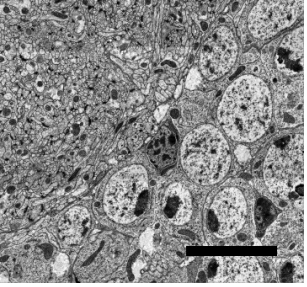


Figure 2.4: Illustration of three warped montages using the moving least squares method (Schaefer et al., 2006). Each tile's center point was calculated by global registration of the series of montages (a). These points are used as a control-points to warp the independent montage (b). That way, the globally deformed shape is preserved but section montages are continuous. In this figure, warping is exaggerated non-realistically for better visualization.

smooth non-linear deformation that has a similar global appearance. Given that an affine transformation has been assigned to all tiles by registering a series of montages globally, the location of the center point of each individual tile is stored. Then, all montages are registered individually, improving the registration quality within the montage. A smooth non-linear transformation that warps each montage such that the center points of each tile are transferred to the earlier stored location will generate a globally registered series of individually registered montages. I have used the moving least-squares method described by Schaefer et al. (2006). Figure 2.4 illustrates the effect.

2.6 Elastic Alignment and Montaging

Alignment can be achieved by maximizing the overlap of similar image content between adjacent sections. However, there are two unknown components that change image content across the section series: the specimen's shape and independent section distortion introduced during preparation. For volume reconstruction, only the latter must be removed as naively warping one section into another would introduce artificial deformation by compensating for signal change that originates from the shape of the specimen. I cope with this dilemma by exploiting the fact that the biological specimen's shape typically changes smoothly across sections whereas the independent distortion in each section is random and uncorrelated with



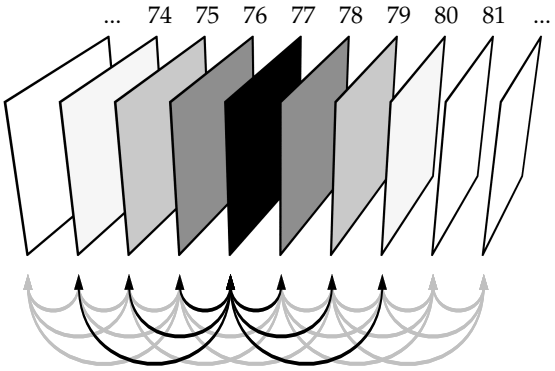


Figure 2.5: Illustration of alignment context in a section series. All sections in the series are aligned not only to their direct neighbors but to all sections in a local neighborhood. Sections are shaded to visualize how the influence of cross-section connections decreases in inverse proportion to the distance between the two sections in the series. That influence is specified by the spring constant k.

neighboring sections. If section distortions were really independent, their average in an infinitely large series would be zero. I. e., registering all to all sections in a large series would enable to remove independent distortions by averaging. Unfortunately, due to the signal changing across sections, this is not possible. Still, sections at small distances can usually be aligned well, particularly in regions where the signal change due to the specimen's shape is small. Consequently, I align all sections not only to their direct neighbors in the series but to all sections in a local neighborhood (Figure 2.5 and Figure 2.13). Averaging of the distortions is achieved by modeling sections as two-dimensional elastic sheets that introduce a cost for non-rigid deformation. All sections are treated as moving targets in a template-free global alignment procedure. This global elastic system locally averages independent distortions and by that restores a very good approximation of the original volume. The elastic constraint is implemented as a spring-connected particle system where each section is represented as a triangular mesh (Figure 2.6 and Video 1).

2.6.1 The Elastic Model

In the elastic section model, vertices are connected by ideal springs according to Hooke's law. A Hookean spring has a relaxed length l at which it exerts no force. Both extending and compressing a Hookean spring results in increasing stress. The stress amplitude is proportional to the difference of the spring's actual length and its relaxed length. Springs connecting the vertices of an image-mesh have a relaxed length correspond-

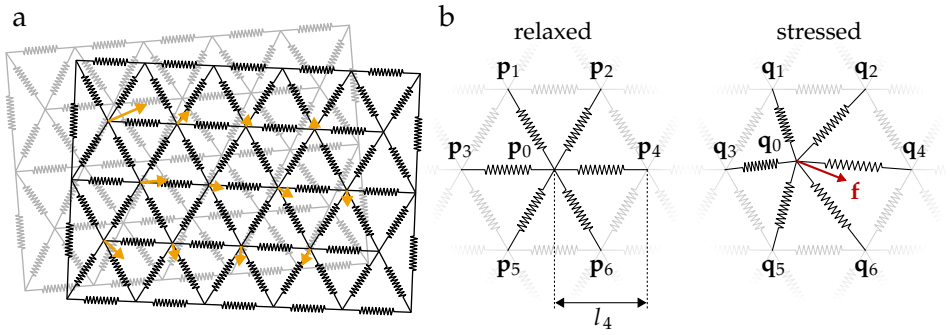
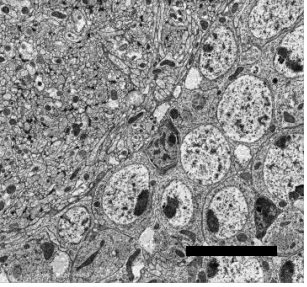


Figure 2.6: Spring connected particles. (a) Sections are modeled as elastic sheets by a 2d spring-connected triangle mesh. Springs within the mesh stabilize the section. Springs across sections are depicted by orange arrows; they have a relaxed length of zero and drag the sections toward alignment. (b) Each vertex $(\mathbf{p}, \mathbf{q})_0$ is connected to other vertices $(\mathbf{p}, \mathbf{q})_i$ by an ideal spring. Each spring has a relaxed length l_i (shown for $(\mathbf{p}, \mathbf{q})_4$). The relaxed length is zero for springs connecting to other images. The force of each spring can be calculated using Hooke's law summing up to the combined force vector $\mathbf{F}$. The desired end-state of the system is where $\mathbf{F} = 0$ for all vertices.

ing to the distance of the vertices in the non-deformed image. Deforming the image-mesh compresses and extends springs and therefore results in stress. Hooke's law enables to model springs with a relaxed length of zero for which no physical equivalent exists. A zero-length spring exerts force proportional to its extension beyond zero-length; it cannot be compressed. Zero-length springs can be used to connect points that should be positioned at the same location. Accordingly, corresponding locations between two overlapping images (tiles in a montage or sections in a section series) are connected by zero-length springs. These springs aim to warp the images towards perfect overlap. In contrast, the non-zero length springs building the triangular mesh for each section image prefer a locally rigid transformation of each image. That way, the system penalizes arbitrary warp and distributes deformation among all images.

Each image is tessellated into a mesh of regular triangles with each vertex being connected to its neighboring vertices by a spring whose relaxed length is the original edge length of the triangle (Figure 2.6). For those vertices of the mesh on image $\mathcal{I}_1$ overlapping image $\mathcal{I}_2$, I identify their corresponding location in image $\mathcal{I}_2$ by block matching (see Section 2.6.2). The vertex is then connected into the mesh on image $\mathcal{I}_2$ by a zero-length spring with its target end being located at an arbitrary place inside a triangle of the target mesh. Note that this 'passive' end does not contribute to the de-



formation of the mesh on image $\mathcal{I}_2$ because it is not connected to any of its vertices by a spring. During relaxation, its location is updated according to the affine transformation defined by the three vertices of the embedding triangle. Vice versa, vertices of the mesh on image $\mathcal{I}_2$ are connected to their corresponding location in image $\mathcal{I}_1$ with their 'passive' ends updated according to the affine transformation of the embedding triangle in the mesh on image $\mathcal{I}_1$.

The stiffness of ideal Hookean springs is specified by the spring constant k. Increasing k for springs spanning the triangle-mesh will lead to less deformed images and also less well aligned solutions. Using too small k effectively eliminates the elastic constraint and will therefore result in arbitrarily warped solutions. I have empirically estimated a spring-constant $k = 0.1$ performing well for our TEM series. During series alignment, the spring-constant for cross-section springs depends on the index distance ds in the series ($k = \frac{1}{\mathrm{d}s}$) giving farther away sections less impact.

Relaxation of the elastic system is done using an iterative solution similar to gradient descent. For each vertex $(\mathbf{p}, \mathbf{q})$, its local coordinate $\mathbf{p}$ in the non-deformed image is stored, i. e. $\mathbf{p}$ is constant. The desired end-state of the system is where for each vertex the forces of all attached springs combine to zero. The force vector $\mathbf{f}_0$ for a vertex $(\mathbf{p}, \mathbf{q})_0$ can be calculated using Hooke's law (Figure 2.6)

$$\mathbf{f} = -\sum_{i=1}^{n} k_i \mathbf{x}_i \quad \text{with} \quad \mathbf{x}_i = \left(1 - \frac{l_i}{|\mathbf{q}_i - \mathbf{q}_0|}\right)(\mathbf{q}_i - \mathbf{q}_0) \tag{2.34}$$

$$\begin{aligned} \text{and} \qquad l_i &= |\mathbf{p}_i - \mathbf{p}_0| && \text{for all section-mesh springs} \\ l_i &= 0 && \text{for all cross-section springs.} \end{aligned}$$

At each iteration, force vectors are calculated for all vertices and then all vertices are moved alongside their force vector. The distance of the move is the length of the force vector divided by the length of the largest force vector in the entire system. That way, the maximum step size per iteration is one pixel. All 'passive' spring ends are moved according to the affine transformation specified by the embedding triangle, preserving their relative location in the triangle. The solution typically converges within a few hundred iterations.

A triangle of springs has two families of cost minima in the plane: (1) at rigid transformations and (2) at rigid transformations flipped. That is, for all local deformations smaller than the size of a triangle, the mesh will drag towards a rigid transformation. For larger deformation, it may fold. The resolution of the triangle meshes should thus be chosen such that the expected local deformation does not exceed the side-length of a triangle.

The mesh resolution chosen for our experiments (24 in the *C. elegans* series and 48 in the *Drosophila* series, Section 2.7) is a trade-off between alignment accuracy and execution speed. Since correspondences are searched

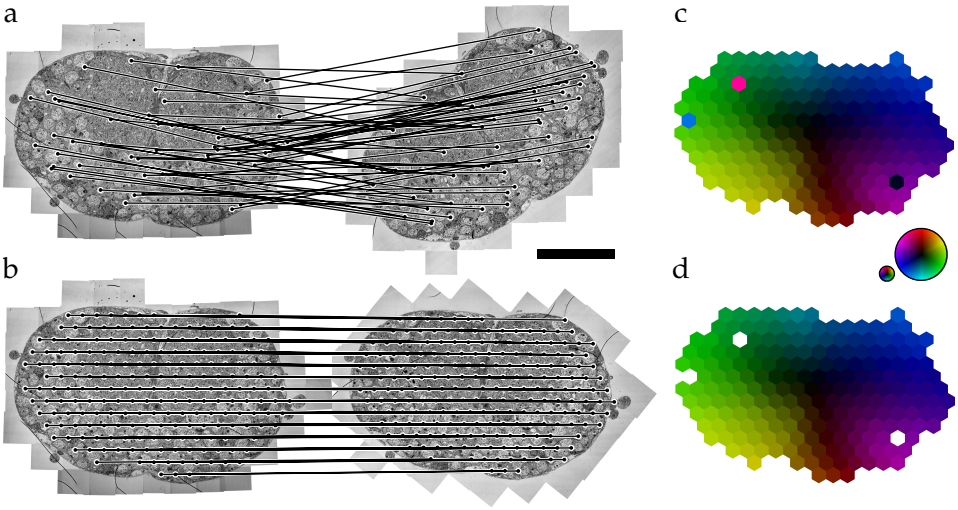
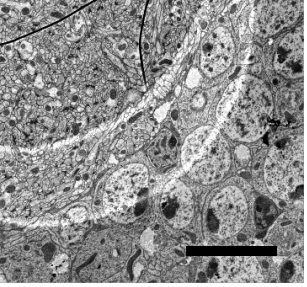


Figure 2.7: Block matching on approximately aligned images. (a) Corresponding SIFT landmarks in two adjacent electron microscopy sections are connected by lines. (b) These landmarks are used to calculate an initial approximate alignment, and the remaining local deformation is estimated by block matching, visualized here by lines connecting the corresponding locations. (c) The resulting deformation field is displayed as color-intensity encoded displacement vectors. (d) Spurious matches show up as 'outlier' colors and are automatically rejected using local and global filters. Scale bar 20 µm=5,000 px; radius of the original orientation-length scale (small circle) 2 µm=500 px, magnified copy for better visualization.

for each vertex of the mesh by block matching, increasing the resolution of the mesh improves accuracy but increases runtime. Around discontinuities like section folds or tears (see Figure 2.1c,d,g,j), a region of up to the side-length of a triangle suffers from potentially inaccurate alignment. The error decreases with increasing distance to the discontinuity and reaches 'normal' values at the boundary to an adjacent triangle if for all its vertices a block match could be identified. Although the method in its present form does not explicitly model discontinuous deformation, it performs robustly in their presence. A discontinuous deformation in one section of a series will not compromise the alignment of the series because it will be overruled by all sections in their local neighborhood.

2.6.2 Block Matching

For each vertex of the spring-mesh, the corresponding location in other sections is searched by pairwise block matching. Since block matching will



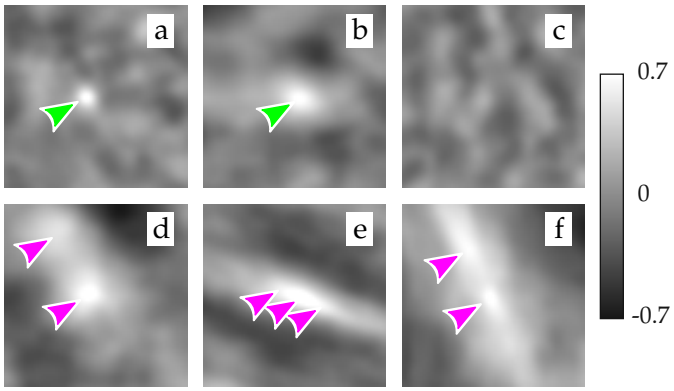


Figure 2.8: Local block matching filters. Shown is the NCC coefficient r of a block of a image $\mathcal{I}_1$ over a reference image $\mathcal{I}_2$ for all translational offsets in a square region with the coordinate origin in the center. Coefficients r are displayed as gray values (see scale). The candidate for the sought after translational offset is the location with maximal r (a,b). Candidates are rejected if either r was below a given threshold (c; not similar), there was more than one maximum with very similar r (d,f; ambiguous), or the maximum is not well localized in both dimensions (e,f; an edge pattern that fits everywhere alongside the edge).

identify only local translational offsets, sections need to be in approximate alignment which is estimated beforehand using the landmark-based approach as described in Section 2.5. The local vicinity around each vertex of the section spring-mesh is inspected for an optimal match. I use the NCC coefficient r of a patch around the vertex in image $\mathcal{I}_1$ and the overlapping patch in the other image $\mathcal{I}_2$ as quality measure for a match. The location with maximal r specifies the offset of the vertex relative to the initial approximate alignment. Block matching is executed on reasonably down-scaled versions of the images. The ideal scaling factor depends on the application and quality of the signal. In ssTEM series, the disparity between lateral and axial resolution suggests a scaling factor of 0.1 by which isotropic resolution is approximated. To overcome the reduced accuracy of the estimated offset, I use the method developed by Brown and Lowe (2002) to calculate an approximate sub-pixel offset (see Section 2.5.1 and Equation (2.12)).

Similar to matching local image features, block matching creates a substantial number of spurious matches, most prominently in regions where the signal is obstructed by artifacts. Such matches would result in significant misalignment and artificial deformation. I identify and reject such spurious matches using a set of filters that include local properties of features and block matches as well as global geometric constraints imposed by the supported transformation (Figures 2.7 and 2.8).

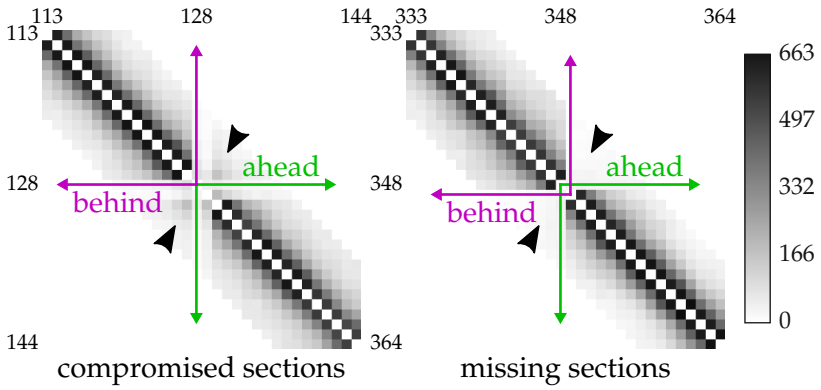
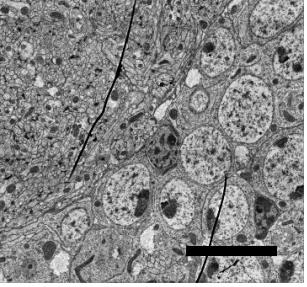


Figure 2.9: The number of filtered block matches can be used to detect artifacts and missing sections. Shown are two areas along the diagonal cropped from an all-to-all matrix showing the number of block matches that have passed the filters for each pair of sections from the *Drosophila* series. Numbers are encoded as gray values (see scale). Both panels show that the number of matches decreases with increasing distance of two sections in the series. The left panel shows the effect of two compromised sections (128 and 129) where more than 50 % of the surface was covered with artifacts. Compromised sections will have a reduced number of matches with all sections behind and ahead in the series. Sections behind and ahead of the compromised sections generate the number of matches that corresponds to their relative distance (arrows). The right panel shows the effect of a gap of five sections between sections 348 and 349, lost during collection or imaging. The number of matches is consistently reduced for the section before the gap in forward direction and for the section after the gap in inverse direction (arrows).

Correlation threshold Block matches with an NCC coefficient r below a user specified threshold are rejected (Figure 2.8c). The NCC coefficient r ranges from -1.0 to 1.0 with $r = 1.0$ indicating perfect linear dependency, $r = 0.0$ indicating no linear dependency and $r = -1.0$ indicating inverse linear dependency.

Edge responses Similar to the edge filter for interest point detections as described in Section 2.5.1 (Harris and Stephens, 1988; Lowe, 2004), block matches that were detected on top of elongated structures (edges, ridges, stripes) pose the risk of poor localization alongside the structure (Figure 2.8d,f). In the correlation surface, such detections have a large (orthogonal to the ridge) and a small (alongside the ridge) principal curvature and can thus be identified by a large ratio between the two values. Detections with a ratio beyond a given threshold are rejected.



Ambiguous matches Lowe (2004) proposed to compare the distances of the reference to the best and second-best match as a filter for local feature descriptor matching. For a distinctive match, the ratio between the two distances is most likely significantly lower than 1.0 whereas for a wrong match, many matches are expected to have very similar distances leading to a ratio close to 1.0. During block matching, I use the filter to reject matches where multiple offsets result in similar correlation (Figure 2.8e,f).

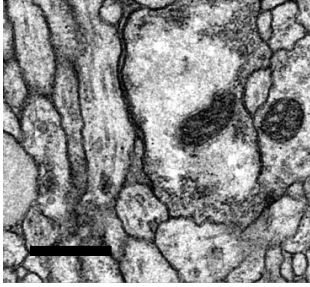
Using local methods exclusively will lead either to false positives being accepted when using too soft constraints or to many correct matches being rejected when using too hard constraints. An alternative is to accept a reasonable ratio of false positives and filter them later using global constraints. False positive feature matches were filtered using a combination of RANSAC with a robust M-estimator with respect to a simple global transformation model (see Section 2.5.2). For block matches, I use a local variation of the described robust estimator. Each match $(\mathbf{p}_0, \mathbf{q}_0)$ is inspected individually for being an 'outlier'. To that end, all other block matches $(\mathbf{p}_i, \mathbf{q}_i)$ are used to estimate a global transformation T (e.g. an affine transformation) by means of weighted least-squares (Schaefer et al., 2006). To that end, each match $(\mathbf{p}_i, \mathbf{q}_i)$ is weighted by a Gaussian radial distribution function (RDF) $\omega : \mathbb{R}^n \to \mathbb{R}$ centered at the reference $(\mathbf{p}_0, \mathbf{q}_0)$

$$\arg\min_T \sum_{i=1}^{n} \omega_i \left| T(\mathbf{p}_i) - \mathbf{q}_i \right|^2 \quad \text{with} \quad \omega_i = e^{\frac{|\mathbf{p}_i - \mathbf{p}_0|}{-2\sigma^2}} \tag{2.35}$$

Choosing a larger standard deviation σ for the RDF ω requires the deformation field to be smoother. A match is rejected if its transfer error with respect to the estimated model is larger than a given threshold or if it is larger than k times the average transfer error. The average transfer error is accumulated from all matches weighted by the RDF ω accordingly. The filter is applied in a loop until no match has been removed.

Naturally, the fraction of matches passing the filters degrades with increasing distance between the pair of compared sections in the series, constituting a coarse deformation-invariant distance metric for two sections. It can be used to correct the order of the series, to estimate the number of missing sections, and the amount of noise in a section (Figure 2.9).

Finally, all vertices for which corresponding locations in other sections could be identified are connected by zero-length springs to those sections. That way, each vertex connects to 0 to $2k$ sections where k is limiting the range of sections to be tested (Figure 2.5). This distance in the series to which cross-section connections spread is limited by how rapidly the biological structure changes across sections (for ~50 nm Transmission Electron Microscopy section series (ssTEM) it is typically $k = 7 \pm 5$ sections).



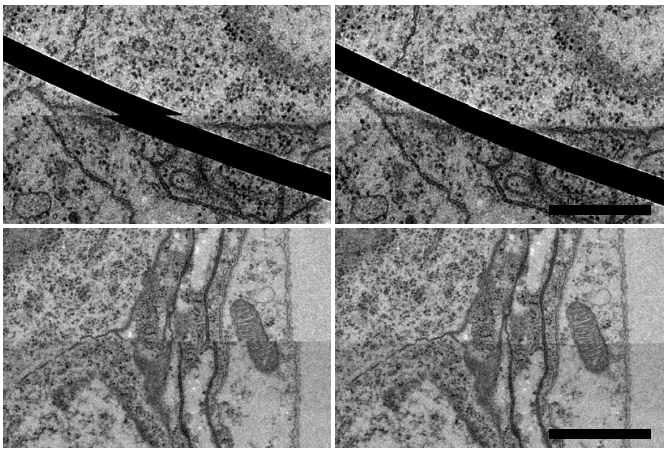


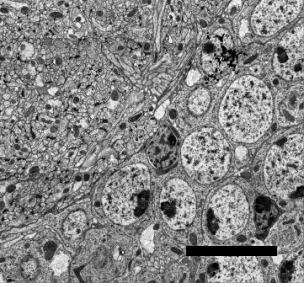
Figure 2.10: Elastic montaging compensates for non-rigid deformation. Lens distortion effects and heat-induced deformation of sections during imaging renders montaging with a rigid transformation per tile insufficient. Only the elastic approach can compensate for non-linear tile-to-tile distortion while minimizing the non-rigid deformation simultaneously. Note that the shown stitching errors do not result from non-optimal rigid alignment but exclusively from non-rigid deformation. Scale bars 500 nm = 125 px top row, 1 µm = 250 px, bottom row.

Similarly to elastically aligning a series of deformed serial sections, the method can be used to assemble montages from deformed overlapping image tiles covering a single section, compensating for non-linear tile-to-tile distortion (Figure 2.10). Taken together, a single framework can be used to reconstruct massive series of tiled sections.

Block Matching with Integral Images

The current implementation estimates local offsets by naïvely matching blocks in the spatial domain. Each vertex and therefore each block matching operation is processed independently and can be (and is) executed in parallel with others. This is an efficient solution for a set of sparse vertices with non-overlapping blocks and search areas.

Nonetheless, I have investigated a method to estimate a dense deformation field for a pair of images for each individual pixel using NCC as similarity metric. In this situation, matching a block for each pixel individually is excessively redundant since both blocks and search areas overlap to a large extent. I found that it is possible to remove all redundancies through the use of *integral images*, also called *summed-area tables* (Crow, 1984). While the approach is not yet used in the application for serial section registra-



tion, I expect it to be a useful tool for future improvements, and therefore report it here. Besides delivering a dense deformation field as estimated by block matching, it enables to test several block sizes at once for a runtime cost that is only linear to the number of sizes tested. In future work, I will investigate how this approach can be used to improve the quality of the deformation field, e. g. at local discontinuities and at locations that contain dominant staining artifacts.

Integral images have been introduced by Crow (1984) as a technique to improve texture rendering speed at multiple scales in perspective projections. The technique has since then been used for a number of applications. The most popular examples are fast normalization of the cross-correlation coefficient in template matching (Lewis, 1995), the Viola-Jones object detection framework (Viola and Jones, 2004), and the Speeded Up Robust Feature (SURF) transform (Bay et al., 2008). In an integral image $\mathcal{I}(x, y)$, each pixel stores the sum of all pixels of the original image $\mathcal{J}(x, y)$ in the rectangle between the pixel's coordinate and the origin. The sum of pixels in any box defined by its top-left and bottom-right coordinates (t, l) and (b, r) can be calculated by the sum

$$\sum_{(b,r)}^{(t,l)} \mathcal{J}(x, y) = \mathcal{I}(b, r) - \mathcal{I}(t-1, r) - \mathcal{I}(b, l-1) + \mathcal{I}(t-1, l-1). \quad (2.36)$$

It is obvious that this principle can be transferred into higher dimensions as Equation (2.36) applies the fundamental theorem of calculus

$$\int_a^b f'(x)dx = f(b) - f(a). \quad (2.37)$$

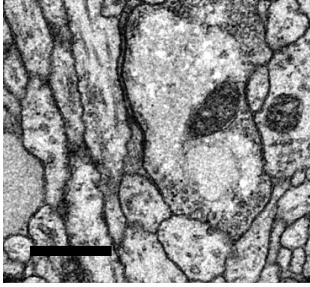
on a double integral (Tapia, 2011). In this thesis, however, I am working with two-dimensional images for which Equation (2.36) can be applied directly.

We deal with a situation where the intensities in two overlapping image regions $\mathcal{X}$ and $\mathcal{Y}$ might vary in brightness and contrast. Therefore, a simple estimate like e. g. SSD cannot be used as a similarity measure because it is not invariant with respect to a linear transformation. The NCC (also *Pearson Product-Moment Correlation Coefficient*, PMCC) $\rho_{\mathcal{X},\mathcal{Y}}$ is an appropriate measure for linear dependency

$$\rho_{\mathcal{X}\mathcal{Y}} = \frac{\sigma_{\mathcal{X}\mathcal{Y}}}{\sigma_{\mathcal{X}}\sigma_{\mathcal{Y}}} \quad (2.38)$$

which, for $\mathcal{X}$ and $\mathcal{Y}$ being a finite sample with n elements each (two pixel blocks) gives the correlation coefficient $r_{\mathcal{X}\mathcal{Y}}$

$$r_{\mathcal{X}\mathcal{Y}} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad \text{with} \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (2.39)$$



that can be transformed yielding a set of independent sums. For the numerator, that is

$$\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y}) = \sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}x_i\bar{y} - \sum_{i=1}^{n}y_i\bar{x} + \sum_{i=1}^{n}\bar{x}\bar{y} \tag{2.40}$$

$$= \sum_{i=1}^{n}x_iy_i - \bar{y}\sum_{i=1}^{n}x_i - \bar{x}\sum_{i=1}^{n}y_i + n\bar{x}\bar{y} \tag{2.41}$$

$$= \sum_{i=1}^{n}x_iy_i - \frac{1}{n}\sum_{i=1}^{n}y_i\sum_{i=1}^{n}x_i \tag{2.42}$$

For the denominator, it is handy to multiply with $\frac{n}{n}$ first

$$r_{\mathcal{XY}} = \frac{\sum_{i=1}^{n}x_iy_i - \frac{1}{n}\sum_{i=1}^{n}y_i\sum_{i=1}^{n}x_i}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{2.43}$$

$$= \frac{n\sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}y_i\sum_{i=1}^{n}x_i}{n\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{2.44}$$

$$= \frac{n\sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}y_i\sum_{i=1}^{n}x_i}{\sqrt{n\sum_{i=1}^{n}(x_i-\bar{x})^2}\sqrt{n\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{2.45}$$

because

$$n\sum_{i=1}^{n}(x_i-\bar{x})^2 = n\sum_{i=1}^{n}(x_i^2 - 2x_i\bar{x} + \bar{x}^2) \tag{2.46}$$

$$= n\sum_{i=1}^{n}x_i^2 - 2n\bar{x}\sum_{i=1}^{n}x_i + \left(\sum_{i=1}^{n}x_i\right)^2 \tag{2.47}$$

$$= n\sum_{i=1}^{n}x_i^2 - 2\left(\sum_{i=1}^{n}x_i\right)^2 + \left(\sum_{i=1}^{n}x_i\right)^2 \tag{2.48}$$

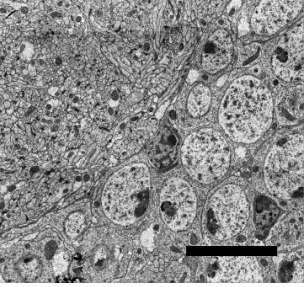
$$= n\sum_{i=1}^{n}x_i^2 - \left(\sum_{i=1}^{n}x_i\right)^2 \tag{2.49}$$

yielding

$$r_{\mathcal{XY}} = \frac{n\sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i}{\sqrt{n\sum_{i=1}^{n}x_i^2 - \left(\sum_{i=1}^{n}x_i\right)^2}\sqrt{n\sum_{i=1}^{n}y_i^2 - \left(\sum_{i=1}^{n}y_i\right)^2}} \tag{2.50}$$

which means that we can calculate $r_{\mathcal{XY}}$ for each block at a fix offset of two images from five summed-area tables at constant time. In order to find an extremum, it is sufficient to estimate $r^2_{\mathcal{XY}}$ and the sign of $r_{\mathcal{XY}}$. Then, the calculation of the two square roots can be avoided

$$r^2_{\mathcal{XY}} = \frac{a^2}{\left(n\sum_{i=1}^{n}x_i^2 - \left(\sum_{i=1}^{n}x_i\right)^2\right)\left(n\sum_{i=1}^{n}y_i^2 - \left(\sum_{i=1}^{n}y_i\right)^2\right)} \tag{2.51}$$



with

$$a = n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \quad \text{and} \quad \operatorname{sgn}(r_{\mathcal{XY}}) = \operatorname{sgn}(a) \tag{2.52}$$

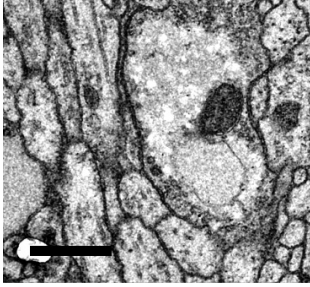
In order to test another offset, only the first sum in Equation (2.42) needs to be updated. Lewis (1995) suggested to perform this convolution operation in the Fourier domain for exhaustive matching of a single template against an image. That has the implication that each block has to be processed individually. For a dense block matching application with relatively small search radius as described here, it is more efficient to implement an outer loop over all possible offsets (e. g. 61×61 for a search radius of 300 px at a scale factor of 0.1) in the spatial domain. At each iteration the NCC coefficient for each pixel or selected blocks is calculated. If the calculated value is larger than the previously found maximum (per pixel or block), then this maximum and the corresponding offset are stored.

2.7 Results

I have applied the elastic alignment method to two outstanding ssTEM datasets (Table 2.1) and to an array tomography section series imaged by light microscopy. Sample preparation, sectioning and imaging have been performed by collaborators. The following protocols are excerpts from corresponding publications and personal communications and added here for better understanding of particular properties of each data set.

The *C. elegans* dataset shows a three-fold stage embryo, the latest embryonic stage during *C. elegans* embryogenesis immediately before hatching. It was generated in 2003 by Richard Fetter in the laboratory of Cori Bargmann. Timed embryo collections were prepared by *high-pressure freezing* (HPF) and *freeze-substitution* (FS). Substitution was performed in 1 % OsO_4 with 0.2 % uranyl acetate in 98 % acetone and 2 % methanol for 72 h at -90°C. Samples were then warmed to room temperature and embedded in Eponate 12 resin (Ted Pella, Inc.). 50 nm serial sections were cut using an ultra-microtome with a diamond knife and imaged on a JEOL 1200EX TEM using Kodak 4489 film (3.25″×4″). Negatives were digitized with an Epson flat bed scanner at 600 dpi resulting in digital images of 6,160×4,640 pixels each representing one section at a resolution of 4 nm per pixel (Figure 2.11a). The entire series consists of 803 section images (protocol by Chuang et al., 2007, and Richard Fetter, personal communication).

The *Drosophila melanogaster* dataset shows ~1.5 segments of the ventral nerve cord of a first instar larva (Ito et al., 1995, Figure 2.12), the earliest larval stage during *Drosophila* development, immediately after hatching. It was generated by Richard Fetter and Albert Cardona at the Janelia Farm Research Campus. Samples were dissected by removing the posterior end



	C. elegans three-fold embryo	*D. melanogaster* 1st instar larval VNC segment
TEM recording mode	scanned from film	digital imaging of overlapping tiles
Number of sections	803	458
Image tiles per section	1	> 70
Tile size	$6,160 \times 4,640$ px	$2,048 \times 2,048$ px
Bits per pixel	8	16
Section canvas size	$6,160 \times 4,640$ px	$\approx 22,000 \times 17,000$ px
Lateral resolution	4 nm/pixel	4 nm/pixel
Section thickness	50 nm	45 nm
Missing sections	25*	$2 + 5$
Total number of images	803	33,051 (with content) 77,017 total**
Size in GB	21	258 (with content) 602 total**
Processing time	$\approx$ 12 hours	$\approx$ two weeks (one week for elastic alignment)

Table 2.1: Overview of reconstructed ssTEM datasets. *The record of the series declares missing sections at 9 different places, ranging from 1 up to 7 sections missing. Without knowing that in prior, we were able to identify the four larger gaps and their approximate sizes by analyzing the ratio of block matches that passed local and global filters (Figure 2.9). **Many tiles imaged on the periphery of the specimen contain only background. The feature-based method as described in Section 2.5 identifies such tiles as being disconnected from the montage and automatically removes them.

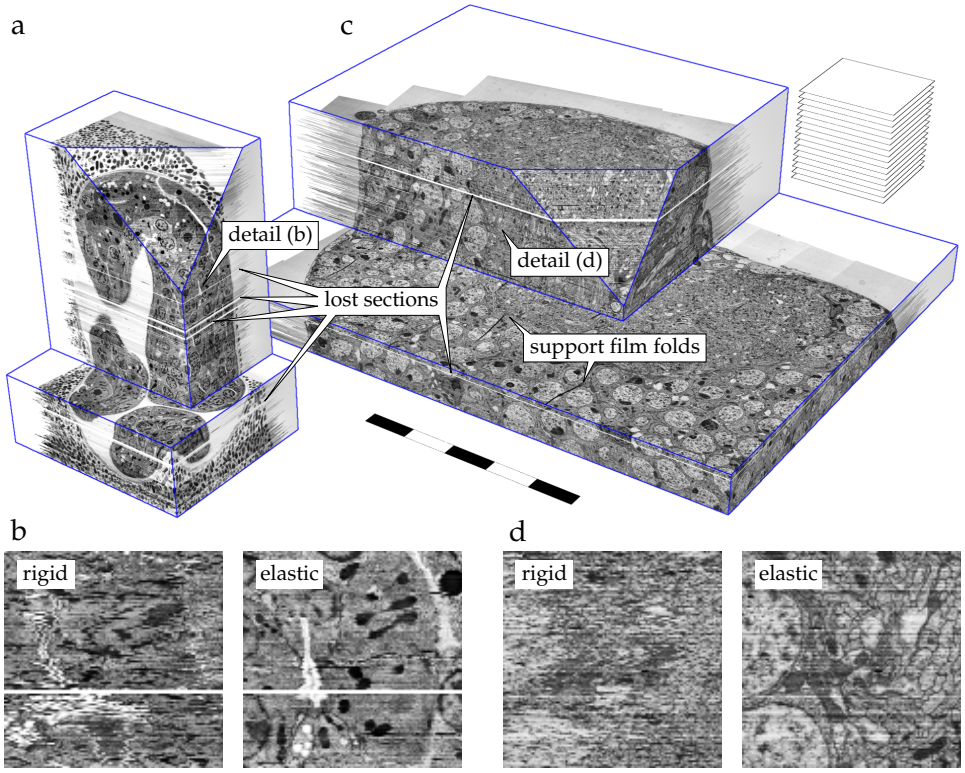


Figure 2.11: Reconstruction of two large TEM section series. Sections were scanned from film negatives (a,b) or assembled from many overlapping digital camera images (c,d) using the elastic alignment method in montaging mode. Parts of the reconstructed volumes are shown as arbitrarily sliced 3d renderings (a,c). The planar resolution (scale bar 25 µm = 6, 250 px at ~4 nm per pixel) is ~10× higher than the axial resolution (40–50 nm per section). The orientation of the section series is orthogonal to the horizontal plane (see staple). Specimens shown are a threefold *C. elegans* embryo (803 sections; a,b) and 1.5 segments of the ventral nerve cord of a first-instar *Drosophila melanogaster* larva (458 sections, each section consists of ~70 overlapping image tiles; c,d). Details of orthogonal cross-sections at isotropic resolution (b,d) demonstrate how well the elastic alignment method recovers the continuity of the specimen from significantly distorted section series compared to a pure rigid alignment.

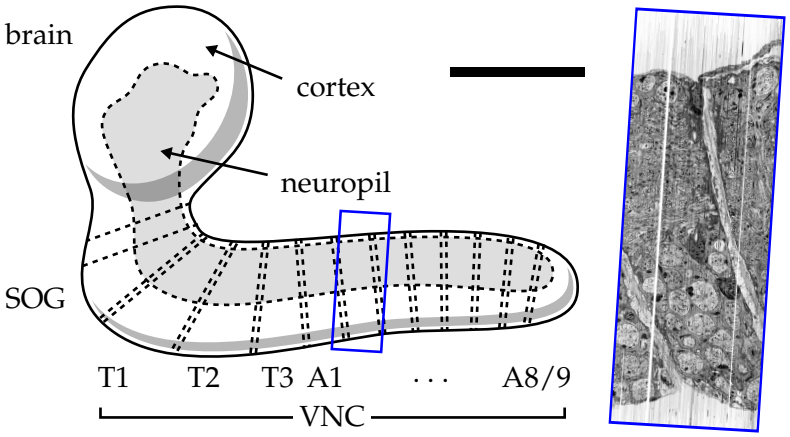
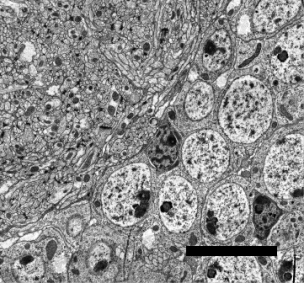


Figure 2.12: Anatomical context of the *Drosophila* larval ventral nerve cord TEM section series. Cartoon based on a figure from (Ito et al., 1995). The blue box marks the approximate area from which the series of sections originates (note that the exact location of the series within abdominal segments is not known). The inset shows the sagital plane at the midline through the elastically aligned volume. VNC – ventral nerve cord, SOG – subesophageal ganglion, T1–3 – thoracic segments, A1–9 – abdominal segments.

of first instar larvae by traction. They were fixed in 1 % glutaraldehyde in 0.1 M Na-cacodylate buffer, pH 7.4 for 1 h at room temperature on a tissue rotator, rinsed with buffer, post-fixed with 1 % OsO_4 in 0.1 M Na-cacodylate buffer for 1 h at room temperature, rinsed with distilled H_20 and stained *en bloc* with 1 % aqueous uranyl acetate overnight at 4°C. Samples were then dehydrated in an ethanol/propylene oxide series, embedded in Eponate 12 resin and polymerized in a 60°C oven for 48 h. Serial 50 nm sections were cut using a Diatome diamond knife and a Leica UC6 ultramicrotome, and picked up on Pioloform coated Synaptek slot grids stabilized with 2 nm carbon. The sections were stained with uranyl acetate and Sato's lead (Sato, 1968; Hanaichi et al., 1986). Sections were imaged with an FEI Tecnai T20 TEM using a Gatan 895 4,096×4,096 pixels digital camera at 4.4 nm/px resolution (2.2 nm/px, bin = 2) as a regular grid of 10 % overlapping image tiles using Leginon (Suloway et al., 2005) for automatic image acquisition (protocol by Richard Fetter and Albert Cardona, personal communication). Typically about 150 tiles were recorded per section covering a canvas of about 22,000×17,000 pixels. More than half of the acquired tiles, mostly on the periphery of the ventral nerve cord, lacked any image content and were automatically removed during the alignment procedure.

The light microscopy dataset was generated using array tomography (Micheva and Smith, 2007) by Forrest Collman, Nick Weiler, Kristina Micheva and Stephen Smith. It shows a fragment from a barrel cortex of an adult



Line H YFP mouse (Feng et al., 2000) expressing YFP in a subset of layer 5b pyramidal cells. The brain was removed from the anesthetized animal and placed in Hank's Balanced Salt Solution (HBSS) at 4°C. The region of interest was dissected and fixed in 4 % paraformaldehyde and 2.5 % sucrose in phosphate-buffered saline (PBS) using rapid microwave irradiation and ColdSpot at 15°C, left in the fixative for 30 min at room temperature, rinsed in 3.5 % sucrose in PBS, quenched in 50 mM glycine in PBS, dehydrated in graded ethanol series (up to 95 %), and embedded in LRWhite resin. 70 nm serial sections were cut with a Diatome diamond knife and an ultramicrotome and mounted on glass slides where they were repeatedly eluted and stained with different antibodies and imaged with a fluorescence microscope (protocol by Micheva and Smith, 2007). The exemplary series that was provided to me comprises 43 serial sections at 100 nm/px resolution.

During serial sectioning, inevitably, some sections are lost. For the *C. elegans* series this happened in nine different places and for the *Drosophila* series it happened twice.

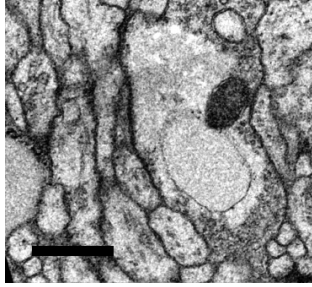
I aligned all three data sets using the presented method on a standard desktop computer with a quad-core Intel® Xeon® CPU with 2.67 GHz and 24 GB of memory, Ubuntu Linux 10.04 LTS installed.

The *C. elegans* series was pre-aligned rigidly using SIFT landmark correspondences. Then, elastic alignment was performed exploring a neighborhood of up to 6 sections for each section. Figure 2.11c and Videos 4–7 show the dramatic improvement of the alignment after the elastic method was applied both in terms of overall specimen outer shape and the internal structure. The reconstruction of the *C. elegans* series was computed in approximately 12 hours and the result can be interactively viewed at various scales in CATMAID[5] (Saalfeld et al., 2009). The nearly uninterrupted continuity of the series through the complete organism will enable unambiguous segmentation of nuclei and anatomical ultra-structure with unprecedented resolution which will make it a perfect reference for segmentation of light microscopy data (Long et al., 2009).

For the array tomography series, I have used the same protocol as for the *C. elegans* series. Elastic alignment was performed after SIFT-based rigid pre-alignment (Videos 12 and 13). In a 3d rendering (Video 14), it becomes apparent that given the ultra-thin sections and the limits of resolution imposed on light-microscopy, in this dataset, the axial resolution (70 nm/px) is better then the lateral one.

The *Drosophila* series was imaged as a series of mosaics of overlapping tiles. Unfortunately, no lens-calibration montage was generated such that the combination of heat-induced deformation during image acquisition and uncompensated lens-distortion resulted in displacements of up to 50 pixels when only a rigid transformation was used to stitch the mon-

[5] http://fly. mpi-cbg.de/c-elegans



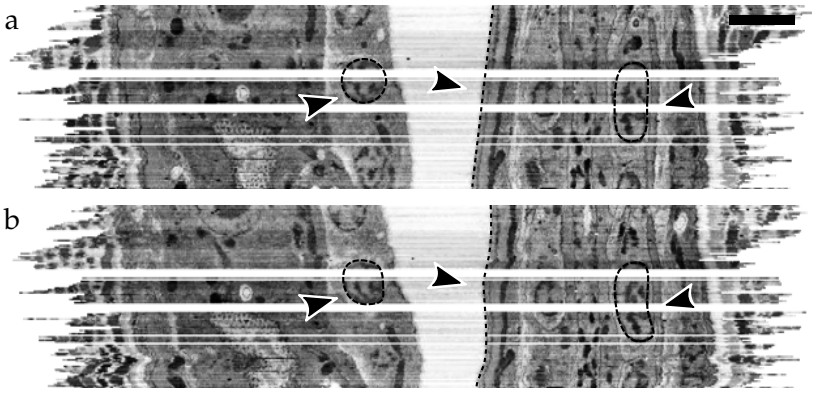
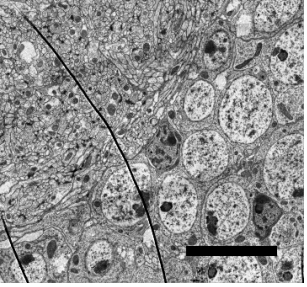


Figure 2.13: The effect of aligning larger neighborhoods shown by comparing elastic alignment exploring eight adjacent sections or only direct neighbors. An axial virtual slice of a subset of sections *C. elegans* series (two parts of the same folded embryo are visible) that were aligned elastically connecting each section either to a local neighborhood of 8 sections (a) or only to its directly adjacent sections (b). Alignment using a larger local neighborhood (a) outperforms alignment where only direct neighbors were used (b); the nuclei adopt smooth oval shape even in the presence of missing sections (white bands across) and vertical features are continuous and straight (arrowheads). The apparently jagged edges on each side are the result of each section image covering a slightly different field of view at arbitrary orientation. Consequently, sections overlap only partially in the aligned series. Scale bar 2 µm = 500 px lateral, 50 px axial.

tages (Figure 2.10). It was possible to remove these stitching problems in the overlapping regions between tiles. However, it is expected that lens-induced deformation will have a compromising effect to series alignment. As discussed in Section 2.6.2, elastic alignment becomes more robust and efficient when initialized in proximity to the desired solution. The search radius for block matching is then smaller which means that it can be explored quicker and the likelihood to find spurious matches is greatly reduced. The according alignment protocol consists of a series of steps, each building upon previous results as initialization:

1. The series of montages was registered using the global tile-wise method based on SIFT-features described in Section 2.5.4. For each individual tile, a rigid body transformation was estimated that globally minimized the sum of square landmark correspondence residuals. The resulting center coordinates of each tile were stored. Out of 77,017 images, 33,499 images were automatically recognized as non-empty connected content and aligned, all other tiles were excluded from the dataset. I have later manually removed 448 tiles that were not show-



ing parts of the specimen but were connected to section montages through texture from artifacts and background, resulting in 33,051 images all contributing to the reconstruction of the specimen.

2. Then, all section mosaics were montaged individually, not connecting tiles across sections. Again the global landmark-based method and a rigid body transformation for each individual tile were used. The resulting montages have much smaller remaining residuals than the global configuration calculated in step 1, because the non-linear section-to-section distortion is excluded.

3. These montages were then elastically montaged, compensating for both lens-distortion and heat-induced non-linear tile-to-tile deformation of the section. The result were seamlessly stitched montages.

4. The series of seamlessly stitched montages was then warped such that the center points of each tile were at the location as stored in step 1 (see Section 2.5.7). This warped series was a very good initialization for subsequent elastic alignment.

5. Finally, the series of warped sections was aligned elastically, exploring a search radius of 400 px per block in a neighborhood of up to 8 sections.

The execution time for the landmark based methods was in total approximately one week. The elastic alignment steps were done in less than a week. Overall, execution time was dominated by rendering the transformed images at multiple intermediate steps.

Similarly to the *C. elegans* dataset, I observed dramatic improvement of the alignment over the landmark-based attempts after the elastic method was applied both in terms of the ventral nerve cord's outer shape and the internal structure down to the resolution sufficient for distinguishing individual synapses in the axial direction (Figure 2.18 and Videos 8–11). I expect the result to improve further in future project when we can explicitly compensate for lens-distortion.

2.8 Evaluation

To further substantiate the benefits of the elastic alignment method for recovering the biological shape of the imaged specimen, I have evaluated the quality of my elastic serial section alignment method using manually generated annotation and artificial ground truth. I compare rigid, affine, tilewise rigid, and elastic alignment, demonstrating that the elastic method clearly outperforms all other methods in the quality of the reconstruction.

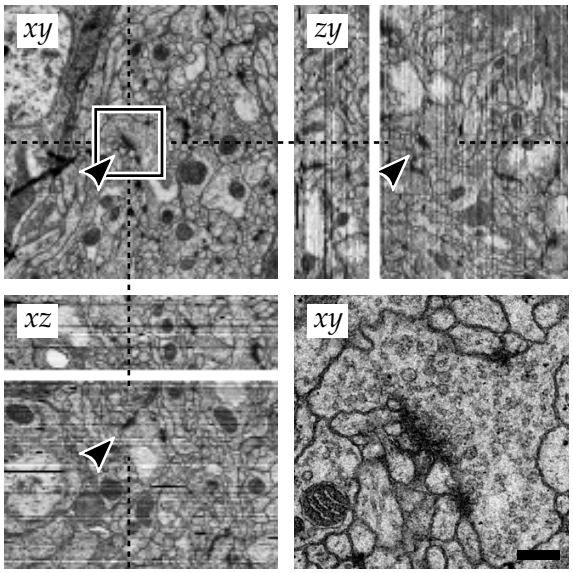


Figure 2.14: Synapses are detectable in both lateral and axial views after elastic alignment. A synapse (arrowheads) is shown in lateral (xy) and two axial (zy, xz) views of the elastically aligned *Drosophila* series. The top left lateral and both axial views were generated from the section series scaled down to isotropic resolution. The area highlighted by a black square is shown at the original lateral resolution of 4 nm/px (xy, bottom right). While single vesicles can be distinguished only in the high resolution image of the section, it is possible to identify synapses by their increased presynaptic density at axial resolution. The white bands visible in the axial panels are missing sections. The dashed lines indicate the locations of the respective axial sections. Scale bar 200 nm = 50 px.

2.8.1 Manual Skeleton Traces

Albert Cardona has traced the skeletons of several individual neurons from their cell bodies to the neuropil where they branch and engage in synaptic connections. Tracing was performed manually, using the TrakEM2 software (Cardona et al., 2010, 2012), on a previous version of the dataset that had been aligned using manually corrected sequential affine transformations. After rigid and elastic registration, respectively, the manual traces were computationally transferred into the re-aligned dataset and visualized (Figure 2.15a–d). While rigid alignment suffers from characteristic jitter of traced neuronal profiles, the elastically aligned dataset is smooth, much better reflecting shape details of the biological tissue (Figure 2.15 insets).

Jitter as introduced by insufficient alignment contributes significantly to the total length of skeleton traces. I therefore report the scale-normalized

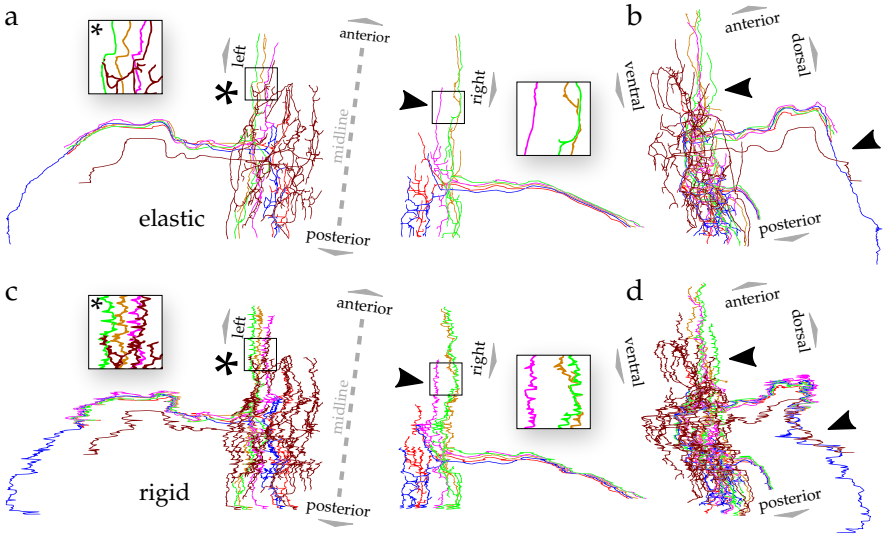
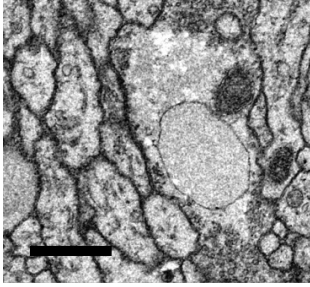


Figure 2.15: The quality of serial section alignment impacts reconstructed shapes. Manually created skeleton traces of exemplary neuronal arbors are compared for elastic (a,b) and rigid (c,d) series alignment. Traces are shown in two perspective projections: dorsal view (a,c) and lateral view from left to right (b,d). The section plane is orthogonal to the projection plane. Therefore, longitudinal branches expose jitter where alignment insufficiently compensates for low scale distortion (c,d; arrows and inset). This jitter is to a large extent removed by elastic alignment (a,b; arrows and inset). * marks a noticeable misalignment due to a gap of 5 lost sections (inset). Note that in the rigidly aligned series (c,d), the spot cannot be distinguished from general jitter.

total length l of the skeleton traces shown in Figure 2.15 as an alignment quality criterion (Figure 2.16). The total length l is the sum of all edge lengths $|\mathbf{p} - \mathbf{q}|$ normalized by a local scale factor s

$$l = \sum_{\forall(\mathbf{p},\mathbf{q})} \sqrt{\frac{(x_\mathbf{p} - x_\mathbf{q})^2 + (y_\mathbf{p} - y_\mathbf{q})^2}{s^2} + (z_\mathbf{p} - z_\mathbf{q})^2}. \tag{2.53}$$

The local scale factor s is the average scale factor of the contributing sections. The scale factor of a section is the average scale factor of all image tiles in the section and the scale factor of an image tile is estimated through a least-squares approximation of its non-linear elastic transformation by a similarity transformation (scale, rotation, translation). Shorter total length l implies improved alignment. Scale normalization makes the length measure invariant to global scaling. Without scale normalization, globally reducing the size of all (or a range of) sections would reduce the total length, rendering it useless as a criterion for alignment quality.



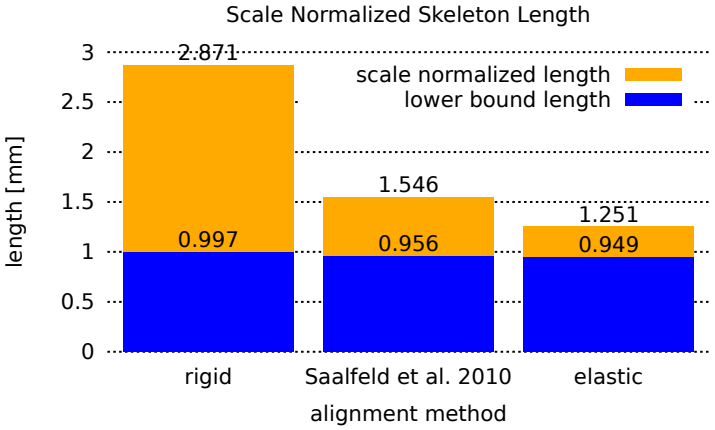
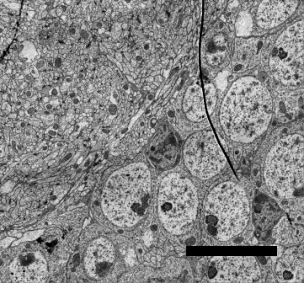


Figure 2.16: Comparison of three alignment methods applied to the *Drosophila* series using the total length of manual skeleton annotation. The first method is a sequential alignment using a rigid transformation per section estimated from automatically extracted features (rigid). The second method estimates a rigid transformation per each mosaic tile globally minimizing the total square displacement of automatically extracted feature correspondences (Section 2.5.1, Saalfeld et al., 2010). The third is our new elastic method initialized with the result of the second method. We report the total length of manually traced neuronal arbor skeletons with all edge lengths normalized by a local scale factor s (Equation (2.53)). The local scale factor s for each skeleton edge $(\mathbf{p}, \mathbf{q})$ is the average scale factor of the contributing sections at $\mathbf{p}$ and $\mathbf{q}$ relative to the original image size. We compare the total skeleton length with an approximate lower bound skeleton length calculated from all edges between branch and end nodes of the skeleton replaced by straight lines (Figure 2.22).

Elastic alignment reduces the total skeleton length of the neuronal arbors shown in Figure 2.15 from 2.87 mm in the rigidly aligned series and 1.55 mm using the tile-wise rigid method to 1.25 mm. This is a decrease of 56.4 % compared with a rigid series alignment and of 19.1 % compared with the tile-wise rigid method.

I compared the scale-normalized skeleton length l with a lower bound length f as suggested by Albert Cardona as an approximation of the minimal expected 'real' length of a traced skeleton. The lower bound length f is the length of the skeleton after all edges between branch- and end-points have been replaced by straight lines (Figure 2.17). Since now only branch and end points suffer from alignment errors, the lower bound length f is robust with respect to insufficient section-to-section alignment. This robustness is reflected in the observation that elastic alignment decreases f by only 4.8 % compared with a rigid series alignment and 0.7 % compared



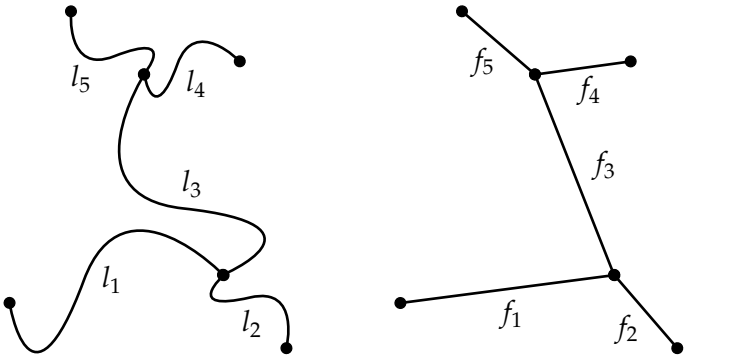


Figure 2.17: We compared the total skeleton length l with a lower bound skeleton length f that serves as an approximation of the minimal possible skeleton length and is robust with respect to alignment errors. The lower bound length f is the sum of the length of all edges between branch and end points of the skeleton replaced by straight lines.

with the tile-wise rigid method. On the other hand, the percental difference between l and f serves as an indicator for overall alignment quality. It is reduced to 31.8 % by elastic alignment, compared to 188.0 % for the rigid series alignment and 61.7 % for the tile-wise rigid method demonstrating the superior alignment results achieved with the elastic method. It is important to note that jitter in the manually generated skeleton-traces comes not solely from insufficient alignment but as well from inaccurate manual operation. This is particularly relevant for the annotations used here since they were performed on poorly aligned data and annotation speed had a higher priority than accurate localization of each profile's center point. My prediction is therefore that these skeleton traces cannot be used to report qualitative improvement over the current series alignment because the manual error already outweighs the alignment error.

2.8.2 Artificially Generated Ground Truth

I have quantitatively evaluated the accuracy of the proposed alignment methods using artificially generated ground truth. Using the Open Source Raytracer POV-Ray,[6] I have generated a synthetic volume that has the shape of a distorted ball filed with volumetric texture resembling membranes and blob-like structures as present in biological tissue. I have artificially sectioned the volume at a section thickness of 2 px and generated two series of

[6] http://www.povray.org

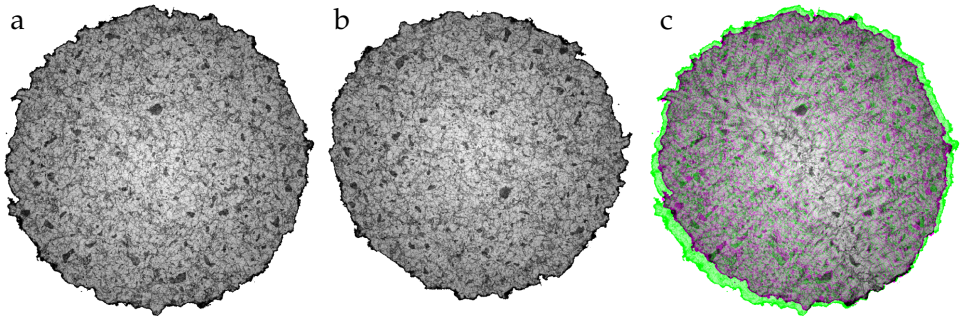
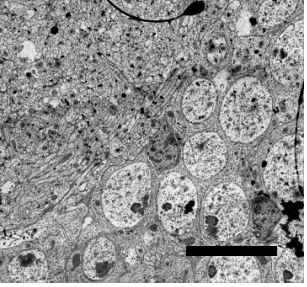


Figure 2.18: An exemplary section of the artificial evaluation series. The original section (a) was deformed non-linearly, rotated and shifted. Aligned by a rigid body transformation and displayed as a green-magenta color-merge, the deformation becomes apparent (c).

400 sections, each $2,000 \times 2,000$ px. Evaluation series A repeats the same section 400 times (Video 2). In this series, texture displacement is the exclusive result of deformation since no 'biological' signal changes alongside the z-axis. Evaluation series B consists of 400 serial sections including the signal change induced by the volume (Video 3) and as such is a more realistic test case. I have artificially distorted all sections of both series using randomized smooth non-linear transformations using a moving least-squares affine transformation (Schaefer et al., 2006) for four control points at random source locations in either of the four quadrants of the image displaced by a maximum distance of 50 px. That induced section-to-section pairwise local deformation of up to 200 px relative to a rigid least-squares approximation. Each section was then rotated by a random angle and shifted in a random direction by up to 150 px. Both evaluation series were aligned using a rigid transformation per section, a globally regularized affine transformation per section and the elastic alignment method.

I report the average scale factor of each section relative to ground truth for all three alignment methods (Figure 2.19). Rigid series alignment per definition preserves the average section scale that has been introduced by non-linear deformation. Both regularized affine and elastic alignment recover the original scale of all sections across the entire series. The elastic method performs better as it can compensate for non-linear deformation.

I then compare alignment precision using a sample of straight lines projected along the z-axis through the ground-truth series. These lines, if perfectly reconstructed should again be straight lines along the z-axis. Only points covered by the 'specimen' are considered because background is not expected to and does not need to be aligned. For all lines, I report the absolute displacement in the x, y-plane relative to ground truth in each z-section



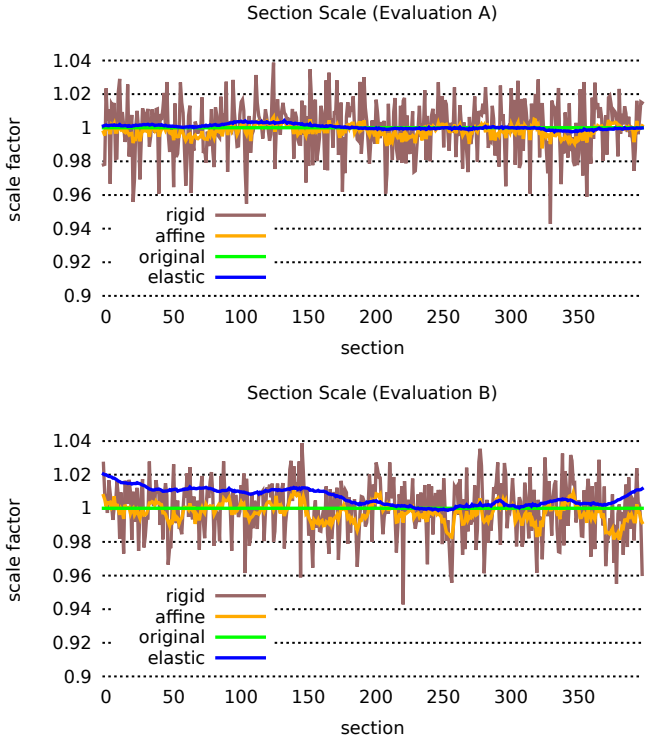
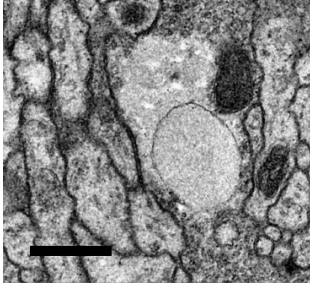


Figure 2.19: The average scale factor of a section relative to the non-deformed ground-truth is estimated through a least-squares approximation of the artificially introduced non-linear deformation by a similarity transformation (scale, rotation, translation). Since non-linear deformation has been introduced to all sections independently, no systematic scale change is expected across the series. All three methods preserve this property (Videos 2 and 3). Elastic alignment almost perfectly recovers the original scale of all sections. As expected, the performance is better for series A where no signal change compromises the measured motion vectors.

(Figures 2.20 to 2.22). Ground truth and reconstruction result were previously aligned by a 2d-rigid transformation to compensate for a global rotation and translational offset. Elastic alignment clearly outperforms rigid and regularized affine alignment in its ability to recover the original shape of the 'specimen' while at the same time effectively removing section to section jitter.



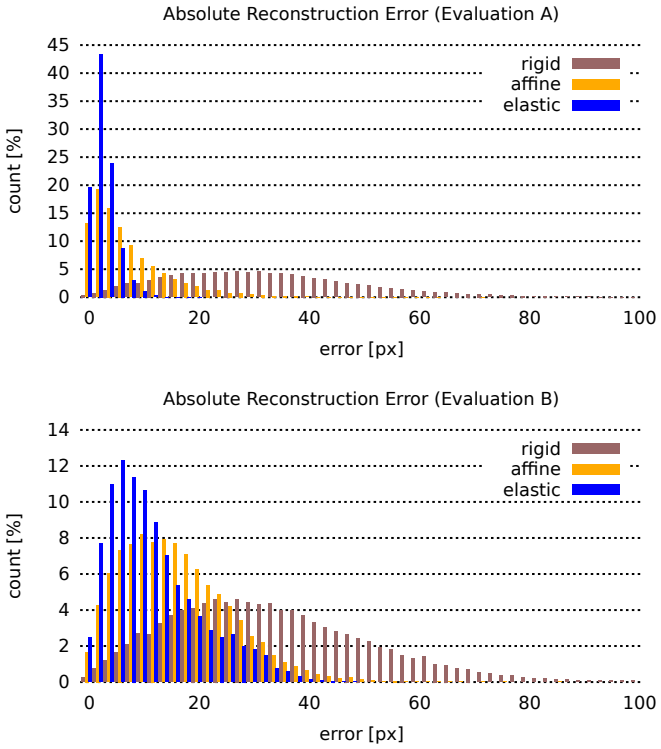
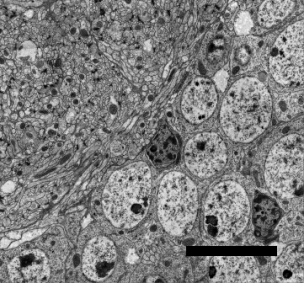


Figure 2.20: Histograms of absolute point displacements after alignment of evaluation series A and B. We have measured the absolute displacement in the x, y-plane relative to ground truth of each z-point of a sample of straight lines projected through the volume along the z-axis. Elastic alignment outperforms both rigid and affine alignment. As expected, the result is slightly better in evaluation series A where no signal change compromises the measured motion vectors (left graph).



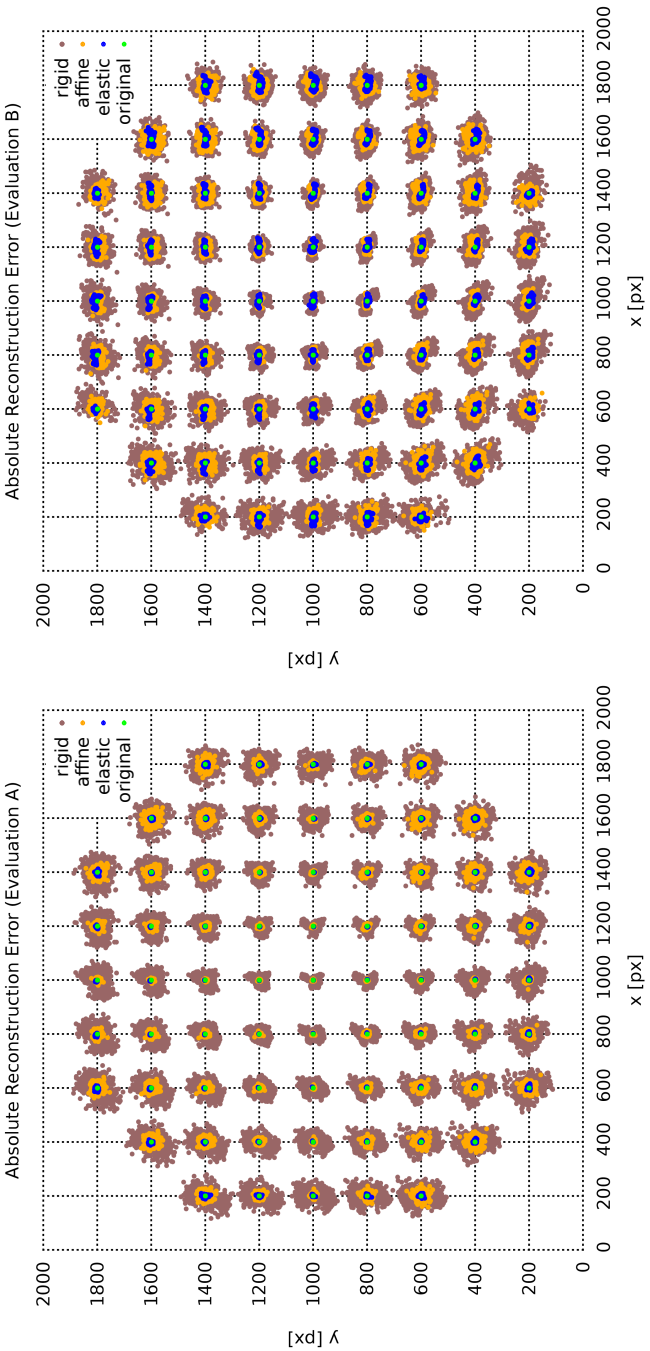
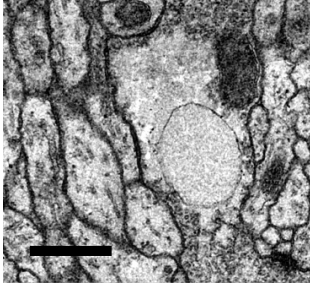


Figure 2.21: Projection of the x, y-location of each z-point of a sample of straight lines projected through the volume along the z-axis. Elastic alignment very well recovers the original location of all points along the lines. As expected, the result is slightly better in evaluation series A where no signal change compromises the measured motion vectors (left graph). The displacement is minimal in the center of the volume and increases towards the periphery for all methods used. This effect is comparatively minimal in case of the elastic alignment.



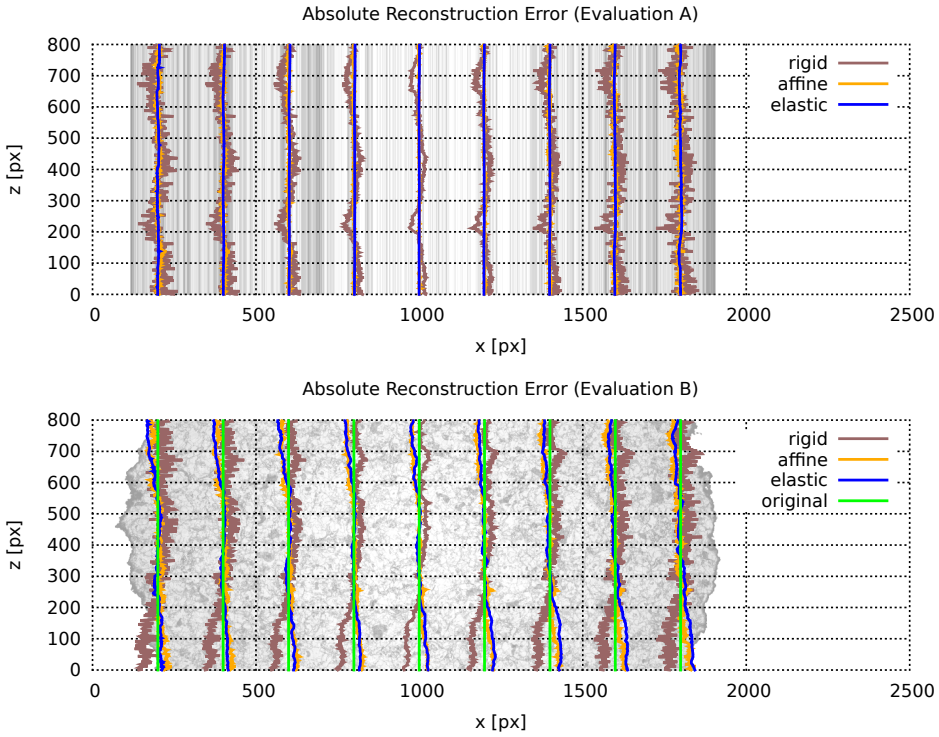
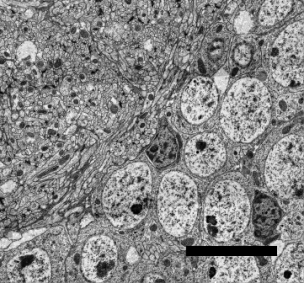


Figure 2.22: Projection of the x, z-location of each z-point of a sample of straight lines projected through the volume along the z-axis. Elastic alignment very well recovers the original location of all points along the lines. As expected, the result is slightly better in evaluation series A where no signal change compromises the measured motion vectors (top graph). The displacement is minimal in the center of the volume and increases towards the periphery for all methods used. This effect is comparatively minimal in case of the elastic alignment. The background of the plot shows the corresponding x, z-section of the respective original evaluation series (note that in the top graph the texture of the volume does not change along the z-axis as all sections are the same). We have deliberately omitted the original lines (green) in the top graph as they almost perfectly occlude the elastic alignment result.

Finally, I report section-to-section pairwise alignment using again the reconstructed lines (Figures 2.23 and 2.24). Since the lines were originally projected along the z-axis, any x, y-displacement at two adjacent z-coordinates constitutes an error in pairwise alignment. Affine series alignment results in lower pairwise alignment error than rigid series alignment. Elastic alignment outperforms both methods significantly.



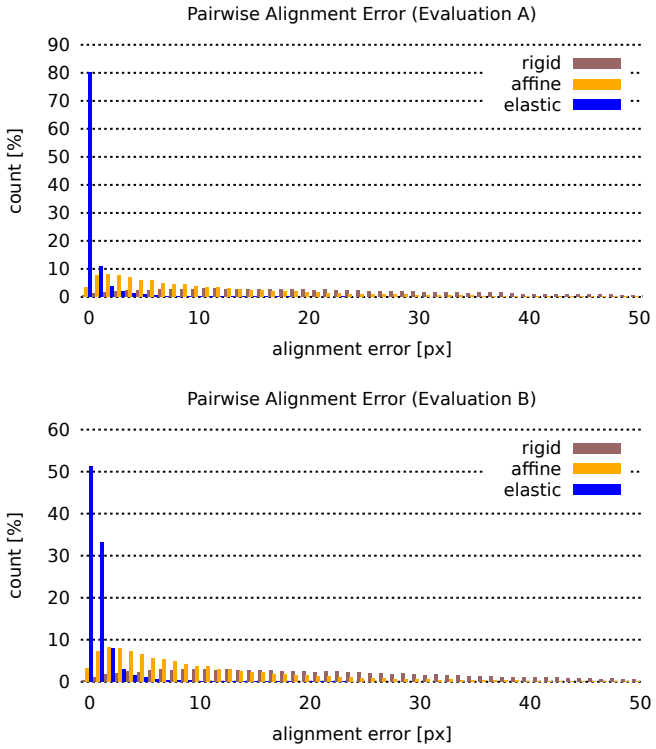


Figure 2.23: Histograms of section-to-section pairwise point displacements after alignment of evaluation series A and B. We have measured the pairwise displacement in the x, y-plane between adjacent sections for a sample of straight lines projected through the volume along the z-axis. Elastic alignment outperforms both rigid and affine alignment significantly. As expected, the result is better in evaluation series A where no signal change compromises the measured motion vectors (top graph). Note that in evaluation series A for 80 % of all sampled locations the pairwise displacement is below 1 pixel and for 90 % below 2 pixels. In evaluation series B for 50 % of all sampled locations the pairwise displacement is below 1 pixel and for 80 % it is below 2 pixels.

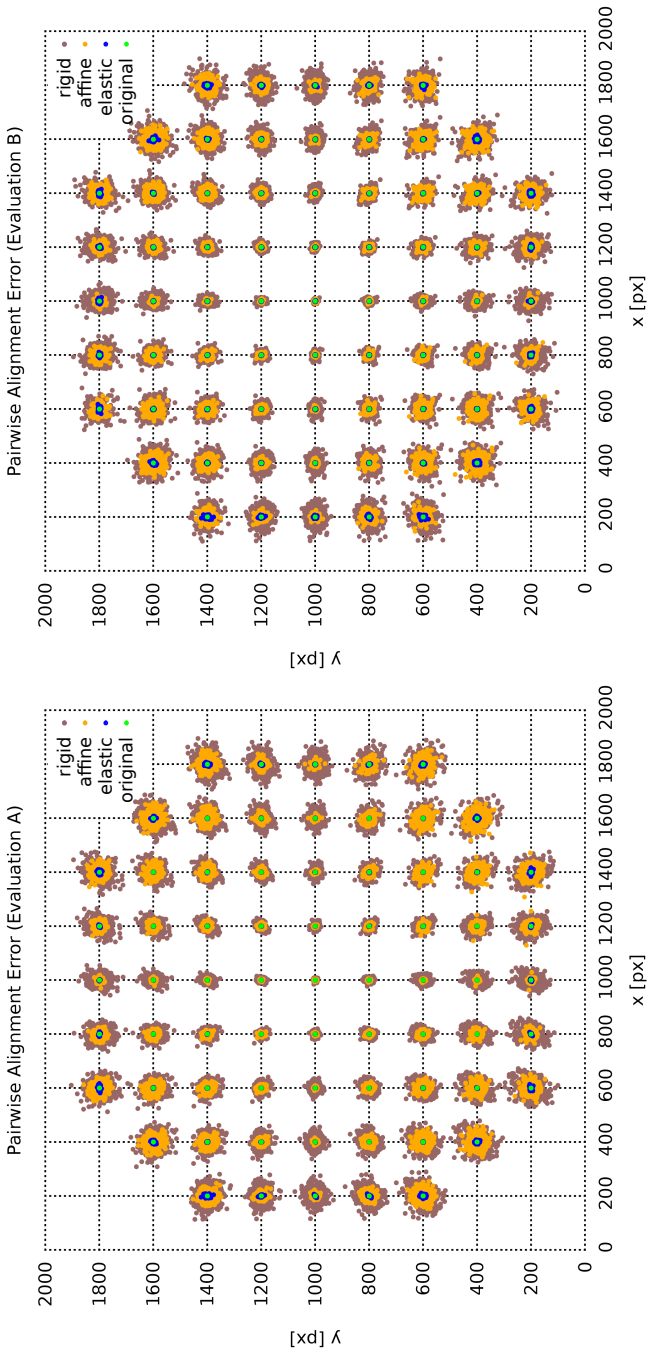
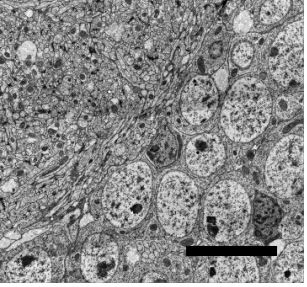


Figure 2.24: Visualization of section-to-section pairwise point displacements after alignment of evaluation series A and B. The pairwise displacement for each z-point of a sample of straight lines projected through the volume along the z-axis are displayed centered at the original x, y-location of the corresponding line. Elastic alignment outperforms both rigid and affine alignment significantly. As expected, the result is better in evaluation series A where no signal change compromises the measured motion vectors (left graph). The pairwise displacement is minimal in the center of the volume and increases towards the periphery for all methods used. This effect is comparatively minimal in case of elastic alignment.



2.9 Local Contrast Enhancement

Digital cameras as used in todays electron microscopes typically capture pixel intensities with a resolution of 12 bits ($2^{12} = 4096$ gray levels) or 16 bits ($2^{16} = 65,536$ gray levels). With the noise present in electron micrographs, the 'biological signal', that is the texture showing intracellular structures stained with heavy metal atoms, is usually encoded at substantially lower resolution and can be interpreted sufficiently at 5 or 6 bits depth. It is thus possible to utilize the excess bit range to capture a much larger dynamic range than that covering the staining signal. As depicted in Figure 2.1, some electron micrographs in a large section series will suffer from a range of artifacts, the captured image integrates both artifacts and the staining signal. Where artifacts dominate the contrast of an image it is difficult to map the interesting signal into the narrow contrast range used for visual display. Interestingly, some artifacts such as support film folds, dirt, variance in section thickness or staining intensity are largely homogeneous or very smooth and thus enable the signal of interest to be recovered by filtering low frequencies in various ways. Since structures of interest are small compared with the global image size, frequencies with a wavelength larger than the objects of interest can be removed. By that, local contrast can be improved and the influence of artifacts reduced.

Local contrast enhancement has been demonstrated to improve the performance of human curators in image annotation (Pisano et al., 1998). I have implemented two methods for local contrast enhancement, the Contrast Limited Adaptive Histogram Equalization (CLAHE) by Pizer et al. (1987) and a contrast-stretching high-pass filter using block-filters.

CLAHE is an extension of *Adaptive Histogram Equalization* (AHE) which is an extension of *histogram equalization*. Histogram equalization works as follows: Given an image with n pixels and k gray-values (256 in an unsigned 8 bit image), the k-bin histogram $p_x(i)$ of the entire image is collected and normalized such that each entry represents the sampled probability of its corresponding gray-value

$$p_x(i) = \frac{n_i}{n} \qquad \text{for} \qquad 0 \leq i < k. \tag{2.54}$$

Now, the cumulative distribution function $P_x(i)$ is created by integrating the histogram

$$P_x(i) = \sum_{j=0}^{i} p_x(j). \tag{2.55}$$

To each pixel, the value given by $P_x(i)$ is assigned, scaling it back into the original scale

$$y(x) = P_x(x) * (k-1). \tag{2.56}$$

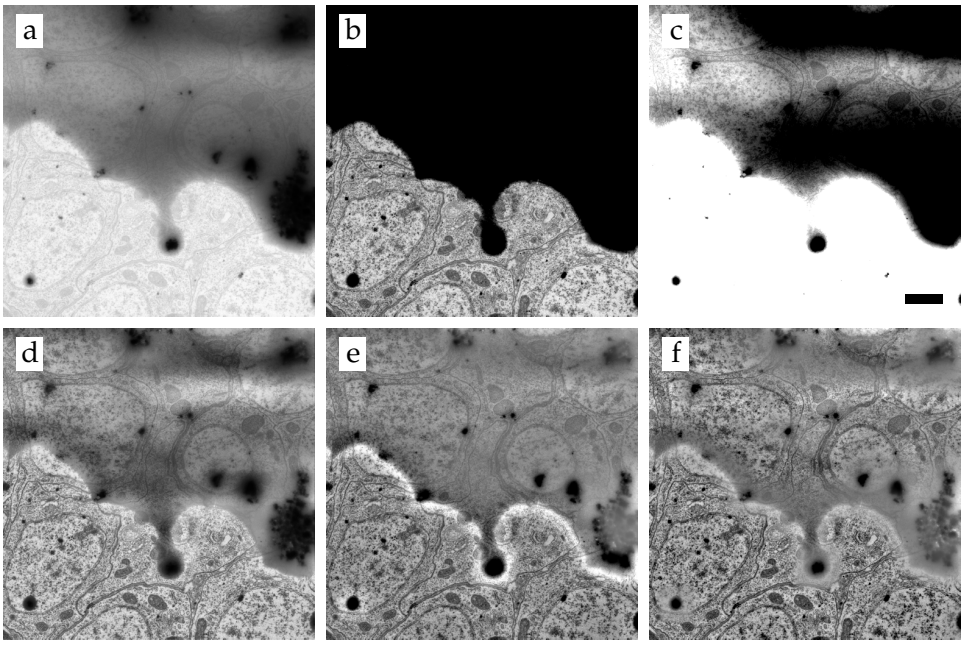
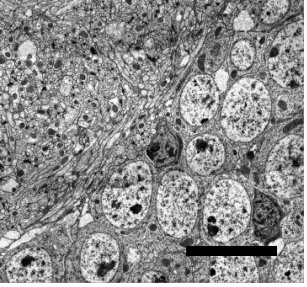


Figure 2.25: Comparison of three local contrast enhancement methods at an electron micrograph from the *Drosophila* series with a large area covered by strong staining precipitate. (a) shows the entire dynamic range captured by the camera. The staining signal covers only a small fraction of the range compared with the artifact. (b,c) show how globally narrowing the range will show the signal in particular regions of the image but make it invisible in others. (d) shows the image processed by fast CLAHE, (e) shows the image after filtering low frequencies, and (f) shows the image after filtering low frequencies and adjusting the dynamic range according to the local standard-deviation. (e,f) are realized using the integral block filter. (d,e) require that the desired contrast range is adjusted in prior which (f) does automatically. Scale bar 1 µm = 250 px.

In case that the original number of gray-values was larger than the number of histogram bins or in case that real values for the pixels are used, the desired pixel value can be transferred by linearly interpolating between the two adjacent entries in $P_x(i)$.

AHE performs the same operation with an independent histogram for each pixel that is collected not from the entire image but from a block of given size around the pixel location. Since collecting the histogram and calculating the $P_x(i)$ is a relatively expensive operation, it is undesired to evaluate it independently at each pixel. Pizer et al. (1987) suggested therefore to evaluate histograms only in non-overlapping blocks and linearly interpolate $P_x(i)$ for all pixels between. They found that this provides a



good approximation of the independently evaluated function while greatly decreasing runtime.

CLAHE adds to AHE the option to set an upper limit m for the slope of the transfer function. Since the transfer function is the integral of the histogram, this can be achieved by cropping the original histogram at a threshold of $m/(k-1)$. In order to preserve the mean intensity, the cropped area is equally distributed to all non-saturated histogram bins. With a maximum slope of $m = 1.0$, CLAHE re-generates the original image. Slope limits $m > 1.0$ lead to increasing histogram equalization. Using the approximation as described for AHE, CLAHE can be executed at quick enough speed for real-time high-resolution video processing on a modern desktop computer (see Figure 2.25d).

For high-pass filtering I employed again the integral image approach (Crow, 1984). A low-pass can be approximated with a mean-box-filter. Since the mean $\bar{x}$ of all pixels x_i in a box of pixels $\mathcal{X} = \{x_1, \ldots, x_n\}$ is

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{2.57}$$

the sum of a box of arbitrary size can be calculated with just three additions and one multiplication from an integral image. Subtracting the filtered image from the original image results in an approximate high-pass (see Figure 2.25e). In addition, I want to adjust the local contrast in each box such that it saturates the available contrast range (e. g. $[0, k-1]$). To that end, I calculate the mean $\bar{x}$ and the standard-deviation σ_x in each block and map the range $[\bar{x} - a\sigma_x, \bar{x} + a\sigma_x]$ into $[0, k-1]$ with a specifying the contrast range of interest with respect to σ_x. For each pixel the transferred value is then

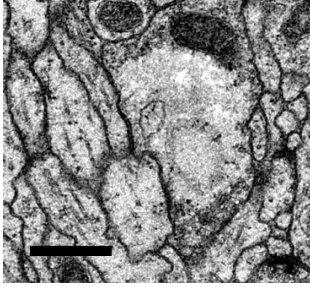
$$y = \frac{k-1}{2} \left(\frac{x - \bar{x}}{a\sigma_x} + 1 \right). \tag{2.58}$$

As already shown in Section 2.6.2, the standard deviation can be calculated using integral images because

$$\sigma_x = \sqrt{\frac{1}{n} \left(\sum_{i=1}^{n} x_i^2 - \left(\frac{1}{n} \sum_{i=1}^{n} x_i \right)^2 \right)} \tag{2.59}$$

where we have two independent sums that can be calculated using two integral images, one over x_i and one over x_i^2 (see Figure 2.25f).

Enhancing local contrast in tiles of ssTEM series can be beneficial in situations where artifacts dominate the interesting signal (see Figure 2.11 and Figure 2.25a). In the *Drosophila* series, I have applied CLAHE to four sections (128, 129, 149, and 152) where this was the case, resulting in much improved montages.



2.10 Matching and Registering Point Clouds

In collaboration with Stephan Preibisch, we found that the methods that I had developed for landmark-based registration of section series and montages (Section 2.5) could be applied to a related problem, registration of multi-view SPIM recordings. The collaboration lead to a new method for matching overlapping point clouds and an efficient solution for the registration problem. Preibisch performed the experiments and evaluation and described the method elaborately in his PhD thesis (Preibisch, 2010). I will therefore only briefly describe and discuss the major principles here and refer to his work for further reading.

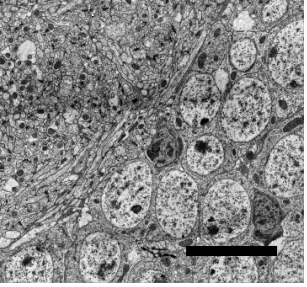
2.10.1 SPIM

Single Plane Illumination Microscopy (SPIM; Huisken et al., 2004) is an optical sectioning technique. Instead of sectioning the volume by narrowing the focal plane for both excitation and imaging in the axial direction (as in confocal microscopy; Minsky, 1957), in SPIM, a specially focused laser is used to illuminate only a very thin sheet of the specimen that can then be imaged using a normal wide-field setup with a digital camera. The detection axis is perpendicular to the light-sheet. The specimen is then moved through the light-sheet generating a 3d image by scanning a stack of 2d images. Thanks to no point or line scanning being involved, the method's major advantage is imaging speed which makes it particularly interesting for detailed studies of developmental processes in real-time (Huisken et al., 2004; Huisken and Stainier, 2009; Keller et al., 2008, 2010; Truong et al., 2011; Tomer et al., 2012; Krzic et al., 2012).

As a consequence of the illumination and detection axes being orthogonal, SPIM images suffer from signal quality degradation in both directions independently. For samples of suboptimal transparency such as e. g. *Drosophila* embryos, this effect prevents them from being imaged *in toto* at homogeneous quality using a single scan. The SPIM setup, however, enables the sample to be rotated such that 3d images from multiple directions (views) can be recorded, each suffering from signal degradation in a different direction. Ideally, these views should enable to reconstruct a 3d image that does not suffer from signal degradation in any direction.

2.10.2 Matching and Registration

The sample is embedded in a column of rigid agarose that can be moved and rotated in front of the lens. With this fragile setup, both the rotation axis and the angle of rotation are highly uncertain which requires the views to be registered based on their image content. Stephan Preibisch et al. (2008) examined the applicability of intensity-based registration methods used for



medical image registration (Preibisch et al., 2008; Preibisch, 2010). While their results were initially promising, it turned out that the approach did not deliver correct results reliably suffering from degradation artifacts in the images. In addition, the method has high computational demands, and it has no reporter that provides confidence whether or not registration was successful.

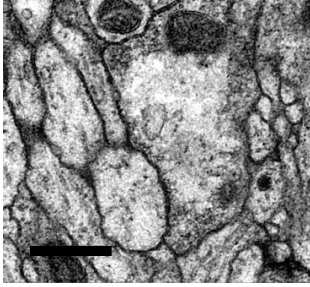
The solution for this problem was to focus on a sample of interest points in all views that could be detected and localized reliably. To this end, we embedded fluorescent beads in the agarose column such that the specimen was surrounded by a random cloud of fluorescent spots. Focusing on these beads instead of the specimen makes registration independent from signal degradation in the specimen.

Since beads are small bright spots in the image, they can be detected easily. We use the DoG detector (Lindeberg, 1998; Lowe, 2004) with approximate sub-pixel localization (Brown and Lowe, 2002) as described in Section 2.5.1, resulting in a point cloud with all beads and many other small spots in the specimen being detected. Bead detections are consistent in all views, the detections within the specimen, to a large extent, are not. The remaining question was thus how to match consistent bead detections across views and reject the spurious detections in the specimen.

We have achieved this by using the local configuration of the n nearest neighbors of each point as a local descriptor for the point. In the set of all detected interest points, the n nearest neighbors of each point are identified (using a kd-tree; Bentley, 1975), sorted by their Euclidean distance to the point, and stored. Matching is performed by comparing each of these n-point clouds extracted from one view to those extracted in another view. Since matching must be invariant with respect to an unknown rigid transformation, a well suited similarity measure is the sum of square residuals after the two point clouds have been registered using a 3d rigid body transformation. We calculate this rigid body transformation using Johannes Schindelin's open source implementation of the closed form least-squares solution by Horn (1987).

We found this matching approach to perform very well. False positives could be effectively rejected using a constant threshold for the residuals, Lowe's (2004) best to second-best ratio, and robust estimators as described in Section 2.5.2. The identified matches across all views are then used to calculate a 3d rigid body or affine transformation for each view that minimizes the global sum of square residuals. The problem is equivalent with that formulated in Equation (2.28) and solved using the same iterative optimizer (see Section 2.5.4).

Since matching point cloud descriptors is invariant with respect to a rigid body transformation, no prior information about the relative orientation of two overlapping views is required. Furthermore, using robust estimators including RANSAC to find consistent sets of 'inliers' in sets with



many 'outliers', it is possible to recognize whether or not two views overlap and to identify point matches despite that the overlapping region may be very small.

The approach is much more robust and by an order of magnitude faster than the earlier tested intensity based method. However, matches in two large point clouds are identified by brute force testing all-to-all local point cloud descriptors which is computationally expensive and by far not an instant operation.

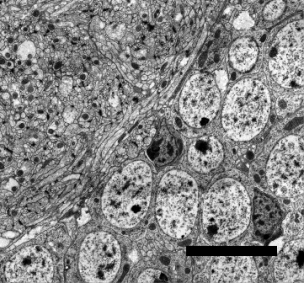
Preibisch has then developed a method inspired by *geometric hashing* (Lamdan and Wolfson, 1988) to avoid brute force comparison. The method works with 3d point cloud descriptors consisting of three nearest neighbors to one reference point. All local point cloud descriptors are mapped into a standard reference coordinate system. The mapping is the 3d rigid body transformation that moves the reference point to the origin of the coordinate system, the third-nearest neighbor onto its x-axes, and the second-nearest neighbor onto its x, y-plane, both in the positive half or quadrant respectively. In this coordinate system, the local point cloud descriptor is uniquely defined by a six-dimensional vector, since for the third-nearest neighbor, the y- and z-coordinate are zero and can be omitted, and for the second-nearest neighbor, the z-coordinate is zero and can be omitted. Matching is then implemented as nearest neighbor search in this six-dimensional descriptor space which can be performed efficiently using a kd-tree (Bentley, 1975).

Our approach for matching point clouds differs significantly from approaches developed for matching 3d shapes in overlapping surface scans (e. g. shape context Belongie et al., 2002; Frome et al., 2004). Shape matching is designed for a situation where distinctive shapes are sampled with a large number of points, however, a unique match for each individual point is not necessarily available. In our case, points are distributed randomly and do not form distinctive shapes at that level of observation. Other than in surface scans, points are expected to be detected reliably, such that a unique match for each point can be found.

We have applied the bead-based registration method to several large data sets from SPIM and *Confocal Laser Scanning Microscopy* (CLSM; see Preibisch et al., 2010a). The largest data set was a 3d time series covering the embryonic development of *Drosophila melanogaster* from gastrulation to the mature embryo imaged at 249 time points from five different angles (see Figure 1.3).

2.11 Summary

I have developed a method for volume restoration from series of ultra-thin microscopy sections. The method consists of two major components:



1. a landmark-based approach that estimates a rigid body or regularized affine transformation for each image using SIFT feature matches, globally minimizing the sum of square residuals, and

2. an elastic method that redundantly connects each section to a local range of sections by NCC-based block matching and then aligns the entire series with an elastic constraint implemented as a spring-connected particle system. Springs across and within sections serve concurrent purposes; while cross-section connections drag the vertex towards series alignment, springs in the triangle mesh within sections tend towards maintaining a rigid transformation of the sections and penalize distortion. Relaxing this system leads to a series of smoothly aligned sections with the required bending energy distributed equally among all sections.

Both components of the method avoid error propagation and artificial distortion by globally minimizing local non-rigid deformation. The two important alternative methods implementing such a global elastic constraint were proposed by Guest and Baldock (2001) and Wirtz et al. (2004); Schmitt et al. (2007). Both methods combine search for an elastic alignment and a pixel-based pairwise similarity estimate between adjacent sections in an iterative solution. Initial linear pre-alignment of the section series is achieved by variants of principal component analysis. My method differs from these approaches in four key areas:

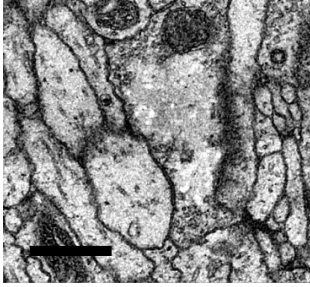
1. I compare and align not only adjacent sections but all sections in a local neighborhood (Figure 2.5 and Figure 2.13).

2. I use landmark correspondences from SIFT matches (Lowe, 2004) to calculate an initial approximate alignment (Saalfeld et al., 2010, Figure 2.7a,b). By that, my approach does not depend on a globally distinctive shape and can be applied to any kind of textured image data, even if it is partially occluded by artifacts.

3. I separate pairwise correspondence search from the elastic alignment, yielding an efficient solution for even very large data (Figure 2.7b).

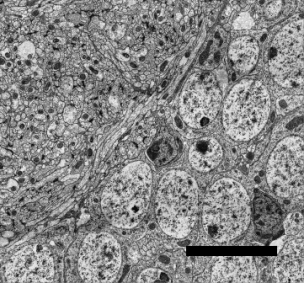
4. I have implemented a set of filters to robustly exclude staining artifacts and otherwise corrupted image regions from contributing to the alignment (Figure 2.7c,d and Figure 2.8).

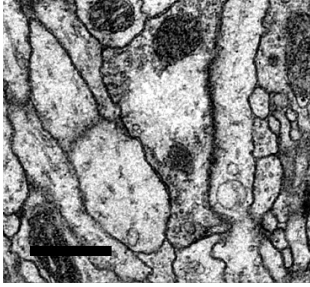
In order to evaluate my method quantitatively, I have generated synthetic volumes that mimic the properties of biological tissue, sectioned them and introduced artificial deformation to the sections. I have measured the alignment error using a sample of straight lines projected through the volume along the z-axis. The elastic method clearly outperforms sequential



rigid and affine alignment in its ability to recover the straight lines (Figures 2.19 to 2.24 and Videos 2 and 3).

Together with Preibisch, we have transferred the principles that I have developed for landmark-based registration of ssTEM mosaics to an application for registering 3d SPIM recordings based on randomly distributed fluorescent beads. The approach includes a new method to match local configurations of points in random point clouds and was successfully applied to register large 3d recordings from SPIM and CLSM.





Chapter 3

Implementation

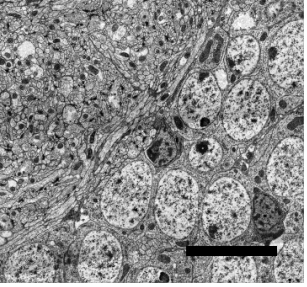
> I'm writing code. And I wrote a big library and I'm not gonna bore you with that 'cause it's highly equipped. But it does a huge number of things, it's like ImgLib [...]
>
> Gene Myers

A variety of software tools for processing and annotating bio-medical image data are publicly available. These tools provide processing and annotation capabilities beyond what can be achieved with commercial or free photo-editing software, and were developed for special demands arising in bio-medical research. While the following list is not complete, it comprises the most important tools that can be used or were explicitly designed for processing large serial section data sets imaged with electron microscopy:

IMOD (Kremer et al., 1996) includes tools for automatic montaging, automatic affine pairwise section registration, and manual adjustment. IMOD is the most popular tool set in the world of electron microscopy. It has a strong focus on the generation of high quality electron tomography. IMOD is open source software licensed under the terms of the General Public License.

Reconstruct (Fiala, 2005) focuses on microscopy section series. It includes tools for manual landmark based sequential section registration and has its main focus on manual annotation. Reconstruct is open source software licensed under the terms of the General Public License.

CMTK (http://www.nitrc.org/projects/cmtk) is a set of command line tools that implement advanced automatic registration and image processing techniques. CMTK is is open source software licensed under the terms of the General Public License.



NCR toolset (Anderson et al., 2009) is a set of command line tools and a graphical user interface for processing, aligning, and montaging of large microscopy section series. The NCR toolset is free to use software, the source code may be available on request, no specific license is given.

Viking (Anderson et al., 2011) is a desktop client for collaborative manual annotation of extremely large microscopy section series that are stored at a remote location. Annotations are geometric primitives associated with terms from a project-specific ontology, and stored in a centralized remote database. Viking is open source software licensed under the terms of the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported license.

CATMAID (Saalfeld et al., 2009; Longair et al., unpublished manuscript) is a browser-based client for collaborative manual annotation of extremely large microscopy section series that are stored at a remote location. Annotations are geometric primitives associated with terms from a project-specific ontology, and stored in a centralized remote database. CATMAID is is open source software licensed under the terms of the General Public License.

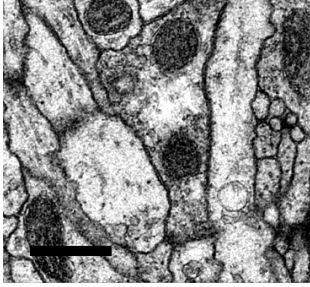
TrakEM2 (Cardona et al., 2012) is an ImageJ-based (Rasband, 1997–2012) desktop application for manual annotation, processing, aligning, and montaging of large microscopy section series. Annotations are geometric primitives or series of free-form 2d annotations associated with terms from a project-specific hierarchy. TrakEM2 is open source software licensed under the terms of the General Public License.

Amira (http://www.amira.com) is a commercial software suite that provides tools for image processing, segmentation and registration of up to four-dimensional data both manual and automatic.

Imaris (http://www.bitplane.com/go/products/imaris) is a commercial software suite that provides tools for image processing, segmentation and registration of up to four-dimensional data both manual and automatic.

Today, only IMOD, CATMAID, Viking, and TrakEM2 can handle data sets of several hundreds of Gigabytes and only TrakEM2 incorporates an advanced method for global elastic serial section alignment and montaging, the one presented in this thesis.

Based on ImageJ, TrakEM2 is implemented in the Java programming language (Arnold et al., 2005; Gosling et al., 2005). Therefore, in order to make my methods available through its interface, it was a pragmatic choice to implement them in Java.



I have developed an Open Source Java library under the name *mpicbg*.[1] The library comprises my implementations of SIFT (Lowe, 2004) and a variation of multi-scale oriented patches (MOPS; Brown et al., 2005), integral images (Crow, 1984), block matching, a collection of image filters, and, most importantly, a generic framework for n-dimensional coordinate transformations, estimators, and optimizers for a multitude of transformation models. All methods described in this thesis are implemented in the mpicbg-library. Naturally, the library is a dependency of TrakEM2.

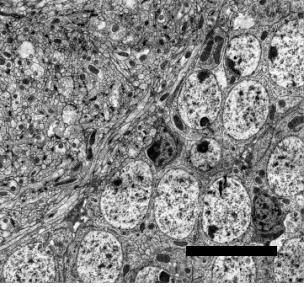
Implementing image processing methods using ImageJ has revealed a number of severe limitations that mainly stem from the history of the platform. ImageJ supports only a limited number of pixel data types (unsigned 8 bit integer, unsigned 16 bit integer, signed 32 bit float and unsigned 24 bit RGB color). Its internal data model is a two-dimensional image with all pixel values mapped into an array of primitive types. Up to five-dimensional data can be stored as a series of two-dimensional images with a convoluted access logic. These limitations make it complicated to implement generic and extensible methods for higher-dimensional image processing. In collaboration with Preibisch and Pietzsch, we have therefore developed ImgLib2 (Pietzsch et al., 2012), a library for generic n-dimensional data representation and manipulation with focus on image processing

ImgLib2, TrakEM2, and the mpicbg-library are packaged with Fiji (Schindelin et al., 2012), an ImageJ distribution targeting the bio-image informatics community.

3.1 The mpicbg-library and TrakEM2

The TrakEM2[2] software was developed by Albert Cardona et al. (2012) for rapidly browsing, modifying, and annotating ssTEM data sets of several hundred Gigabytes on a desktop computer. TrakEM2 enables real-time browsing through such data sets by virtualizing display of images. The series of TEM montages consists of several tens or hundreds of thousands of individual 2d images. Each image is associated with a sequence of transformations including lens-distortion correction, elastic deformation and others which means that, in principle, transformed images can be generated on-the-fly. However, in order to accelerate loading speed and to improve the browsing experience, a transformed representation of each image is pre-processed and stored as a pyramid of scales. During browsing the data set, only the images visible in the current canvas need to be loaded at the scale required for display. A TrakEM2 project is stored as an XML-file that contains a hierarchy of project specific annotation terms, associated geometric primitives, and the file-path and sequence of transformations for

[1] http://pacific.mpi-cbg.de/cgi-bin/gitweb.cgi?p=mpicbg.git
[2] http://www.ini.uzh.ch/ acardona/trakem2.html



each image.

During the course of this project, I have substantially contributed to the development of TrakEM2. I have designed and implemented its mechanism for rendering transformed images. I have improved its efficiency and precision in generating high-quality image representations at all possible scales. Finally, I have contributed to the development of its user interface, particularly for interactive manual image deformation, on-the-fly image enhancement and color-composition. In this Section, I will describe and document these contributions.

I have started implementing the mpicbg library during my diploma project (Saalfeld, 2008) including the implementation of SIFT, the least-squares solution for 2d rigid body transformation models, RANSAC and a first version of the iterative optimizer (see Section 2.5). In contrast to claims in the Internet,[3] these implementations do not depend on ImageJ but only provide a thin wrapper layer making it easily accessible for applications that are based on ImageJ.

During the course of this project, I have generalized and improved the library to its current state. In this Section, I will describe it's major components. For a complete documentation, I refer to the Javadocs at the supplementary DVD.

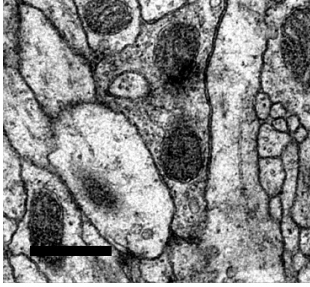
3.1.1 Coordinate Transformations

I designed and implemented a transformation framework that enables to implement and describe virtually any kind of geometric transformation from the function space $\mathbb{R}_n \rightarrow \mathbb{R}_m$. The interface `CoordinateTransform` serves as a basis for arbitrary coordinate transfer functions $\mathbb{R}_n \rightarrow \mathbb{R}_m$ (see Figure 3.1). It is a limitation of the Java programming language that constant values cannot be used as generic parameters. Therefore, defining an n-dimensional vector interface with a fixed number of dimensions at compile time would require definition of a separate class or interface for each n and comes with additional cost for accessing the scalars. I avoid these shortcomings by passing real value coordinates as `float[]` arrays thus leaving open the number of dimensions to be transferred. That way, I cannot assure that the passed vectors have the appropriate number of dimensions at compile time, which, in my opinion, is outweighed by optimal performance, flexibility and ease of use. At critical places, assertions are used for debugging assistance.

The interface declares two methods to apply the transfer function to coordinates

`apply(float[]):float[]` Creates the transferred coordinate and leaves

[3] http://www.jidul.com/wp/tools/sift-algorithm/?lang=en
http://code.google.com/p/preplab/source/browse/trunk/src/imagelib/Sifter.java



the passed coordinate untouched.

`applyInPlace(float[]):void` Replaces the passed coordinate values by the transferred values. This method offers optimal performance because no objects are created. The programmer is required to make sure that the length of the passed vector is the maximum of the numbers of input and output domain $\max(n, m)$.

For use in TrakEM2 projects, the interface is extended to export into and load from XML using the XML element `ict_transform` with the attributes `class` and `data`

```
<!ELEMENT ict_transform EMPTY>
<!ATTLIST ict_transform class CDATA #REQUIRED>
<!ATTLIST ict_transform data CDATA #REQUIRED>
```

The `class` attribute holds the class name of the `CoordinateTransform` as generated by `java.lang.Class.getCanonicalName()` and will be used to create an instance of the `CoordinateTransform` through reflection. This requires that each `CoordinateTransform` can be generated with an empty constructor which usually means that its parameters are equivalent with the identity transform. The methods declared by the extended interface are

`toDataString():String` Creates a `String` representation of the properties of the `CoordinateTransform` that is suitable to re-create itself through `init(String)`.

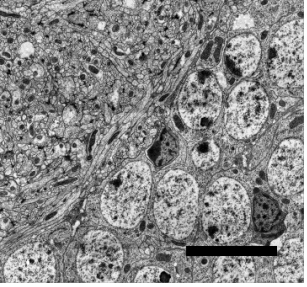
`toXML(String):String` Creates an XML-entity of the `CoordinateTransform` for saving into a TrakEM2 project file with an optional indentation string for formatting purposes.

`init(String):void` Initializes the properties of the `CoordinateTransform` from a data string as generated by `toDataString()`.

Invertible functions as required for inverse coordinate transfer implement the interface `InvertibleCoordinateTransform` that declares the respective methods

`applyInverse(float[]):float[]` Creates the inversely transferred coordinate and leaves the passed coordinate untouched.

`applyInverseInPlace(float[]):void` Replaces the passed coordinate by the inversely transferred coordinate. This method offers optimal performance because no objects are created. The programmer is required to make sure that the length of the passed vector is the maximum of the numbers of input and output domain $\max(n, m)$.



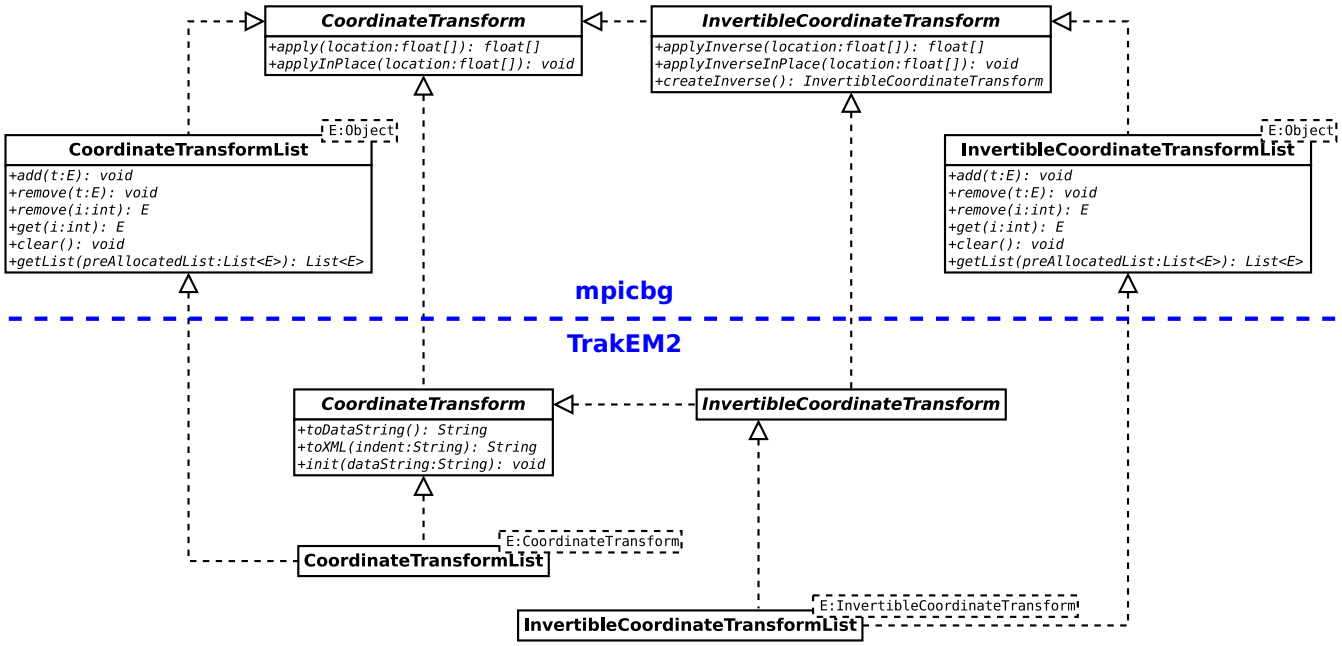


Figure 3.1: UML diagram of the interface hierarchy for coordinate transformations and lists of coordinate transformations. TrakEM2 interfaces extend the respective mpicbg interfaces by XML import and export. For better visualization, most of the inheritance graph on the mpicbg side is omitted.

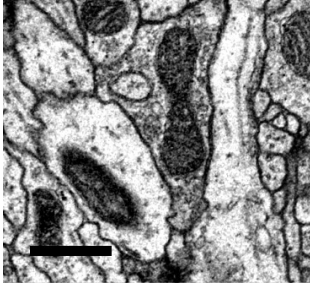
`createInverse():InvertibleCoordinateTransform` Creates the inverse of this `InvertibleCoordinateTransform` as a new instance.

The extended TrakEM2 interface exports the XML element `iict_transform` using the same attributes as `ict_transform`.

3.1.2 Concatenating Coordinate Transformations

Rendering a transformed image means transferring data from one pixel grid into another. For non-integer transformations, pixel values in the target image are generated from data at coordinates between source pixels. Such data is generated through interpolation that, regardless which method is used, is an approximation. Therefore, serial image transformation results in degrading quality of the output. I overcome this issue with framework that, instead of transforming images sequentially, concatenates the coordinate transfer functions and generates the transformed image from pixel data in the source image at coordinates as estimated by sequential coordinate transfer.

For many classes of transfer functions, analytic concatenation is pos-



sible (e.g. all affine transformations). Still, in a mixed environment such analytic concatenation is not necessarily available. I support concatenation of arbitrary transfer functions through sequential coordinate transfer. The concatenation of `CoordinateTransforms` is a sequence of `CoordinateTransforms` with the output of one transfer function serving as input for its successor. The respective interface is the `CoordinateTransformList` that itself is a `CoordinateTransform` and, thus, can be used for transparent coordinate transfer (see Figure 3.1). Its `applyInPlace(float[])` method will transfer the passed coordinate calling `applyInPlace(float[])` on all elements of the list from first through last. The interface has a generic parameter `E extends CoordinateTransform` that can be used to further specify its elements. In addition to the methods inherited from `CoordinateTransform`, the interface declares a set of methods to add, remove and access particular elements

`add(E):void` Adds a `CoordinateTransform` of type `E` to the list.

`remove(E):void` Removes a `CoordinateTransform` of type `E` from the list if the list contains that `CoordinateTransform`. If the list does not contain the `CoordinateTransform`, nothing will change.

`remove(int):E` Removes the `CoordinateTransform` of type `E` at the specified index from the list and returns it.

`get(int):E` Returns the `CoordinateTransform` of type `E` at the specified index.

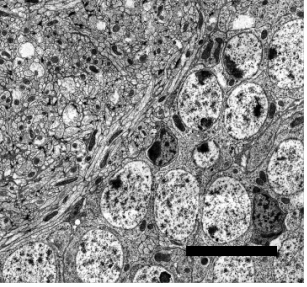
`clear():void` Clears the list.

`getList(List<E>):List<E>` Returns a `java.util.List<E>` that holds all `CoordinateTransforms` of type `E`. An optionally passed preallocated list will be used.

The TrakEM2 interface exports the XML element `ict_transform_list` with an arbitrary number of `CoordinateTransform` children

```
<!ELEMENT ict_transform_list (ict_transform|iict_transform|
          ict_transform_list|iict_transform_list)*>
```

The invertible counterpart of `CoordinateTransformList` is the interface `InvertibleCoordinateTransformList` that is an `InvertibleCoordinateTransform` accordingly. Inverse coordinate transfer is performed by calling the `applyInverseInPlace(float[])` method of all list members in inverse order, from last through first. `InvertibleCoordinateTransformList` can only contain `InvertibleCoordinateTransforms`. It exports to the XML element `iict_transform_list` with an arbitrary number of `InvertibleCoordinateTransform` children



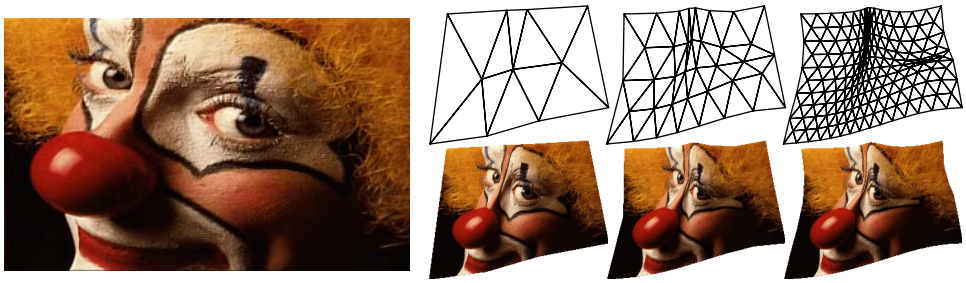
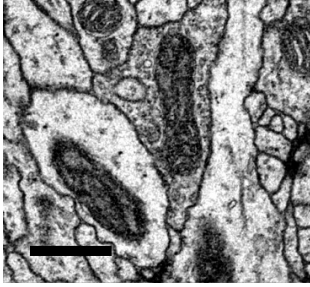


Figure 3.2: Approximate Inversion of non-invertible coordinate transfer functions through triangular meshes is used for rapid image rendering. The resolution of the mesh specifies the accuracy of the approximation. In the figure, the same original image is transformed using meshes at different resolutions to approximate the same moving least-squares transformation. In TrakEM2, the mesh resolution for rendering is a parameter of the project, 32 being default. I found this a reasonable trade-off between accuracy and speed for the usually smooth non-linear deformations involved.

```
<!ELEMENT iict_transform_list (iict_transform|
          iict_transform_list)*>
```

3.1.3 Triangular Meshes for Approximate Inversion of Non-Invertible Coordinate Transformation

In the common case that an `InvertibleCoordinateTransform` is not available but a non-invertible `CoordinateTransform`, rendering a target image efficiently still requires inverse coordinate transfer. I realize this through triangular tessellation as implemented in the class `TransformMesh`. A `TransformMesh` is created passing a `CoordinateTransform` and the desired resolution specified as the number of triangles per row (Figure 3.2 visualizes the effect of the resolution parameter). It creates a regular triangular mesh whose vertices are then transferred according to the `CoordinateTransform`. The three vertices of each triangle specify a unique 2d affine transformation which can be directly inverted. The set of affine transformations per triangle approximates the `CoordinateTransform` and can thus be used for piece-wise approximate inversion. For TrakEM2, the `TransformMesh` automatically estimates the bounding box of the mesh after transformation and adds an offset at the integer coordinates immediately below the top left real coordinates of the bounding box. The bounding box can be accessed as a `java.awt.Rectangle` through the public method `getBoundingBox()`. While the `TransformMesh` implements the mpicbg interface `InvertibleCoordinateTransform`, it cannot be exported as an XML element and, there-



fore, not be used as a transformation for TrakEM2 displayable elements. This decision is justified by the fact that `TransformMesh` is not a directly controlled transformation but, as a side-effect, introduces an implicit offset that makes sense for image rendering but changes the actual coordinate transfer. This is not a limitation—any arbitrary coordinate transformation for a set of vertices can be realized by a `MovingLeastSquaresTransform` (see below) with a control-point for each vertex.

Forward coordinate transfer is realized by consecutive execution of the `applyInPlace(float[])` method of the `CoordinateTransform` and adding the offset. Inverse coordinate transfer is realized by first searching the triangle that contains the passed coordinate in transferred space minus offset and then calling the `applyInverseInPlace(float[])` method of the affine transformation specified by this triangle.

3.1.4 Available Coordinate Transformations

I have implemented the following coordinate transformations:

`TranslationModel2D` 2d shift

`TranslationModel3D` 3d shift

`RigidModel2D` 2d rotation and shift

`SimilarityModel2D` 2d isotropic scale, rotation and shift

`AffineModel2D` 2d shear, non-isotropic scale, rotation and shift

`AffineModel3D` 3d shear, non-isotropic scale, rotation and shift

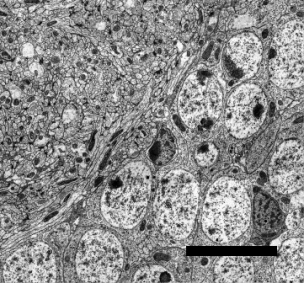
`MovingLeastSquaresTransform` *n*d deformation using one of the above models estimated locally by mean of moving least-squares fit (Schaefer et al., 2006) for a set of corresponding control points

`TransformMesh` 2d triangular mesh for approximate inversion of non-invertible coordinate transformations

I have assisted the adaptation of the following existing implementations that are both non-invertible and thus rendered by `TransformMesh`:

`CubicBSplineTransform` 2d cubic b-Spline deformation implemented by Arganda-Carreras et al. (2006). Through that implementation, bUnwarpJ (Arganda-Carreras et al., 2006) became available as a non-linear pairwise registration tool in TrakEM2.

`NonLinearTransform` 2d polynomial transformation by Kaynig et al. (2008). Through that implementation, non-linear lens-deformation correction



(Kaynig et al., 2008) became available in TrakEM2. I have further extended the original implementation by the authors combining lens-model estimation with globally optimal montages through an iterative scheme achieving more accurate results (see Section 2.5.6).

3.1.5 Rendering Transformed 2d-Images

Image rendering is implemented for ImageJ's `ImageProcessor` class and realized by classes implementing the interface `Mapping` whose public methods are

`map(ImageProcessor,ImageProcessor):void` Maps a source image into a target image of the same type using nearest neighbor interpolation.

`mapInterpolated(ImageProcessor,ImagePr...):void` Maps a source image into a target image of the same type using bilinear interpolation.

Both methods are realized using the `ImageProcessor` methods

```
putPixel(int,int,int):void
```

```
getPixel(int,int):int
```

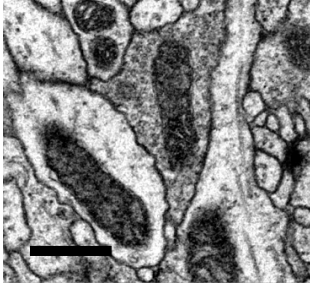
```
getPixelInterpolated(int,int):int
```

with the latter introduced into ImageJ after a patch that I have developed. By that, all ImageJ pixel types are supported, the only requirement is that source and target image are of the same type.

For an `InvertibleCoordinateTransform`, the mapping can be implemented directly through inverse coordinate transfer. However, for a `TransformMesh`, executing the triangle search for each pixel coordinate results in very slow rendering. Therefore, I have introduced `TransformMeshMapping` that implements `Mapping`. `TransformMeshMapping` iterates over all triangles and, for all pixels that are inside the transferred triangle, calculates the source coordinate through inverse application of the triangle's affine. That way, for each triangle, only the pixels in its bounding box need to be checked for being inside the triangle or not. Furthermore, rendering the contents of a triangle is independent for each triangle and thus executed in parallel.

3.1.6 Rendering Transformed 3d-Images

With the goal to project 3d image data (such as Confocal Microscopy stacks) into the aligned stack of serial EM sections, I have developed the interface `mpicbg.ij.stack.Mapping` for rendering a 2d-slice from a transformed 3d-volume. The interface is built for the ImageJ classes `ImageProcessor` and



`ImageStack`. The interface is currently implemented by the class `InverseTransformMapping` for 3d-coordinate transformations that can expose their inverse. The respective mapping methods are

`map(ImageStack,ImageProcessor):void` Renders a 2d-slice of the transformed volume by nearest neighbor interpolation.

`mapInterpolated(ImageStack,ImageProcessor):void` Renders a 2d-slice of the transformed volume by tri-linear interpolation.

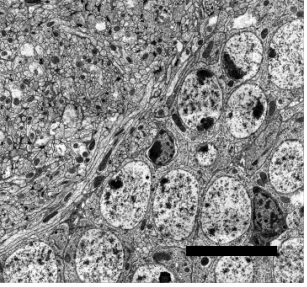
`setSlice(float):void` Sets the z-position of the 2d-slice in target space, default is $z = 0$.

In TrakEM2, the coordinate transfer function is pre-processed for optimal performance and rendering quality. Both average scale and the 2d-affine component in the target plane are extracted. The 2d-affine including scale is propagated to the 2d-affine executed by Java AWT. The slice itself is rendered at an effective scale close to 1.0. For scales $s < 1$ the image is smoothed by a Gaussian filter with $\sigma = \sqrt{0.25/s^2 - 0.25}$ effectively suppressing frequencies higher than the target resolution limit.

3.1.7 Manual Deformation with Moving Least Squares

For the purpose of manual alignment and correction of automatic alignments, we have implemented a manual deformation tool for control point based deformation using the Moving Least Squares technique as described by Schaefer et al. (2006). The tool can be applied to an arbitrary selection of images in a layer. For rapid rendering, each consecutive (non-interrupted) series of patches is exported into an AWT image with some overhead left and right of the screen. Each navigation event in the canvas (pan,zoom) triggers re-creation of these temporary images. Without exposing the full resolution patches to the tool, these images allow to generate a similar appearing preview of the current deformation process in real-time. The user will now add control points and drag them to the desired location. While dragging, the preview images are updated reflecting the current state of the transformation. The tool uses a `TranslationModel2D` for one point, a `SimilarityModel2D` for two points, and an `AffineModel2D` for three or more point. That is, non-linear deformation can be observed after the fourth point has been dragged. Similar to the existing linear transformation mode, the non-linear transformation mode can be canceled or applied. On-apply, all selected patches are rendered using the deformation at full resolution. Since TrakEM2 requires the 2d-affine component of a patch's transformation chain last and independent from the rest, the transformation applied to each patch consists of three components:

1. The inverse of the patch's affine including bounding box compensation for already existing coordinate transforms.



2. The moving least-squares transform as introduced to the tool.
3. The patch's affine including bounding box compensation for already existing coordinate transforms.

Finally, the patch's 2d-affine is updated compensating for the offset introduced by the current bounding box of the patch.

3.1.8 Transparency

TrakEM2 renders its images using the Java AWT 2d framework. Images in Java AWT support transparency through alpha-channels. The actual image data is loaded and processed and through ImageJ bindings. The mappings as described above are implemented for the ImageJ's `ImageProcessor` and `ImageStack` thus supporting all types supported by ImageJ (e. g. floating point accuracy for each pixel which is not available in Java AWT). Unfortunately, ImageJ does not support transparency. Therefore, transparency and images are stored as independent data. While Java AWT supports affine transformation natively, all other classes of transformation need to be rendered into a rectangular image that is then passed to Java AWT. I have implemented another specialization of the `Mapping` interface, the `TransformMeshMappingWithMasks` which transfers an image and the corresponding mask without duplicating the effort for transferring the coordinates. The class specifies the inner class `ImageProcessorWithMask` that holds an `ImageProcessor` (the actual pixel data), a mask (the alpha transparency channel) and a mask for out of bounds coordinates. The latter is required because both the image and the alpha channel can be interpolated whereas a specific location is either inside or outside exclusively. The class adds the equivalents to the two `Mapping` methods for `ImageProcessorWithMask`

```
map(ImageProcessorWithMask,ImageProcessorWithMask):void
```

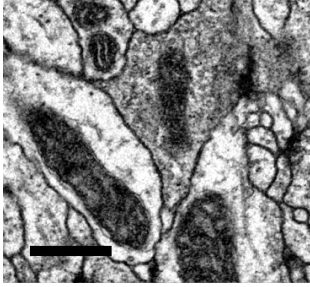
```
mapInterpolated(ImageProcessorWithMask,ImageProces[...]):void
```

3.1.9 Registration and Alignment

I have implemented a set of alignment tools that realize registration through minimizing the sum of square residuals of corresponding landmarks (see Section 2.5) and the elastic method based on block matching described in Section 2.6. Following, a brief description of the parameters that the implementations expose to the user.

Scale Invariant Interest Point Detector

SIFT interest points are blob-like structures as detected by the Difference of Gaussian (DoG) detector (Lowe, 2004, and Section 2.5.1). Basically, DoG



builds upon a Gaussian scale space that is built by smoothing the original image with normalized Gaussian kernels with growing standard deviation σ. The scale-interval between σ and 2σ is called a scale-octave (as used in music scale where an octave means a factor of two between the base frequencies of two sounds).

initial gaussian blur The least blurred image will have an effective blur defined by that parameter. It's size is defined by the parameter **maximum image size**. Setting this parameter to values ≤ 1.0 will result in both, **maximum image size** and this parameter multiplied by 2.

steps per scale octave Specifies the number of scales generated per octave, that is the resolution of the scale space alongside the scale dimension.

minimum image size Per each scale octave, the effective image size is reduced by a factor of 2. This parameter specifies the upper bound of the scale space.

maximum image size If larger than that parameter, the image is downscaled to that size before generating the scale space. This parameter specifies the lower bound of the scale space.

Local Descriptor

The local SIFT descriptor is built from a grid of orientation histograms, each generated from a 4×4 gradient patch extracted from the scale and rotation invariant region around the detection. Matching is nearest neighbor search in this local descriptor space.

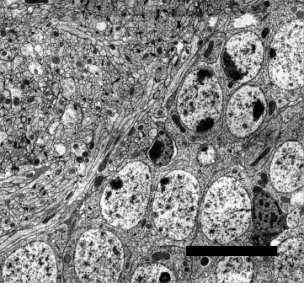
feature descriptor size The side length of the grid and thus square root of the number of histograms used for each descriptor. I found larger descriptors (size=8) to perform better for cross-section alignment to the cost of increasing runtime for matching.

feature descriptor orientation bins The number of bins per orientation histogram. I have never found changing this parameter from the default value=8 to improve the result.

closest/next closest ratio The ratio between the distance to nearest neighbor and second-nearest neighbor in local descriptor space as a measure for distinctiveness of the match. Lower values will reject more features as ambiguous.

Geometric Consensus Filters

Estimating correspondences by matching local feature descriptors usually results in a significant fraction of false positives. Such false positives need



to be rejected since they would adulterate the estimated transformation model. As described in Section 2.5.2, I use a combination of RANSAC and a robust trimming estimator for that purpose. Both methods depend on fast least-squares model estimators which I have implemented after Schaefer et al. (2006) for `TranslationModel2D`, `TranslationModel3D`, `RigidModel2D`, `SimilarityModel2D`, `AffineModel2D`, and `AffineModel3D`.

maximal alignment error During RANSAC, random minimal subsets of corresponding points are used to estimate an approximate candidate transformation which is then tested calculating the residual of all other matches. Matches resulting in a residual larger than this parameter will be considered 'outliers'. The value should be chosen such that it covers at least the expected deformation relative to the **expected transformation** that is calculated. The largest set of 'inliers' found is then used to further refine the corresponding transformation by a least squares fit on the entire 'inlier' set.

minimal inlier ratio Reject a model identified by RANSAC if the number of 'inliers' relative to all correspondences is less than this parameter.

minimal number of inliers Reject a model identified by RANSAC if the number of 'inliers' is smaller than this parameter.

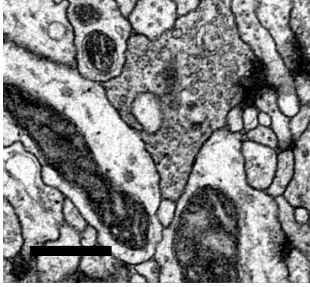
expected transformation Specifies the transformation class that is expected to align the extracted landmark correspondences up to the **maximal alignment error**. This transformation is not used for alignment but for filtering 'outliers' only.

ignore constant background Will result in transformations and the corresponding 'inlier' sets being rejected in case that the estimated transformation is close to identity. 'Close' is specified in pixels in the field **tolerance**. The correspondence candidates are searched for another transformation. Using this, dominant textured background patterns that are constant among all images are successfully ignored.

Alignment

desired transformation The transformation model that is calculated and applied during alignment. Note that this transformation is not necessarily equal to the **expected transformation** as specified for geometric consensus filters.

correspondence weight Sets a weight to the correspondences estimated using this set of parameters. Changing this value is sensible in situation where multiple parameters sets are used, e.g. for global montaging and alignment of series of overlapping tiles.



regularizer The transformation model that is used as a regularizer to the **desired transformation** (see Section 2.5.5). Typically, this will be a rigid body transformation. The estimated transformation is that linearly interpolating between the **desired transformation** and **regularizer** with the weight of the **regularizer** specified by the parameter **lambda** in the range $[0, 1]$. Accordingly, **lambda**=0 means that the result is the **desired transformation**, and **lambda**=1 means the result is the **regularizer**. Instead of setting **lambda**=0, **regularize** may be unchecked.

Block Matching

scale Specifies the scale factor at which matching is performed. The scale should be selected such that (1) high level noise is effectively invisible, and (2) for series alignment, such that approximately isotropic resolution is achieved.

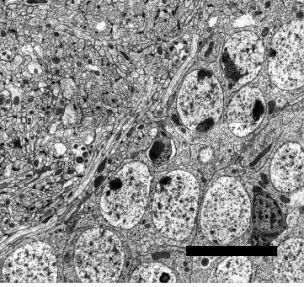
search radius Specifies the radius of the region that is explored for the best local offset. This radius must include the largest non-linear deformation expected relative to the **expected transformation** specified before. At the same time it should be kept as small as possible to increase speed and to reduce the occurrence of spurious matches.

block radius Specifies the radius of the block template used for block matching. It should be large enough to include recognizable texture that is robust with respect to section-to-section shape change of the specimen. Too large values will result in an increase in runtime and a deformation field too smooth to cover lower scale non-linear distortions. Usually, setting both **search radius** and **block radius** to similar values is a good choice.

resolution Specifies the number of vertices in a long row of the spring mesh as depicted in Figure 2.6. TrakEM2 does not require all image tiles in a montage to have the same size. Therefore, it asks for the **triangle side-length** instead. That way, the individual mesh resolution for each image can be calculated such that all tiles are modeled by meshes whose triangles are of approximately equal size.

minimal PMCC r Is the threshold for the correlation coefficient. Matches resulting in a correlation coefficient below **minimal PMCC r** will be rejected (see Section 2.6.2 and Figure 2.8).

maximal curvature ratio Specifies the threshold to reject edge responses as described in Section 2.6.2 and Figure 2.8. The value must be > 1. Higher values will lead to more matches alongside elongated structures accepted.



maximal second-best r / best r Specifies the threshold to reject matches as ambiguous. Higher values will lead to more potentially ambiguous matches accepted (see Section 2.6.2 and Figure 2.8).

approximate local transformation Specifies the transformation model that is used to model a smooth deformation field described by a moving least-squares fit to all block matches using a Gaussian RDF Section 2.6.2 and Equation (2.35) whose standard deviation is specified by the parameter **local region sigma**. Block Matches that do not conform with this deformation field up to an absolute or relative **maximal local displacement** will be rejected as 'outliers'. The relative threshold is specified as a multitude of the average residual weighted by the same Gaussian RDF.

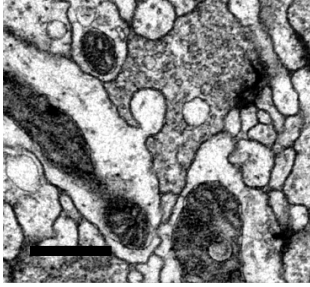
Automatic Alignment Modes

I have implemented three separate automatic alignment modes covering the common needs for handling montages from serial section EM data sets.

Align Layers Serial section alignment, either using SIFT landmark correspondences or the elastic method or both in combination. Feature correspondences or block matches respectively are extracted for each pair of sections in a user-specified local range, the transformation for each section is optimized using the method described in Section 2.5.4 or 2.6 respectively. Optionally, one section can be selected as a reference, serving as a global anchor. In addition to be applied to a complete series of sections, the tool can be used to align a subset of sections, propagating the transformation to the end or beginning of the whole series. This option is useful for fixing gaps in otherwise successful alignments.

Montage Estimates a globally optimal montage on all selected tiles in a section. The sum of square residuals for all pairwise feature correspondences is minimized, or the elastic method is used.

Align multi-layer mosaic Implements global registration of series of montages as described in Section 2.5.4 (Saalfeld et al., 2010). Feature correspondences for overlapping tile pairs within and across sections are estimated and the global sum of square residuals minimized. In order to reduce the number of cross-section pairs to be tested, adjacent sections are registered with a linear transformation beforehand. Therefore, each section montage is generated independently first. A section does not have to be a single montage. Each montage in one section is registered with each montage in the consecutive section, ideally being combined into a single global montage through the section series.



Global optimization is performed again after a new section is added into the global montage, thus initializing the next optimization with a good estimate. Parameter sets for finding correspondences within and across sections are independent. I suggest to set a significantly higher weight to the correspondences across sections than to those within section montages (~20×). That way, intra-section matches drag the section towards rigidity without enforcing it to be rigid. In other words: Intra-section correspondences act as a regularizer for the deforming section alignment. Increasing the weight of cross-section correspondences decreases the influence of the regularizer.

The resulting globally optimal, but locally discontinuous, configuration of tile-wise linearly transformed layers can be warped using the method described in Section 2.5.7 such that continuity within the layer is restored. This method is available through the **deform layers** option and, alternatively, through a post-processing script. See Figure 2.4 for a visualization of the effect of warping.

Optimization

maximal iterations Specifies the maximal number of iterations the optimizer or particle-spring simulation will perform.

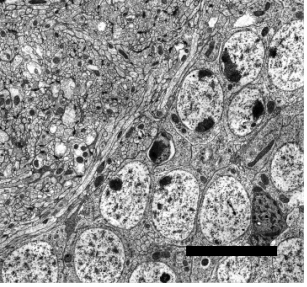
maximal plateauwidth Optimization runs until the cost is not changing any more. It must not change at least for the number of iterations specified by this parameter.

filter outliers In the unlikely case that a model supported by false positives had not been filtered by the robust estimator, it can be rejected here by not being in consensus with the rest of the globally optimized solution. The solution is equivalent with the robust trimming estimator as described in Section 2.5.2 just that the model to be estimated is the global set of transformations $\boldsymbol{T}$. After optimization, the mean residual $\bar{\epsilon}$ is calculated for all tile- or section-pairs and the correspondences connecting pairs with residuals larger than $c = k\bar{\epsilon}$ are removed. k is specified through the parameter **mean factor**.

stiffness Specifies the spring constant k for all springs in the image mesh. Larger values will result in less deformed, less perfectly aligned results. Very small values may result in unevenly distributed deformation due to premature 'relaxation' of the simulation.

Composites

Driven by the need to overlay Optical Microscopy data to the EM sections, I have implemented a set of Composites implementing the `java.awt.Com-`



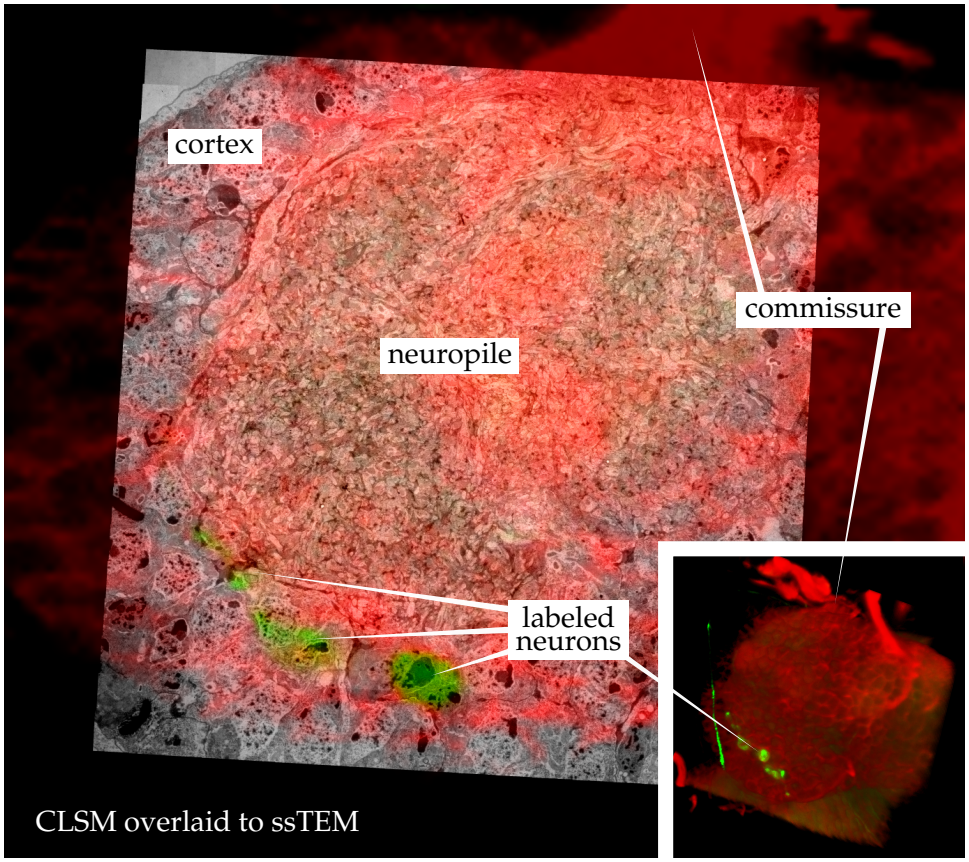
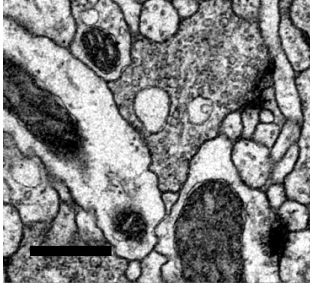


Figure 3.3: Fluorescent 3d stack imaged with the Confocal Laser Scanning Microscope (CLSM) mapped into and overlaid to a TEM series, displayed in the TrakEM2 canvas. The confocal 3d stack is registered to the TEM series by a 3d affine transformation. The respective slice for the current display is generated on the fly. The overlay mode for the confocal stack is YCbCr, combining the gray values from TEM with the Cb and Cr channels from the confocal stack.



`posite` interface. The `Composite` interface specifies how to combine two `Rasters` into a third, freely accessing the independent channels of a pixel. I have implemented the following five `Composites` supporting a continuous opacity parameter and alpha channel that, in TrakEM2, are available as a property of each displayable object:

AddARGBComposite Adds R, G, and B channels independently. Useful for overlay of two independent channels of fluorescent microscopy images.

ColorYCbCr After linear transformation of RGB into YCbCr space, create an image whose luminance channel is that of the second raster and whose Cb and Cr channels are that of the first raster. Useful for overlaying fluorescently labeled optical microscopy images and EM (see Figure 3.3 for an example).

DifferenceARGBComposite Shows the absolute difference of each color channel of two rasters. Useful to judge the accuracy of alignment.

MultiplyARGBComposite Multiplies R, G, and B channels independently. For meaningful results, the source raster is scaled by a factor of 255 such that its values range between 0.0 and 1.0. Useful for using one image as a filter or mask for another.

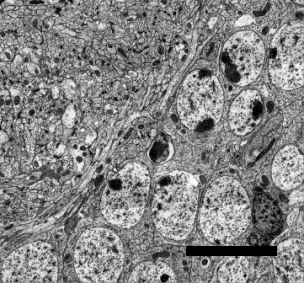
SubtractARGBComposite Subtracts R, G, and B channels independently. Useful for combining two bright-field images or overlapping EM sections.

3.1.10 Summary

I have designed and implemented an extendable framework for coordinate transformations and rendering of transformed 2d and 3d images. Transformations are embedded as properties of image data into TrakEM2 projects allowing fully virtual image transformation with the original image data preserved. Only for purpose of faster navigation, the final result of a transformation chain is saved as a preview image at multiple scales. I have assisted the adaptation of two additional coordinate transformations, cubic b-Splines (Arganda-Carreras et al., 2006) and polynomial transformations (Kaynig et al., 2008) and included an improved version of the lens correction method of Kaynig et al. (2008) into TrakEM2.

In collaboration with Albert Cardona, we have implemented a transformation tool for manual deformation of selected images using the moving least-squares method (Schaefer et al., 2006). This tool can be used to manually correct automatic registration results or for entirely manual alignment.

Additionally, I have implemented a set of Composite modes for color overlay of image data using the AWT framework and several filters that



became useful for display and pre-processing images in the TrakEM2 interface and as standalone plugins for ImageJ.

Extending my earlier work (Saalfeld, 2008) significantly, I have designed and implemented SIFT-based automatic registration methods and made them available in TrakEM2. Similarly, I have designed and implemented the elastic alignment and montaging method described in this thesis and made it available in TrakEM2. I have successfully applied these implementations to reconstruct three large section series at unprecedented quality (see Section 2.7).

3.2 ImgLib2

Many algorithmic concepts from computer vision and image processing are applicable to the analysis of biological image data. However, re-using existing code is often extremely difficult because it is implemented for a specific data type, limited image size, or fixed number of dimensions, e. g., small 2d grayscale images. Biological imaging techniques generate images of varying dimensionality and a multitude of sample types (e. g. wavelength, frequency spectra, diffusion tensors) with varying precision. Improvements in imaging speed and resolution result in gigantic datasets that require well-designed strategies for data handling (e. g. tiled or compressed storage, streaming access). Writing code that is re-usable across many combinations of dimensionality, sample type, and storage strategy is challenging and requires an appropriate abstraction layer.

In collaboration with Tobias Pietzsch and Stephan Preibisch, we have designed and implemented ImgLib2, an open source framework for n-dimensional data representation and manipulation with focus on image processing. ImgLib2 achieves code re-usability through a generic interface architecture that abstracts from dimensionality, sample type, and storage strategy. It is highly extensible, providing developers with great flexibility in adding new sample types and image representations that will seamlessly work with existing algorithms, and vice versa. ImgLib2 shares basic concepts with the C++ frameworks ITK (Yoo et al., 2002) and Vigra (Köthe, 2000) for n-dimensional, generic image processing, and it is the first framework that introduces generic programming to the Java image processing community (Preibisch et al., 2010a).

3.2.1 Architecture

The ImgLib2 core design is based on three main concepts: *Accessibles* (i.e., images), *Accessors*, and *Types*. We define an image as any mapping from a subset of n-dimensional Euclidean coordinate space to a generic pixel value type $f : \Omega \rightarrow \mathbb{T}$ with $\Omega \subseteq \mathbb{R}^n$.

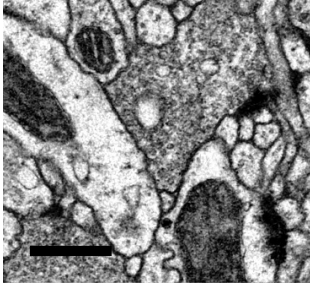


Image properties are expressed by *Accessible* interfaces (collections): Coordinates can be integer or real-valued, the coordinate domain can be either bounded or infinite. The image may support random access at arbitrary coordinates, iteration of all samples, or other access strategies such as search.

Consider a conventional pixel image. It comprises samples of a specific value type in bounded n-dimensional space, arranged on an integer grid. It is both, random-accessible (at arbitrary integer coordinates) and iterable. Importantly, ImgLib2 supports concepts beyond the conventional pixel image, e.g. infinite, procedurally generated images or continuous images interpolated from sparsely sampled data.

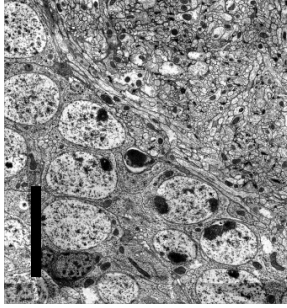
The following list gives an overview of the currently available *Accessible* interfaces and sets them in relation with the corresponding functional mapping

$\Omega = \mathbb{R}^n$	`RealRandomAccessible<T>`
$\Omega = [\mathbf{a}, \mathbf{b}]; \mathbf{a}, \mathbf{b} \in \mathbb{R}^n$	`RealRandomAccessibleRealInterval<T>`
$\Omega = \{\mathbf{x}_1 \dots \mathbf{x}_n\}; \mathbf{x}_i \in \mathbb{R}^n$	`IterableRealInterval<T>`
	`NearestNeighborSearch<T>`
	`KNearestNeighborSearch<T>`
	`RadiusNearestNeighborSearch<T>`
$\Omega = \mathbb{Z}^n$	`RandomAccessible<T>`
$\Omega = [\mathbf{a}, \mathbf{b}]; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$	`RandomAccessibleInterval<T>`
$\Omega = \{\mathbf{x}_1 \dots \mathbf{x}_n\}; \mathbf{x}_i \in \mathbb{Z}^n$	`IterableInterval<T>`

with $[\mathbf{a}, \mathbf{b}]; \mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ denoting all real coordinates in an n-dimensional box spanned by two vectors, and $[\mathbf{a}, \mathbf{b}]; \mathbf{a}, \mathbf{b} \in \mathbb{Z}^n$ denoting all integer coordinates in an n-dimensional box spanned by two vectors (e. g. a pixel raster, see Figure 3.4).

Access to sample (pixel) values and coordinates is provided by *Accessor* interfaces (see Figure 3.5). These exist in variants for integer and real coordinates, iterating and random access, and neighbor search.

It is important to understand that coordinate access is not required to be combined with sample value access, e. g. a `Point` at integer coordinates is not associated with a sample but implements the interfaces `Localizable` and `Positionable`. Only random accessors (`RandomAccess<T>` and `RealRandomAccess<T>`) enable to set the coordinates of a sampling accessor explicitly while all other accessors enable to read the coordinates associated with a particular sample. This becomes particularly clear with iterating accessors (`Cursor<T>` and `RealCursor<T>`). While the order of iteration is subject to implementation, specialized for each memory layout to minimize access time, the iterator provides reading access to both the sample value and the associated coordinate.



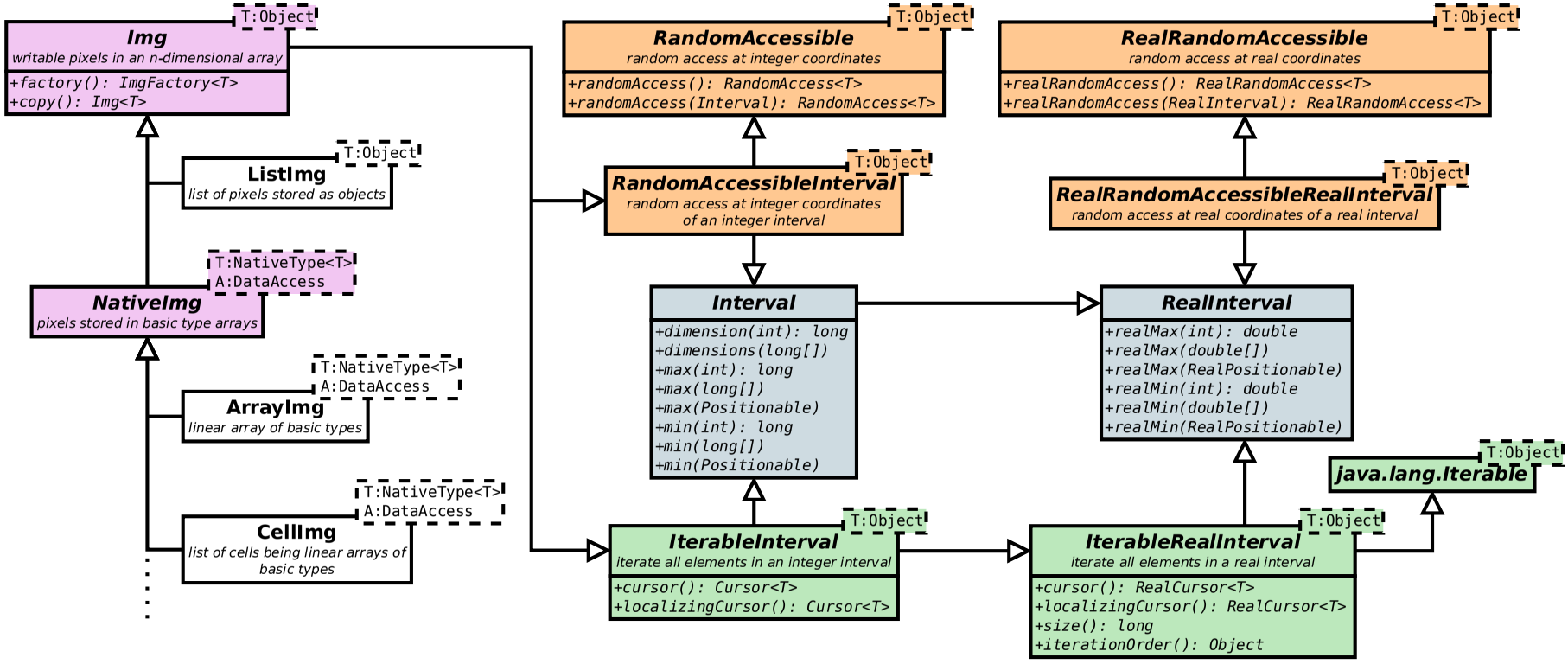
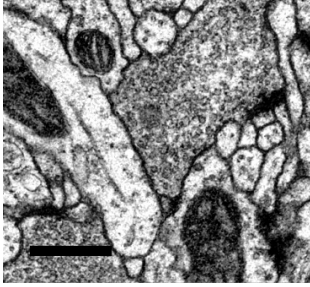


Figure 3.4: On the right, basic ImgLib2 collections interfaces. An image is a mapping from a subset of n-dimensional Euclidean coordinate space (domain) to a generic pixel Type. We distinguish integer and real precision coordinates, bounded and unbounded, randomly accessible and iterable domains, either continuous or discrete with values arranged on a grid or sparsely sampled. A conventional pixel image is a `RandomAccessibleInterval<T>` and `IterableInterval<T>`: it comprises a bounded grid of pixels of generic Type T that are random-accessible and iterable. Note, that the ImgLib2 image interfaces support many concepts that go beyond the classic pixel image. For example, a continuous `RealRandomAccessible<T>` may be procedurally generated or interpolated from discrete data such as pixel images or sparsely sampled datasets. On the left, exemplary concrete implementations to store n-dimensional pixel data in a regular grid. `ListImg` stores pixels as object instances and can be used for any kind of object but not for large numbers of pixels. `ArrayImg` maps pixels into a single Java primitive type array. `CellImg` splits the coordinate space into a grid of rectangular cells, each mapping pixels into a Java basic type array. For mapping into arrays, `NativeType` pixels are required.



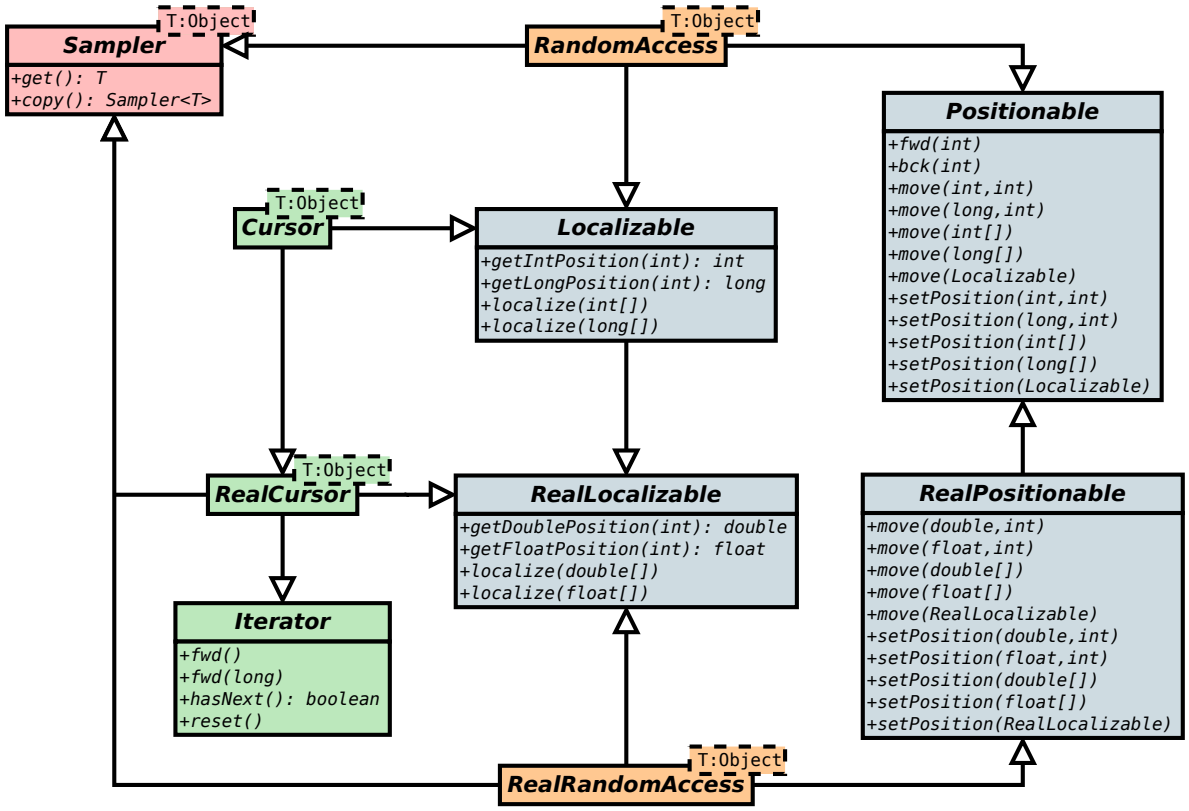
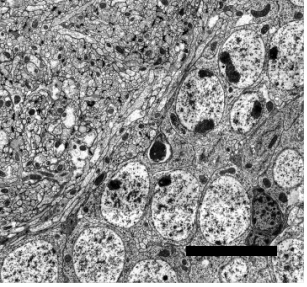


Figure 3.5: Interfaces for access to pixel values and coordinates. `Sampler<T>` provides access to pixel values from a generic pixel value domain. `Localizable` and `RealLocalizable` provide read access to integer and real precision coordinates. `Positionable` and `RealPositionable` provide write access to integer and real precision coordinates. `RandomAccess` and `RealRandomAccess` combine pixel value access and random coordinate access. With `Iterator`, all pixels can be visited once. The order of traversal is subject to special purpose implementation and expected to minimize access time. `Cursor` and `RealCursor` combine pixel access by iteration and localization. Typically, there are two variants of Iterators available. One that calculates its location per each iteration and one that calculates it only per localization request. The former is more efficient when localization occurs frequently, the latter otherwise.

Accessors provide value access via *Types*. ImgLib2 has a hierarchy of *Type* interfaces that describe algebraic properties of families of concrete types (see Figure 3.6). Examples are `Comparable` types or `NumericType` types that support basic arithmetic operations (e. g. $<, >, +, -, *, /$).

Access patterns and type properties allow fine-grained specification of algorithmic requirements. An algorithm that is built using appropriate in-



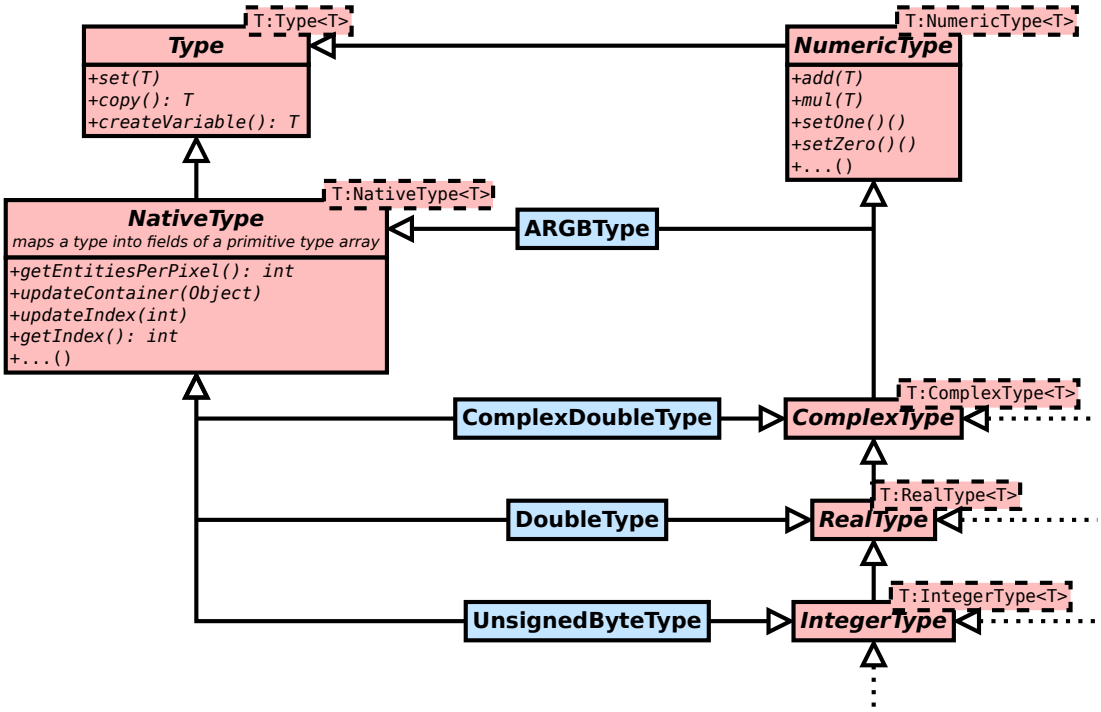


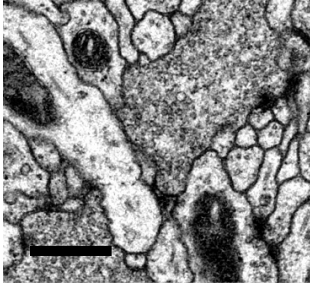
Figure 3.6: A fragment of the ImgLib2 pixel type hierarchy. Pixel type interfaces serve two purposes: Firstly, they specify a set of algebraic operations to be used for pixel processing. Re-usability of algorithms can be maximized by implementing them using the minimal set of operations required (e.g. addition and multiplication for convolution). Secondly, pixel types that implement the `NativeType<T>` interface can be mapped into an array of Java primitive types. Such proxy types avoid the memory and runtime overhead that would be inevitable when storing individual pixels as Java object instances.

terfaces applies to any specific image implementing those interfaces. Re-usability of algorithms is maximized by specifying them for the minimal set of required properties. Consider, for example, summing all pixel values in an image. This can be implemented in two lines of Java code for, e. g., a gray-level image stored as a `byte[]` array. However, it has to be re-implemented, over and over, for every combination of data type, dimensionality, and storage layout. ImgLib2 enables a generic two-lines solution

```
for (T value : image)
  sum.add(value);
```

where we specify that `image` implements `Iterable<T>` and that `T` extends `NumericType<T>`.

The same code handles all pixel images with appropriate value type, virtual views into such images, sparsely sampled data-sets, procedural im-



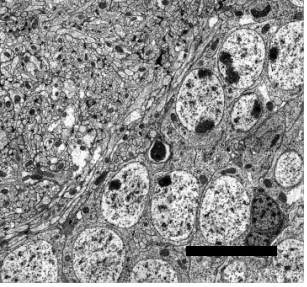
# dim	size [px]	Java [s]	ImageJ [s]	ArrayImg [s]	CellImg [s]
1	100m	0.080/1.14	0.081/1.25	0.081/2.17	0.39/2.21
2	8192^2	0.054/1.13	0.054/1.32	0.055/1.90	0.26/1.89
2	50000^2	×	×	×	9.79/70.6
3	384^3	0.046/1.35	0.049/1.49	0.046/2.01	0.22/2.02
6	28^6	0.386/21.5	×	0.385/27.7	1.91/28.6

Table 3.1: Performance of per-pixel operations on ImgLib2 data structures (`ArrayImg` and `CellImg`), native Java arrays and ImageJ's `ImagePlus`. Values before / are for inverting all pixels in a one- to six-dimensional `float` image. The ImgLib2 implementation achieves native performance with `ArrayImg` while being able to handle images with $> 2^{31}$ pixels with `CellImg` (third row). Values after / are for calculating the center-of-mass in an one- to six-dimensional `byte` image. Here, the ImgLib2 code is on average 1.6× slower than native arrays (caused by an inner loop over the number of dimensions). We consider this a reasonable abstraction penalty as the ImgLib2 code supports any dimensionality, image and value type, while native arrays and ImageJ images require specialized implementations for each supported dimensionality and value type. The trade-off in this example is 20 lines of code (loc) for ImgLib2 vs. 260 loc for native *byte* arrays and ImageJ.

ages, etc. In Java, this level of generality requires pixels to be objects. Storing simple pixel values (e.g., bytes) as individual objects, however, comes with significant memory overhead. Conversely, creating new objects per pixel access introduces significant runtime overhead and triggers frequent garbage collection. Both approaches do not scale well with large images. To address this issue, ImgLib2 uses *proxy types* to access pixel data that can be mapped into Java primitive type arrays (byte[], float[], etc). In this way, an accessor can re-use one proxy instance for all pixel accesses. In the above example, a proxy of type T is instantiated once and then re-used in every iteration, changing only internal state. This virtualization pattern has no performance overhead compared to direct array access, thanks to the optimizations performed by Java's just-in-time compiler (JIT).

3.2.2 Implementation

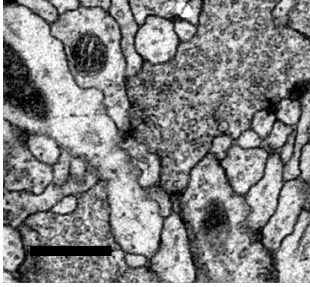
ImgLib2 incorporates common value types (`BitType`, `UnsignedByteType`, `ARGBType`, `ComplexFloatType`, ...) efficiently implemented as proxies that map into Java primitive type arrays. Various implementations for pixel data in a discrete n-dimensional grid (conventional pixel images) are provided: `ListImg` stores pixels as individual object instances and thus sup-



ports arbitrary value types, but does not scale to large numbers of pixels. `ArrayImg` maps proxy types into a single primitive type array, providing optimal performance and memory efficiency. However, Java arrays are limited to a size of 2^{31} (e.g. a square 2d image with maximally 46,340 pixels side-length) which is easily exceeded in todays microscopy recordings (e. g. Bock et al., 2011). `CellImg` splits the coordinate domain into a n-dimensional grid of cells, each mapping into one primitive type array. This enables significantly larger images (2^{62} px) at slightly reduced performance. In generic code we can transparently switch between image implementations using image factories. This allows performance tuning for specific datasets without any modification to the algorithm implementation. In Table 3.1, we compare the performance of ImgLib2 generic code and special purpose (fixed dimensionality and value type) implementations for Java primitive type arrays and ImageJ (benchmark sources in the Appendix).

By design, ImgLib2 permits virtualization of sample access. We use this extensively for providing read and write access into alien data structures, e. g. AWT `BufferedImage` or ImageJ's `ImagePlus` without copying memory content. In addition, we have implemented accessors that perform on-the-fly coordinate and value transformations without copying the underlying data. The `Views` framework creates accessibles that provide coordinate-transforming accessors. Integer coordinate transformations include slicing, windowing, axes permutations, and 90° rotations. Consecutive transformations are reduced and simplified, yielding accessors with optimal performance. For real coordinates we currently support n-dimensional affine transformations, however, the framework is open for future extension to arbitrary coordinate transformations. Interpolating and rasterizing views convert between discrete and continuous coordinate spaces. Finally, some algorithms (e. g. convolution) require access to pixels outside of the image which are usually created by padding or mirroring. This is achieved by extending views, whose accessors generate outside values on demand. Note, that views may be cascaded and act both as input and output for pixel processing. Similarly, the `Converter` framework realizes transparent transformation of values. For instance, a `FloatType` image can be addressed as `ByteType` using an arbitrary mapping function.

ImgLib2 uses Bio-Formats (Linkert et al., 2010) to read and write a large number of image file formats. Interoperability with ImageJ is provided by non-copying wrappers of ImageJ data structures as ImgLib2 accessibles and vice versa. ImgLib2 comprises a growing collection of generic image processing algorithms such as FFT, convolution, edge detection, segmentation, *etc*. Sparsely and irregularly sampled data is supported, stored either as a sample-list or in a kd-tree. Both implement interfaces for nearest-neighbor search, allowing extrapolation of sparse data into a continuous image that provides an interpolated sample value at all real coordinates.



3.2.3 Discussion

ImgLib2 is an open-source Java library for *n*-dimensional data representation and manipulation with focus on image processing. It aims at minimizing code duplication by cleanly separating pixel-algebra (how sample values are manipulated), data access (how sample coordinates are traversed), and data representation (how the samples are stored, laid out in memory, or paged to disc), promoting generic implementations. Algorithms can be implemented for classes of pixel types and generic access patterns by which they become independent of the specific dimensionality, pixel type, and data representation. ImgLib2 focuses on flexible and efficient image storage and access. It illustrates that an elegant high-level programming interface can be achieved without sacrificing performance. It provides efficient implementations of common data types, storage layouts, and algorithms. ImgLib2 relies on virtual access to both sample values and coordinates, facilitating parallelizability and extensibility.

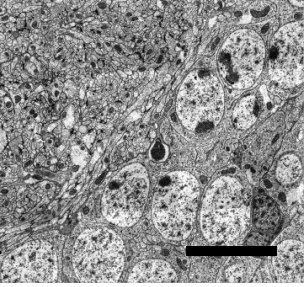
ImgLib2 aims to connect software projects through an interface design that is easily adapted to existing data structures. As a high-performance generic library it is a powerful tool for developing cutting-edge scalable applications for biological image analysis. It is easily integrated into other projects providing an ideal basis for sharing interoperable, generic algorithms.

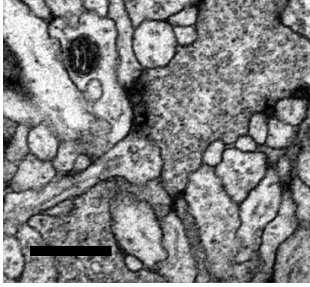
ImgLib2 is a core part of the scijava effort[4]. It is the data model underlying ImageJ2 (Rueden et al., 2010), the KNIME Image Processing toolbox (Berthold et al., 2009), and an increasing number of Fiji-Plugins (Schindelin et al., 2012). The Open Microscopy Environment (OME) plans to make use of the ImgLib2 API in the Bio-Formats library as well as the OMERO server.

ImgLib2 is licensed under BSD. Documentation and source code are available on-line[5] and in a public git repository.[6]

ImgLib2 today comprises a number of additional modules extending the core functionality that I have described here. These modules have benefited from a number of contributors from the open source community, notably C. Rueden, B. DeZonia, C. Dietz, M. Horn, L. Kamentsky, A. Cardona, J. Schindelin, G. Harris, L. Lindsey, M. Longair, J.-Y. Tinevez, N. Perry, J. Funke, S. Jaensch. I want to use this opportunity to thank all contributors for their interest and impact on the further development of the library.

[4] http://scijava.github.com
[5] http://imglib2.net
[6] https://github.com/imagej/imglib





Chapter 4

Discussion

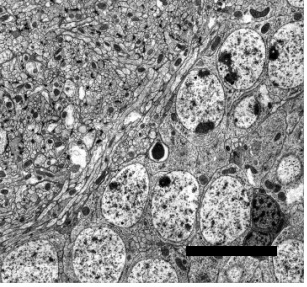
In this thesis, I have described a new method for image-based restoration of the original volume from ultra-thin serial microscopy sections. My method restores the volume by elastically aligning all sections minimizing their non-rigid deformation globally. It is capable of removing the independent distortion of individual sections to a large extent by averaging alignment across a range of sections instead of aligning directly adjacent sections only. Addressing volume restoration from section series by image alignment will unavoidably lead to erroneous deformation because the image signal across sections is moving not only by deformation of the sections but also due to the shape of the specimen. This renders the concatenation of sequential pairwise alignments (Thévenaz et al., 1998; Arganda-Carreras et al., 2006; Anderson et al., 2009) problematic for large sections series as the error accumulates along the series and leads to increasing artificial deformation (see Theorem 1). Global minimization of non-rigid deformation explicitly minimizes the amount of this erroneous deformation and is therefore an appropriate approach to address the problem.

The two most related approaches to my method are those by Guest and Baldock (2001) and Wirtz et al. (2004); Schmitt et al. (2007), both combining an intensity based similarity estimate between adjacent sections and a global elastic constraint for searching the optimal alignment of long series of serial sections in an iterative solution. They achieve initial alignment by variants of principal component analysis. As discussed earlier in Section 2.11, my method differs from these approaches in that:

1. it aligns not only adjacent sections but local ranges of sections, effectively averaging independent section deformation;

2. it estimates initial pre-alignment using invariant local image features, making the approach independent of globally distinctive shapes;

3. it separates pairwise alignment from the elastic solution, yielding an efficient solution for very large series;

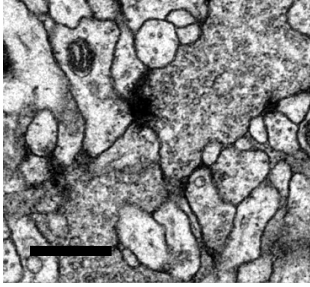


4. it includes a set of filters to explicitly exclude artifacts from contributing to the alignment.

The attempt to remove independent section deformation by averaging is related to the approach by Ju et al. (2006) who suggested to smooth the warping field of each individual section in the series across larger neighborhoods. Other than Ju et al. (2006), I estimate the deformation field between two sections by direct comparison instead of concatenating pairwise deformations. While the transitive approach of Ju et al. (2006) seems compelling for it's lower computational cost, it fully incorporates shape change induced by the specimen. In their approach, this effect is limited by constraining the transformation model to be non-linear only along the y-axis. Direct comparison, while similarly suffering from signal change and motion induced by the specimen, will explicitly fail where this effect becomes dominant. I. e., regions with dramatic signal change do not affect alignment, because the corresponding block matches do not pass the quality filters (see Section 2.6.2). That makes my approach more robust with respect to signal change that is not the target of compensation. At the same time, I support a more general class of non-linear distortions which makes my approach applicable to a broader range of reconstruction problems.

The combination of the elastic approach with alignment and recognition based on robust local image features has interesting implications. Local feature matches enable to register images that are largely occluded by artifacts, lack the globally distinctive properties required for PAT-based methods, or overlap only by a small fraction. In this work, I have suggested to use sets of feature matches to first estimate an affine transformation (or a sub-class thereof) for individual images and subsequently align them using the elastic method. As we have seen in the *Drosophila* data set, this basic sequential principle can be extended. I have calculated a single rigid body transformation not for an entire section but instead for individual tiles of the sections and then used this information to warp the elastically montaged sections (see Sections 2.5.7 and 2.7). With this 'local' initialization, the search radius for subsequent elastic alignment was greatly reduced because the warped montages were already an approximation of the desired elastic alignment.

I can imagine an even more interesting application: Consider a situation where sections were not only deformed elastically but physically split or folded into several pieces, each collected at an entirely different orientation. Consider now that this canvas was recorded as a single image, including several fragments of the section. While in such a situation some part of the section will be lost inevitably, one can use local image features to identify the individual pieces each conforming to an approximately rigid or affine transformation. The approach is straight-forward. Features are extracted in two adjacent section images and the best consensus set is ex-



tracted with RANSAC and the robust estimator (see 2.5.2). This consensus set is then removed from the set of features and the second-best consensus set is estimated. This process is repeated until no more consensus sets can be identified. Each consensus set corresponds to one piece of the corrupted section. It can be used to crop the region and to align it with the series as an individual section.

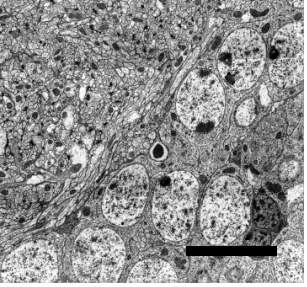
This can be generalized to a coarse-to-fine approach that will become particularly relevant as section series become orders of magnitudes larger (Bock et al., 2011). First, local sets of feature matches and later block matching can be used to hierarchically approximate pair-wise registration and by that narrow the search space for the subsequent operation. My implementation already supports this transparently, e. g. block matching has as an input parameter an initial transformation, although, for the data described in this thesis, it has not been necessary.

Thanks to the robust estimators and filters described in Sections 2.5.2 and 2.6.2, my method is robust with respect to the substantial amount of artifacts present in TEM section series which makes it, to my knowledge, the only available method today that can automatically montage and align such data sets without manual intervention. I however expect even better and more efficient results from exploiting dense deformation fields rather than sparse sets of block matches (see Section 2.6.2). This will become most interesting for identifying and modeling locally discontinuous deformations such as tears and folds.

The elastic section model, as it is currently implemented, is a naïve iterative simulation. I will investigate how the solution of this system can be improved and solved more efficiently. This will become more relevant as data will grow in size.

Although this opportunity has not been used in the presented work, the elastic model already enables to vary the elastic properties across the section. This includes that springs may break when stretched beyond some threshold, enabling the model to become locally discontinuous. The model can also fold where compression exceeds the length of a spring. I am curious how these properties can be exploited to model the corresponding effects in microscopy section series, namely folding, tearing and variable elasticity depending on variance in section thickness.

In Section 2.6.2, I have briefly discussed that the number of 'correct' block matches decreases with increasing distance of two sections in the series. I have shown that this can be used as a deformation invariant similarity metric for two sections which lends itself to correct for mistakes in section order and to estimate the number of sections missing at particular locations (see Figure 2.9). By manual inspection, I was able to identify the major gaps in both the *Drosophila* series and the *C. elegans* series and to estimate their approximate size (see Table 2.1). I believe that it will be relatively straight-forward to formulate an analytic solution to estimate not



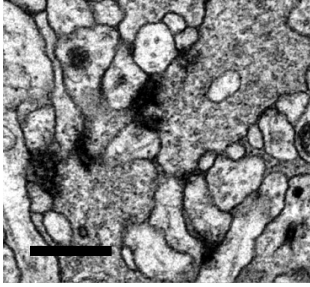
only the size of gaps but to deliver an approximate estimate for the relative thickness of each individual section. This estimate will greatly improve the quality of the volume reconstruction as we know from experiments that the thickness of ultra-thin sections varies significantly along the sections (Rick Fetter, personal communication).

I have implemented the described alignment and image enhancement methods in in the Java programming language and included it into my open source library mpicbg that I distribute with Fiji (Schindelin et al., 2012). I provide a number of standalone plugins for Fiji, and I have integrated landmark-based and elastic montaging and elastic series alignment into the TrakEM2 open source software that is the major component included in Fiji (Cardona et al., 2012; Schindelin et al., 2012, described in Section 3.1). TrakEM2 virtualizes access to all image tiles and provides an interactive point-and-click environment to montage or register subsets of the data and to manually alter the results of automatic elastic registration. In TrakEM2, my registration methods complements many other tools to organize, align, adjust, segment, visualize and analyze large and small electron microscopy datasets. Each image can be transformed by a sequence of arbitrary transformations without degrading its quality by consecutive rendering steps. That is, elastic montaging and series alignment can be executed on image data that was previously corrected for lens-distortion (Kaynig et al., 2010a, and Section 2.5.6) or pre-aligned using a different method (Saalfeld et al., 2010, and Section 2.5.4).

Thanks to virtualized image access in TrakEM2, my methods can be applied to process large data sets of several hundreds of Gigabytes on a conventional desktop computer. At this time, all expensive independent operations are parallelized for optimal use of a single computer with multiple CPU cores accessing shared RAM. Further scalability may be achieved by distributing the dominant operation: independent image-to-image pairwise block matching and filtering to a cluster of independent computers with non-shared memory. To establish the required infrastructure for this is subject of future improvements of the implementation.

I have demonstrated the applicability of my elastic method on two outstanding TEM data sets, one series showing a three-fold *C. elegans* embryo in its entirety and one series showing 1.5 abdominal segments of the *Drosophila melanogaster* first instar larva. Both series were reconstructed at a quality that has not been achieved before. The lab of Albert Cardona has by now annotated the majority of all neuronal processes and synaptic connections in the *Drosophila* series which will provide interesting insights into the structure of local processing units in the larval nervous system. This work would not have been possible without my contribution.

I have demonstrated successfully that the method can be applied directly to other imaging modalities than TEM, using an array tomography series kindly provided by the lab of Stephen Smith. The aligned series has



been of unprecedented quality. My method will enable to utilize array tomography for reconstruction of anatomical ultrastructure at significantly improved quality.

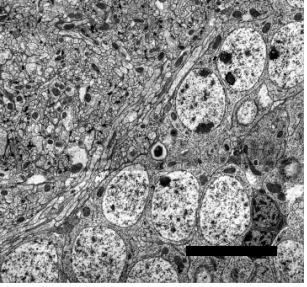
The vast majority of automatic segmentation attempts on serial section TEM has until today been performed by firstly segmenting sections individually and then linking segmented profiles across section (e. g. Mishchenko, 2009; Chklovskii et al., 2010; Kaynig et al., 2010b; Knowles-Barley et al., 2011; Funke et al., 2012). The linking operation usually incorporated substantial efforts to resolve alignment errors. With a section series reconstructed as presented in this work, automatic segmentation attempts can process the volume as a three-dimensional structure which, I believe, will greatly improve the achieved results.

The artificial section series that I have used to evaluate the proposed elastic alignment method will enable to evaluate future developments quantitatively. I consider this an important contribution as Rohlfing (2012) correctly demonstrates that the common evaluation of registration based on intensity based surrogates can be fooled by carefully designed though simple approaches. I. e., it is possible to achieve great evaluation scores although the solution is completely inappropriate. In my artificial section series, ground truth is available per definition which enables quantitative evaluation without any kind of surrogates.

The extendability of the implementation, the accessibility through a popular platform familiar to biologists, the reliability on commodity hardware and the scalability to large data sets make my work a unique and extraordinarily useful tool for today and future challenges in high-resolution reconstructions of large biological specimens from series of ultra-thin microscopy sections. At the time of writing, my method has already been used actively by several laboratories for large scale anatomical reconstruction from serially sectioned volumes.

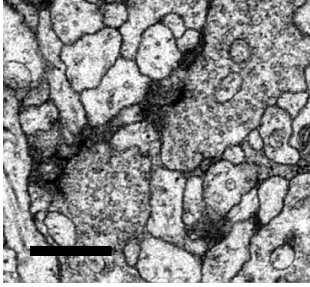
Beyond volume reconstructions from serial microscopy sections, I have contributed substantially to other projects. In collaboration with Preibisch (2010), we have developed a new method for matching clouds of randomly distributed points 2.10 and applied it to register SPIM recordings from multiple angles and across time. The approach is generic in that it can be applied to virtually any kind of repeatable detections. It will thus find application in many more fields than SPIM recordings in the future. As it currently relies on a strict distance-based order of local point clouds, the approach is, in the most general case, invariant to similarity transformations. I am curious whether it will be possible to achieve invariance with respect to affine transformations.

Together with Preibisch and Pietzsch, we have designed and implemented ImgLib2, an open-source Java library for n-dimensional data representation and manipulation with focus on image processing. ImgLib2 is essentially a complete re-design of the earlier proposed ImgLib library



(Preibisch et al., 2010b) that we had developed together with Preibisch. It has been my genuine contribution to cleanly separate coordinate access from pixel access and to generalize coordinate excess beyond discrete pixel coordinates into the real domain. In that sense, ImgLib2 has an enormous expressive power that enables to directly implement concepts beyond those available in other currently available generic image processing libraries (Yoo et al., 2002; Ibáñez et al., 2003; Köthe, 2000). ImgLib2 has already attracted developers from many important open source projects in biological image processing such as the NIH-funded team developing ImageJ2 (Rueden et al., 2010), the Fiji community (Schindelin et al., 2012), the developers of the KNIME Image Processing toolbox (Berthold et al., 2009), the developers of the Open Microscopy Environment (OME; Allan et al., 2012), the developers of CellProfiler (Kamentsky et al., 2011) and many more.

I am looking forward to make heavy use of ImgLib2 for my subsequent projects. In the immediate future I will investigate methods to automatically register sparsely labeled 3d CLSM recordings of the *Drosophila* first instar larval CNS with the aligned ssTEM series. I have already generated these recordings during a summer visit at Janelia Farm last year. The combination of light and electron microscopy will provide interesting insight which neurotransmitters are available in particular neurons and by that enable more advanced interpretation of the reconstructed structures.



Bibliography

E. K. Abbe. Beiträge zur Theorie des Mikroskops und der mikroskopischen Wahrnehmung, I. Die Construction von Mikroskopen auf Grund der Theorie. *Archiv für Mikroskopische Anatomie*, 9(1):413–468, 1873.

M. D. Adams, S. E. Celniker, R. A. Holt, et al. The genome sequence of *Drosophila melanogaster*. *Science*, 287(5461):2185–2195, 2000.

B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell*. Garland Science, New York, NY, 4 edition, 2002. ISBN 0-8153-3218-1.

C. Allan, J.-M. Burel, J. Moore, et al. OMERO: flexible, model-driven data management for experimental biology. *Nature Methods*, 9(3):245–253, 2012.

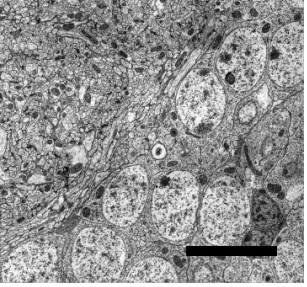
N. M. Alpert, J. F. Bradshaw, D. Kennedy, and J. A. Correia. The principal axes transformation—a method for image registration. *Journal of Nuclear Medicine*, 31(10):1717–1722, 1990.

J. R. Anderson, B. W. Jones, J.-H. Yang, M. V. Shaw, C. B. Watt, P. Koshevoy, J. Spaltenstein, E. Jurrus, K. UV, R. T. Whitaker, D. Mastronarde, T. Tasdizen, and R. E. Marc. A computational framework for ultrastructural mapping of neural circuitry. *PLoS Biology*, 7(3):e1000074, 2009.

J. R. Anderson, S. Mohammed, B. Grimm, B. W. Jones, P. Koshevoy, T. Tasdizen, R. Whitaker, and R. E. Marc. The Viking viewer for connectomics: scalable multi-user annotation and summarization of large volume data sets. *Journal of Microscopy*, 241(1):13–28, 2011.

I. Arganda-Carreras, C. O. S. Sorzano, R. Marabini, J. M. Carazo, C. O. de Solorzano, and J. Kybic. Consistent and elastic registration of histological sections using vector-spline regularization. In *Computer Vision Approaches to Medical Image Analysis*, volume 4241 of *Lecture Notes in Computer Science*, pages 85–95, 2006.

I. Arganda-Carreras, C. O. S. Sorzano, P. Thévenaz, A. Muñoz-Barrutia, J. Kybic, R. Marabini, J. M. Carazo, and C. O. de Solórzano. Non-rigid



consistent registration of 2D image sequences. *Physics in Medicine and Biology*, pages 6215–6242, 2010.

K. Arnold, J. Gosling, and D. Holmes. *The Java™Programming Language*. Addison Wesley Professional, 4 edition, 2005.

R. Bajcsy and S. Kovačič. Multiresolution elastic matching. *Computer Vision, Graphics, and Image Processing*, 46(1):1–21, 1989.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. KNIME – the konstanz information miner: version 2.0 and beyond. *ACM SIGKDD Explorations Newsletter*, 11(1):26–31, 2009.

D. D. Bock, W.-C. A. Lee, A. M. Kerlin, M. L. Andermann, G. Hood, A. W. Wetzel, S. Yurgenson, E. R. Soucy, H. S. Kim, and R. C. Reid. Network anatomy and in vivo physiology of visual cortical neurons. *Nature*, 471: 177–182, 2011.

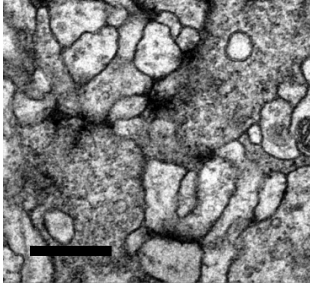
F. L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, 1989.

G. J. Born. Die Plattenmodellirmethode. *Archiv für Mikroskopische Anatomie*, 22:584–599, 1883.

J. J. Bozzola and L. D. Russell. *Electron Microscopy*. Principles and Techniques for Biologists. Jones & Bartlett Learning, Sudbury, MA, 2 edition, 1999.

K. L. Briggman and D. D. Bock. Volume electron microscopy for neuronal circuit reconstruction. *Current Opinion in Neurobiology*, 22(1):154–161, 2012.

K. L. Briggman, M. Helmstaedter, and W. Denk. Wiring specificity in the direction-selectivity circuit of the retina. *Nature*, 471:183–188, 2011.



C. Broit. *Optimal registration of deformed images*. PhD thesis, University of Pennsylvania, 1981.

L. G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

M. Brown and D. G. Lowe. Invariant features from interest point groups. In *In British Machine Vision Conference*, pages 656–665, 2002.

M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

M. Brown, R. Szeliski, and S. Winder. Multi-image matching using multi-scale oriented patches. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 510–517, 2005.

J.-A. Campos-Ortega and V. Hartenstein. *The Embryonic development of* Drosophila melanogaster. Springer-Verlag, Berlin, 1985.

A. Cardona, S. Saalfeld, S. Preibisch, B. Schmid, A. Cheng, J. Pulokas, P. Tomančák, and V. Hartenstein. An integrated micro- and macroarchitectural analysis of the *Drosophila* brain by computer-assisted serial section electron microscopy. *PLoS Biology*, 8(10):e1000502, 2010.

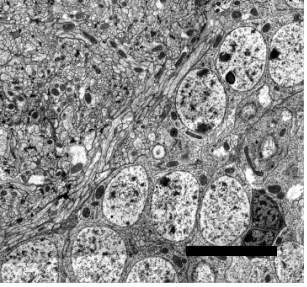
A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomančák, V. Hartenstein, and R. J. Douglas. Trakem2 software for neural circuit reconstruction. *PLoS ONE*, 7(6):e38011, 2012.

B. L. Chen, D. H. Hall, and D. B. Chklovskii. Wiring optimization can relate neuronal structure and function. *Proceedings of the National Academy of Sciences of the United States of America*, 103(12):4723–4728, 2006.

W. Cheung and G. Hamarneh. *n*-SIFT: *n*-dimensional scale invariant feature transform. *IEEE Transactions on Image Processing*, 18(9):2012–2021, 2009.

D. B. Chklovskii, S. Vitaladevuni, and L. K. Scheffer. Semi-automated reconstruction of neural circuits using electron microscopy. *Current Opinion in Neurobiology*, 20(5):667–75, 2010.

C.-F. Chuang, M. K. VanHoven, R. D. Fetter, V. K. Verselis, and C. I. Bargmann. An innexin-dependent cell network establishes left-right neuronal asymmetry in *C. elegans*. *Cell*, 129(4):787–799, 2007.



A. Collignon, F. Maes, D. Delaere, D. Vandermeulen, P. Suetens, and G. Marchal. Automated multi-modality image registration based on information theory. In Y. Bizais, C. Barillot, and R. Di Paola, editors, *Proceedings XIVth international conference on information processing in medical imaging*, volume 3 of *Computational imaging and vision*, pages 263–274, 1995.

D. Cremers and C. Schnörr. Motion competition: Variational integration of motion segmentation and shape regularization. In L. Van Gool, editor, *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, pages 472–480, 2002.

F. C. Crow. Summed-area tables for texture mapping. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 207–212, 1984.

A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1403–1410, 2003.

L. de Broglie. A tentative theory of light quanta. *Philosophical Magazine and Journal of Science*, 47(278):446–458, 1924.

W. Denk and H. Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biology*, 2(11):e329, 2004.

J. Duchon. Interpolation des fonctions de deux variables suivant le principe de la flexion des plaques minces. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 10(3):5–12, 1976.

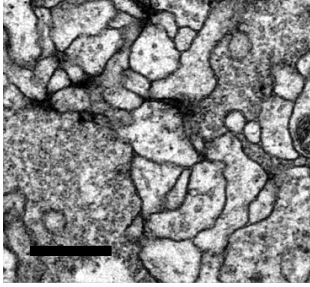
G. Feng, R. H. Mellor, M. Bernstein, C. Keller-Peck, Q. T. Nguyen, M. Wallace, J. M. Nerbonne, J. W. Lichtman, and J. R. Sanes. Imaging neuronal subsets in transgenic mice expressing multiple spectral variants of GFP. *Neuron*, 28(1):41–51, 2000.

J. C. Fiala. Reconstruct: a free editor for serial section microscopy. *Journal of Microscopy*, 218(1):52–61, 2005.

R. D. Fields and B. Stevens-Graham. New insights into neuron-glia communication. *Science*, 298(5593):556–562, 2002.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973.



B. Flach and R. Sara. Joint non-rigid motion estimation and segmentation. In R. Klette and J. Žunić, editors, *Proceedings of the 10th international conference on Combinatorial Image Analysis*, volume 3322 of *IWCIA'04*, pages 631–638, 2004.

B. Flach, E. Kask, D. Schlesinger, and A. Skulish. Unifying registration and segmentation for multi-sensor images. In L. Van Gool, editor, *Proceedings of the 24th DAGM Symposium on Pattern Recognition*, pages 190–197, 2002.

C. C. Fowlkes, C. L. L. Hendriks, S. V. E. Keranen, G. H. Weber, O. Rubel, M.-Y. Huang, S. Chatoor, A. H. DePace, L. Simirenko, C. Henriquez, A. Beaton, R. Weiszmann, S. Celniker, B. Hamann, D. W. Knowles, M. D. Biggin, M. B. Eisen, and J. Malik. A quantitative spatiotemporal atlas of gene expression in the *Drosophila* blastoderm. *Cell*, 133:364–374, 2008.

A. Frome, D. Huber, R. Kolluri, T. Bülow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2004.

J. Funke, B. Andres, F. Hamprecht, A. Cardona, and M. Cook. Efficient automatic 3D-reconstruction of branching neurons from EM data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

J. C. Gee. On matching brain volumes. *Pattern Recognition*, 32(1):99–111, 1999.

J. C. Gee, D. R. Haynor, L. L. Briquer, and R. K. Bajcsy. Advances in elastic matching theory and its implementation. In P. Cinquin, R. Kikinis, , and S. Lavallée, editors, *CVRMed-MRCAS '97*, 1997.

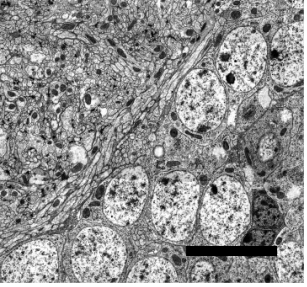
A. Goshtasby. Template matching in rotated images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7(3):338–344, 1985.

J. Gosling, B. Joy, G. Steele, and G. Bracha. *The Java™Language Specification*. Addison-Wesley, 3 edition, 2005.

E. Guest and R. Baldock. Automatic reconstruction of serial sections using the finite element method. *Bioimaging*, 3(4):154–167, May 2001.

M. G. L. Gustafsson. Nonlinear structured-illumination microscopy: Wide-field fluorescence imaging with theoretically unlimited resolution. *PNAS*, 102(37):13081–13086, 2005.

M. G. L. Gustafsson, D. A. Agard, and J. W. Sedat. I^{5}M: 3D widefield light microscopy with better than 100 nm axial resolution. *Journal of Microscopy*, 195(1):10–16, 1999.



M. G. L. Gustafsson, L. Shao, P. M. Carlton, C. J. R. Wang, I. N. Golubovskaya, W. Z. Cande, D. A. Agard, , and J. W. Sedat. Three-dimensional resolution doubling in wide-field fluorescence microscopy by structured illumination. *Biophysical Journal*, 94(12):4957–4970, 2008.

D. H. Hall and R. L. Russell. The posterior nervous system of the nematode caenorhabditis elegans: serial reconstruction of identified neurons and complete pattern of synaptic interactions. *The Journal of Neuroscience*, 11 (1):1–22, 1991.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1), 2009.

T. Hanaichi, T. Sato, M. Hoshin, and N. Mizuno. A stable lead stain by modification of Sato's method. In *Proc. XIth International Congress on Electron Microscopy*, pages 2181–2182, 1986.

C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

V. Hartenstein. *Atlas of* Drosophila *Development*. Cold Spring Harbor Laboratory Press, 1993.

V. Hartenstein, S. Spindler, W. Pereanu, and S. Fung. The development of the *Drosophila* larval brain. In G. M. Technau, editor, *Brain Development in* Drosophila melanogaster, volume 628 of *Advances in Experimental Medicine and Biology*, pages 1–31. Springer New York, 2008.

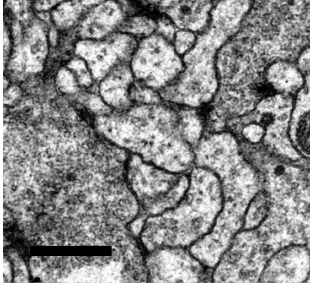
K. J. Hayworth, N. Kasthuri, R. Schalek, and J. W. Lichtman. Automating the collection of ultrathin serial sections for large volume TEM reconstructions. *Microscopy and Microanalysis*, 12(Suppl. 02):86–87, 2006.

S. W. Hell and J. Wichmann. Breaking the diffraction resolution limit by stimulated emission: stimulated-emission-depletion fluorescence microscopy. *Optics Letters*, 19(11):780–782, 1994.

J. A. W. Heymann, M. Hayles, I. Gestmann, L. A. Giannuzzi, B. Lich, and S. Subramaniama. Site-specific 3D imaging of cells and tissues with a dual beam microscope. *Journal of Structural Biology*, 155(1):63–73, 2006.

L. S. Hibbard and R. A. Hawkins. Objective image alignment for three-dimensional reconstruction of digital autoradiograms. *Journal of Neuroscience Methods*, 26(1):55–74, 1988.

W. Hoppe. Towards three-dimensional "electron microscopy" at atomic resolution. *Naturwissenschaften*, 61(6):239–249, 1974.



B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987.

B. Huang, W. Wang, M. Bates, and X. Zhuang. Three-dimensional super-resolution imaging by stochastic optical reconstruction microscopy. *Science*, 319(5864):810–813, 2008.

P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

P. J. Huber. *Robust Statistics*. John Wiley & Sons. Inc., 1981.

C. L. Hughes and J. B. Thomas. A sensory feedback circuit coordinates muscle activity in *Drosophila*. *Molecular and Cellular Neuroscience*, 35(2): 383–396, 2007.

J. Huisken and D. Y. R. Stainier. Selective plane illumination microscopy techniques in developmental biology. *Development*, 136(12):1963–1975, 2009.

J. Huisken, J. Swoger, F. Del Bene, J. Wittbrodt, and E. H. K. Stelzer. Optical sectioning deep inside live embryos by Selective Plane Illumination Microscopy. *Science*, 305(5686):1007–1009, 2004.

L. Ibáñez, W. Schroeder, L. Ng, J. Cates, and the *Insight Software Consortium*. *The ITK Software Guide*. Kitware, Inc., 2003.

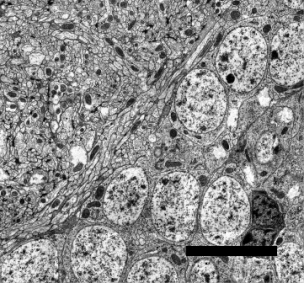
K. Ito, J. Urban, and G. M. Technau. Distribution, classification, and development of *Drosophila* glial cells in the late embryonic and early larval ventral nerve cord. *Rouxs Arch. Dev. Biol.*, 204(5):284–307, 1995.

T. Ju, J. Warren, J. Carson, M. Bello, I. Kakadiaris, Wah Chiu, C. Thaller, and G. Eichele. 3d volume reconstruction of a mouse brain from histological sections using warp filtering. *Journal of Neuroscience Methods*, 156(1–2): 84–100, 2006.

T. Kadir, A. Zisserman, and M. Brady. An affine invariant salient region detector. In *Proceedings of the 8th European Conference on Computer Vision*, pages 404–416, 2004.

A. T. Kalinka, K. M. Varga, D. T. Gerrard, S. Preibisch, D. L. Corcoran, J. Jarrells, U. Ohler, C. M. Bergman, and P. Tomančák. Gene expression divergence recapitulates the developmental hourglass model. *Nature*, 468 (7325):811–814, 2010.

L. Kamentsky, T. R. Jones, A. Fraser, M.-A. Bray, D. J. Logan, K. L. Madden, V. Ljosa, C. Rueden, K. W. Eliceiri, and A. E. Carpenter. Improved



structure, function and compatibility for CellProfiler: modular high-throughput image analysis software. *Bioinformatics*, 27(8):1179–1180, 2011.

V. Kaynig, B. Fischer, and J. M. Buhmann. Probabilistic image registration and anomaly detection by nonlinear warping. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

V. Kaynig, B. Fischer, E. Müller, and J. M. Buhmann. Fully automatic stitching and distortion correction of transmission electron microscope images. *Journal of Structural Biology*, 171(2):163–173, 2010a.

V. Kaynig, T. Fuchs, and J. M. Buhmann. Neuron geometry extraction by perceptual grouping in ssTEM images. In *Conference on Computer Vision and Pattern Recognition*, pages 2902–2909, 2010b.

P. J. Keller, A. D. Schmidt, J. Wittbrodt, and E. H. Stelzer. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science*, 322(5904):1065–1069, 2008.

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

M. Knoll and E. Ruska. Das Elektronenmikroskop. *Zeitschrift für Physik*, 78 (5):318–339, 1932.

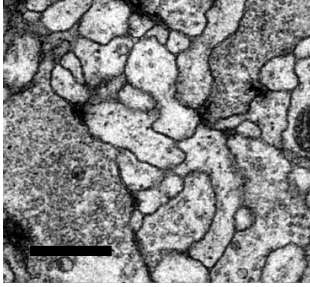
G. Knott, H. Marchman, D. Wall, and B. Lich. Serial section scanning electron microscopy of adult brain tissue using focused ion beam milling. *The Journal of Neuroscience*, 28(12):2959–2964, 2008.

S. Knowles-Barley, N. J. Butcher, I. A. Meinertzhagen, and J. D. Armstrong. Biologically inspired EM image alignment and neural reconstruction. *Bioinformatics*, 27(16):2216–2223, 2011.

P. A. Koshevoy, T. Tasdizen, and R. T. Whitaker. Implementation of an automatic slice-to-slice registration tool. SCI Institute Technical Report UUSCI-2006-018, University of Utah, 2006.

P. A. Koshevoy, T. Tasdizen, and R. T. Whitaker. Automatic assembly of tem mosaics and mosaic stacks using phase correlation. SCI Institute Technical Report UUSCI-2007-004, University of Utah, 2007.

U. Köthe. STL-style generic programming with images. *C++ Report Magazine*, 12(1):24–30, 2000.



J. R. Kremer, D. N. Mastronarde, and J. R. McIntosh. Computer visualization of three-dimensional image data using IMOD. *Journal of Structural Biology*, 116:71–76, 1996.

U. Krzic, S. Gunther, T. E. Saunders, S. J. Streichan, and L. Hufnagel. Multiview light-sheet microscope for rapid in toto imaging. *Nature Methods*, 9(7):730–733, 2012.

C. D. Kuglin and D. C. Hines. The phase correlation image alignment method. In *IEEE Conference on Cybernetics and Society*, pages 163–165, 1975.

Y. Lamdan and H. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision*, pages 238–249, 1988.

M. Landgraf, T. Bossing, G. M. Technau, and M. Bate. The origin, location, and projections of the embryonic abdominal motorneurons of *Drosophila*. *The Journal of Neuroscience*, 17(24):9642–9655, 1997.

M. Landgraf, N. Sánchez-Soriano, G. M. Technau, J. Urban, and A. Prokop. Charting the *Drosophila* neuropile: a strategy for the standardised characterisation of genetically amenable neurites. *Developmental Biology*, 260 (1):207–225, 2003.

S. Lang, V. J. Dercksen, B. Sakmann, and M. Oberlaender. Simulation of signal flow in 3D reconstructions of an anatomically realistic neural network in rat vibrissal cortex. *Neural Networks*, 24(9):998–1011, 2011.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

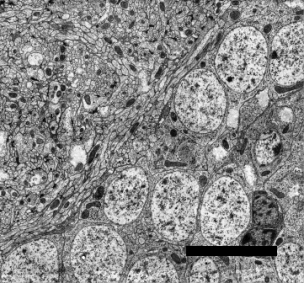
D. Levin. The approximation power of moving least-squares. *Mathematics of Computation*, 67(224):1517–1531, 1998.

J. P. Lewis. Fast template matching. In *Vision Interface*, volume 95, pages 120–123, 1995.

J. W. Lichtman and J. R. Sanes. Ome sweet ome: what can the genome tell us about the connectome? *Current Opinion in Neurobiology*, 18(3):346–353, 2008.

T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998. ISSN 0920-5691.

M. Linkert, C. T. Rueden, C. Allan, J.-M. Burel, W. Moore, A. Patterson, B. Loranger, J. Moore, C. Neves, D. MacDonald, A. Tarkowska, C. Sticco,



E. Hill, M. Rossner, K. W. Eliceiri, and J. R. Swedlow. Metadata matters: access to image data in the real world. *The Journal of Cell Biology*, 189(5): 777–782, 2010.

J. Livet, T. A. Weissman, H. Kang, R. W. Draft, J. Lu, R. A. Bennis, J. R. Sanes, and J. W. Lichtman. Transgenic strategies for combinatorial expression of fluorescent proteins in the nervous system. *Nature*, 450:56–62, 2007.

F. Long, H. Peng, X. Liu, S. K. Kim, and E. Myers. A 3D digital atlas of *C. elegans* and its application to single-cell analyses. *Nature Methods*, 6: 667–672, 2009.

M. Longair, S. Gerhard, C. Schneider-Mizell, S. Saalfeld, and A. Cardona. Unpublished manuscript.

D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1150–1157, 1999.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

W. Lukosz and M. Marchand. Optischen Abbildung unter Überschreitung der beugungsbedingten Auflösungsgrenze. *Optica Acta: International Journal of Optics*, 10(3):241–255, 1963.

H. Markram. The Blue Brain Project. *Nature Reviews Neuroscience*, 7:153–160, 2006.

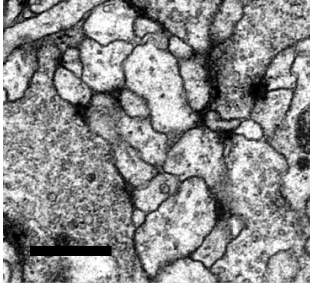
J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In P. L. Rosin and D. Marshall, editors, *Proceedings of the British Machine Vision Conference*, volume 1, pages 384–393, 2002.

W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, 1943.

P. McQuilton, S. E. St. Pierre, J. Thurmond, and the FlyBase Consortium. FlyBase 101 – the basics of navigating FlyBase. *Nucleic Acids Research*, 40 (D1):D706–D714, 2012.

K. D. Micheva and S. J. Smith. Array tomography: A new tool for imaging the molecular architecture and ultrastructure of neural circuits. *Neuron*, 55(1):25–36, 2007.

K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. In *International Conference on Computer Vision and Pattern Recognition*, volume 2, pages 257–263, June 2003.



K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, 2004.

M. Minsky. Microscopy apparatus. US patent 3.013.467, 1957.

Y. Mishchenko. Automation of 3D reconstruction of neural tissue from large volume of conventional serial section transmission electron micrographs. *Journal of Neuroscience Methods*, 176(2):276–289, 2009.

A. F. Möbius. *Der barycentrische Calcul ein neues Hüfsmittel zur analytischen Behandlung der Geometrie*. Verlag von Johann Ambrosius Barth, Leipzig, 1827.

J. Modersitzki. *FAIR: Flexible Algorithms for Image Registration*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009.

A. Nern, B. D. Pfeiffer, K. Svoboda, and G. M. Rubin. Multiple new site-specific recombinases for use in manipulating animal genomes. *Proceedings of the National Academy of Sciences*, 108(34):14198–14203, 2011.

S. Ourselin, A. Roche, G. Subsol, X. Pennec, and N. Ayache. Reconstructing a 3D structure from serial histological sections. *Image and Vision Computing*, 19(1–2):25–31, 2001.

G. H. Patterson. Fluorescence microscopy below the diffraction limit. *Semin Cell Dev Biol*, 20(8):886–893, 2009.

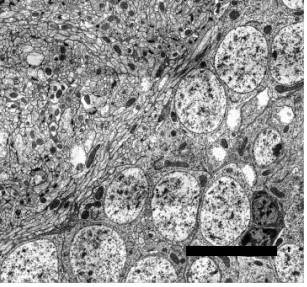
S. R. P. Pavani, M. A. Thompson, J. S. Biteen, S. J. Lord, N. Liu, R. J. Twieg, R. Piestun, and W. E. Moerner. Three-dimensional, single-molecule fluorescence imaging beyond the diffraction limit by using a double-helix point spread function. *Proceedings of the National Academy of Sciences*, 106 (9):2995–2999, 2009.

B. D. Pfeiffer, A. Jenett, A. S. Hammonds, T.-T. B. Ngo, S. Misra, C. Murphy, A. Scully, J. W. Carlson, K. H. Wan, T. R. Laverty, C. Mungall, R. Svirskas, J. T. Kadonaga, C. Q. Doe, M. B. Eisen, S. E. Celniker, and G. M. Rubin. Tools for neuroanatomy and neurogenetics in *Drosophila*. *Proceedings of the National Academy of Sciences*, 105(28):9715–9720, 2008.

B. D. Pfeiffer, J. W. Truman, and G. M. Rubin. Using translational enhancers to increase transgene expression in *Drosophila*. *Proceedings of the National Academy of Sciences*, 109(17):6626–6631, 2012.

T. Pietzsch. *Towards Dense Visual SLAM*. PhD thesis, Technische Universität Dresden, 2011.

T. Pietzsch, S. Preibisch, P. Tomančák, and S. Saalfeld. ImgLib2 – generic image processing in Java. Manuscript submitted for publication, 2012.



E. D. Pisano, S. Zong, B. M. Hemminger, M. DeLuca, R. E. Johnston, K. Muller, M. P. Braeuning, and S. M. Pizer. Contrast limited adaptive histogram equalization image processing to improve the detection of simulated spiculations in dense mammograms. *Journal of Digital Imaging*, 11 (4):193–200, 1998.

S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld. Adaptive histogram equalization and its variations. *Computer Vision, Graphics, and Image Processing*, 39(3):355–368, 1987.

S. Preibisch. *Reconstruction of Multi-View Microscopic Acquisitions*. PhD thesis, Technische Universität Dresden, 2010.

S. Preibisch, T. Rohlfing, M. Hasak, and P. Tomančák. Mosaicing of single plane illumination microscopy images using groupwise registration and fast content-based image fusion. In *Proceedings of SPIE*, 2008.

S. Preibisch, S. Saalfeld, T. Rohlfing, and P. Tomančák. Bead-based mosaicing of single plane illumination microscopy images using geometric local descriptor matching. In J. P. W. Pluim and B. M. Dawant, editors, *Proceedings of SPIE*, volume 7259, 2009a.

S. Preibisch, S. Saalfeld, and P. Tomančák. Globally optimal stitching of tiled 3d microscopic image acquisitions. *Bioinformatics*, 25(11):1463–1465, 2009b.

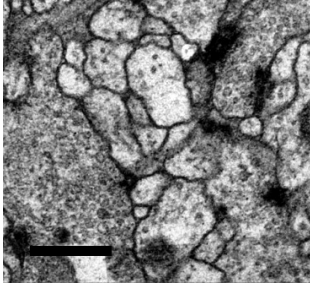
S. Preibisch, S. Saalfeld, J. Schindelin, and P. Tomančák. Software for bead-based registration of selective plane illumination microscopy data. *Nature Methods*, 7:418–419, 2010a.

S. Preibisch, P. Tomančák, and S. Saalfeld. Into ImgLib—generic image processing in Java. In *Proceedings of the ImageJ User and Developer Conference*, 2010b.

S. Ramón y Cajal. *Revista trimestral de Histología normal y patológica*. 1891.

G. Randall, A. Fernández, O. Trujillo-Cenoz, G. Apelbaum, M. Bertalmío, L. Vázquez, F. Malmierca, and P. Morelli. Neuro3d: an interactive 3D reconstruction system of serial sections using automatic registration. In C. J. Cogswell, J.-A. Conchello, T. Wilson, T. T. Lu, and J. M. Lerner, editors, *Proceedings of SPIE*, volume 3261, pages 117–126, 1998.

W. Rasband. ImageJ: Image processing and analysis in Java [version 1.46p], 1997–2012. URL `http://rsb.info.nih.gov/ij/`. Accessed June 19, 2012.



J.-P. Revel and M. J. Karnovsky. Hexagonal array of subunits in intercellular junctions of the mouse heart and liver. *The Journal of Cell Biology*, 33(3): C7–C12, 1967.

E. S. Reynolds. The use of lead citrate at high pH as an electron-opaque stain in electron microscopy. *The Journal of Cell Biology*, 17:208–212, 1963.

M. Rivera-Alba, S. N. Vitaladevuni, Y. Mishchenko, Z. Lu, S. Takemura, L. Scheffer, I. A. Meinertzhagen, D. B. Chklovskii, and G. G. de Polavieja. Wiring economy and volume exclusion determine neuronal placement in the *Drosophila* brain. *Current Biology*, 21:2000–2005, 2011.

J. D. Robertson. Membrane structure. *The Journal of Cell Biology*, 91(3):189s–204s, 1981.

T. Rohlfing. Image similarity and tissue overlaps as surrogates for image registration accuracy: Widely used but unreliable. *IEEE Transactions on Medical Imaging*, 31(2):153–163, 2012.

C. Rueden, G. Harris, B. DeZonia, A. Grislis, R. Lentz, L. Kamentsky, A. Fraser, J. Schindelin, A. Cardona, S. Saalfeld, S. Preibisch, J.-Y. Tinevez, P. Tomančák, A. Carpenter, R. Oldenbourg, and K. Eliceiri. Imagejdev: Next generation ImageJ. In *Proceedings of the ImageJ User and Developer Conference*, 2010.

S. Saalfeld. Automatic inter-slice registration of tiled Transmission Electron Microscopy (TEM) images. Diploma thesis, Technische Universität Dresden, 2008.

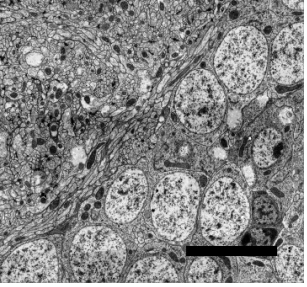
S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. CATMAID: Collaborative annotation toolkit for massive amounts of image data. *Bioinformatics*, 25(15):1984–1986, 2009.

S. Saalfeld, A. Cardona, V. Hartenstein, and P. Tomančák. As-rigid-as-possible mosaicking and serial section registration of large sstem datasets. *Bioinformatics*, 26(12):i57–i63, 2010.

S. Saalfeld, R. Fetter, A. Cardona, and P. Tomančák. Elastic volume reconstruction from series of ultra-thin microscopy sections. *Nature Methods*, 9 (7):717–720, 2012.

T. Sato. A modified method for lead staining of thin sections. *Journal of Electron Microscopy*, 17(2):158–159, 1968.

S. Schaefer, T. McPhail, and J. Warren. Image deformation using moving least squares. *ACM Transactions on Graphics*, 25(3):533–540, 2006.



J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak, and A. Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9: 676–682, 2012.

A. Schmid, A. Chiba, and C. Q. Doe. Clonal analysis of *Drosophila* embryonic neuroblasts: neural cell types, axon projections and muscle targets. *Development*, 126(21):4653–4689, 1999.

O. Schmitt, J. Modersitzki, S. Heldmann, S. Wirtz, and B. Fischer. Image registration of sectioned brains. *International Journal of Computer Vision*, 73(1):5–39, 2007.

I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. Part A: On the problem of smoothing of graduation. a first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, 4:45–99, 1946.

C. S. Sherrington. *The Integrative Action of the Nervous System*. Yale University Press, New Haven, 1906.

G. Shtengel, J. A. Galbraith, C. G. Galbraith, J. Lippincott-Schwartz, J. M. Gillette, S. Manley, R. Sougrat, C. M. Waterman, P. Kanchanawong, M. W. Davidson, R. D. Fetter, and H. F. Hess. Interferometric fluorescent superresolution microscopy resolves 3d cellular ultrastructure. *Proceedings of the National Academy of Sciences*, 106(9):3125–3130, 2009.

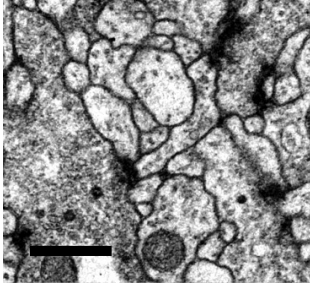
F. S. Sjöstrand. Ultrastructure of retinal rod synapses of the guinea pig eye as revealed by three-dimensional reconstructions from serial sections. *Journal of Ultrastructure Research*, 2(1):122–170, 1958.

O. Sporns, G. Tononi, and R. Kötter. The human connectome: A structural description of the human brain. *PLoS Computational Biology*, 1(4): e42, 2005.

C. Suloway, J. Pulokas, D. Fellmann, et al. Automated molecular microscopy: The new Leginon system. *Journal of Structural Biology*, 151(1): 41–60, 2005.

E. Tapia. A note on the computation of high-dimensional integral images. *Pattern Recognition Letters*, 32(2):197–201, 2011.

T. Tasdizen, P. Koshevoy, B. C. Grimm, J. R. Anderson, B. W. Jones, C. B. Watt, R. T. Whitaker, and R. E. Marc. Automatic mosaicking and volume assembly for high-throughput serial-section transmission electron microscopy. *Journal of Neuroscience Methods*, 193(1):132–144, 2010.



G. M. Technau and J. Urban. Einblicke in die frühe Entwicklung des Zentralen Nervensystems am *Drosophila*-Modell. *Forschungsmagazin der Johannes Gutenberg Universität Mainz*, 21:12–15, 2005.

P. Thévenaz, U. E. Ruttimann, and M. Unser. A pyramid approach to subpixel registration based on intensity. *IEEE Transactions on Image Processing*, 7(1):27–41, 1998.

P. Tomančák, A. Beaton, R. Weiszmann, E. Kwan, S. Shu, S. E. Lewis, S. Richards, M. Ashburner, V. Hartenstein, S. E. Celniker, and G. M. Rubin. Systematic determination of patterns of gene expression during *Drosophila* embryogenesis. *Genome Biology*, 3(12):research0088.1–0088.14, 2002.

R. Tomer, K. Khairy, F. Amat, and P. J. Keller. Quantitative high-speed imaging of entire developing embryos with simultaneous multiview light-sheet microscopy. *Nature Methods*, 9(7):755–763, 2012.

J. W. Truman and M. Bate. Spatial and temporal patterns of neurogenesis in the central nervous system of *Drosophila melanogaster*. *Developmental Biology*, 125(1):145–157, 1988.

J. W. Truman, H. Schuppe, D. Shepherd, and D. W. Williams. Developmental architecture of adult-specific lineages in the ventral CNS of *Drosophila*. *Development*, 131(20):5167–5184, 2004.

J. W. Truman, W. Moats, J. Altman, E. C. Marin, and D. W. Williams. Role of Notch signaling in establishing the hemilineages of secondary neurons in *Drosophila melanogaster*. *Development*, 137(1):53–61, 2010.

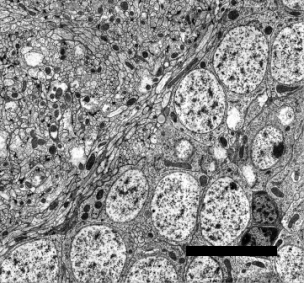
T. V. Truong, W. Supatto, D. S. Koos, J. M. Choi, and S. E. Fraser. Deep and fast live imaging with two-photon scanned light-sheet microscopy. *Nature Methods*, 8(9):757–760, 2011.

S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. Briggman, W. Denk, and H. S. Seung. Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22(2):511–538, 2010.

L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii. Structural properties of the *Caenorhabditis elegans* neuronal network. *PLoS Computational Biology*, 7(2):e1001066, 2011.

A. Verkhratsky and H. Kettenmann. Calcium signalling in glial cells. *Trends in Neurosciences*, 19(8):346–352, 1996.

P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.



P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, 24(2):137–154, 1997.

W. Wein, S. Brunke, A. Khamene, M. R. Callstrom, and N. Navab. Automatic CT-ultrasound registration for diagnostic imaging and image-guided intervention. *Medical Image Analysis*, 12(5):577–585, 2008.

J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system of the nematode Caenorhaditis elegans. *Philosophical Transactions of the Royal Society of London – Series B: Biological Sciences*, 314 (1165):1–340, 1986.

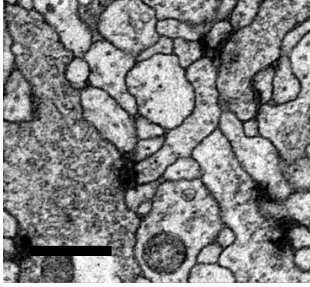
S. Wirtz, B. Fischer, J. Modersitzki, and O. Schmitt. Superfast elastic registration of histologic images of a whole rat brain for 3D reconstruction. In *Proceedings of SPIE*, volume 5370, 2004.

I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, second edition, 2005.

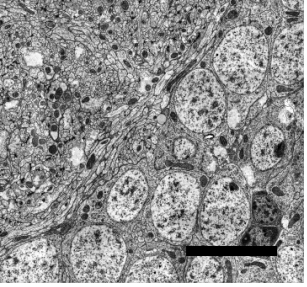
T. S. Yoo, M. J. Ackerman, W. E. Lorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxas, and R. Whitaker. Engineering and algorithm design for an image processing API: A technical report on ITK - the insight toolkit. In J. Westwood, editor, *Proceedings of Medicine Meets Virtual Reality*, pages 586–592, 2002.

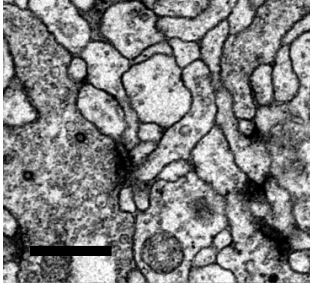
A. Younossi-Hartenstein, C. Nassif, P. Green, and V. Hartenstein. Early neurogenesis of the *Drosophila* brain. *The Journal of Comparative Neurology*, 370:313–329, 1996.

M. Zlatic, F. Li, M. Strigini, W. Grueber, and M. Bate. Positional cues in the *Drosophila* nerve cord: Semaphorins pattern the dorso-ventral axis. *PLoS Biology*, 7(6):e1000135, 2009.



Appendices





Source Code for Benchmarks

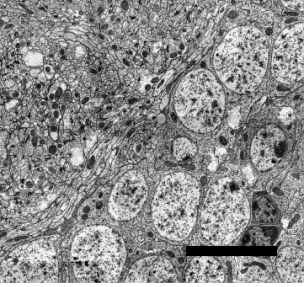
In the following we give the complete source code for the benchmarks presented in Section 3.2. We benchmarked two ImgLib2 image types against ImageJ's `ImagePlus` and primitive Java type arrays. Two sets of benchmarks were performed:

Invert image. The task is to invert the value of every pixel in an image. This does not require access to pixel coordinates, just iteration. This can be implemented concisely and efficiently for ImgLib2, ImageJ, as well as primitive arrays. For primitive arrays, the value type is fixed (we use `float` for the benchmark), whereas both the ImageJ and ImgLib2 implementations provide some degree of type independence. The ImageJ code supports `float` as well as 8 and 16 bit unsigned integer value types (by conversion to `float`). The ImgLib2 code supports all `NumericType` value types (these support the necessary arithmetic operations).

Compute center-of-mass of image. The task is to compute a weighted sum of all pixel coordinates in the image, where the weights are the pixel intensities. This requires access to pixel coordinates as well as values. This can be implemented concisely in a dimensionality-independent way for ImgLib2. However, for ImageJ and primitive arrays we have to implement special cases for each supported dimensionality. The primitive array implementation handles 1d to 6d images. ImageJ handles 1d to 5d images. The ImgLib2 code handles all dimensionalities. For primitive arrays, the value type is fixed (we use `byte` for the benchmark), whereas both the ImageJ and ImgLib2 implementations provide some degree of type independence. The ImageJ code supports `float` as well as 8 and 16 bit unsigned integer value types (by conversion to `float`). The ImgLib2 code supports all `RealType` value types (these support the necessary arithmetic operations).

Invert for ImgLib2

```
1  package benchmark;
2
3  import ...
6  public class InvertImgLib2Implementation {
7    /**
8     * generic implementation for all ImgLib2
9     * {@link IterableInterval}s.
10    */
11   public final static <T extends NumericType<T>>
```



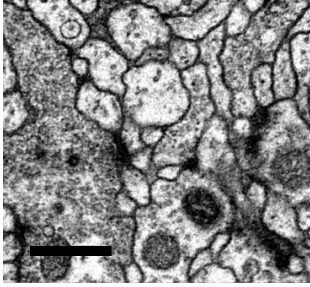

```
12        void invert(final IterableInterval<T> img) {
13      for (final T t : img)
14        t.mul(-1);
15    }
16  }
```

Invert for ImageJ's ImagePlus

```
1   package benchmark;
2
3   import ...

7   public class InvertImageJImplementation {
8     /**
9      * Invert implementation for ImagePlus. Pixel coordinates
10     * are not required so this works for 1d to 5d images.
11     */
12    public final static void invert(final ImagePlus imp) {
13      final ImageStack stack = imp.getStack();
14      final int numSlices = stack.getSize();
15      for (int s = 1; s <= numSlices; ++s) {
16        final ImageProcessor ip = stack.getProcessor(s);
17        final int size = ip.getPixelCount();
18        for (int i = 0; i < size; i++)
19          ip.setf(i, -ip.getf(i));
20      }
21    }
22  }
```

Invert for Java float[] Array

```
1   package benchmark;
2
3   public class InvertFloatNativeImplementation {
4     /**
5      * Invert implementation for native float[] array. Pixel
6      * coordinates are not required so this works for all
7      * dimensions.
8      */
9     public final static void invert(final float[] img) {
10      for (int i = 0; i < img.length; i++)
11        img[i] = -img[i];
```



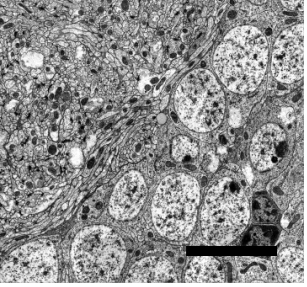

```
  }
}
```

Center-Of-Mass for ImgLib2

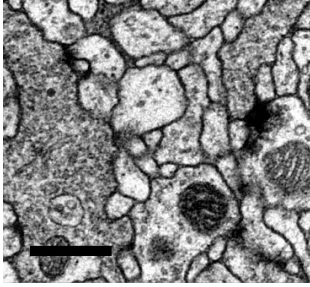
```
package benchmark;

import ...

public class CenterOfMassImgLib2Implementation {
  /**
   * generic implementation for all ImgLib2
   * {@link IterableInterval}s.
   */
  public final static <T extends RealType<T>> double[]
      findCenterOfMass(final IterableInterval<T> img) {
    final RealSum[] realSums = new RealSum[img.numDimensions()];
    for (int d = 0; d < realSums.length; ++d)
      realSums[d] = new RealSum();
    final RealSum s = new RealSum();
    final Cursor<T> cursor = img.localizingCursor();
    while (cursor.hasNext()) {
      final double w = cursor.next().getRealDouble();
      s.add(w);
      for (int d = 0; d < realSums.length; ++d)
        realSums[d].add(cursor.getDoublePosition(d) * w);
    }

    final double[] centerOfMass = new double[realSums.length];
    final double sum = s.getSum();
    for (int d = 0; d < realSums.length; ++d)
      centerOfMass[d] = realSums[d].getSum() / sum;

    return centerOfMass;
  }
}
```

Center-Of-Mass for ImageJ's ImagePlus

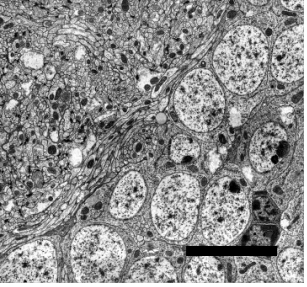
```
package benchmark;

import ...
```



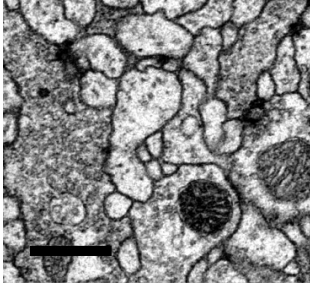

```
8  public class CenterOfMassImageJImplementation {
9    /**
10    * special purpose implementation for 1d
11    * {@link ImagePlus}.
12    */
13   private final static double[] findCenterOfMass1D(
14       final ImagePlus imp) {
15     final RealSum centerOfMass0 = new RealSum(),
16                   sum = new RealSum();
17
18     final ImageProcessor ip = imp.getProcessor();
19     final int size0 = ip.getPixelCount();
20
21     for (int d0 = 0; d0 < size0; ++d0) {
22       final double value = ip.getf(d0);
23       centerOfMass0.add(value * d0);
24       sum.add(value);
25     }
26     final double s = sum.getSum();
27
28     return new double[]{centerOfMass0.getSum() / s};
29   }
30
31   /**
32    * special purpose implementation for 2d
33    * {@link ImagePlus}.
34    */
35   private final static double[] findCenterOfMass2D(
36       final ImagePlus imp) {
37     final RealSum centerOfMass0 = new RealSum(),
38                   centerOfMass1 = new RealSum(),
39                   sum = new RealSum();
40
41     final ImageProcessor ip = imp.getProcessor();
42     final int size0 = ip.getWidth();
43     final int size1 = ip.getHeight();
44
45     int i = 0;
46     for (int d1 = 0; d1 < size1; ++d1)
47       for (int d0 = 0; d0 < size0; ++d0) {
48         final double value = ip.getf(i++);
49         centerOfMass0.add(value * d0);
50         centerOfMass1.add(value * d1);
```



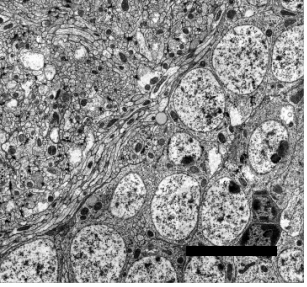
```
51          sum.add(value);
52        }
53      final double s = sum.getSum();
54
55      return new double[]{centerOfMass0.getSum() / s,
56                          centerOfMass1.getSum() / s};
57    }
58
59    /**
60     * special purpose implementation for 3d
61     * {@link ImagePlus}.
62     */
63    private final static double[] findCenterOfMass3D(
64        final ImagePlus imp) {
65      final RealSum centerOfMass0 = new RealSum(),
66                    centerOfMass1 = new RealSum(),
67                    centerOfMass2 = new RealSum(),
68                    sum = new RealSum();
69
70      final int size0 = imp.getWidth();
71      final int size1 = imp.getHeight();
72      final int size2 = imp.getNChannels();
73
74      final ImageStack stack = imp.getStack();
75      for (int d2 = 0; d2 < size2; ++d2) {
76        final ImageProcessor ip = stack.getProcessor(
77            imp.getStackIndex(d2 + 1, 1, 1));
78        int i = 0;
79        for (int d1 = 0; d1 < size1; ++d1)
80          for (int d0 = 0; d0 < size0; ++d0) {
81            final double value = ip.getf(i++);
82            centerOfMass0.add(value * d0);
83            centerOfMass1.add(value * d1);
84            centerOfMass2.add(value * d2);
85            sum.add(value);
86          }
87      }
88      final double s = sum.getSum();
89
90      return new double[]{centerOfMass0.getSum() / s,
91                          centerOfMass1.getSum() / s,
92                          centerOfMass2.getSum() / s};
93    }
```




```
94
95    /**
96     * special purpose implementation for 4d
97     * {@link ImagePlus}.
98     */
99    private final static double[] findCenterOfMass4D(
100       final ImagePlus imp) {
101     final RealSum centerOfMass0 = new RealSum(),
102                   centerOfMass1 = new RealSum(),
103                   centerOfMass2 = new RealSum(),
104                   centerOfMass3 = new RealSum(),
105                   sum = new RealSum();
106
107     final int size0 = imp.getWidth();
108     final int size1 = imp.getHeight();
109     final int size2 = imp.getNChannels();
110     final int size3 = imp.getNSlices();
111
112     final ImageStack stack = imp.getStack();
113     for (int d3 = 0; d3 < size3; ++d3) {
114       for (int d2 = 0; d2 < size2; ++d2) {
115         final ImageProcessor ip = stack.getProcessor(
116             imp.getStackIndex(d2 + 1, d3 + 1, 1));
117         int i = 0;
118         for (int d1 = 0; d1 < size1; ++d1)
119           for (int d0 = 0; d0 < size0; ++d0) {
120             final double value = ip.getf(i++);
121             centerOfMass0.add(value * d0);
122             centerOfMass1.add(value * d1);
123             centerOfMass2.add(value * d2);
124             centerOfMass3.add(value * d3);
125             sum.add(value);
126           }
127       }
128     }
129     final double s = sum.getSum();
130
131     return new double[]{centerOfMass0.getSum() / s,
132                         centerOfMass1.getSum() / s,
133                         centerOfMass2.getSum() / s,
134                         centerOfMass3.getSum() / s};
135   }
136
```



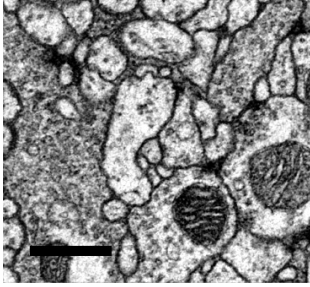
```
137     /**
138      * special purpose implementation for 5d
139      * {@link ImagePlus}.
140      */
141     private final static double[] findCenterOfMass5D(
142         final ImagePlus imp) {
143       final RealSum centerOfMass0 = new RealSum(),
144                     centerOfMass1 = new RealSum(),
145                     centerOfMass2 = new RealSum(),
146                     centerOfMass3 = new RealSum(),
147                     centerOfMass4 = new RealSum(),
148                     sum = new RealSum();
149
150       final int size0 = imp.getWidth();
151       final int size1 = imp.getHeight();
152       final int size2 = imp.getNChannels();
153       final int size3 = imp.getNSlices();
154       final int size4 = imp.getNFrames();
155
156       final ImageStack stack = imp.getStack();
157       for (int d4 = 0; d4 < size4; ++d4) {
158         for (int d3 = 0; d3 < size3; ++d3) {
159           for (int d2 = 0; d2 < size2; ++d2) {
160             final ImageProcessor ip = stack.getProcessor(
161                 imp.getStackIndex(d2 + 1, d3 + 1, d4 + 1));
162             int i = 0;
163             for (int d1 = 0; d1 < size1; ++d1)
164               for (int d0 = 0; d0 < size0; ++d0) {
165                 final double value = ip.getf(i++);
166                 centerOfMass0.add(value * d0);
167                 centerOfMass1.add(value * d1);
168                 centerOfMass2.add(value * d2);
169                 centerOfMass3.add(value * d3);
170                 centerOfMass4.add(value * d4);
171                 sum.add(value);
172               }
173           }
174         }
175       }
176       final double s = sum.getSum();
177
178       return new double[]{centerOfMass0.getSum() / s,
179                           centerOfMass1.getSum() / s,
```



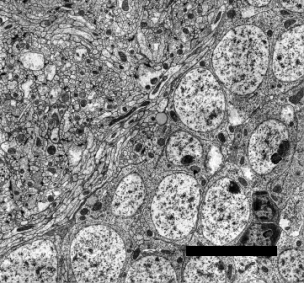

```
                        centerOfMass2.getSum() / s,
                        centerOfMass3.getSum() / s,
                        centerOfMass4.getSum() / s};
  }

  /**
   * special purpose implementation for 1d to 5d ImagePlus
   * dispatches to the 1d to 5d versions.
   *
   * @param numDimensions
   *          number of dimensions, because it is impossible
   *          to retrieve this from the ImagePlus.
   */
  public final static double[] findCenterOfMass(
      final ImagePlus imp, final int numDimensions) {
    if (numDimensions == 1)
      return findCenterOfMass1D(imp);
    else if (numDimensions == 2)
      return findCenterOfMass2D(imp);
    else if (numDimensions == 3)
      return findCenterOfMass3D(imp);
    else if (numDimensions == 4)
      return findCenterOfMass4D(imp);
    else if (numDimensions == 5)
      return findCenterOfMass5D(imp);
    else
      throw new IllegalArgumentException(
          "only 1D to 5D images are supported");
  }
}
```

Center-Of-Mass for Java byte[] Array

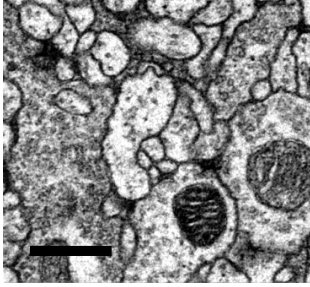
```
package benchmark;

import net.imglib2.util.RealSum;

public class CenterOfMassByteNativeImplementation {
  /**
   * special purpose implementation for primitive
   * byte[] array 1d.
   */
  private final static double[] findCenterOfMass(
```



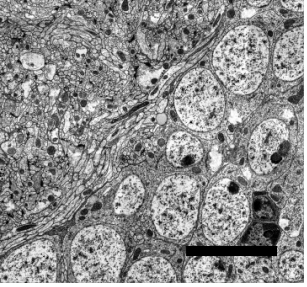

```
11        final byte[] img, final int size0) {
12      final RealSum centerOfMass0 = new RealSum(),
13                    sum = new RealSum();
14
15      for (int d0 = 0; d0 < size0; ++d0) {
16        final double value = img[d0] & 0xff;
17        centerOfMass0.add(value * d0);
18        sum.add(value);
19      }
20      final double s = sum.getSum();
21
22      return new double[]{centerOfMass0.getSum() / s};
23    }
24
25    /**
26     * special purpose implementation for primitive
27     * byte[] array 2d.
28     */
29    private final static double[] findCenterOfMass(
30        final byte[] img, final int size0, final int size1) {
31      final RealSum centerOfMass0 = new RealSum(),
32                    centerOfMass1 = new RealSum(),
33                    sum = new RealSum();
34
35      int i = 0;
36
37      for (int d1 = 0; d1 < size1; ++d1)
38        for (int d0 = 0; d0 < size0; ++d0) {
39          final double value = img[i++] & 0xff;
40          centerOfMass0.add(value * d0);
41          centerOfMass1.add(value * d1);
42          sum.add(value);
43        }
44      final double s = sum.getSum();
45
46      return new double[]{centerOfMass0.getSum() / s,
47                          centerOfMass1.getSum() / s};
48    }
49
50    /**
51     * special purpose implementation for primitive
52     * byte[] array 3d.
53     */
```



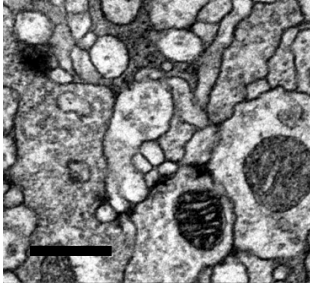

```
54    private final static double[] findCenterOfMass(
55        final byte[] img, final int size0, final int size1,
56        final int size2) {
57      final RealSum centerOfMass0 = new RealSum(),
58                    centerOfMass1 = new RealSum(),
59                    centerOfMass2 = new RealSum(),
60                    sum = new RealSum();
61
62      int i = 0;
63
64      for (int d2 = 0; d2 < size2; ++d2)
65        for (int d1 = 0; d1 < size1; ++d1)
66          for (int d0 = 0; d0 < size0; ++d0) {
67            final double value = img[i++] & 0xff;
68            centerOfMass0.add(value * d0);
69            centerOfMass1.add(value * d1);
70            centerOfMass2.add(value * d2);
71            sum.add(value);
72          }
73      final double s = sum.getSum();
74
75      return new double[]{centerOfMass0.getSum() / s,
76                          centerOfMass1.getSum() / s,
77                          centerOfMass2.getSum() / s};
78    }
79
80    /**
81     * special purpose implementation for primitive
82     * byte[] array 4d.
83     */
84    private final static double[] findCenterOfMass(
85        final byte[] img, final int size0, final int size1,
86        final int size2, final int size3) {
87      final RealSum centerOfMass0 = new RealSum(),
88                    centerOfMass1 = new RealSum(),
89                    centerOfMass2 = new RealSum(),
90                    centerOfMass3 = new RealSum(),
91                    sum = new RealSum();
92
93      int i = 0;
94
95      for (int d3 = 0; d3 < size3; ++d3)
96        for (int d2 = 0; d2 < size2; ++d2)
```



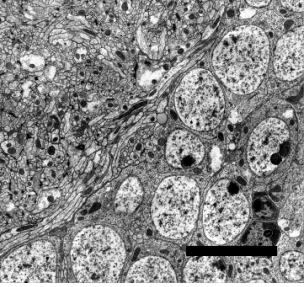

```
97          for (int d1 = 0; d1 < size1; ++d1)
98            for (int d0 = 0; d0 < size0; ++d0) {
99              final double value = img[i++] & 0xff;
100             centerOfMass0.add(value * d0);
101             centerOfMass1.add(value * d1);
102             centerOfMass2.add(value * d2);
103             centerOfMass3.add(value * d3);
104             sum.add(value);
105           }
106     final double s = sum.getSum();
107
108     return new double[]{centerOfMass0.getSum() / s,
109                         centerOfMass1.getSum() / s,
110                         centerOfMass2.getSum() / s,
111                         centerOfMass3.getSum() / s};
112   }
113
114   /**
115    * special purpose implementation for primitive
116    * byte[] array 5d.
117    */
118   private final static double[] findCenterOfMass(
119       final byte[] img,  final int size0, final int size1,
120       final int size2, final int size3, final int size4) {
121     final RealSum centerOfMass0 = new RealSum(),
122                   centerOfMass1 = new RealSum(),
123                   centerOfMass2 = new RealSum(),
124                   centerOfMass3 = new RealSum(),
125                   centerOfMass4 = new RealSum(),
126                   sum = new RealSum();
127
128     int i = 0;
129
130     for (int d4 = 0; d4 < size4; ++d4)
131       for (int d3 = 0; d3 < size3; ++d3)
132         for (int d2 = 0; d2 < size2; ++d2)
133           for (int d1 = 0; d1 < size1; ++d1)
134             for (int d0 = 0; d0 < size0; ++d0) {
135               final double value = img[i++] & 0xff;
136               centerOfMass0.add(value * d0);
137               centerOfMass1.add(value * d1);
138               centerOfMass2.add(value * d2);
139               centerOfMass3.add(value * d3);
```




```
140               centerOfMass4.add(value * d4);
141               sum.add(value);
142             }
143     final double s = sum.getSum();
144
145     return new double[]{centerOfMass0.getSum() / s,
146                         centerOfMass1.getSum() / s,
147                         centerOfMass2.getSum() / s,
148                         centerOfMass3.getSum() / s,
149                         centerOfMass4.getSum() / s};
150   }
151
152   /**
153    * special purpose implementation for primitive
154    * byte[] array 6d.
155    */
156   private final static double[] findCenterOfMass(
157       final byte[] img, final int size0, final int size1,
158       final int size2, final int size3, final int size4,
159       final int size5) {
160     final RealSum centerOfMass0 = new RealSum(),
161                   centerOfMass1 = new RealSum(),
162                   centerOfMass2 = new RealSum(),
163                   centerOfMass3 = new RealSum(),
164                   centerOfMass4 = new RealSum(),
165                   centerOfMass5 = new RealSum(),
166                   sum = new RealSum();
167
168     int i = 0;
169
170     for (int d5 = 0; d5 < size5; ++d5)
171       for (int d4 = 0; d4 < size4; ++d4)
172         for (int d3 = 0; d3 < size3; ++d3)
173           for (int d2 = 0; d2 < size2; ++d2)
174             for (int d1 = 0; d1 < size1; ++d1)
175               for (int d0 = 0; d0 < size0; ++d0) {
176                 final double value = img[i++] & 0xff;
177                 centerOfMass0.add(value * d0);
178                 centerOfMass1.add(value * d1);
179                 centerOfMass2.add(value * d2);
180                 centerOfMass3.add(value * d3);
181                 centerOfMass4.add(value * d4);
182                 centerOfMass5.add(value * d5);
```




```
183                 sum.add(value);
184               }
185     final double s = sum.getSum();
186
187     return new double[]{centerOfMass0.getSum() / s,
188                         centerOfMass1.getSum() / s,
189                         centerOfMass2.getSum() / s,
190                         centerOfMass3.getSum() / s,
191                         centerOfMass4.getSum() / s,
192                         centerOfMass5.getSum() / s};
193   }
194
195   /**
196    * special purpose implementation for primitive
197    * byte[] arrays 1D to 6D.
198    * dispatches to the 1d to 6d versions.
199    */
200   public final static double[] findCenterOfMass(
201       final byte[] img, final int[] dimensions) {
202     if (dimensions.length == 1)
203       return findCenterOfMass(img, dimensions[0]);
204     else if (dimensions.length == 2)
205       return findCenterOfMass(img, dimensions[0],
206           dimensions[1]);
207     else if (dimensions.length == 3)
208       return findCenterOfMass(img, dimensions[0],
209           dimensions[1], dimensions[2]);
210     else if (dimensions.length == 4)
211       return findCenterOfMass(img, dimensions[0],
212           dimensions[1], dimensions[2], dimensions[3]);
213     else if (dimensions.length == 5)
214       return findCenterOfMass(img, dimensions[0],
215           dimensions[1], dimensions[2], dimensions[3],
216           dimensions[4]);
217     else if (dimensions.length == 6)
218       return findCenterOfMass(img, dimensions[0],
219           dimensions[1], dimensions[2], dimensions[3],
220           dimensions[4], dimensions[5]);
221     else
222       throw new IllegalArgumentException(
223           "only 1D to 6D images are supported");
224   }
225 }
```



Statements for the Opening of Doctorate Proceedings

1. I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such.

2. I received assistance from the following persons in conjunction with the selection and evaluation of materials and creation of the manuscript: Stephan Preibisch, Tobias Pietzsch. This interdisciplinary project was in parts a collaboration with biologists and computer scientists with the following contributions:

   (a) The TEM section series were prepared and recorded by Albert Cardona und Richard Fetter. The array tomograhy series was prepared and recorded by Forrest Collman, Nick Weiler, Kristina Micheva und Stephen Smith.

   (b) Albert Cardona generated the skeleton traces of neuronal arbors that I used for evaluation of my method. He suggested and implemented the *lower bound length* as a measure for further comparison.

   (c) The method to match and register randomized point clouds is a collaboration with Stephan Preibisch and described in detail in his PhD thesis. I have designed and implemented the methods for robust model estimation and optimization. We have together designed and implemented the method for point detection, and we have together conceived the idea to match distance-sorted local point clouds by least-squares fit with respect to an invariance model. All other work has been performed by Stephan Preibisch.

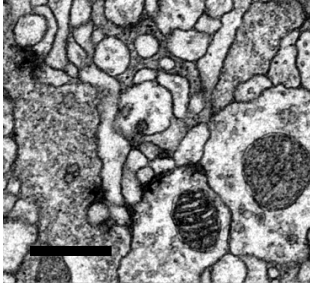
   (d) The library ImgLib2 is a collaboration with equal contribution together with Stephan Preibisch and Tobias Pietzsch.

3. No further persons were involved in the intellectual creation of the presented work. I have in particular not taken recourse to the assistance of a commercial doctorate advisor. No third parties have received remuneration or payment in kind from me, neither directly nor indirectly, for work in connection with the contents of the presented thesis.

4. This paper has not previously been presented in identical or similar form to any other German or foreign examination board and has not yet been published.

5. I confirm that I accept the applicable Doctorate Regulations of the Faculty of Computer Science of the Dresden University of Technology.

Dresden, August 6, 2013



Erklärungen zur Eröffnung des Promotionsverfahrens

1. Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

2. Bei der Auswahl und Auswertung des Materials sowie bei der Herstellung des Manuskripts habe ich Unterstützungsleistungen von folgenden Personen erhalten: Stephan Preibisch, Tobias Pietzsch. Diese interdisziplinäre Arbeit ist in Teilen eine Kooperation mit Biologen und Informatikern, die hier im Detail aufgezählt sind:

   (a) Die TEM-Schnittserien wurden von Albert Cardona und Richard Fetter präpariert und aufgenommen. Die Array-Tomography-Serie wurde von Forrest Collman, Nick Weiler, Kristina Micheva und Stephen Smith präpariert und aufgenommen.

   (b) Albert Cardona hat die Skelett-Rekonstruktionen der Neuriten generiert, die ich für die Qualitätsbewertung meiner Methode benutzt habe. Er hat außerdem die *lower bound length* als Vergleichsmaß vorgeschlagen und implementiert.

   (c) Die Methode zur Registrierung von randomisierten Punktwolken ist eine Kooperation mit Stephan Preibisch und in dessen Dissertation detailiert beschrieben. Ich habe die Methoden zur robusten Modellbestimung und Optimierung entwickelt und implementiert. Gemeinsam haben wir den Punktdetektor entwickelt und die Idee, nach Abstand sortierte Punktwolken als Deskriptoren zu verwenden und mittels Least-Squares-Fits bezüglich eines Invarianz-Modells zu vergleichen. Alle weiteren Arbeiten in diesem Projekt hat Stephan Preibisch entwickelt und implementiert.

   (d) Die Bildverarbeitungsbibliothek ImgLib2 ist eine Kollaboration zu gleichen Teilen mit Stephan Preibisch und Tobias Pietzsch.

3. Weitere Personen waren an der geistigen Herstellung der vorliegenden Arbeit nicht beteiligt. lnsbesondere habe ich nicht die Hilfe eines kommerziellen Promotionsberaters in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

4. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt und ist auch noch nicht veröffentlicht worden.

5. Ich bestätige, dass ich die geltende Promotionsordnung der Fakultät Informatik der Technischen Universität Dresden anerkenne.

Dresden, den 6. August 2013