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 Zlatić, 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 Zlatić 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

**Contents**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>xi</td>
</tr>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Electron Microscopy</td>
<td>3</td>
</tr>
<tr>
<td>1.2 The fruitfly <em>Drosophila melanogaster</em></td>
<td>7</td>
</tr>
<tr>
<td>2 Registration and Alignment</td>
<td>9</td>
</tr>
<tr>
<td>2.1 Serial Section Microscopy</td>
<td>10</td>
</tr>
<tr>
<td>2.2 Mosaics</td>
<td>12</td>
</tr>
<tr>
<td>2.3 Related Work</td>
<td>13</td>
</tr>
<tr>
<td>2.4 The Transformation Model</td>
<td>19</td>
</tr>
<tr>
<td>2.5 Landmark-Based Alignment and Montaging</td>
<td>20</td>
</tr>
<tr>
<td>2.5.1 Scale Invariant Feature Transform (SIFT)</td>
<td>21</td>
</tr>
<tr>
<td>2.5.2 A Robust Estimator</td>
<td>25</td>
</tr>
<tr>
<td>2.5.3 Affine Transformation Models</td>
<td>29</td>
</tr>
<tr>
<td>2.5.4 Global Solution</td>
<td>30</td>
</tr>
<tr>
<td>2.5.5 Regularization</td>
<td>32</td>
</tr>
<tr>
<td>2.5.6 Correct lens distortion</td>
<td>33</td>
</tr>
<tr>
<td>2.5.7 Warping with Moving Least-Squares</td>
<td>34</td>
</tr>
<tr>
<td>2.6 Elastic Alignment and Montaging</td>
<td>35</td>
</tr>
<tr>
<td>2.6.1 The Elastic Model</td>
<td>36</td>
</tr>
<tr>
<td>2.6.2 Block Matching</td>
<td>39</td>
</tr>
<tr>
<td>2.7 Results</td>
<td>46</td>
</tr>
<tr>
<td>2.8 Evaluation</td>
<td>52</td>
</tr>
<tr>
<td>2.8.1 Manual Skeleton Traces</td>
<td>53</td>
</tr>
<tr>
<td>2.8.2 artificially Generated Ground Truth</td>
<td>56</td>
</tr>
<tr>
<td>2.9 Local Contrast Enhancement</td>
<td>64</td>
</tr>
<tr>
<td>2.10 Matching and Registering Point Clouds</td>
<td>67</td>
</tr>
<tr>
<td>2.10.1 SPIM</td>
<td>67</td>
</tr>
<tr>
<td>2.10.2 Matching and Registration</td>
<td>67</td>
</tr>
<tr>
<td>2.11 Summary</td>
<td>69</td>
</tr>
</tbody>
</table>
## 3 Implementation

3.1 The mpicbg-library and TrakEM2 .......................... 75
   3.1.1 Coordinate Transformations ............................. 76
   3.1.2 Concatenating Coordinate Transformations ............. 78
   3.1.3 Triangular Meshes for Approximate Inversion of Non-Invertible Coordinate Transformation ........... 80
   3.1.4 Available Coordinate Transformations ................. 81
   3.1.5 Rendering Transformed 2d-Images ....................... 82
   3.1.6 Rendering Transformed 3d-Images ....................... 82
   3.1.7 Manual Deformation with Moving Least Squares ........ 83
   3.1.8 Transparency .......................................... 84
   3.1.9 Registration and Alignment ............................ 84
   3.1.10 Summary .............................................. 91

3.2 ImgLib2 ..................................................... 92
   3.2.1 Architecture ........................................... 92
   3.2.2 Implementation ........................................ 97
   3.2.3 Discussion ............................................ 99

## 4 Discussion ................................................. 101

## Appendices .................................................. 123
**List of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Main components of a Transmission Electron Microscope.</td>
<td>4</td>
</tr>
<tr>
<td>1.2</td>
<td>Electron microscopy examples.</td>
<td>5</td>
</tr>
<tr>
<td>1.3</td>
<td><em>Drosophila</em> embryogenesis.</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Typical ssTEM artifacts.</td>
<td>11</td>
</tr>
<tr>
<td>2.2</td>
<td>The Difference of Gaussian detector.</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>SIFT-feature correspondences.</td>
<td>24</td>
</tr>
<tr>
<td>2.4</td>
<td>Warping montages.</td>
<td>35</td>
</tr>
<tr>
<td>2.5</td>
<td>Aligning wider context in the series.</td>
<td>36</td>
</tr>
<tr>
<td>2.6</td>
<td>Spring connected particles.</td>
<td>37</td>
</tr>
<tr>
<td>2.7</td>
<td>Block matching approximately aligned images.</td>
<td>39</td>
</tr>
<tr>
<td>2.8</td>
<td>Local block matching filters.</td>
<td>40</td>
</tr>
<tr>
<td>2.9</td>
<td>The number of filtered block matches as a deformation invariant similarity metric.</td>
<td>41</td>
</tr>
<tr>
<td>2.10</td>
<td>Elastic montaging compensates for non-rigid deformation.</td>
<td>43</td>
</tr>
<tr>
<td>2.11</td>
<td>Reconstruction of two large TEM section series.</td>
<td>48</td>
</tr>
<tr>
<td>2.12</td>
<td>Anatomical context of the <em>Drosophila</em> series.</td>
<td>49</td>
</tr>
<tr>
<td>2.13</td>
<td>The effect of connecting larger neighborhoods.</td>
<td>51</td>
</tr>
<tr>
<td>2.14</td>
<td>Synapses detectable in axial views after elastic alignment.</td>
<td>53</td>
</tr>
<tr>
<td>2.15</td>
<td>Serial section alignment impacts reconstructed shapes.</td>
<td>54</td>
</tr>
<tr>
<td>2.16</td>
<td>Comparison of three alignment methods applied to the <em>Drosophila</em> series using the total length of manual skeleton annotation.</td>
<td>55</td>
</tr>
<tr>
<td>2.17</td>
<td>Total skeleton length and lower bound skeleton length</td>
<td>56</td>
</tr>
<tr>
<td>2.18</td>
<td>Artificial evaluation data.</td>
<td>57</td>
</tr>
<tr>
<td>2.19</td>
<td>Average section scale after alignment.</td>
<td>58</td>
</tr>
<tr>
<td>2.20</td>
<td>Histograms of absolute point displacements after alignment.</td>
<td>59</td>
</tr>
<tr>
<td>2.21</td>
<td>Projection of absolute point displacements after alignment.</td>
<td>60</td>
</tr>
<tr>
<td>2.22</td>
<td>Orthogonal projection of absolute point displacements after alignment.</td>
<td>61</td>
</tr>
<tr>
<td>2.23</td>
<td>Histograms of section-to-section pairwise point displacements after alignment.</td>
<td>62</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.24</td>
<td>Visualization of section-to-section pairwise point displacements after alignment</td>
<td>63</td>
</tr>
<tr>
<td>2.25</td>
<td>Local Contrast Enhancement</td>
<td>65</td>
</tr>
<tr>
<td>3.1</td>
<td>Simplified interface diagram</td>
<td>78</td>
</tr>
<tr>
<td>3.2</td>
<td>Approximate inversion through Triangular Meshes</td>
<td>80</td>
</tr>
<tr>
<td>3.3</td>
<td>YCbCr Overlay</td>
<td>90</td>
</tr>
<tr>
<td>3.4</td>
<td>Basic ImgLib2 collection interfaces and images</td>
<td>94</td>
</tr>
<tr>
<td>3.5</td>
<td>Interfaces for access to pixel values and coordinates</td>
<td>95</td>
</tr>
<tr>
<td>3.6</td>
<td>A fragment of the ImgLib2 pixel type hierarchy</td>
<td>96</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Overview of reconstructed ssTEM datasets .............. 47
3.1 Performance of per-pixel operations on ImgLib2 data struc-
tures ......................................................... 97
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 *Drosophila* 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](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-
processing (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 today’s 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 Project1 (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 Project2 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 Project3.

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://openconnectome.me/
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 $\lambda$ be the wavelength of the light source, $n$ the refractive index of the medium separating specimen and lenses, $\theta$ 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 $\Delta r$ is

$$
\Delta r = \frac{0.61 \lambda}{A_N} = \frac{0.61 \lambda}{n \sin \theta} \quad (1.1)
$$

It is possible to work around this physical limitation by embedding additional information into 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-
CHAPTER 1. INTRODUCTION

Electron cannon Condenser lens Grid with specimen Objective lens Projector lens Phosphor screen

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

\[ \lambda = \frac{h}{mv} \] with \[ m_e = 9.11 \times 10^{-31} \text{ kg} \]
\[ h = 6.63 \times 10^{-34} \text{ Js} \] (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 \( \frac{2}{3}c \) (\( c \) = speed of light) which corresponds to a De Broglie wavelength of \( \sim 0.004 \text{ nm} \) 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-
1.1. ELECTRON MICROSCOPY

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.

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.
CHAPTER 1. INTRODUCTION

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 (OsO₄) which also serves as a secondary fixative and is applied to the sample as a whole; uranyl acetate (UO₂(CH₃COO)₂ · 2H₂O) which can be applied en bloc and to individual sections; and lead (Pb) compounds such as Reynolds’ 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 an 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
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 best annotated genome of multicellular organisms, publicly available at FlyBase (McQuilton et al., 2012). 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 dendritic branches. Chemical synapses are generally polyadic, one pre-synaptic terminal communicates to two or

http://flybase.org/
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 Olympiads5) render the Drosophila larva an ideal model to decipher the formation and function of a complex biological nervous system at synaptic resolution.

5Fly Olympiad: http://www.janelia.org/team-project/fly-olympiad
Larval Olympiad: http://www.janelia.org/team-project/larval-olympiad
Chapter 2

Registration and Alignment

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.
CHAPTER 2. REGISTRATION AND ALIGNMENT

Ordinary 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\footnote{Completely Useless Registration Tool (CURT), available as part of Rohlfing’s Computational Morphometry Toolkit (CMTK), \url{http://nitrc.org/projects/cmtk}} generates outstanding results in terms
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. 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

$$\text{cost} = \text{deformation} - \text{similarity}$$

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

---

\(^2\)The perfect counter example is again Rohlfing’s CURT, however, for practically useful transformations, the rule generally holds.
2.3. RELATED WORK

dimensional real space $\mathbb{R}^m$. Let further $T : \mathbb{R}^n \to \mathbb{R}^m$ be a transformation $T(x) = y$ and $I[T] : \mathbb{R}^m \to \mathbb{I}$ and $J : \mathbb{R}^m \to \mathbb{J}$ 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 $D[I,J]$ be a dissimilarity metric defined over both pixel value domains and $S[T]$ denote a regularizer, typically the cost for deformation. Then, generalizing the formulation by Modersitzki (2009), Equation (2.1) constitutes the optimization problem

$$\arg \min_T D[I[T], J] + S[T].$$

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 Kovačić, 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-

---

3http://www.sci.utah.edu/software/129-nctoolset.html
2.3. RELATED WORK

Solution 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 \( \epsilon(a_i) = \epsilon(x_i) + \epsilon(y_i) \) 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 \( x_n = x_0 + \sum_{i \in \{1...n\}} d(x_i) \). The respective uncertainty \( \epsilon(x_n) \) accumulates accordingly \( \epsilon(x_n) = \epsilon(x_0) + \sum_{i \in \{1...n\}} \epsilon(d(x_i)) \). 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

\[
R_s : \mathbb{R}^2 \to \mathbb{R}^2 \quad \text{with} \quad R = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix}, \quad t = \begin{pmatrix} x \\ y \end{pmatrix}, \quad (2.3)
\]
\( \mathbf{R} \) being a \( 2 \times 2 \) 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 \( D_s : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \). Imaging sections with the microscope introduces lens-distortion \( L : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) 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 \( \mathbf{t}_s \) for each tile, and the heat-induced deformation of the section during imaging which is an unknown non-rigid deformation \( D_t : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \). The combined geometric transformation \( T_t : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \) mapping an image coordinate \( \mathbf{p} \) into its two-dimensional world-coordinate \( \mathbf{q} \) is thus

\[
\mathbf{q} = T_t(\mathbf{p}) = D_s(R_s(L(\mathbf{p}) + \mathbf{t}_s)). \tag{2.4}
\]

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

\[
T_t(\mathbf{p}) = D_s(D_t'(R_s(L(\mathbf{p}) + \mathbf{t}_s))). \tag{2.5}
\]

which enables to join the translation of section and tile

\[
T_t(\mathbf{p}) = D_s(D_t'(L(\mathbf{p}) + \mathbf{t}_s)). \tag{2.6}
\]

Knowing little about the deformations \( D_s \) and \( D_t' \), they may be subsumed to a more general unknown deformation \( D_{st'} \)

\[
T_t(\mathbf{p}) = D_{st'}(R_s L(\mathbf{p}) + \mathbf{t}_{st}). \tag{2.7}
\]

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

\[
S_s(\mathbf{p}) = D_s(R_s L(\mathbf{p}) + \mathbf{t}_s) \tag{2.9}
\]

### 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 \( x \in \mathbb{Z}^n \) be an \( n \)-dimensional discrete coordinate vector and \( I : \mathbb{Z}^n \to \mathbb{R} \) be a gray-scale image. Let further \( G : \mathbb{Z}^{n+1} \to \mathbb{R} \) 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 $L(i, x) = G(i, x) \ast I(x)$. Two adjacent Gaussians $G(i, x)$ and $G(i+1, x)$ are separated by a constant factor $k$ such that $\sigma_{i+1} = k \sigma_i$. $D(i, x) = L(i+1, x) - L(i, x)$ is a constantly scaled approximation of $\sigma^2 \nabla^2 G(i, x)$. Extrema of $D(i, x)$ 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).
2.5. LANDMARK-BASED ALIGNMENT AND MONTAGING

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

\[
G(i, x) = \frac{1}{2\pi \sigma_i^2} e^{-\frac{|x|^2}{2\sigma_i^2}}.
\]  

(2.10)

Then a scale pyramid \( \mathcal{L} : \mathbb{Z}^{n+1} \rightarrow \mathbb{R} \) is generated by convolution of the image with the series of Gaussian kernels \( \mathcal{L}(i, x) = G(i, x) * I(x) \). The DoG pyramid \( \mathcal{D} : \mathbb{Z}^{n+1} \rightarrow \mathbb{R} \) is built by subtracting adjacent entries of the scale pyramid \( \mathcal{D}(i, x) = \mathcal{L}(i + 1, x) - \mathcal{L}(i, x) \).\(^4\) Local extrema are found by comparison of each value in \( \mathcal{D}(i, x) \) 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. \cite{Brown2002} use the Taylor expansion and finite first and second order derivatives at the discrete extremum-detection \( \mathcal{D} \) to parameterize the quadric

\[
\mathcal{D}(x) = D + \frac{\partial D}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x
\]

(2.11)

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

\[
\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x}.
\]

(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 \( \alpha \) and \( \beta \) 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 $\text{tr}(H)$ and the determinant $\det(H)$ of the Hessian:

$$\frac{\text{tr}(H)^2}{\det(H)} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r + 1)^2}{r}.$$  (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 $4 \times 4$ 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 $8 \times 8$ 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 straightforward 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 $C = \{ (p_1, q_1), \ldots, (p_m, q_m) \}$ with $(p, q)$ denoting a pair of corresponding interest point coordinates $p, q \in \mathbb{R}^n$ 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 $T : \mathbb{R}^n \rightarrow \mathbb{R}^m$ minimizes the sum of a function $\rho : \mathbb{R} \rightarrow \mathbb{R}$ over the residuals of all samples

$$
\arg \min_T \sum_{i=1}^{n} \rho(|T(p_i) - q_i|).
$$

(2.14)

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

$$
\arg \min_T \sum_{i=1}^{n} |T(p_i) - q_i|^2
$$

(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

\[
\text{arg min}_T \sum_{i=1}^{n} \rho(|T(p_i) - q_i|) \quad \text{with} \quad \rho(x) = \begin{cases} 
\frac{x^2}{2} & \text{if } x \leq c, \\
0 & \text{if } x > c
\end{cases}
\quad (2.16)
\]

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 \( c \)

\[
\bar{\epsilon} = \frac{1}{n} \sum_{i=1}^{n} \omega(|T(p_i) - q_i|) \quad \text{with} \quad \omega(x) = \begin{cases} 
x & \text{if } x \leq c, \\
0 & \text{if } x > c
\end{cases}
\quad (2.17)
\]

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

<table>
<thead>
<tr>
<th>Input:</th>
<th>Set of data points $C = {(p_1, q_1), \ldots, (p_n, q_n)}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Maximal multiple $k$ of mean residual $\bar{e}$</td>
</tr>
<tr>
<td>Output:</td>
<td>Set of inlier data points $I \subseteq C$</td>
</tr>
<tr>
<td></td>
<td>Model $T$ best fitting to $I$</td>
</tr>
</tbody>
</table>

1: $I \leftarrow C$  \hspace{1cm} $\triangleright$ Initialize best set of inliers.  
2: $\bar{e} \leftarrow \infty$  \hspace{1cm} $\triangleright$ Initialize threshold.  
3: while $|I|$ is changing do  
4: \hspace{1cm} $T \leftarrow$ least-squares estimate for $I$  \hspace{1cm} $\triangleright$ Update the model.  
5: \hspace{1cm} calculate $\bar{e}$ for $I$ and $T$  
6: \hspace{1cm} for all $(p, q) \in C$ do  
7: \hspace{2cm} if $|Tp - q| > k\bar{e}$ then  
8: \hspace{3cm} $I \leftarrow I \cup \{(p, q)\}$  
9: \hspace{2cm} end if  
10: \hspace{1cm} end for  
11: 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 $m = |C_{\text{min}}|$ to estimate a hypothesis

$$p = (1 - r^m)^k.$$  \hspace{1cm} (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 $C = \{(p_1, q_1), \ldots, (p_n, q_n)\}$
Maximal number of iterations $k$
Maximal accepted error $\epsilon_{max}$
Minimal required inlier ratio $r$

Output: Set of inlier point matches $I \subseteq C$
Model $T$ best fitting to $I$

1: $I \leftarrow \emptyset$ \hspace{1em} $\triangleright$ Initialize best set of inliers.
2: for $k$ iterations do
3: $C_{min} \leftarrow$ choose random minimal subset of $\in C$ to solve $T$
4: $T \leftarrow$ estimate for $C_{min}$ \hspace{1em} $\triangleright$ Generate hypothesis.
5: $C_+ \leftarrow \emptyset$ \hspace{1em} $\triangleright$ Initialize consensus set.
6: for all $(p, q) \in C$ do
7: if $|Tp - q| \leq \epsilon_{max}$ then
8: $C_+ \leftarrow C_+ \cap \{(p, q)\}$
9: end if
10: end for
11: if $|C_+| > |I|$ then
12: $I \leftarrow C_+$
13: end if
14: end for
15: if $|I|/|C| \geq r$ then
16: $T \leftarrow$ fit to $I$ \hspace{1em} $\triangleright$ Refine the model with respect to all inliers.
17: end if

$m$ and an expected lower bound for the inlier ratio $r$

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

In a large dataset, the desired lower bound for pairwise failure can be extremely small by which $k$ becomes excessively large. In case that $k \geq |C|^m$ 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

\[ A : \mathbb{R}^n \to \mathbb{R}^n \]

\[ x \mapsto Ax + t, \]  

(2.20)

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

**Translations** where \( A = I \);

**Rigid body transformations** where \( A \) is a rotation matrix, i.e. orthogonal

\[ A^T A = AA^T = I \]

and \( \det A = 1 \);

**Similarity transformations** where \( A \) complies with

\[ A^T A = AA^T = v^2 I, \]

with \( v \) being a scale factor such that \( R = \frac{1}{v} A \) 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 \( C = \{(w_1, p_1, q_1), \ldots, (w_n, p_n, q_n)\} \)

\[ \arg \min_{A,t} \sum_{i=1}^{n} w_i |Ap_i + t - q_i|^2. \]  

(2.21)

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

\[ t = \bar{q} - A \bar{p} \quad \text{with} \]

\[ \bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}, \]

\[ \bar{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}. \]  

(2.22)
which is already the solution for the translation model and enables to rewrite Equation (2.21) to
\[
\arg\min_{A} \sum_{i=1}^{n} w_i |A\hat{p}_i - \hat{q}_i|^2 \quad \text{with} \quad \hat{p}_i = p_i - \bar{p}, \quad \hat{q}_i = q_i - \bar{q}. \tag{2.23}
\]

The least-squares solution for \( A \) in general affine transformations is accordingly
\[
A = \sum_{i=1}^{n} w_i \hat{q}_i \hat{p}_i^T \left( \sum_{j \in \{1...n\}} w_j \hat{p}_j \hat{p}_j^T \right)^{-1}. \tag{2.24}
\]

For similarity transformations, Schaefer et al. (2006) have embedded the constraint \( A^T A = AA^T = \sigma^2 I \) in Equation (2.24) yielding a compact solution for the two-dimensional case
\[
A = \frac{1}{\mu} \sum_{i=1}^{n} \left( -\hat{q}_i^T \hat{p}_i \right) \left( \hat{p}_i - \hat{p}_i^\perp \right). \tag{2.25}
\]
where \( \perp \) denotes a 90° rotation such that \((x, y)^\perp = (-y, x)\) and \( \mu \) is the variance of all weighted \( \hat{p}_i \)
\[
\mu = \sum_{i=1}^{n} w_i \hat{p}_i^T \hat{p}_i. \tag{2.26}
\]

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 \( \mu \) such that \( M \) 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 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 $C = \{C_{11}, C_{12}, \ldots, C_{tt}\}$

Output: Optimal set of transformations $T = \{T_1, \ldots, T_t\}$

1: $\epsilon \leftarrow \infty$ \Comment{Initialize the average residual.}
2: $T \leftarrow \{I, \ldots, I\}$ \Comment{Initialize all transformations with the identity.}
3: repeat
4: for all $a \leq t$ do
5:  $T \leftarrow$ estimate for $C_a = C_{a1} \cup \cdots \cup C_{at}$
6: for all $(p, q) \in C_a$ do
7:  $p \leftarrow T(a)$ \Comment{Update landmarks in $C_a$.}
8: end for
9: $T_a \leftarrow T \cdot T_a$ \Comment{Concatenate transformations.}
10: end for
11: $\epsilon \leftarrow$ calculate global cost
12: until $\epsilon$ 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 $C_{ab} = \{(p_1, q_1), \ldots, (p_c, q_c)\}$ 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 $T = \{T_1, \ldots, T_t\}$ is that minimizing the cost

$$\arg\min_T \sum_{a=1}^t \left( \sum_{b=1}^t \left( \sum_{(p, q) \in C_{ab}} \rho(|T_a(p) - T_b(q)|) \right) \right). \quad (2.27)$$

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

$$\arg\min_T \sum_{a=1}^t \left( \sum_{b=1}^t \left( \sum_{(p, q) \in C_{ab}} |T_a(p) - T_b(q)|^2 \right) \right). \quad (2.28)$$

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
CHAPTER 2. REGISTRATION AND ALIGNMENT

terminates on convergence of the global cost $\epsilon$ as a function of $i$. In the least-squares formulation, this is the sum of all square residuals. As convergence criteria, I require $\epsilon$ 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 $x$ to the linearly interpolated location $q = T(p) = (1 - \lambda)A(p) + \lambda R(p)$. 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

$$A_H = \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix}. \quad (2.29)$$

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

$$q = Tp = (1 - \lambda)Ap + \lambda Rp \quad (2.30)$$

$$= Ap - \lambda Ap + \lambda Rp$$

$$= (A - \lambda A + \lambda R)p$$

$$= ((1 - \lambda)A + \lambda R)p$$

$$T = (1 - \lambda)A + \lambda R. \quad (2.31)$$

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 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 $\rho$ on the landmark residuals is

$$\arg\min_{A,L} \sum_{a=1}^{t} \left( \sum_{b=1}^{t} \left( \sum_{(p,q) \in C_{ab}} \rho \left( \|A_a(L(p)) - A_b(L(q))\| \right) \right) \right) \quad (2.32)$$
simultaneously estimating the affine transformations $A = \{A_1, \ldots, A_t\}$ 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 \cite{Huber1964} instead of the sum of square residuals as a more robust estimator with respect to ‘outliers’. They described and implemented the method for $3 \times 3$ montages of images with $\sim 50\%$ 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 piece-wise 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
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 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.

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 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.
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-
2.6. ELASTIC ALIGNMENT AND MONTAGING

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 \((p, q)_0\) is connected to other vertices \((p, q)_i\) by an ideal spring. Each spring has a relaxed length \(l_i\) (shown for \((p, q)_4\)). 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 \(F\). The desired end-state of the system is where \(F = 0\) for all vertices.

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 \(I_1\) overlapping image \(I_2\), I identify their corresponding location in image \(I_2\) by block matching (see Section 2.6.2). The vertex is then connected into the mesh on image \(I_2\) 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 $I_2$ 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 $I_2$ are connected to their corresponding location in image $I_1$ with their ‘passive’ ends updated according to the affine transformation of the embedding triangle in the mesh on image $I_1$.

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 $d_s$ in the series ($k = \frac{1}{d_s}$) giving farther away sections less impact.

Relaxation of the elastic system is done using an iterative solution similar to gradient descent. For each vertex $(p, q)$, its local coordinate $p$ in the non-deformed image is stored, i.e. $p$ 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 $f_0$ for a vertex $(p, q)_0$ can be calculated using Hooke’s law (Figure 2.6)

$$f = -\sum_{i=1}^{n} k_i x_i \quad \text{with} \quad x_i = \left(1 - \frac{l_i}{|q_i - q_0|}\right) (q_i - q_0) \quad (2.34)$$

and $l_i = |p_i - p_0|$ for all section-mesh springs

$l_i = 0$ for all cross-section springs.

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 of alignment accuracy and execution speed. Since correspondences are searched
2.6. ELASTIC ALIGNMENT AND MONTAGING

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
 CHAPTER 2. REGISTRATION AND ALIGNMENT

Figure 2.8: Local block matching filters. Shown is the NCC coefficient \( r \) of a block of an image \( I_1 \) over a reference image \( I_2 \) 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 \( I_1 \) and the overlapping patch in the other image \( I_2 \) 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 \((p_0, q_0)\) is inspected individually for being an ‘outlier’. To that end, all other block matches \((p_i, q_i)\) 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 \((p_i, q_i)\) is weighted by a Gaussian radial distribution function (RDF) \(\omega : \mathbb{R}^n \rightarrow \mathbb{R}\) centered at the reference \((p_0, q_0)\):

\[
\arg \min_T \sum_{i=1}^n \omega_i |T(p_i) - q_i|^2 \quad \text{with} \quad \omega_i = e^{-\frac{|p_i - p_0|^2}{2\sigma^2}} \quad (2.35)
\]

Choosing a larger standard deviation \(\sigma\) for the RDF \(\omega\) 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 \(\omega\) 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 2\(k\) 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).
2.6. ELASTIC ALIGNMENT AND MONTAGING

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 naively 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 $I(x, y)$, each pixel stores the sum of all pixels of the original image $J(x, y)$ 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

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

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

$$
\int_a^b f'(x) \, dx = f(b) - f(a). \tag{2.37}
$$
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

$$
\rho_{\mathcal{X}, \mathcal{Y}} = \frac{\sigma_{\mathcal{X}\mathcal{Y}}}{\sigma_{\mathcal{X}} \sigma_{\mathcal{Y}}} \tag{2.38}
$$

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

$$
r_{\mathcal{X}, \mathcal{Y}} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \quad \text{with} \quad \bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{2.39}
$$
that can be transformed yielding a set of independent sums. For the numerator, that is

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

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

\[
= \sum_{i=1}^{n} x_i y_i - \frac{1}{n} \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i \tag{2.42}
\]

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

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

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

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

because

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

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

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

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

yielding

\[
r_{XY} = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{\sqrt{n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2} \sqrt{n \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2}} \tag{2.50}
\]

which means that we can calculate \( r_{XY} \) 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 \( r_{XY}^2 \) and the sign of \( r_{XY} \). Then, the calculation of the two square roots can be avoided

\[
r_{XY}^2 = \frac{a^2}{\left( n \sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2 \right) \left( n \sum_{i=1}^{n} y_i^2 - (\sum_{i=1}^{n} y_i)^2 \right)} \tag{2.51}
\]
with
\[
a = n \sum_{i=1}^{n} x_i y_i - n \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i \quad \text{and} \quad \text{sgn}(r_{XY}) = \text{sgn}(a)
\] (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...
2.7. RESULTS

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

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.
2.7. RESULTS

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 – sub-esophageal 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 % OsO₄ in 0.1 M Na-cacodylate buffer for 1 h at room temperature, rinsed with distilled H₂O 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 \cite{Feng2000} 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 \cite{Micheva2007}). 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 \textit{C. elegans} series this happened in nine different places and for the \textit{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\textsuperscript{®} Xeon\textsuperscript{®} CPU with 2.67 GHz and 24 GB of memory, Ubuntu Linux 10.04 LTS installed.

The \textit{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 \textit{C. elegans} series was computed in approximately 12 hours and the result can be interactively viewed at various scales in CATMAID\textsuperscript{5} \cite{Saalfeld2009}. 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 \cite{Long2009}.

For the array tomography series, I have used the same protocol as for the \textit{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 \textit{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-

\footnote{http://fly.mpi-cbg.de/c-elegans}
2.7. RESULTS

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.

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, tile-wise rigid, and elastic alignment, demonstrating that the elastic method clearly outperforms all other methods in the quality of the reconstruction.
2.8. EVALUATION

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.

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

$$l = \sum_{(p,q)} \sqrt{\frac{(x_p - x_q)^2 + (y_p - y_q)^2 + (z_p - z_q)^2}{s^2}}. \quad (2.53)$$

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 $(p, q)$ is the average scale factor of the contributing sections at $p$ and $q$ 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...
2.8.1  Registration and Alignment

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
2.8. EVALUATION

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 × 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.
2.8. EVALUATION

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.
CHAPTER 2. REGISTRATION AND ALIGNMENT

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 today's 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

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

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

\[
y(x) = P_x(x) \ast (k - 1). \tag{2.56}
\]
2.9. LOCAL CONTRAST ENHANCEMENT

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

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.
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 histo-
gram, this can be achieved by cropping the original histogram at a thresh-
old 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 maxi-
mum slope of \( m = 1.0 \), CLAHE re-generates the original image. Slope lim-
its \( m > 1.0 \) lead to increasing histogram equalization. Using the approx-
imation 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

\[
\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i
\]

(2.57)

the sum of a box of arbitrary size can be calculated with just three addi-
tions and one multiplication from an integral image. Subtracting the fil-
tered 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 \( \sigma_x \) 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 \( \sigma_x \). For each pixel the transferred value is
then

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

(2.58)

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

\[
\sigma_x = \sqrt{\frac{1}{n} \left( \sum_{i=1}^{n} x_i^2 - \left( \frac{1}{n} \sum_{i=1}^{n} x_i \right)^2 \right)}
\]

(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 2.1l
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 nearest neighbors of each point as a local descriptor for the point. In the set of all detected interest points, the 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
2.11. SUMMARY

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 2. REGISTRATION AND ALIGNMENT
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 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. 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 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
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, 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 its 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
```

---

3.1. THE MPICBG-LIBRARY AND TRAKEM2

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

```xml
<!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).
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

\begin{verbatim}
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 con-
    tain the CoordinateTransform, nothing will change.
remove(int):E Removes the CoordinateTransform of type E at the speci-
    fied 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.
\end{verbatim}

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

\begin{verbatim}
<!ELEMENT ict_transform_list (ict_transform|iict_transform|
    ict_transform_list|iict_transform_list)*>
\end{verbatim}

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.

```xml
<!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  nd 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
CHAPTER 3. IMPLEMENTATION

(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

\[
\begin{align*}
\text{map}(&\text{ImageProcessor}, \text{ImageProcessor}): \text{void} & \quad \text{Maps a source image into a target image of the same type using nearest neighbor interpolation.} \\
\text{mapInterpolated}(&\text{ImageProcessor}, \text{ImageProcessor}): \text{void} & \quad \text{Maps a source image into a target image of the same type using bilinear interpolation.}
\end{align*}
\]

Both methods are realized using the ImageProcessor methods

\[
\begin{align*}
\text{putPixel}(\text{int, int, int}): \text{void} \\
\text{getPixel}(\text{int, int}): \text{int} \\
\text{getPixelInterpolated}(\text{int, int}): \text{int}
\end{align*}
\]

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
3.1. THE MPICBG-LIBRARY AND TRAKEM2

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

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

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

\texttt{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 TranslationMode2D for one point, a SimilarityMode2D for two points, and an AffineMode2D 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:

```java
map(ImageProcessorWithMask, ImageProcessorWithMask): void
mapInterpolated(ImageProcessorWithMask, ImageProcessorWithMask): 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 $\sigma$. The scale-interval between $\sigma$ and $2\sigma$ 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. Its size is defined by the parameter **maximum image size**. Setting this parameter to values $\leq 1.0$ 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 down-scaled 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 $4 \times 4$ 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 [Schafer 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 \( \text{local region sigma} \). 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 (~20x). 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 $T$. 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.
3.1. THE MPICBG-LIBRARY AND TRAKEM2

composite 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 $f : \Omega \rightarrow T$ with $\Omega \subseteq \mathbb{R}^n$. 
3.2. IMGLIB2

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

$$\Omega = \mathbb{R}^n \quad \text{RealRandomAccessible}\langle T \rangle$$
$$\Omega = [a, b]; a, b \in \mathbb{R}^n \quad \text{RealRandomAccessibleRealInterval}\langle T \rangle$$
$$\Omega = \{x_1 \ldots x_n\}; x_i \in \mathbb{R}^n \quad \text{IterableRealInterval}\langle T \rangle$$
$$\text{NearestNeighborSearch}\langle T \rangle$$
$$\text{KNearestNeighborSearch}\langle T \rangle$$
$$\text{RadiusNearestNeighborSearch}\langle T \rangle$$

$$\Omega = \mathbb{Z}^n \quad \text{RandomAccessible}\langle T \rangle$$
$$\Omega = [a, b]; a, b \in \mathbb{Z}^n \quad \text{RandomAccessibleInterval}\langle T \rangle$$
$$\Omega = \{x_1 \ldots x_n\}; x_i \in \mathbb{Z}^n \quad \text{IterableInterval}\langle T \rangle$$

with $[a, b]; a, b \in \mathbb{R}^n$ denoting all real coordinates in an $n$-dimensional box spanned by two vectors, and $[a, b]; a, b \in \mathbb{Z}^n$ 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\langle T \rangle and RealRandomAccess\langle T \rangle) 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\langle T \rangle and RealCursor\langle T \rangle). 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 randomly 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* 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-
implementations apply to any specific image implementing those interfaces. Reusability 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

```java
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-
3.2. IMGLIB2

<table>
<thead>
<tr>
<th># dim</th>
<th>size [px]</th>
<th>Java [s]</th>
<th>ImageJ [s]</th>
<th>ArrayImg [s]</th>
<th>CellImg [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100m</td>
<td>0.080/1.14</td>
<td>0.081/1.25</td>
<td>0.081/2.17</td>
<td>0.39/2.21</td>
</tr>
<tr>
<td>2</td>
<td>8192²</td>
<td>0.054/1.13</td>
<td>0.054/1.32</td>
<td>0.055/1.90</td>
<td>0.26/1.89</td>
</tr>
<tr>
<td>2</td>
<td>50000²</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>9.79/70.6</td>
</tr>
<tr>
<td>3</td>
<td>384³</td>
<td>0.046/1.35</td>
<td>0.049/1.49</td>
<td>0.046/2.01</td>
<td>0.22/2.02</td>
</tr>
<tr>
<td>6</td>
<td>28⁶</td>
<td>0.386/21.5</td>
<td>×</td>
<td>0.385/27.7</td>
<td>1.91/28.6</td>
</tr>
</tbody>
</table>

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.

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 today’s 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 $90^\circ$ 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. **IMGLIB2**

### 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.](http://scijava.github.com) 2010), the KNIME Image Processing toolbox ([Berthold et al.](http://imglib2.net) 2009), and an increasing number of Fiji-Plugins ([Schindelin et al.](https://github.com/imagej/imglib) 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](http://scijava.github.com)
[^5]: [http://imglib2.net](http://imglib2.net)
[^6]: [https://github.com/imagej/imglib](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 its 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 straightforward. 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 java 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 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 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.
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


113


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,


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**

```java
package benchmark;

import ...

public class InvertImgLib2Implementation {
    /**
     * generic implementation for all ImgLib2
     * @link IterableInterval)s.
     */
    public final static <T extends NumericType<T>>
```
void invert(final IterableInterval<T> img) {
    for (final T t : img)
        t.mul(-1);
}

Invert for ImageJ's ImagePlus

package benchmark;
import ...

public class InvertImageJImplementation {

    /**
     * Invert implementation for ImagePlus. Pixel coordinates
     * are not required so this works for 1d to 5d images.
     */
    public final static void invert(final ImagePlus imp) {
        final ImageStack stack = imp.getStack();
        final int numSlices = stack.getSize();
        for (int s = 1; s <= numSlices; ++s) {
            final ImageProcessor ip = stack.getProcessor(s);
            final int size = ip.getPixelCount();
            for (int i = 0; i < size; i++)
                ip.setf(i, -ip.getf(i));
        }
    }
}

Invert for Java float[] Array

package benchmark;

public class InvertFloatNativeImplementation {

    /**
     * Invert implementation for native float[] array. Pixel
     * coordinates are not required so this works for all
     * dimensions.
     */
    public final static void invert(final float[] img) {
        for (int i = 0; i < img.length; i++)
            img[i] = -img[i];
    }
}
package benchmark;

import ...;

public class CenterOfMassImgLib2Implementation {
  /**
   * generic implementation for all ImgLib2 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;
  }
}

package benchmark;

import ...;

public class CenterOfMassImageJ's ImagePlus {
  /**
   * generic implementation for all ImageJ's ImagePlus.
   */
  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;
  }
}
public class CenterOfMassImageJImplementation {

    private final static double[] findCenterOfMass1D(final ImagePlus imp) {
        final RealSum centerOfMass0 = new RealSum(),
        sum = new RealSum();

        final ImageProcessor ip = imp.getProcessor();
        final int size0 = ip.getPixelCount();

        for (int d0 = 0; d0 < size0; ++d0) {
            final double value = ip.getf(d0);
            centerOfMass0.add(value * d0);
            sum.add(value);
        }

        final double s = sum.getSum();

        return new double[]{centerOfMass0.getSum() / s};
    }

    private final static double[] findCenterOfMass2D(final ImagePlus imp) {
        final RealSum centerOfMass0 = new RealSum(),
        centerOfMass1 = new RealSum(),
        sum = new RealSum();

        final ImageProcessor ip = imp.getProcessor();
        final int size0 = ip.getWidth();
        final int size1 = ip.getHeight();

        for (int d1 = 0; d1 < size1; ++d1)
            for (int d0 = 0; d0 < size0; ++d0) {
                final double value = ip.getf(i++);
                centerOfMass0.add(value * d0);
                centerOfMass1.add(value * d1);
            }

        return new double[]{centerOfMass0.getSum() / s,
                             centerOfMass1.getSum() / s};
    }

sum.add(value);
}
final double s = sum.getSum();
return new double[]{centerOfMass0.getSum() / s,
        centerOfMass1.getSum() / s};
}

/**
 * special purpose implementation for 3d
 * @link ImagePlus
 */
private final static double[] findCenterOfMass3D(
        final ImagePlus imp) {
    final RealSum centerOfMass0 = new RealSum(),
            centerOfMass1 = new RealSum(),
            centerOfMass2 = new RealSum(),
            sum = new RealSum();

    final int size0 = imp.getWidth();
    final int size1 = imp.getHeight();
    final int size2 = imp.getNChannels();

    final ImageStack stack = imp.getStack();
    for (int d2 = 0; d2 < size2; ++d2) {
        final ImageProcessor ip = stack.getProcessor(
                imp.getStackIndex(d2 + 1, 1, 1));
        int i = 0;
        for (int d1 = 0; d1 < size1; ++d1)
            for (int d0 = 0; d0 < size0; ++d0) {
                final double value = ip.getf(i++);
                centerOfMass0.add(value * d0);
                centerOfMass1.add(value * d1);
                centerOfMass2.add(value * d2);
                sum.add(value);
            }
    }
    final double s = sum.getSum();

    return new double[]{centerOfMass0.getSum() / s,
            centerOfMass1.getSum() / s,
            centerOfMass2.getSum() / s};
}
/**
 * special purpose implementation for 4d
 * {@link ImagePlus}.
 */
private final static double[] findCenterOfMass4D(final ImagePlus imp) {
    final RealSum centerOfMass0 = new RealSum(),
    centerOfMass1 = new RealSum(),
    centerOfMass2 = new RealSum(),
    centerOfMass3 = new RealSum(),
    sum = new RealSum();

    final int size0 = imp.getWidth();
    final int size1 = imp.getHeight();
    final int size2 = imp.getNChannels();
    final int size3 = imp.getNSlices();

    final ImageStack stack = imp.getStack();
    for (int d3 = 0; d3 < size3; ++d3) {
        for (int d2 = 0; d2 < size2; ++d2) {
            final ImageProcessor ip = stack.getProcessor(imp.getStackIndex(d2 + 1, d3 + 1, 1));
            int i = 0;
            for (int d1 = 0; d1 < size1; ++d1)
                for (int d0 = 0; d0 < size0; ++d0) {
                    final double value = ip.getf(i++);
                    centerOfMass0.add(value * d0);
                    centerOfMass1.add(value * d1);
                    centerOfMass2.add(value * d2);
                    centerOfMass3.add(value * d3);
                    sum.add(value);
                }
        }
    }
    final double s = sum.getSum();
    return new double[]{centerOfMass0.getSum() / s,
                        centerOfMass1.getSum() / s,
                        centerOfMass2.getSum() / s,
                        centerOfMass3.getSum() / s};
}
/**
 * special purpose implementation for 5d
 * {@link ImagePlus}.
 */

private final static double[] findCenterOfMass5D(
    final ImagePlus imp) {
    final RealSum centerOfMass0 = new RealSum(),
    centerOfMass1 = new RealSum(),
    centerOfMass2 = new RealSum(),
    centerOfMass3 = new RealSum(),
    centerOfMass4 = new RealSum(),
    sum = new RealSum();
    final int size0 = imp.getWidth();
    final int size1 = imp.getHeight();
    final int size2 = imp.getNChannels();
    final int size3 = imp.getNSlices();
    final int size4 = imp.getNFrames();
    final ImageStack stack = imp.getStack();
    for (int d4 = 0; d4 < size4; ++d4) {
        for (int d3 = 0; d3 < size3; ++d3) {
            for (int d2 = 0; d2 < size2; ++d2) {
                final ImageProcessor ip = stack.getProcessor(
                    imp.getStackIndex(d2 + 1, d3 + 1, d4 + 1));
                int i = 0;
                for (int d1 = 0; d1 < size1; ++d1)
                    for (int d0 = 0; d0 < size0; ++d0) {
                        final double value = ip.getf(i++);
                        centerOfMass0.add(value * d0);
                        centerOfMass1.add(value * d1);
                        centerOfMass2.add(value * d2);
                        centerOfMass3.add(value * d3);
                        centerOfMass4.add(value * d4);
                        sum.add(value);
                    }
            }
        }
    }
    final double s = sum.getSum();
    return new double[] {centerOfMass0.getSum() / s,
        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(
final byte[] img, final int size0) {
    final RealSum centerOfMass0 = new RealSum(),
    sum = new RealSum();
    for (int d0 = 0; d0 < size0; ++d0) {
        final double value = img[d0] & 0xff;
        centerOfMass0.add(value * d0);
        sum.add(value);
    }
    final double s = sum.getSum();
    return new double[]{centerOfMass0.getSum() / s};
}

/**
 * special purpose implementation for primitive
 * byte[] array 2d.
 */
private final static double[] findCenterOfMass2d(
    final byte[] img, final int size0, final int size1) {
    final RealSum centerOfMass0 = new RealSum(),
    centerOfMass1 = new RealSum(),
    sum = new RealSum();
    int i = 0;
    for (int d1 = 0; d1 < size1; ++d1)
        for (int d0 = 0; d0 < size0; ++d0) {
            final double value = img[i++] & 0xff;
            centerOfMass0.add(value * d0);
            centerOfMass1.add(value * d1);
            sum.add(value);
        }
    final double s = sum.getSum();
    return new double[]{centerOfMass0.getSum() / s,
    centerOfMass1.getSum() / s};
}

/**
 * special purpose implementation for primitive
 * byte[] array 3d.
 */
private final static double[] findCenterOfMass(
    final byte[] img, final int size0, final int size1,
    final int size2) {
    final RealSum centerOfMass0 = new RealSum(),
    centerOfMass1 = new RealSum(),
    centerOfMass2 = new RealSum(),
    sum = new RealSum();

    int i = 0;

    for (int d2 = 0; d2 < size2; ++d2)
        for (int d1 = 0; d1 < size1; ++d1)
            for (int d0 = 0; d0 < size0; ++d0) {
                final double value = img[i++] & 0xff;
                centerOfMass0.add(value * d0);
                centerOfMass1.add(value * d1);
                centerOfMass2.add(value * d2);
                sum.add(value);
            }
    final double s = sum.getSum();

    return new double[]{centerOfMass0.getSum() / s,
                        centerOfMass1.getSum() / s,
                        centerOfMass2.getSum() / s};
}

/**
 * special purpose implementation for primitive
 * byte[] array 4d.
 */

private final static double[] findCenterOfMass(
    final byte[] img, final int size0, final int size1,
    final int size2, final int size3) {
    final RealSum centerOfMass0 = new RealSum(),
    centerOfMass1 = new RealSum(),
    centerOfMass2 = new RealSum(),
    centerOfMass3 = new RealSum(),
    sum = new RealSum();

    int i = 0;

    for (int d3 = 0; d3 < size3; ++d3)
        for (int d2 = 0; d2 < size2; ++d2)
for (int d1 = 0; d1 < size1; ++d1)
for (int d0 = 0; d0 < size0; ++d0) {
    final double value = img[i++] & 0xff;
    centerOfMass0.add(value * d0);
    centerOfMass1.add(value * d1);
    centerOfMass2.add(value * d2);
    centerOfMass3.add(value * d3);
    sum.add(value);
}

final double s = sum.getSum();

return new double[]{centerOfMass0.getSum() / s,
                    centerOfMass1.getSum() / s,
                    centerOfMass2.getSum() / s,
                    centerOfMass3.getSum() / s};
}

/**
 * special purpose implementation for primitive
 * byte[] array 5d.
 */
private final static double[] findCenterOfMass(
    final byte[] img, final int size0, final int size1,
    final int size2, final int size3, final int size4) {
    final RealSum centerOfMass0 = new RealSum(),
    centerOfMass1 = new RealSum(),
    centerOfMass2 = new RealSum(),
    centerOfMass3 = new RealSum(),
    centerOfMass4 = new RealSum(),
    sum = new RealSum();

    int i = 0;

    for (int d4 = 0; d4 < size4; ++d4)
        for (int d3 = 0; d3 < size3; ++d3)
            for (int d2 = 0; d2 < size2; ++d2)
                for (int d1 = 0; d1 < size1; ++d1)
                    for (int d0 = 0; d0 < size0; ++d0) {
                        final double value = img[i++] & 0xff;
                        centerOfMass0.add(value * d0);
                        centerOfMass1.add(value * d1);
                        centerOfMass2.add(value * d2);
                        centerOfMass3.add(value * d3);
centerOfMass4.add(value * d4);
sum.add(value);
}

final double s = sum.getSum();

return new double[]{centerOfMass0.getSum() / s,
centerOfMass1.getSum() / s,
centerOfMass2.getSum() / s,
centerOfMass3.getSum() / s,
centerOfMass4.getSum() / s};
}

/**
 * special purpose implementation for primitive
 * byte[] array 6d.
 */

private final static double[] findCenterOfMass(final byte[] img, final int size0, final int size1,
final int size2, final int size3, final int size4,
final int size5) {
final RealSum centerOfMass0 = new RealSum(),
centerOfMass1 = new RealSum(),
centerOfMass2 = new RealSum(),
centerOfMass3 = new RealSum(),
centerOfMass4 = new RealSum(),
centerOfMass5 = new RealSum(),
sum = new RealSum();

int i = 0;

for (int d5 = 0; d5 < size5; ++d5)
  for (int d4 = 0; d4 < size4; ++d4)
    for (int d3 = 0; d3 < size3; ++d3)
      for (int d2 = 0; d2 < size2; ++d2)
        for (int d1 = 0; d1 < size1; ++d1)
          for (int d0 = 0; d0 < size0; ++d0) {
            final double value = img[i++] & 0xff;
            centerOfMass0.add(value * d0);
            centerOfMass1.add(value * d1);
            centerOfMass2.add(value * d2);
            centerOfMass3.add(value * d3);
            centerOfMass4.add(value * d4);
            centerOfMass5.add(value * d5);
sum.add(value);
}

final double s = sum.getSum();

return new double[] {centerOfMass0.getSum() / s,
    centerOfMass1.getSum() / s,
    centerOfMass2.getSum() / s,
    centerOfMass3.getSum() / s,
    centerOfMass4.getSum() / s,
    centerOfMass5.getSum() / s};

/**
 * special purpose implementation for primitive
 * byte[] arrays 1D to 6D.
 * dispatches to the 1d to 6d versions.
 */

public final static double[] findCenterOfMass(
    final byte[] img, final int[] dimensions) {
    if (dimensions.length == 1)
        return findCenterOfMass(img, dimensions[0]);
    else if (dimensions.length == 2)
        return findCenterOfMass(img, dimensions[0],
            dimensions[1]);
    else if (dimensions.length == 3)
        return findCenterOfMass(img, dimensions[0],
            dimensions[1], dimensions[2]);
    else if (dimensions.length == 4)
        return findCenterOfMass(img, dimensions[0],
            dimensions[1], dimensions[2], dimensions[3]);
    else if (dimensions.length == 5)
        return findCenterOfMass(img, dimensions[0],
            dimensions[1], dimensions[2], dimensions[3],
            dimensions[4]);
    else if (dimensions.length == 6)
        return findCenterOfMass(img, dimensions[0],
            dimensions[1], dimensions[2], dimensions[3],
            dimensions[4], dimensions[5]);
    else
        throw new IllegalArgumentException(
            "only 1D to 6D images are supported");
}
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 tomography 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.


   (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.


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


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