From THE DEPARTMENT OF BIOSCIENCES AND NUTRITION
Karolinska Institutet, Stockholm, Sweden

SPATIO-TEMPORAL MODELING OF MORPHOLOGY AND GENE-EXPRESSIONS DURING *C. ELEGANS* EMBRYOGENESIS USING A NEW IMAGING FRAMEWORK

Johan Henriksson

Karolinska Institutet

Stockholm 2012

Published by Karolinska Institutet.

ISBN 978-91-7457-628-3

Printed by
REPROPRINT AB
Stockholm 2012
www.reproprint.se
Gårdsvägen 4, 169 70 Solna

Abstract

Many aspects of ontogenesis, the process whereby an organism develops from the fertilized egg to a grown adult, are still poorly understood, despite having been studied since Aristotle's time. The advent of modern biology with the sequencing of entire genomes and mapping of all genes offers the promise that one day we will understand how genes function to regulate cell division and differentiation. These advances offer new possibilities to solve the question that humanity has been asking since the beginning of time, how is a new life created?

The aim of this thesis is to establish methods to study when and where particular genes are expressed in the model organism *Caenorhabditis elegans* by means of microscopy. Ultimately, this will aid our understanding of embryology in more complex organisms and in human. The usefulness of the software is however not limited to embryology but is also useful for microscopy in general.

In Paper I we present our open source software platform, Endrov (http://www.endrov.net), for image processing and microscopy. It is strongly modular and meant to allow tight integration with all kinds of infrastructure (microscopes, cameras, controllers, etc.) needed for modern biological and non-biological imaging. Endrov supports modern visualization, over 100 file formats, can connect to the OMERO image server and can control many modern light microscopes. We have also implemented 140 image processing algorithms and several types of annotations, e.g., for particle tracking, 3D surfaces, point-text annotation, neuron/vascular 4D annotation and more. A novel feature is the concept of laziness, which makes Endrov scale for large data sets and consequently makes it easier to prototype algorithms. The entire application is written in Java and consists of about 150 000 lines of code.

In Paper II we have developed procedures to record wild-type *C. elegans* embryos in 3D over time (4D). Endrov was used to analyze the data and we generated a model of normal embryonic development complete up to the 150 cell-stage. The variance in cell positioning and division timing was calculated and the cell-cell contacts estimated. When compared to a previous model made from a squeezed embryo, our model has a higher time resolution and is more reproducible.

In Paper III we have generated a revised list of all homeobox genes in *C. elegans*, showing that out of 103 homeobox genes, 70 are co-orthologous to human homeobox genes. A number of modules have been developed for analyzing and quantifying expression data. These are not only suitable for *C. elegans* but can also be used for other biological model systems. Special features include automatic adjustment of exposure time during recording resulting in an increased dynamic range beyond the limits of the microscope camera, and annotation of the cells in 3D rendered space. We have applied the new framework to examine homeobox gene expression patterns and provide an analysis of the patterns that we have recorded.

In Paper IV we have extended the applicability of Endrov by using it to study the mitochondrial polymerase gamma (*polg-1*) in *C. elegans*. *polg-1* deletion mutant alleles were analyzed for phenotypes using methods such as measuring lifespan and brood size, qPCR for mtDNA content and transcript levels, and light and TEM microscopy. The main findings were that homozygous *polg-1* mutant animals develop normally, although later in the adult stage they exhibit compromised gonadal function, sterility, as well as reduced viability due to rupture at the vulva. 3D modeling of the gonad revealed structural abnormalities in the germline. Further, the few descendants that are generated are severely compromised and die during embryogenesis. We can deduce that while mtDNA copy number is a limiting factor for development, the maternally contributed mitochondria are enough to sustain development to adulthood.

In summary, we have created the Endrov software framework that allows processing of large multidimensional microscopy image data sets and demonstrated that it can be applied to a wide range of problems.

List of Publications

I. **Henriksson J**, Hench J, Bürglin TR.
Endrov – An integrated platform for biological image analysis and processing
(manuscript)

II. Hench J, **Henriksson J**, Lüppert M, Bürglin TR.
Spatio-temporal reference model of Caenorhabditis elegans embryogenesis with cell contact maps
Dev Biol, 2009, vol 333 issue 1, page 1-13

III. Hench* J, **Henriksson* J**, Abou-Zied, A, Lüppert, M, Dethlefsen, J, Tong, Y.G, Mukherjee, K, Tang, L, Baillie, D, and Bürglin TR.
Spatio-temporal expression analysis of homeobox genes during C. elegans embryogenesis using a new imaging framework
(manuscript)
*equal contribution

IV. Bratic I, Hench J, **Henriksson J**, Antebi A, Bürglin TR, Trifunovic A.
Mitochondrial DNA level, but not active replicase, is essential for Caenorhabditis elegans development.
Nucleic Acids Res, 2009, 37(6) 1817-1828

Additional Publications

Xue-Franzén Y, Johnsson A, Brodin D, **Henriksson J**, Bürglin TR, Wright AP. *Genome-wide characterisation of the Gcn5 histone acetyltransferase in budding yeast during stress adaptation reveals evolutionarily conserved and diverged roles.* BMC Genomics. 2010 Mar 25;11:200.

I have initiated and supervised several related master thesis projects:

Ahmed Mehadi
Implementing efficient iterative 3D deconvolution for microscopy

Adeel A Nasir
Implementation and comparative study of image thresholding methods

Adam Kull
Ray tracing of volumetric data in real time

Arvid Johansson, David Johansson
Introductory study for the development of a virtual microscope for education within life sciences

Niklas Frisk
Using OpenCL for image analysis

Javier Fernandez
Image processing to detect worms

Christian Ziethen
Segmentation of EM neuron images (in progress)

Stanislav Goryachev
Machine learning for candidate gene extraction (in progress)

Kristina Lend
Whole genome comparisons of X-box promoter motif sequences (in progress)

Contents

List of abbreviations

4D	In this thesis: 3D plus time
API	Application Programming Interface
BSD	Berkeley Software Domain
ceh	*C. elegans* homeobox gene
CoPAn	Contact-preventing angle
DIC	Differential Interference Contrast (or Nomarski)
FFT	Fast Fourier Transform
FRET	Förster Resonance Energy Transfer
FSF	Free Software Foundation
FTP	File Transfer Protocol
GFP	Green Fluorescent Protein
GNU	GNU is Not Unix
GPL	GNU General Public License
GUI	Graphical User Interface
HOCHOB	Homeobox cysteine homeobox (a domain)
JPEG	Image file format, Joint Photographics Expert Group
l_2	Space of vectors $\{v : \sum_i \lvert v_i \rvert^2 < \infty\}$
LZW	Lempel-Ziv-Welch, compression algorithm
O(...)	Big-Oh notation, complexity of operation
OME	Open Microscopy Environment
OST	Native file format for Endrov, Open Spatio-Temporal
OOP	Object-Oriented Programming
OpenGL	Open Graphics Library
PNG	Image file format, Portable Network Graphics
PSF	Point-Spread Function
RFP	Red Fluorescent Protein
RNAi	RNA interference
RNAseq	RNA sequencing
SDK	Software Development Kit
SNR	Signal-to-Noise Ratio
SQL	Structured Query Language, also referring to relational databases
TEM	Transmission electron microscopy
TF	Transcription Factor
TIFF	Image file format, Tagged Image File Format
VTK	The Visualization Toolkit, a library for 3D graphics
XML	File format, Extensible Markup Language

1 INTRODUCTION

How does life begin? This is one of the oldest questions in biology. While not being the first, Aristotle proposed early on that the mixture of a woman's blood and a man's semen would create a fetus. This followed in line with several philosophers who believed that the man provided the human essence, and the mother was only a nurturer. But what is the form of the human essence? Science was divided into two camps: preformation and epigenesis. Preformation is the idea that all life has been created simultaneously and that humans grow from miniatures, the homunculus. Epigenesis, on the other hand, suggests that life starts from an unorganized lump of material. Epigenesis got a boost when William Harvey discovered the human egg (Harvey, 1651). However, the discovery of the spermatozoa was a backlash for this theory (van Leeuwenhoek, 1677); instead Hartsoeker described what he saw as “little men” in the sperms. Epigenesis recovered when the cell theory (Hooke, 1665) could be combined with the theory of evolution (Darwin, 1859). Haeckel proposed in his recapitulation theory in 1866 that the developmental stages follow the evolutionary development (“ontogeny recapitulates phylogeny”). While this theory turned out to be incorrect, many aspects of the biological processes are shared between bilaterian animals and mammals, because of their evolutionary history. For this reason it makes sense to study simple organisms to understand complex human biology.

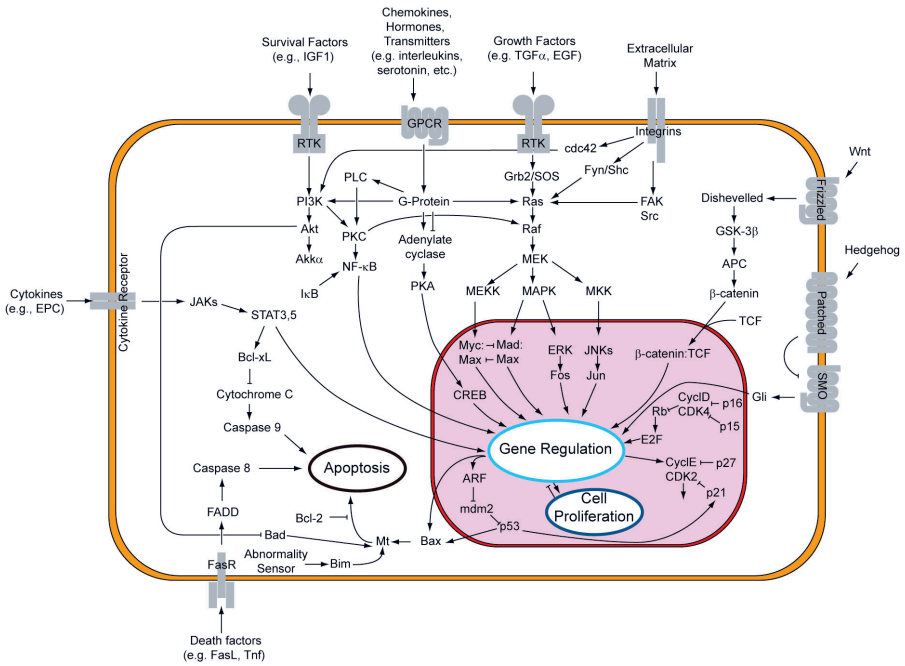

Figure 1: Bilaterian signaling pathways. Arrows indicate a gene up-regulating (activating) another gene, while blunt-ended lines mean down-regulation. Reproduced from KEGG (Kanehisa and Goto, 2000; Kanehisa et al., 2011).

In the 20th century, biologists have discovered proteins, DNA and the genetic code. Cells are now known to be controlled through various interactions between genes. Since every cell has usually the same set of genes, and can in theory perform any function in the body, the identity of a cell is given by the subset of genes that are active. Since mature cells keep their identity, mechanisms exist to lock them into a certain state. There are nonetheless cell types which can change their identity, such as the stem cells in the red bone marrow, which creates new red blood cells and other cell types. When a cell changes from one type to another, it is called cell differentiation.

To understand both how cells maintain their identity/state and how they differentiate, the way genes are regulated has to be understood. Many genes are required for the basic structural integrity and metabolic function of a cell. However, a particular subset of genes (regulatory genes, developmental control genes) play key roles in regulating other genes at the level of the DNA (e.g., transcription factors) or by providing signaling mechanisms within and between cells (Figure 1). A regulatory gene that is expressed (active) can, in different ways, up- or down-regulate other genes (turn them on or off); when several genes are connected in this manner they form a network. Examples of the fundamental bilaterian signaling pathways are shown in Figure 1. Cell extrinsic signals are passed through a cascade of different molecules from receptors in the membrane through the cytoplasm to end up at transcription factors in the nucleus, where they regulate genes. The goal is to know which are the states of the cell and through which mechanisms can the cells evolve into other states? To answer this, the connections of the network have to be found. Given two genes, there are experimental methods to confirm whether they affect each other – but given N genes, there are N^2 possible interactions. If N is large, in the order of 20 000 for humans, there is no efficient way of testing every interaction. The first step is thus to reduce the number of potential interactions by finding out which genes are expressed simultaneously, and in which order. The goal of this thesis is to develop methods for doing so, using microscopy on the simple model system *Caenorhabditis elegans*.

Understanding embryogenesis has yielded many valuable insights. Diseases such as cancer have been linked to mutations in genes coupled to development (Moore, 2009), and the agriculture industry could apply the results to create more efficient crops. Drug design will be easier since the consequences and targets will be easier to predict. Finally it will give us the satisfaction of finally understanding the miracle of life.

1.1 GENE REGULATION

There are several ways in which one gene can regulate the expression of another gene. A transcription factor is a protein that can bind to a particular DNA sequence. After binding, the factor can either block or recruit RNA polymerase, the complex responsible for transcription. This means that the transcription (expression) of the gene is either suppressed or promoted. Transcription coregulators or cofactors are proteins that function in a similar way except that instead of binding to the DNA directly, they bind to a DNA-

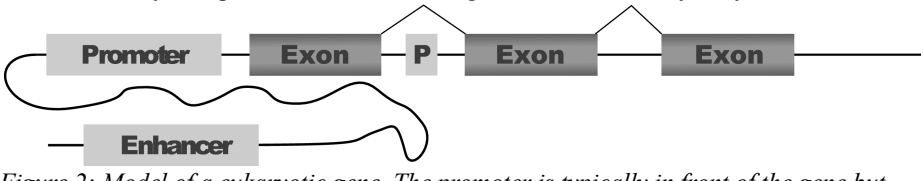

Figure 2: Model of a eukaryotic gene. The promoter is typically in front of the gene but can even be inside an intron (P). The enhancer can be located much further away.

binding transcription factor, or another coregulator. Just like transcription factors, they can influence transcription positively or negatively. There are many places in the genome relative to a certain gene where a transcription factors can bind (Figure 2), such as the promoter region of a gene, typically located in front of the gene, or even in an intron. They can also bind to enhancer regions, which can be located far away from a gene, and even on a the different chromosome (Spilianakis et al., 2005).

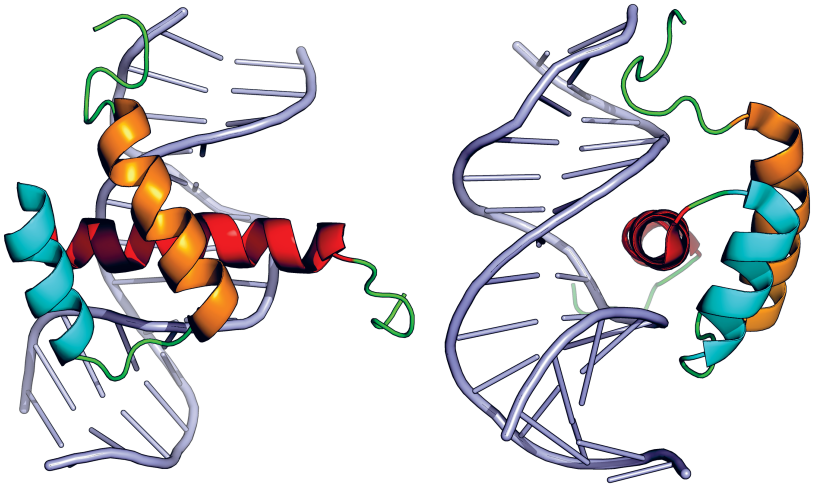

Figure 3: Binding of the Antennapedia homeodomain to DNA (Billeter et al., 1993). Helix 3 (red) entering the grove of the DNA is the recognition site. Produced from PDB:1ADH using PyMol (Schrödinger LLC, 2010).

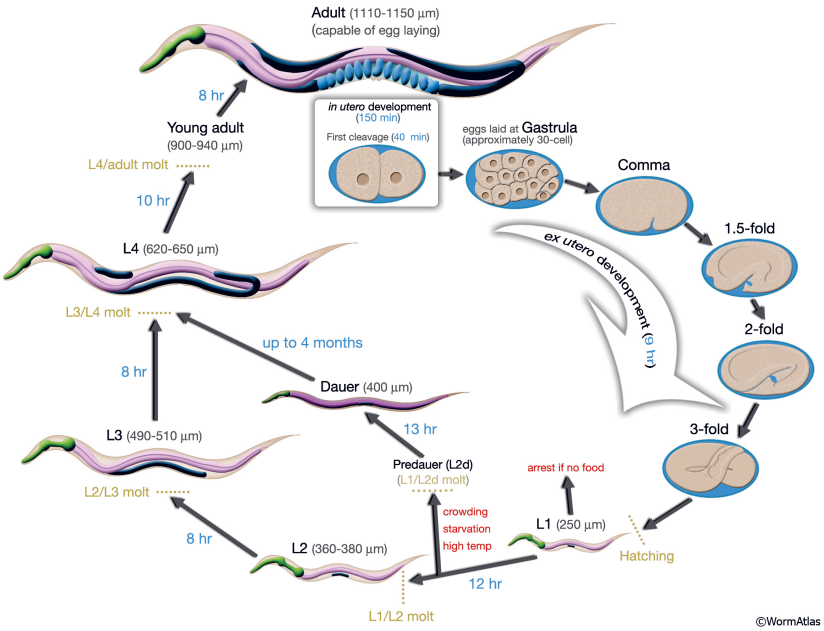

Figure 4: C. elegans life cycle. Reproduced from (Hall and Altun F, 2008)

Histone modifications such as acetylation provide another means of controlling access to the DNA. Acetylation counteracts the folding of DNA into highly compact structures, therefore the transcriptional machinery can bind more easily to the DNA (Eberharter and Becker, 2002). Many others ways of controlling expression in a similar manner exist, including histone/DNA methylation, ubiquitylation, phosphorylation and sumoylation. Gene regulation can also occur post-transcriptionally; for example, RNA interference (RNAi) can silence a gene by targeting an mRNA for degradation (Matzke and Birchler, 2005).

This thesis has particular focus on a family of transcription factors first found in *Drosophila melanogaster* (fruit fly). Mutations in 'homeotic' genes were found to cause body parts to be transformed into other body parts (Lewis, 1978). These genes were found to share a sequence which is now termed the homeobox (McGinnis et al., 1984; Scott et al., 1989). The homeodomain, encoded by the homeobox, consists of 3 helices, about 60 amino acids in total (Figure 3). The third helix contains the major DNA recognition site (Qian et al., 1989). Since their discovery, homeobox genes have been found in most eukaryotes (Bürglin, 2011). In animals, those homeodomain proteins that have been studied play key roles in development, and because these genes are so conserved in evolution, it is likely that many of them play an important role for ontogenesis in all metazoa. Hence they are prime candidates when screening for new genes involved in *C. elegans* embryogenesis.

1.2 CAENORHABDITIS ELEGANS

Much of the progress in biological investigations has been made possible by the usage of model organisms – organisms that share biological functions with humans, but are much easier to work with in a laboratory environment. One of them is *Caenorhabditis elegans*, a nematode commonly found in composts and which is associated with snails (Caswell-Chen et al., 2005). It is non-parasitic, but might be necromenic (breaks down the host after death), and survives by feeding on bacteria.

C. elegans has several advantages as a model organism: The genome sequence is available and the generation time is only 3 days (Figure 4), thus genetics is fast (Wood, 1988). A range of genetic methods have been developed making it is possible to mutate and delete genes and introduce new transgenes. Genes can be silenced by RNAi, either by feeding, soaking or injection. Genome-wide screenings of RNAi have been performed (Kamath et al., 2003) (Kamath et al., 2003). Once the genetic manipulations are done, the strains can be stored by freezing and thawing (Hope, 1999). The worm is transparent and small which is ideal for microscopy. The adult hermaphrodite has a fixed number of only

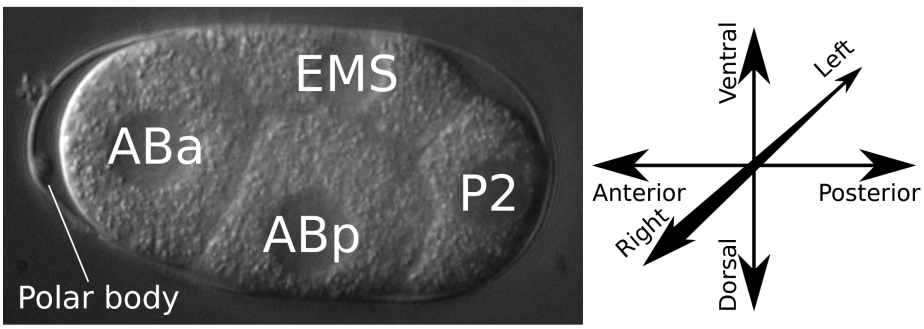

Figure 5: DIC image of 4-cell embryo with coordinate system.

959 somatic nuclei (Sulston and Horvitz, 1977; Kimble and Hirsh, 1979), many of which are equivalents to human organs.

Unlike in mammals, the cells in the nematode *C. elegans* divide the same way in each individual, that is, the lineage is invariant (Sulston and Horvitz, 1977; Sulston et al., 1983). This has profound consequences: each cell can be given a name and the connection between the genetics and cell differentiation can be studied very precisely. An overview of the life-cycle is shown in Figure 4. The first cell division of the *C. elegans* embryo produces already two asymmetric cells, AB and P1, which establishes the anterior-posterior axis, which is set up by the location of the sperm entry point (entry side becomes posterior) (Goldstein and Hird, 1996). The other axes of the coordinate system are set up at the 4-cell stage (Figure 5). At around the 28-cell stage, the embryo enters gastrulation until it reaches the comma stage. At this stage morphogenesis starts and the embryo begins to twitch due to muscle differentiation as it elongates into a folded larvae. After about 14 hours, the larvae (L1 stage) hatches and continues development outside the egg.

1.3 STAINING AND FLUORESCENT TAGGING

To study biological processes they first have to be visualized. Dyes have been used to stain different structures such as lipids and DNA, but they cannot easily be made specific for a certain protein. Immunohistochemistry, using antibodies, has been used extensively to detect where proteins are expressed. One drawback is that antibody production is time consuming, and might not be specific for the protein of interest, thus the antibody must be tested carefully. Another problem is that immunohistochemistry is done on fixed tissues so it is difficult to study the dynamics of expression. In-situ hybridization (ISH) is an alternative technique to follow gene expression at the transcript level, using labeled complementary DNA or RNA. This method is the basis of the NextDB (Kohara lab, http://nematode.lab.nig.ac.jp/), where *C. elegans* embryos of different stages were stained in a systematic fashion. A variant of ISH using a fluorescent label (FISH) also exists (Langer-Safer et al., 1982).

The discovery of Green Fluorescent Protein (GFP) and its application to *C. elegans* was a breakthrough (Chalfie et al., 1994). The method can be used to study living tissues and once a GFP reporter construct has been introduced into an animal, there are no laborious staining protocols. It is possible to identify the cells where a gene is expressed by making a transcriptional fusion – a gene is copied with it's regulatory elements and the coding sequence is replaced with that of GFP. Such reporter constructs are then injected or bombarded into hermaphrodites, to establish transgenic animals. The GFP in the extrachromosomal or integrated arrays will be expressed in cells where the original gene is also expressed (Hobert, 2002). Another method is to generate translational fusions, where the GFP coding region is fused to the end of the protein coding sequence, hence fusing the two proteins. This will cause GFP to be co-localized with the studied protein and provide information about sub-cellular localization. Techniques like FRET (Förster Resonance Energy Transfer) can exploit this further to show if proteins are interacting. There are several fluorescent proteins other than GFP, such as YFP and RFP variants, (Shaner et al., 2004) that fluoresce at other wavelengths, but the spectral overlap limits the usage to about 3 simultaneous fluorophores in practice.

In *C. elegans* a transgene can be inserted in several ways, all by injecting the construct into the gonad or by bombardment with DNA covered gold-particles (Evans, 2010). This creates semi-stable tandem arrays that are heritable, but require selection to be maintained. To improve stability, extrachromosomal arrays can be integrated into a chromosome by gamma, X-ray or UV irradiation (Mello et al., 1991). Newer techniques such as MosTIC (Robert et al., 2009) can be used to ensure that only single copies of a construct are inserted into the genome. Transgenes can be created systematically and large screens have been done using in the order of 2000 transgenic strains (Hunt-Newbury et al., 2007), from which we obtained most of the strains analyzed here (paper III).

There are inherent limitations with fluorescent proteins as markers stemming from the biology. To be seen, the gene first has to be transcribed and then enough of the mRNA has to be translated. This introduces a time lag. Further, the protein can persist stably in the cell, and its descendant cells, until it has been degraded. To address this, GFP variants with PEST sequences causing rapid degradation have been generated (Corish and Tyler-Smith, 1999). Another complication is technical; the GFP protein can be damaged by light from the microscope, causing bleaching. All of these have to be taken into account when interpreting images of GFP.

The process behind fluorescence is shown in Figure 6. Molecules can absorb photons and excite an electron. Some of the energy is passed into a vibrational energy state. Kashas law dictates that before it can proceed to the ground state, it has to reach its lowest excited state by dissipating heat. It is the jump to the ground state that causes the molecule to emit a fluorescent photon. Since the emitted energy is smaller than the absorbed energy, the emitted wavelength is always longer than or equal to the absorbed wavelength (Stokes law). This cycle is only one of many possible; absorption can also occur to higher states. In addition, there are states with low transition probabilities causing the electrons to be effectively trapped ("forbidden states"). Although the electrons will be released eventually, the entrapment is in practice another way of bleaching fluorophores. Understanding the mechanics, and limitations, of fluorescence is vital for using a fluorescent microscope correctly.

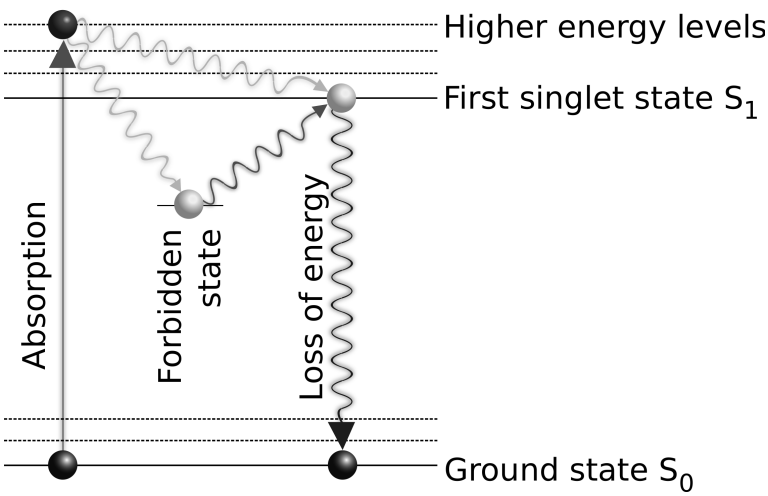

Figure 6: Jablonski diagram showing the energy transitions for fluorescence.

1.4 MICROSCOPY

Microscopy as a field has grown rapidly and a range of techniques are today available off-the-shelf. The classical brightfield microscope where a camera is used to capture the image has been complemented with confocal microscopy, where the sample is scanned point-by-point to generate higher resolution (Laser Scanning Microscopy, LSM). As such, LSM is rather slow. A confocal microscope equipped with a Nipkow spinning disk can scan multiple points simultaneously, making it more comparable in speed to brightfield microscopy. All advanced microscopes offer computer control for automation. For example, the sample can be moved in every direction using a motorized stage (Figure 7). A 3D image can then be generated by moving the focus up and down along the Z-axis and taking a picture at each level. With this method some out-of-focus light will also be acquired. Confocal microscopes have better Z-resolution than brightfield microscopes and are hence more suitable for 3D imaging. It is also possible to move beyond 3D by simultaneously doing recordings over time ("time-lapse"). Multiple samples can also be monitored at the same time by using the XY-stage to move between several positions. This is usually used in conjunction with multi-well plates. The XY-stage can also be used to acquire images of samples that does not fit within the focus – multiple adjacent images are captured and stitched together in software. There are many recent developments; nevertheless, the microscope will always be limited by physical constraints. Hence the user also has to understand some of the theory behind it. Some of it will be given here; a good review of microscopy and different methods is given by (Cox, 2007).

The light path for a typical microscope is shown in Figure 8. A bright-field image is generated by shining light through the sample into the detector. For fluorescence, it is better to shine the light in the opposite direction of the detector. A dichroic mirror only reflecting some of the wavelengths is used for this purpose. Additional filters for emission and excitation are used to select the right wavelength. The light source can be either a bright lamp or a laser, with the laser having the advantage of being monochromatic and coherent. As a consequence, no excitation filter is needed.

The light is normally collected either by a CCD (Charge-Coupled Device) camera or a Photon Multiplier Tube (PMT). The PMT is only used for scanning microscopes. The output, either way, is an array of pixels. The interpretation of these pixels, or read-outs, has been a source of much confusion. A pixel should in this context not be interpreted as a color of a square area on the screen, but rather as an electrical read-out at a single point. For technical reasons, the read-out is not linearly related to photon count, one of them being that the signal is noisy. There might also be very few photons hitting the sensor causing a statistical uncertainty. Electrical noise can also contribute, and this is the main reason why the sensor is typically cooled. In addition there is the background radiation that can cause ionization. There is an upper limitation in how much light a sensor can measure; above this limit the sensor is said to be "saturated". There are several reasons, for example, if the sensor collects too many electrons, they can diffuse away, or make it harder to capture additional electrons. Therefore the read-out will grow less than linearly to the photon count at high counts. Finally there is an upper hard limit of the value sent to the computer, dictated by the digital electronics, known as the "bit depth".

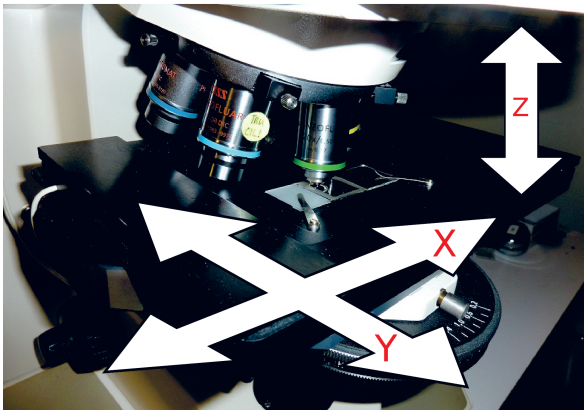

Figure 7: A sample mounted on a microscope. The focus can be changed using the XY- and Z-stages. The different objectives mounted on the turret can be selected from the computer.

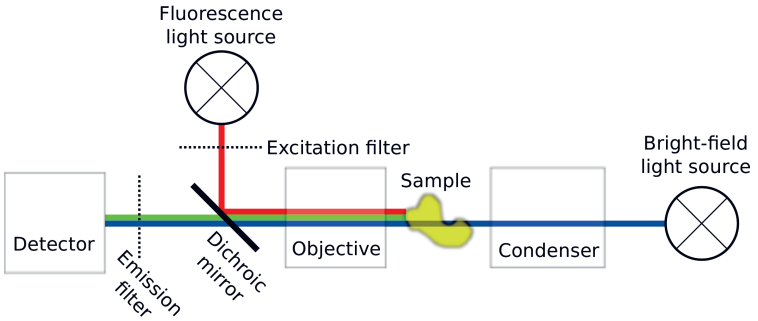

Figure 8: Light path for an ordinary fluorescent microscope. The excitation light (red) stimulates the sample to emit fluorescence (green). To minimize the excitation light reaching the detector, the incoming light is made to travel in the opposite direction of the emission light. A dichroic mirror and two filters separates the wavelengths. A bright-field light source can be fit in this system (blue) and simply shines through the sample (the emission filter and mirror is normally removed in this case).

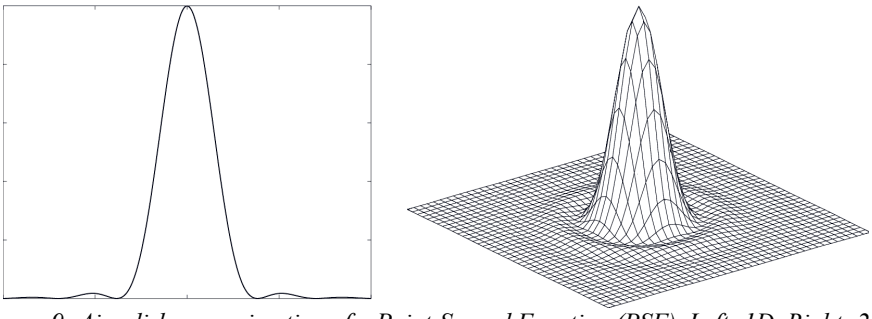

Figure 9: Airy disk approximation of a Point-Spread Function (PSF). Left: 1D, Right: 2D

Light is not collected from a point but from a local volume. The volume is represented by the Point-Spread Function (PSF, Figure 9) and is related to the probability that a photon within a certain range is captured by the sensor. The observed image can be thought of as

$$ObservedImage = Sample \circ PSF$$

where the convolution operator in 3D is defined as

$$f \circ g = \int \mathrm{d}a\,\mathrm{d}b\,\mathrm{d}c\, f(a,b,c)g(x-a, y-b, z-c)$$

A more precise model is that the observed image is a stochastic variable,

$$ObservedImage \sim Poi[Sample \circ PSF] + N[ambientLevel, noise],$$

involving two random distributions – one from the sample following a Poisson distribution as seen as a decay process, and an additional normal distribution that approximates other interference such as electric noise in the detector and background illumination in the room. The signal-to-noise ratio (SNR) becomes higher with increasing light but it comes at the cost of photobleaching and reduced sample viability.

The "real" image of the sample, before the PSF has been applied, can be obtained computationally through deconvolution. The disadvantage is that it takes a long time, requires good knowledge of the PSF and certain assumptions about the image have to be made (Hansen et al., 2006). This means that care has to be taken when interpreting deconvolved images and if deconvolution is to be used then it should be planned before acquiring the images. There is also blind deconvolution that computes the PSF from the image itself, but by making additional assumptions. Several programs exist for the purpose, both open source and proprietary.

There are further limitations beyond the PSF. In particular, the wavelength of light limits the best possible resolution. The famous Abbes formula states that the smallest distance r between two objects that can be resolved for a lens is:

$$r = \frac{\lambda}{2sin\theta}$$

which once the lens has been taken into account becomes the Rayleigh formula:

$$r = \frac{0.612\lambda}{n\sin\theta} = \frac{0.612\lambda}{NA}$$

Here the definition of Numerical Aperture (NA) has been inserted. Classically, the only way to view very small details has been electron microscopy, as the electron has a shorter wavelength. To circumvent this limitation, recent development includes two-photon microscopy which uses non-linear effects to achieve better resolution. Two-photon microscopy also has higher sample penetrability (it can image deeper) and reduced sample damage compared to single-photon confocal microscopy. SPIM (Voie et al., 1993) is particularly promising due to its speed and depth penetrability. For fluorescent samples, techniques such as STED (Stimulated Emission Depletion) and STORM (Stochastic Optical Reconstruction Microscopy) can be used to achieve even higher resolution. These advanced techniques have however not been used in this thesis; interested readers can look at, for example, (Huang et al., 2009) for a review.

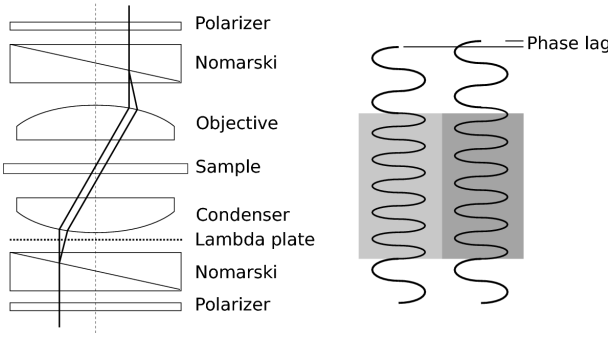

Figure 10: Left: The DIC lightpath. The Nomarski splits the beam into two. Right: The different material opacity causes a phase lag in the two adjacent beams.

Two traditional alternatives to brightfield images are Phase Contrast (PC) and Differential Interference Contrast (DIC). Both techniques offer improved contrast for transparent samples. DIC uses a Nomarski/Wollaston prism to measure local difference in opacity. Light travels at different speeds in different media; hence a difference in opacity will cause two light waves to become out of phase. DIC measures this phase difference. The design is shown in Figure 10. The light is polarized and goes through a Nomarski prism. This prism is composed of two birefringent quartz plates, causing the polarized beams to exit at slightly different angles. This causes the beams to pass through the sample at slightly different locations, each with a different opacity. The beams are then combined together with another prism and polarizer. To get a contrast, a lambda plate is also needed to create the initial phase difference of usually ¼ wavelength. It can be shown that for an ideal DIC the observed intensity is

$$i(x,y) = Csin^2(\Delta x \frac{\partial \phi(x,y)}{x} + \Delta\theta)$$

where Δx is the beam-shift in the x-direction, $\phi(x,y)$ is the opacity, and $\Delta\theta$ is the phase shift from the lambda plate (Preza et al., 1996). It can also be seen that the intensity wraps if the phase difference becomes too large which imposes some limitations on the applicability. This is one of the reasons, besides scattering, why DIC has an upper limit for how thick the sample can be. Besides this, a well-made DIC has excellent Z-resolution compared to PC and ordinary brightfield, which is why it has been used so extensively in *C. elegans*.

1.5 IMAGE AND DATA STORAGE

An important aspect of microscopy is to store images the correct way. Starting with an individual pixel, the measured light from the microscope is not a continuum but quantized as an integer. Integers are normally stored in a fixed number of bits and the number of bits is referred to as the bit depth. Most programs work with 8-bit RGB (red, green, blue) images, and this is the best precision a regular monitor can display, meaning that the largest number that can be presented in a single channel (color) is 2^8-1=255. For microscopy, this has turned out to be limiting or inconvenient – in particular as the value should be thought of as a photon count, not a color on the screen. Therefore many cameras work with 12 or 16 bits instead. When talking about digital storage, such as a hard disk or computer RAM, the unit bytes is used instead. One byte is 8 bits and because of the prevalence of bytes in the computer industry, 8 bits are commonly used. As an

example of size, a grayscale (1 channel/color) 8 bit image of size 512x512 consumes 262 kb memory uncompressed.

Figure 11: JPEG and PNG compression. The uncompressed image is stored as TIFF (100% size) and PNG (60%), Lossy JPEG compression further reduces the size: JPEG 95 (13.3%), JPEG 50 (3.0%), and JPEG 10 (1.0%). Image: (Lenna, 1972)

Microscopy suffers from the curse of dimensionality: size increases very fast when adding new dimensions. While the previous image of 262kb is negligibly small for current computer hardware, a 3D stack of 512x512x200 is 52Mb. If further a movie of 3D stacks is produced, with 1000 time-points, the size increases to 52Gb. Now, imagine recording every sample on a 96-well plate. The final data size becomes 5TB, more than can be fitted on the largest consumer-grade hard-drive at the time of writing. Hence microscopy pushes the limits of storage and methods of reducing data size are needed. The simplest way of saving space is by discarding acquired data. Cropping the image to only view the sample is an efficient way of doing so. Besides cropping, various data compression techniques have to be considered. These techniques are divided into lossy and lossless methods (Sayood, 2000).

Lossy methods imply that some data is discarded to reduce file size. These are the most efficient techniques and are used for DVDs (MPEG-2), JPEG and MP3. The algorithms are designed to keep the data as close to the original as possible, from an aesthetic point of view. The discarded data is mostly noise and other high-frequency components. While these formats can be used for microscopy, they should be used with the greatest care and an understanding of what data is lost, in particular if something is to be quantified. Because compression degrades the image quality it also makes analysis harder and so it should be considered the very last option. See Figure 11 for example compression rates.

Lossless compression implies that no data is lost. Compression always assumes that the data contains correlations and is not white noise (perfectly random data cannot be compressed). The redundancies are removed to conserve space. As an example, consider the string AAAAAABBBB. With the RLE (Run-Length Encoding) algorithm it can be abbreviated as 6A4B. Much more sophisticated methods are used today including for example arithmetic coding and the LZW (Lempel-Ziv-Welch) algorithm. The popular PNG (Figure 11) and ZIP file formats incorporate lossless compression.

General-purpose lossless algorithms does not give very good results compared to domain-specific specialized algorithms. However, a general algorithm can be made specific by augmenting it with a preprocessor that incorporates domain-specific heuristics. In imaging, this can be done by incorporating a predictor (Figure 12). The lossless image format PNG uses the assumption that "neighboring pixels have similar values". A simple example is shown in Table 1 where the predictor stores the difference from the previous value, $y_t=x_t-x_{t-1}$. The output is more redundant than the input, and thus simpler to compress. With this in mind, it is easier to understand the performance of different methods on different images. Experiments in our lab have shown that fluorescent images compress better than DIC images when using PNG (LZW). The intuitive reason is that DIC images are less regular and contain more features that have to be stored.

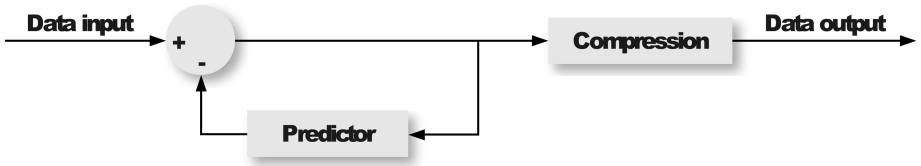

Figure 12: Compression with a predictor. The predictor estimates what the next input from the data sequence will be. By subtracting this value from the input, the compressor is fed more uniform values that are easier to compress.

Input x_t	3	4	5	6	7	8	5	4	3	2
$y_t=x_t-x_{t-1}$	3	1	1	1	1	1	-3	-1	-1	-1

Table 1: Above: Original values. Below: Values after subtraction by predicted value (the previous input is the predicted value). The sequence, which is more uniform, can in turn be fed into a generic compression algorithm.

So far only the pixel data has been considered. A microscopy image is however not just pixels; the information about how the data was captured is called metadata. This includes for example the date, specimen, author, which microscopy technique was used, the exposure time and magnification. The latter are very important for further processing of the data. It is for example impossible to show an accurate 3D rendering of the data without knowing the resolution, and the exposure time is important when quantifying signal intensity. It follows that this information has to be stored somewhere. Most file formats used by home users for images, such as BMP, JPEG, PNG, SVG and so on, are unable to store this information at all. If a microscopy image is ever converted to any of these formats then the metadata is lost permanently. The microscope vendors have therefore invented many of their own formats to handle the increasing requirements on what a format should be able contain. The TIFF file format is often used because it is easy to add extra data in the TIFF header. Unfortunately, this has been done in many different ways, e.g., resolution might be called different things and have different units. Other vendors have made entirely new formats and there are over 100 file formats in existence. A big step forward is the creation of the Bio-formats library which can read an ever increasing list of formats (Linkert et al., 2010). The problem is that users still need to be aware of the issues of metadata when they are writing the files. The Open Microscopy Environment (OME) is currently standardizing what metadata should be included for a recording to be considered complete (Swedlow et al., 2009). They have also produced new file formats, OME-XML and OME-TIFF, which can contain all the metadata they have defined.

Images will in the future likely be stored in databases. One driving reason is that it is infeasible to share 10GB+ files on FTP-sites. Even with the current capacities, transfer is too slow for browsing, and images which cannot be browsed efficiently will not be used to their full potential. Another reason is that it is difficult to organize the images as files. Should they be sorted by date, by strain, by project or by user? A database can organize the data more thoroughly than the file system. A database also simplifies batch processing of images. It is very convenient to tag images for processing and have the application query the database for input. For medical images, the transition to databases has already been completed with the DICOM format and the Picture Archiving and Communication Systems (PACS) standard (DICOM Standards Committee, 2011). However, the DICOM standard covers different needs than for those in basic biology. For biological images, there are at least two big databases; one is Bisque (Kvilekval et al., 2010) but the most commonly used image database is OMERO by OME (Swedlow et al., 2009). The latter is supported by the program developed in this thesis.

1.6 SOFTWARE DEVELOPMENT

Software development can be divided roughly into the following activities: 1. Defining inputs, calculations, and outputs. 2. Choosing programming language, data structures and algorithms. 3. Deciding which tools to deploy. 4. Designing the user-interface. 5. Finally the work has to be planned and after the project, it has to be evaluated. Being an engineering process, software development is both subjective and iterative. Some things are covered by computer science such as the best choice of algorithms. There are methods to evaluate user interfaces but no single correct way of designing them. Many decisions are made by convention and experience rather than through deep evaluation. Some decisions are heavily debated such as which is the best

language, and which working methodology is most efficient. On these matters I can only give my own perspective.

The choice of programming language is no doubt the most opinionated matter. Many languages have been created, for many different purposes. Programmers who have used one language for X often want to use it for Y, just because they have spent effort learning it. Others prefer one language for the syntax or the theory behind it, not because it solves the problem in shortest time. For this thesis, a rather large piece of software has been written. This raises large demands on the ease of maintaining the code. Languages are either statically typed (e.g., C, Java, Haskell) or dynamically typed (e.g., Perl, Python, R). With static typing, many errors can be found during compilation instead of having to run the program. Testing and ensuring that each part of a program is covered is a difficult task, especially for large programs. Thus for large projects, my belief is that only statically typed languages should be used. Further, static typing allows the compiler to make additional assumptions and hence produce faster code. The types are also an essential part of the documentation and therefore makes it easier to understand the source code. There are developments such as the Hindley-Milner type system (Hindley, 1969) and type interference (Damas and Milner, 1982) which enable static typing without the need to always explicitly write out the types. This removes the probably most common argument for dynamic typing, that the programmer has to write less code. I believe that dynamically typed languages should never be used, for any purpose, unless they offer compensating advantages. The main reason to choose another language tend to be that it has a great bulk of already existing code which can be reused.

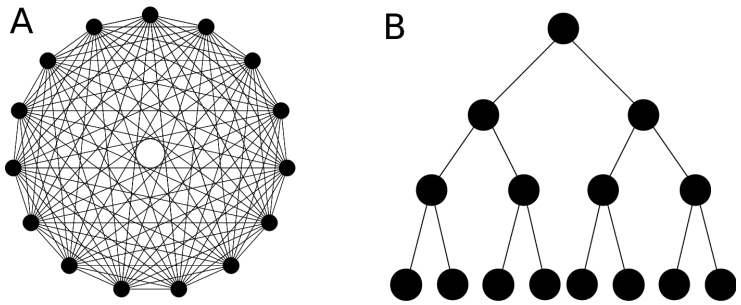

Figure 13: Two extreme examples of the interaction between 15 modules (dots). In A, all modules depend on each other while in B, modules have at most 3 interactions. Good modular software resembles a tree such as this one. Interactions can also be made one-way (calling code versus called code) and this is essential for reusability.

Another opinionated choice is that of tools, which is greatly connected to the choice of language. Over the years the importance has increased of for example the text editor used to write the code. Modern IDEs (Integrated Development Environment) for Java, such as Eclipse (http://www.eclipse.org), supports refactoring for activities such as mass-renaming of functions, encapsulation of variables and repackaging of code. This has to be done to increase the quality of the code, as structure usually deteriorates whenever new features are added. For large projects, this is essential. Other features such as auto-completion, where the editor suggests the programmer what to write, reduces the need for documentation. Finally there are tools such as Ant, Hudson and Maven, which are used for compiling and packaging the software.

The design of software has more consensus and is well-studied academically. Code should be written to be modular and reusable. The motivation for this can be seen from Figure 13. The complexity of a program, and hence the amount of bugs and cost of maintenance, is proportional to the number of module interactions. Also, it is very hard to reuse a module if all its interactions have to be recreated in a new setting. Modularity is achieved by encapsulation, hiding as much of the code as possible. Thus code building on top of another piece of code, is forced to invoke only a small subset of the functions, called the interface. The way encapsulation is done depends on the language. Object-oriented languages such as C++ and Java have language-specific features for hiding code, and there is a large body of "best practice" dictating how certain problems are best mapped to different objects. The concept of "patterns" is well-known for this purpose, although I don't want to advocate for them myself (Gamma et al., 1995). I believe the best code is the code that, doing what it should do, is cheapest to produce, maintain and can be reused the most. Given how difficult this is to measure, experience is in the end the only way to assess what is best practice.

1.7 COMPLEXITY, ALGORITHMS AND DATA STRUCTURES

An important concept when designing a program is the notion of complexity. It is a measure of how long it takes to run an algorithm (computational complexity), and how much memory in requires (memory complexity). Consider the code which sums all values in a list:

```
Input: list v, n elements
sum=0
for all i=1 to n, do:
  sum ← sum+v_i
```

The number of additions required depends on the number of elements; in this case, n times. Next consider the problem of finding how many elements are equal in two lists:

```
Input: list u, n elements, list v, m
elements
sum=0
for all i=1 to n, do:
  for all j=1 to m, do:
    if u_i=v_j then:
      sum ← sum+1
```

In this example there are $m \cdot n$ additions if all elements are equal. Assuming $m = n$, the number of additions for the two algorithms is seen is Figure 14. It is clear that the latter algorithm quickly takes longer time to execute than the former. The idea in complexity theory is that it is the asymptotic growth in time that is important, not the precise time. That is, if it takes $C \cdot n$ and $D \cdot n^2$ time to run the algorithms for some constants C and D, these can be ignored. Instead we write, using "Big-Oh notation", that the complexity is $O(n)$ and $O(n^2)$. This also means that we can compare the time of additions to the time of, e.g., subtractions; it is the growth that is important, not the actual operation. Memory complexity works the same, except it is memory consumption, instead of speed, that is under consideration.

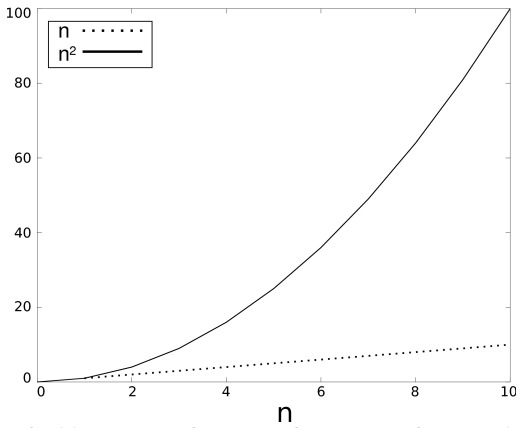

Figure 14: Number of additions as a function of input size, for two algorithms

It is possible to make some general statements about the complexities. $O(1)$ is constant time and nothing is faster – no matter how large the input is, it executes in the same time. It is possible to build a hierarchy of often seen complexities:
$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^2 \log n) < O(n^3) < ...$
Most operations slower than $O(n^5)$, or even $O(n^4)$, are seldom used because they are too slow to consider even for moderate size inputs.

In many cases an algorithm can be replaced with another faster algorithm performing the same function. Revisiting the previous example of counting equal values, it can also be done by this algorithm:

```
Input: list u, n elements, list v, m elements
sum=0
values=<empty hashset>
for all i=1 to n, do:
  add ui to values
for all j=1 to m, do:
   if vi exists in values then:
     sum ← sum+1
```

This algorithm uses a hashset, where inserting and checking if an element exists, is $O(1)$. The total complexity is then $O(n + m)$ which is faster than $O(nm)$. It is beyond this overview to explain how a hashset can be constructed to be this fast, see for example (Data Structures and Algorithms in Java, 2010); the point is merely that by using clever algorithms and data structures, operations can be made several order of magnitudes faster. Even if a program has previously been useful, if it was written with small input sizes in mind, it might be infeasible to run it on large datasets. Since in biology, the datasets are increasing in size fast, especially with the curse of dimensionality, it will be a challenge to write software that can handle the future amounts of data.

One of the foundations for writing fast code is storing the data in appropriate data structures. There are several data structures just for storing elements. See Table 2 for an overview of complexities. Some of them offer additional properties; arrays and linked

lists keeps a given order of the elements. Tree sets, on the other hand, always keeps the elements in sorted order (using an order relationship). There are also other specialized data structures such as Fibonacci heaps which are efficient at keeping a queue of elements. What is the correct choice of data structure depends on which operations the particular algorithm will use, and when.

		Data structure			
		Array	**Linked List**	**Hash set**	**Tree set**
Operation	**Insert**	$O(n)$	$O(1)$	$O(1)$	$O(\log n)$
	Find element	$O(n)$	$O(n)$	$O(1)$	$O(\log n)$
	Delete element	$O(n)$	$O(1)$* / $O(n)$	$O(1)$	$O(\log n)$

*Table 2: Time complexity of Java data structures. * If the element of a linked list has already been localized then it can be removed in $O(1)$.*

Only once the best algorithm and data structure is chosen is it worth trying to improve the speed of individual operations – because it only affects the constant factor, not how it scales. Doing this requires deep understanding of the computer hardware. Modern computers also have multiple CPU cores which brings the intricacy of threaded programming. Further information on this kind of optimizations can be found in for instance (Andrews, 2000) and (Hager and Wellein, 2010).

1.8 IMAGE PROCESSING AND ANALYSIS

It is difficult to give a full overview of image processing as it is done in so many different ways. There is nevertheless a commonly used workflow in biology, in particular for cell cultures (Figure 15). There are three phases: Features are detected, for example landmark points can be found using SURF (Speeded Up Robust Features) (Bay et al., 2008), or shapes can be fit. If the image is to be segmented then it is first simplified by preprocessing, such as evening out areas or amplifying outlines. Segmentation is then done using algorithms that can detect contiguous similar areas. These phases need not all be done nor need they be separate, some algorithms integrate them. Most of the classical techniques are for the local preprocessing. A quick overview is given here.

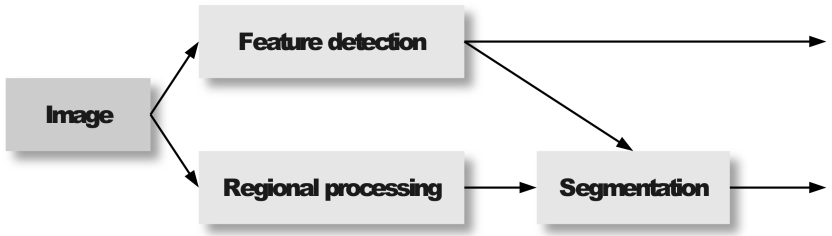

Figure 15: An image processing workflow commonly seen in biology

The techniques used in image processing have been designed from many different philosophies. One set of techniques, based on convolution with a kernel, are used for detecting edges and reducing noise. These can be understood using the theories from

control theory and digital signal processing (DSP), to design low-pass, high-pass and band-pass filters. Such filters can be made to be optimal according to different criteria, given a certain noise model (Hayes, 1996).

Some techniques are based on probability. Statistical hypothesis testing can be used to detect edges (Kurz and Benteftifa Hafed, 1997) and images can be modeled using ideas from statistical physics (Li, 2009). Many techniques are based on transforming the image to other spaces (representations). In particular the Fast Fourier Transform (FFT) algorithm has been a major step forward as it is only $O(n \log n)$ complexity and fits well with the ideas about linear filters (but the list has to have a size that is a power of 2). The Radon transform comes naturally from computerized tomography imaging (Toft, 1996).

There are many techniques which have been purely derived for being fast, and are only based on heuristics. One such is the watershed algorithm (Lantuéjoul and Beucher, 1979) which is commonly used for segmenting images, and uses an analogy to flooding a landscape. Level sets form an extension to it, and can also be used to solve certain real fluid dynamics problems (Sethian, 1999). Another segmentation technique, normalized graph cuts, is based on our ability to efficiently solve large eigenvalue problems (Malik, 2000). It is part of the class of spectral clustering algorithms.

Image processing relies to great extent only on heuristics. A difficult problem is how to exploit *a priori* knowledge. Template matching can be use to find a certain shape within an image, and this shape is the prior knowledge. A more general framework is Machine Learning (ML) where the algorithms can be trained using manually classified data, or by automatically finding classes. Recently the Support Vector Machines (SVM) have gained much attention (Shawe-Taylor and Cristianini, 2004). There are several other machine learning techniques; see (Kotsiantis, 2007) for a review.

A recurring problem is that the final algorithm works well on one set of images, but fails on images acquired under slightly different conditions (different camera, optics, specimen etc). The ability of an algorithm to withstand noise and changes in the input is called “robustness”. Algorithms based on heuristics break if the conditions upon which the heuristics depend are changed. For example, if the size of the interesting features changes. . This can to some extent be solved by pyramidal approaches or by using scale-space theory to find the resolution level of interest (Lindeberg, 1991). Machine learning can overcome some of these problems by training on a new dataset.

The image processing algorithms have been reimplemented many times in different contexts. While the driving force often have been programmers wanting to do it themselves (Not-Invented-Here, NIH), there are also interoperability concerns. Code can only be reused if it operates on existing data structures. In the best case the data can be converted but it impedes performance and sometimes accuracy. Nevertheless, there are now many libraries of image processing algorithms such as ITK (http://www.itk.org/), which collects common image processing operations (Terry S. Yoo, 2002). On the level above algorithms, there are also programs providing a user interface for interacting with the algorithms. ImageJ (Rasband, 1997) is one of the most well-known open source image processing programs. Several commercial programs exist as well such as Bitplane Imaris (http://www.bitplane.com/). A list of existing programs used in biology is given in (Walter

et al., 2010). This thesis introduces the new open source imaging framework Endrov (http://www.endrov.net).

1.9 DATABASES AND SCHEMAS

Any collection of data is per definition a database. The two major structured ways of storing data are currently XML and SQL. XML-files are text files, organizing data in a hierarchical structure while SQL stores data as tables. The data should be kept as organized as possible; this is done by imposing a structure, a "schema", on the data. There are different types of schemas corresponding to different types of data.

One type of schema is the relational schema, primarily used for SQL databases. I also find this to be the most generally applicable method for designing databases. The relational schema consists of two things: attributes (what is stored in the database) and relations (how information can be inferred). Two simple relations are:

```
Person number → First name, Last name
Car number → Person number
```

The first relation states that, given a person number, the name of the person can be found. The attributes on the left side specify the "key" of the relation and once translated into a database, is the natural choice of a table index. Additional information can be inferred by combining the two relationships:

```
Car number → Person number, First name, Last name
```

Knowing the owner of a car can be used to find additional information about the owner. The strength of relational databases lies in the language used to infer information, SQL. There are several open source databases such as PostgreSQL (www.postgresql.org) and MySQL (www.mysql.com), and well-established theories about design (Silberschatz et al., 2010).

XML is a derivative of SGML which is the basis of HTML, used to define the content of websites. As such it has many similarities; the foundation is the tag written as <mytag> … </mytag>, which wraps around other data. In this way XML-files should be thought of as data trees. The data can be stored either as an attribute (attr="...") or as plain text. An example XML-file looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<ost version="3.3">
  <nuclineage id="lin" ostdatecreate="131">
    <data>
      <nuc name="ABalaaapp">
        <child name="ABalaaappl" />
        <child name="ABalaaappr" />
      </nuc>
    </data>
  </nuclineage>
</ost>
```

There are several ways to specify the schema of correct XML, such as DTD. These relationships tell which children a certain node must have, and how many. Schemas for

XML are optional. XML-files have the major disadvantages over SQL-databases that they often must be read entirely into memory before processing, that they consume a lot of space because they are text files, and that the common libraries do not provide database functions such as efficient joining (merging) of tables. Despite this, XML is often used because it is close to object oriented programming. Programs frequently converts between an object-representation and SQL. Because the two paradigms are so different, it is unclear how to do the conversion. This has become known as the impedance mismatch problem. A lot of work has gone into solving it and there are several packages trying to do so (e.g., Hibernate http://www.hibernate.org and Prevayler http://prevayler.org). For this thesis, most data is stored as XML. In the few cases when SQL is used, it is used separately, avoiding the impedance mismatch.

1.10 IDEAS FROM FUNCTIONAL PROGRAMMING

Much of the current development in programming languages comes from the family of functional programming languages, in particular Haskell (www.haskell.org). Some ideas have been used for this thesis (particularly paper I) and should be mentioned.

There are traditionally two ways to evaluate a function in a language: call-by-value and call-by-reference. These are also called eager and lazy evaluation. Eager evaluation is the method most programmers are used to, and is used by languages such as C, Java, Python, Perl and Matlab. The difference is that eager evaluation requires values to be calculated as soon as possible, while lazy evaluation post-pones the evaluation until the raw value is really needed. It is best understood with an example:

Code declaration
X ← 1+2 Y ← X+3 show(Y)

This code just adds 3 numbers, and this is how it executes:

Eager evaluation	Lazy evaluation
X ← 1+2 = 3 Y ← X+3 = 3+3 = 6 show(Y) = show(6) “6” shown to user	X ← 1+2 Y ← X+3 show(Y) = show(X+3) = show(1+2+3) = show(6) “6” shown to user

The point is that to display the calculated value to the user, the value of the Y is needed. Until this point however, it is sufficient to store references to previous calculations (hence, “call-by-reference”). Eager evaluation will on the other hand calculate values as soon as possible, that is, at each assignment. Because it always invokes a function with the raw values, it is called call-by-value.

Lazy evaluation is an old concept (Wadsworth, 1971) but has large implications. Since the language has to keep track of references to calculations, rather than the results of the calculations, efficient compilers have not existed until recently (Jones, 1992). A large problem for a long time was also how to enforce the order of calculations, which is very important for I/O operations, e.g., printing “Hello world” on the screen (several solutions have been found). The two main advantages of lazy evaluation is that many properties of the code are easy to prove, and unnecessary calculations are skipped. The

results of calculations can also easily be cached for later reuse. The big disadvantage is that calculations can pile up, as in the previous example. Storing the references may require more memory than storing the results. If more references are stored than the programmer has intended, then the overuse of memory is called a "space leak" (as opposed to a "memory leak", which is something else). Debugging this kind of errors is still difficult because there are few tools to detect this. Another problem is that some (constant time) computational overhead is induced by keeping track of whether a value has been calculated or not. These problems can be alleviated by forcing eager evaluation at locations where laziness adds no benefit, or space leaks are known to cause problems.

Another concept from functional programming is the lambda. It can be seen as just a way of creating functions without explicit names. This is useful in code that uses many functions as it reduces the burden on the programmer to come up with new unique names. The advantage is purely syntactic. Java has something similar – anonymous classes. Using these it is possible to do something similar to lambdas, albeit with more code. It is not mentioned in the article but Endrov makes extensive use of anonymous classes. Java 1.8 is planned to have proper lambdas as a language feature.

Finally the functional languages have brought better type systems, e.g., Hindley-Milner (Hindley 1969) with type interference, parametric polymorphism and algebraic data types. Features like these are likely to appear also in main-stream languages in the future but are not extensively used for this thesis. Interested readers should have a look at the many online tutorials about Haskell (www.haskell.org).

1.11 IMMUTABILITY AND MEMORY MANAGEMENT

There is one foundation on which laziness rests and that is immutability: the value of a variable never changes. If changes occur then the execution order becomes crucial and since calculations are postponed until later, it becomes very hard to predict the outcome of the execution. Immutability is sometimes also enforced in other languages. Strings (text objects) in Java are immutable, and so are scalars (the simplest values, such as integers). Some data structures can also be made immutable while still keeping the same algorithmic complexity. In particular, tree structures allow subtrees to be shared. To produce a copy of a tree with a modification, only a partial copy of the tree need to be made (Okasaki, 1999). Linked lists can be made immutable in a similar way.

The imperative languages (such as C and Java) usually work with mutable data. It is possible to restore a type of immutability and this is indeed done by some operating systems. When a program is duplicated in memory (used for parallel processing), it would be very efficient if the programs could share their memory to avoid copying everything. Since the two programs are modifying their memory, this cannot easily be done. The Unix-like operating systems, such as BSD, implement copy-on-write pages to accomplish this (McKusick et al., 1996). An example case is shown in Figure 16. The two programs can share their memory as long as they only read the memory. If a program attempts to write to the memory, the operating system will detect this and first duplicate it. Only the necessary parts will be duplicated as the memory is divided into chunks (pages). The mechanism can be extended to also include swapping; that is, memory can be temporarily stored on (swapped to) disk if the program is running low on main memory. Sometimes there is no need to store the page because it can just be read again, e.g., the graphical

icons for the program which are never modified. BSD keeps track of which pages have been modified by storing a “dirty flag”. A page is dirty if it has been modified and need to be stored on disk upon swap, otherwise it is clean and can just be removed.

Programs can implement these mechanisms on their own, on the level of objects. Such objects can thus be used as if they are immutable, despite that for practical reasons they modify their memory.

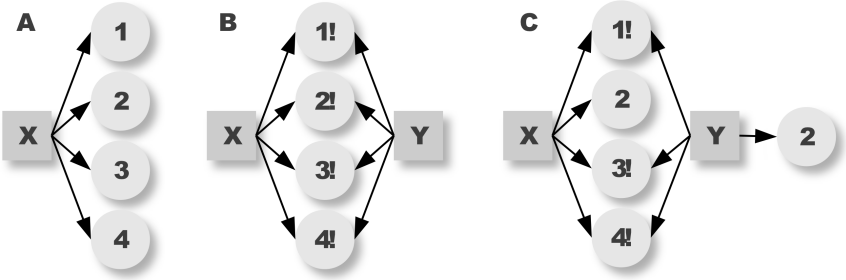

Figure 16: Copy-on-write semantics. When the object X is copied as Y, the sub-objects are marked as copy-on-write (!). In this case, the sub-object 2 is written to by the program. To give the appearance that X and Y each have their own copy, 2 is first copied and no longer marked as copy-on-write. The write operation can then proceed as usual.

1.12 VORONOI DIAGRAMS

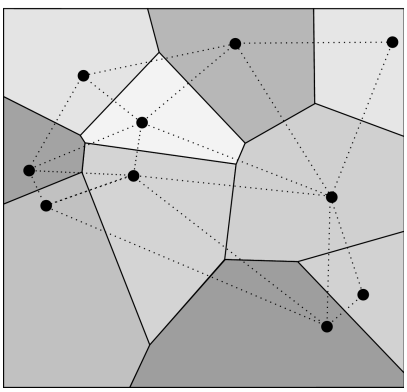

Figure 17: The Voronoi diagram (solid) and Delaunay diagram (dotted).

Voronoi diagrams are a concept within computational geometry (O’Rourke, 1998) and has many uses such as approximating cell shapes. This is used in paper II. The Voronoi diagram consists of regions (Voronoi cells), where each region is made up of the points closest to some certain site S. Formally, each region R_k consists of the points:

$$R_k = \{x \in \Omega : d(x, S_k) < d(x, S_j), j \neq k\}$$

where d is a distance function and S_j are the sites. It can be understood intuitively by looking at Figure 17. If the sites are points and d is the Eulerian distance function, then it can be shown that the regions are polyhedrons, and the boundaries are orthogonal to the lines between the neighboring sites. Another graph (the dual) can be constructed by taking each site as a vertex, and joining vertices whenever they share a boundary. This diagram is called the Delaunay diagram and is a useful compact representation of neighbor relationships.

Both Voronoi and Delaunay graphs can be extended beyond 2D to any number of dimensions. If, instead of points, arbitrary regions are used as sites, then Voronoi is useful for segmentation in image processing. There are numerous other applications, from

material science (computing atom lattice cells) to searching for files. Other applications can be found in (Okabe et al., 2000).

1.13 3D VISUALIZATION

Modern programs working with 3D data are also expected to be able to visualize it. The software developed in this thesis makes extensive use of 3D. The only way to access modern graphics hardware on all operating systems simultaneously is to use the OpenGL library (Shreiner et al., 2011). Using this system requires good knowledge of how the hardware functions. There are other libraries running on top of OpenGL (and DirectX, which is only for Windows) which hide the complexity and perform many of the usual operations, such as VTK (http://www.vtk.org).

Computers are equipped with dedicated 3D graphics hardware to speed up the calculations. A programmer wishing to use the hardware has to adapt their code to the graphics pipeline shown in Figure 18 (older hardware uses another pipeline which will not be discussed here). The software uploads coordinates, the polygons to be rendered, and textures (images on the polygons) to the graphics card. In addition, parts of the program have to run on the card (as opposed to in the CPU) and need be uploaded as well (vertex, geometry and fragment programs). These control aspects such as the camera, lighting, fog and other special effects.

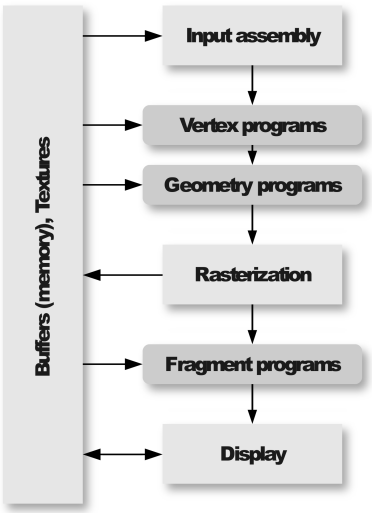

Figure 18: The modern programmable graphics pipeline

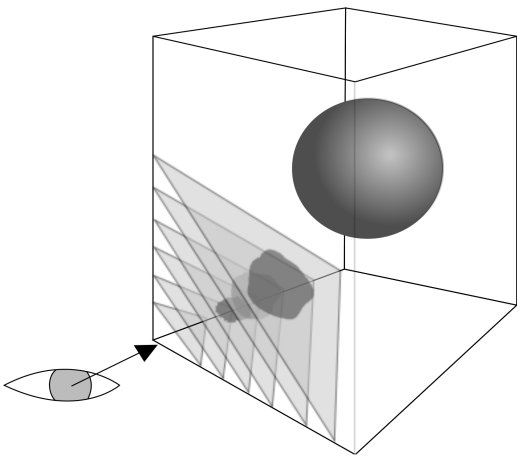

Figure 19: Volume rendering method compatible with modern hardware. The volume is sliced orthogonally to the viewer and the 3D stack is mapped as a 3D texture onto each polygon.

Volume rendering, that is, rendering an image of a 3D stack from the microscope, can be done in hardware (Engel et al., 2006). There are several methods but the one primarily used in the software developed in this thesis is shown in Figure 19. The 3D

stack can be stored as a 3D texture on the graphics card. The image can then be rendered by slicing the volume into polygons, and mapping the 3D texture onto each polygon. If the polygon is transparent wherever there is no sample, the user will be able to see through to the other planes and perceive the collected images as a 3D volume. On older graphics hardware where only 2D textures are supported, the direction of the slicing can be fixed (but this reduces the quality of the view from certain angles).

Another method of volume rendering is to calculate the outline of the 3D object, by defining the object to be the volume where the intensity is higher than a certain threshold. This is called the isosurface and requires no texturing as it is an entirely polygonal representation. The isosurface can be computed using for example marching cubes (Lorensen and Cline, 1987). While the method is fast and gives a crisp display, it is less suitable for modern hardware than 3D textures, and finding a good threshold can be difficult. It is also possible to render isosurfaces using the 3D texturing technique, hence it is more general.

1.14 FREE SOFTWARE AND OPEN SOURCE

Research requires that the results are reproducible. It has been pointed out that it is difficult to exactly reproduce results based only on verbal descriptions. Such descriptions are ambiguous and sometimes leave out important corner cases. Thus the code is needed for full reproducibility (Ince et al., 2012). Together with a need for basic functions as building blocks for new software, much of the bioinformatics software has evolved to become Free Software.

The Free Software movement started already back in the 70's but in particular with Richard Stallman and the Free Software Foundation (FSF) announcing the GNU project in 1983 (http://www.gnu.org). While there is one community, there are two major distinct views. One view is the concept of Free Software, based on the four freedoms by Richard Stallman (http://www.fsf.org):

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor.
- The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

In this view, these freedoms are considered rights on par with the right of free speech. It is hence unethical to write software that is not free. The other major view is that of pragmatists, who does not consider these as rights but still prefer software to be licensed under compatible terms. These rather talk about Open Source, with the emphasis on the source code being available with all the above freedoms. Many, especially companies, considers Open Source to be part of an alternative business model. In this thesis the terms Open Source and Free Software will be assumed to mean the same thing.

No matter the perspective, open source has grown to become part of the mainstream software. GNU/Linux is the most used operating systems on servers and clusters. Regular users have programs such as Firefox, BitTorrent, VLC and LibreOffice

(OpenOffice). Most bioinformatics tools and libraries (R, Perl, Python etc) are open source. Likewise are several programs within imaging, such as ImageJ. Hence most researchers will at some point utilize software that is open source, and should know the implications.

Software is protected by the copyright law which by default offers strong protection of the work. Copyright is obtained automatically upon creation of the work. To allow the work to be spread and used by others, a license has to be applied. The Open Source Initiative (OSI, http://www.opensource.org) maintains a list of many open source-compatible licenses. One of the most common licenses is the GNU General Public License (GPL, http://www.gnu.org), endorsed by FSF, which is copy-left – software derived from copy-left software has to maintain all the original freedoms. Another typical license is the newer BSD license (Berkeley Software Domain, one version can be found at: http://www.opensource.org/licenses/bsd-license.php), which does not even require derivative work to be open source. All open source licenses are variations of, or similar to, these two licenses.

It is possible to mix different types of licenses but care has to be taken with the details. The code produced in this thesis is licensed under the newer BSD license. However, many of the libraries the software uses are under the GPL license. This means that the program as a whole has to be treated under the GPL conditions, e.g., the source code has to be made available. But if only parts of BSD-licensed code is pulled out for reuse in another project then the BSD-conditions apply. Some open source licenses are incompatible. Therefore it is important to understand the licenses if open source code is used to develop new software or one might breach legal requirements.

2 AIMS OF STUDY

The general aim of this thesis is to develop methods to understand how different genes regulate, and are regulated, during *C. elegans* embryogenesis. The scope has grown into developing software for general microscopy and image processing.

The specific aims of the work included in this thesis are to:

1. Develop the software required to analyze the embryo (Paper I)
2. Create a model for *C. elegans* embryogenesis (Paper II)
3. Map primarily the homeobox gene expressions onto the model (Paper III)
4. Expand the applicability of the software, in this case to study the function of mitochondrial polymerase gamma (paper IV)

3 RESULTS

3.1 ENDROV – AN INTEGRATED PLATFORM FOR BIOLOGICAL IMAGE ANALYSIS AND PROCESSING

This paper summarizes my work on writing a program for analyzing embryos. Image processing software already existed when this work begun, but had limitations. SIMI Biocell (Schnabel et al., 1997) was written for Windows 3.1, did not have proper 3D support, was costly, and did not seem to be actively developed. The source code was not available and thus we could not extend it further even had we wanted to. Some in-house software – VirtualWormBase 1.0 - had been written before I joined the laboratory, but it was buggy and did not yet have the features that people needed. Further, due to changes in processors (PowerPC to Intel), compilers, and API calls, a significant amount of work would have been necessary to just update the old code. ImageJ (Rasband, 1997) would have been a natural choice as a platform, but it turned out the foundation would not support further development very well. If a lineaging plugin had been written, it would not have used much of the ImageJ core functionality, and would have been as large as ImageJ itself. Hence there was no gain over creating a stand-alone program for lineaging. Over time, I added some image processing functionality for analyzing our images. After realizing that other researchers were also running into limitations with existing software, I decided to develop the software into an entire platform for all image analysis and microscopy.

The result is the open source software Endrov (http://www.endrov.net). To cover the diverse demands, it is a modular platform with a tight coherent design. Java was chosen for several reasons; it is fast, portable and suitable for large projects. There is a range of available tools and libraries, and many microscopists have seen Java before. Endrov currently consists of about 150 000 lines of code. Most data structures and algorithms have been chosen to have the best computational complexity, making the software suitable for large data sets.

Endrov supports modern visualization and has strong integration of annotation. It is easy to write new types of annotation and integrate it with the existing views, both 2D and 3D. OpenGL was chosen for 3D because it is portable and likely to be supported for a long time. Other 3D libraries such as VTK was ruled out because of installation problems and because the program would no longer be purely written in Java. Advanced 3D rendering functionality may have required modifying VTK itself which is written in C++. Java3D was ruled out because it did not seem to be actively developed. Shaders (vertex and fragment programs) are used to perform as many operations as possible, such as to let the user adjust transfer functions (contrast/brightness) in real-time.

Endrov supports over 100 file formats by integrating the Bio-formats library, and it can interact with the OMERO image server. When work began, Bio-formats did not exist (or we did not know about it), and we created the high-performance file format OST with emphasis on metadata. As of writing, this format is being phased out in favor of OME-TIFF together with a reduced version of OST with only Endrov-specific metadata.

The application includes an interface to control hardware. By integrating with Micro-manager (Edelstein et al., 2010), Endrov is able to control many modern light microscopes. Unlike Micro-manager, the API is object-oriented which makes it easier to control many devices simultaneously. Since the hardware interface is a layer on top of Micro-manager, it is possible to include other devices. These can be written in pure Java. Other systems such as Leginon, for electron microscopes, can be integrated in the future (Suloway et al., 2005).

Endrov separates the code for the algorithms from code for the graphical user interface. The algorithms can hence be invoked from other programs, e.g., custom Java programs. Some of our early algorithms prototyping was done by linking Matlab to Endrov. This design also enables Endrov to run on clusters.

To make the software more useful for image processing, instead of image editing, Endrov has a new workflow; all image operations are based on a new graphical scripting language called Flows. In this language, image sources are connected through image processing blocks to image destinations (virtual channels). Flows can also output analysis data to CSV files and SQL databases. Operations are used through scripts rather than directly through the menus as this forces the user to document every step of the analysis.

When algorithms are developed traditionally, the programmer prototypes them using a small dataset as it otherwise would take too much time to repeatedly run the algorithm. Endrov offers almost the same speed on any size of datasets, speeding up development. It does so by extensive use of lazy evaluation. All image operations are done on the basis of single image planes, that is, the least unit of computation is calculating one plane. If the user requests to see a plane then, if the operation is 2D, only that plane calculated. This makes the calculations in many cases fast enough to be done in real-time.

To work with large data sets, several tricks are employed to avoid loading all data into memory. Most of them are analogies of how operating systems handle memory. One is reading data image on-the-fly and caching it. Another is caching computed data, but allowing it to be thrown away. Endrov investigates in what order computations in Flows are best done such as to be able to remove partial computations as soon as possible. If too much memory is used, images can be swapped to disk, reducing peak memory loads. This mechanism is general and can be used both for processing and for microscopy.

Almost every function supports undo and redo. All operations are embedded in some subclass of UndoOp. The most basic implementation of an operation only requires the programmer to implement redo, at the cost of higher-than-required memory usage. More specific subclasses exist, making assumptions that reduces the overhead. Therefore the programmer can decide the trade-off between implementation effort and performance, making implementation of new functionality faster.

Taken together, Endrov is a well performing framework, covering much of the functionality needed to acquire and analyze images. The robust engineering makes it a solid foundation for the development of new methods in microscopy for the life sciences.

3.2 SPATIO-TEMPORAL REFERENCE MODEL OF *CAENORHABDITIS ELEGANS* EMBRYOGENESIS WITH CELL CONTACT MAPS

This paper presents a new model of *C. elegans* embryogenesis in 3D over time. We mounted the embryos in such a way that the embryos are not distorted. This gives higher reproducibility as the cell coordinates are unchanged, comparing one recording to another (except for natural differences). Embryos which are mounted the traditional way on agarose are also known to rotate at some point during embryogenesis. Our new uncompressed mounting method addresses this problem, and makes it easier to track cells for identification.

DIC has been used extensively to aid in lineaging and cell identification (Bürglin, 2000; Schnabel et al., 1997; Thomas et al., 1996; Thomas and White, 1998) and thus most of our recordings were done with this technique. Some recordings also had Histone::RFP markers to aid in late identification. The Endrov module for lineaging was written and several recordings were manually annotated as far as possible. Because the quality of the DIC image deteriorates with the thickness of the sample, only the top two thirds of the embryo could be annotated. A final model of normal embryonic development complete up to the 150 cell-stage could be generated by numerically fitting the partial annotations together. Since the mounted embryos start at random angles, enough information was present to cover all sides on the model.

It is difficult to detect cell membranes which are perpendicular to the light path with DIC, since it only compares the opacity of two points in the XY-plane. However, a 3D Voronoi diagram with nucleic center points forms a good first approximation. The approximation works well for convex bodies. We have made a procedure with additional heuristics to make up for the *C. elegans* embryo being slightly concave. Hence our model includes cell contacts, for each cell, over time.

The variance in cell positioning and division timing was calculated and the cell-cell contacts estimated using a geometrical model. In this model, 1500 contacts last 2.5 min or longer. When comparing to a previous model made from a squeezed embryo, our model was found to have much higher time resolution and to be more reproducible. Of the cell-cell contacts, 40% were different from one previous model. This was largely attributed to the higher time resolution in our model, but also due to the old model being squeezed.

Finally we released a reduced version of Endrov, with just the features to view the model (The full version of Endrov was reserved for paper I).

3.3 SPATIO-TEMPORAL EXPRESSION ANALYSIS OF HOMEOBOX GENES DURING *C. ELEGANS* EMBRYOGENESIS USING A NEW IMAGING FRAMEWORK

The goal of this paper was to apply and expand our previous imaging framework to extract the gene expression patterns of primarily homeobox genes. Using PSI-blast and manual curation, we have generated a revised list of all homeobox genes for *C. elegans.* Out of 103 homeobox genes, 70 are co-orthologous to human homeobox genes. In

C. elegans, a novel class of double homedomain motifs, termed HOCHOB, was discovered. There are further a large group of genes undergoing rapid evolution. In particular there is a group of locally duplicating genes on chromosome II.

A recording technique was developed to extend the effective sensitivity range of the microscope. Instead of imaging with fixed exposure time, the optimal exposure time is calculated from the previous image and automatically applied. Therefore over- and underexposure can be avoided. The exposure time is only changed whenever the signal is about to move out of the current range, which simplifies post-processing. A linear model turns out to cause jumps in expression level whenever the exposure time is changed. This was remedied by assuming that the average expression level is unchanged during exposure time transitions.

We have found that RFP::Histone markers can cause unexpected phenotypes (paper II). The fluorescent images on our bright-field microscope also have a larger PSF, with particularly bad Z-resolution, compared to the confocal microscope which has been used in other studies (Murray et al., 2008). Thus our recordings rely on DIC for cell identification. DIC has previously been used for lineaging but we did not find it feasible to manually annotate all recordings, in particular as the non-compressed embryos are too thick for using DIC (paper II). Hence we investigated alternative methods to extract the expression patterns.

Strains with GFP promoter reporter constructs were either made or acquired from others. We recorded 440 embryos, out of which 132 recordings were of sufficient quality for further analysis, covering 68 genes. Recordings of additional genes are available as summary movies for qualitative inspection. We defined several resolutions: Single-cell (SC) assigns pixels to the closest annotated cell, that is, it uses a Voronoi approximation. Time (T) is the average expression over time. Anterior-Posterior-Time (APT) considers slices the embryo along the AP-axis over time. Finally we considered dividing the embryo into voxels (XYZ) over time.

Normalizing time, in case not all cells were annotated, was done by manually annotating several stages of development (4 cells, gastrulation, comma and 2-fold). Time was linearly interpolated between these stages, and extrapolated outside. Further the positions of the first 4 cells were annotated to define a coordinate system. Non-compressed embryos do not rotate (or rotates little), motivating the assumption that this coordinate system stays fixed. We also tried to super-impose our model of normal development to obtain approximate single-cell resolution.

All recordings were mapped to the same reference model and clustered using Pearson, l_2 and the Manders coefficients. The different metrics and coordinate systems were compared by assessing the quality of the clusterings. A clustering is good if recordings of the same gene cluster together. APT and Pearson correlation is the most reproducible method. Our approximate single-cell resolution works remarkably well given its crudeness.

Endrov has been extended to extract the expression patterns as described, and to visualize them on the model. The patterns are available for viewing on the model (paper

II), individually or overlapped for comparison. The expression can also be visualized on the lineage. To make it easier to see which features expressing cells give rise to, the full lineage has been imported from Wormbase in addition to a 3D model of the larvae (Chris Grove, http://caltech.wormbase.org/virtualworm/). The user can select an interesting cell and highlight its daughter cells, on the lineage and in 3D.

In conclusion, we have designed one of the most sophisticated lineaging software tools to date. It supports fast manual annotation of embryos and can import automatically lineaged embryos from other programs, to be adjusted. Expression patterns can be extracted and compared with the expression profiles we have generated so far. We have shown that approximate single-cell resolution can be achieved without lineaging. There are also useful alternative representations which might be useful for other model organisms lacking a lineage. All-in-all, our Endrov-based framework should be a strong foundation for further investigation of *C. elegans* embryogenesis.

3.4 MITOCHONDRIAL DNA LEVEL, BUT NOT ACTIVE REPLICASE, IS ESSENTIAL FOR *CAENORHABDITIS ELEGANS* DEVELOPMENT.

In this paper, Endrov and our microscopy setup was used for a specific purpose. The article concerns the function of mitochondrial polymerase gamma (*polg-1*) in *C. elegans*. A mutant with predicted partial deletion of *polg-1* was analyzed for phenotypes. The mtDNA copy number was measured by qPCR at several stages throughout development. While within a factor 2 lower during embryogenesis, the copy number increases tremendously in wild-type during the larval stages but not in the mutant. To evaluate the somatic versus germline component of the mtDNA copy number, a proliferation deficient *glp-4* strain was used. The germline was found to contribute greatly to the copy number in wild-type, and the *polg-1* mutant follows the same trend as the *glp-4* mutant. However, the *polg-1* mutant fails to maintain the copy number throughout adulthood. The mitochondrial transcript levels were measured in adulthood. The *polg-1* mutants have decreased transcript levels, but not as decreased as the mtDNA copy number. This matches the expectations of up-regulation of mitochondrial RNA and protein stability as a consequence of mtDNA deficiency. Measurement of defecation rate shows that *polg-1* mutants have shorter cycles at increased age compared to wild-type. This is in contrast with other mutants with affected mitochondrial metabolism, having longer cycles.

The morphology of mitochondria was studied using the muscle-specific marker, *myo-3*::MTS::GFP. The *polg-1* mutant has disorganized and fused mitochondria. We think this could be an adaptive response. Transmission electron microscopy (TEM) showed a small increase in the number of mitochondria at day 6 of adulthood in the mutant. The mutant gonad further has reduced number of mitochondria, and those remaining are enlarged and deformed. The number is reduced even further at later life-stages. The Endrov software was used to visualize the gonad morphology in 3D. A volume rendering of a Hoechst 33258 staining and annotation of the cells shows that the rachis are gone and hence the nuclei are spread equally over the gonad. Endrov was also used to count the number of nuclei in arrested embryos. The progeny of the mutants arrests early. Our microscopy setup (paper II) was used to measure the speed of development. Mutant progeny grows exponentially slower after the 4-cell stage. The offspring could not be rescued by crossing mutant hermaphrodites with wild-type males. The mutants die by

gonadal and intestinal protrusion, occurring after the 6th day of adulthood. The mutant phenotype can be reproduced by RNAi over three consecutive generations. Moreover, such phenotype could be partially suppressed by crossing with another strain having egg-laying deficiency.

Our conclusion is that the maternal contribution of mitochondria is sufficient for the next generation to undergo embryogenesis. The bulk of mtDNA replication happens in the gonad. In most other model systems, polymerase gamma-deficient mutants do not survive embryogenesis. Thus this paper establishes *C. elegans* as an interesting model system for mtDNA inheritance and maintenance.

4 DISCUSSION

4.1 FUTURE IMPROVEMENTS TO MICROSCOPY

The holy grail of embryonic recordings is to be able to extract expression patterns from single cells. Our software Endrov provides good means of annotating cells manually, or improving existing annotations. Superimposing the wild type model works well as a first approximation, but it is insufficient for studying or screening for developmental defects. To improve precision and annotation speed, lineaging has to be made automatic.

Automatic lineaging with DIC has been tried by several groups (including ours). The best attempt so far reaches only 24 cells (Hamahashi et al., 2005). Given that it requires training for someone to do it manually, it appears extremely difficult to move beyond 30 cells. DIC has to be replaced with something else. The most straight-forward method is using the RFP::Histone mutants on a microscope with better resolution (smaller PSF). This been done and an automatic lineaging algorithm exists (Murray et al., 2008). There is however a problem with this method; our RFP-strain accumulates random phenotypes over time. Others have not seen this problem in their strains (Zhirong Bao, personal communication). Unless another RFP-strain can be made and proven to be without this problem, it might be possible to compensate by producing additional recordings, and also creating more strains for a given gene of interest.

There are alternatives to using RFP::Histone. A marker for the outer membrane, rather than the histones, would likely have less epigenetic/mutagenic effects. It would also make it possible to map cell-cell contacts without approximations. It might be worth investigating label-free microscopy techniques such as CARS (Coherent anti-Stokes Raman spectroscopy). The usefulness depends on if it is possible to detect suitable structures (such as membranes) with good resolution, at high enough speed.

In this thesis we have primarily considered homeobox genes. In general, these are expressed rather late toward comma stage. Annotating cells late near this stage is unfortunately difficult and tedious. Further, beyond the comma stage the worm starts twitching as it elongates. At this point our microscope is too slow to acquire useful images. Also laser-scanning confocal microscopes are too slow; at minimum a spinning disk confocal is required. Better yet is to investigate SPIM (Voie et al., 1993) as it has better promise of handling the thick sample, and may have good enough Z-resolution without using a confocal technique. Indeed this has already been used on *C. elegans* embryos (Wu et al., 2011). This RFP::Histone-based setup scans 30 volumes per minute and is claimed to be fast enough for automatic lineaging. From the images I have seen, I have some doubts about the noise level but it may be possible to improve the microscopy even further.

An alternative way to reduce motion blur is to slow down the sample. Although it may affect the expression patterns, *C. elegans* embryos can be cooled and are viable down to 8°C (Rabin and Podbilewicz, 2000), but also a moderate cooling may be sufficient. If the purpose is only to reduce the motion blur during acquisition then one could even envision that the cooling could be done temporarily with a fast-cycling Peltier device.

4.2 INTERPRETATION OF EXPRESSION PATTERNS

To what extent can the transcriptional reporters be trusted? The underlying assumption is that the gene reporter has the same dynamics as the endogenous gene, but there are many reasons why this may not be entirely true:

- The transgene does not have all regulatory elements: To produce the reporter construct, 3kb of the upstream sequence is taken in our case and assumed to contain all regulatory elements. In *C. elegans*, most elements are indeed nearby, although in most other animals, promoters tend to be larger. Moreover, any intronic or 3' regulatory elements would be missing. The missing elements could also be negative regulatory elements, so their absence could lead to ectopic expression patterns.
- Differences between RNA and protein levels: The protein of the gene of interest remains active for a while after the gene has been down-regulated. The function will cease only once the protein has been degraded. However, the RNA and protein of the reporter gene might be degraded at a different rate. Similarly, a transcribed RNA may be prevented from translation by, e.g., micro-RNAs.
- Epigenetics may also affect the expression pattern. If an adult worm is starved then its embryos have a weaker signal (experience from the lab). This was avoided by proper maintenance, but it suggests that there might be additional epigenetic factors such as temperature, light exposure, pH and the buffers used.
- Low expression levels: Insufficient amounts of reporter protein may be created, because of low up-regulation of the gene, or too low copy number of the reporter construct. In this case the background fluorescence of the embryo can obscure the signal.
- Early stages of development are largely driven by maternal contribution and hence the gene might not be expressed during embryogenesis – but the corresponding protein might still be present and active.

Thus the data has to be interpreted with care. Both false positives and negatives are possible – there can be no expression at all, or there can be expression when there should be none. The *C. elegans* community is aware of these limitations (Hunt-Newbury et al., 2007), so it is important to view each expression pattern as that of a particular reporter construct.

If a gene is of particular interest then the expression should be confirmed with an alternative method. The first stop is the existing microarray data which can verify if a gene is expressed at a certain developmental stage, but not in which cell (Yanai and Hunter, 2009). If the transcriptional reporter is regulated improperly then a translational reporter might be more accurate. Recombineering in fosmids can be used to generate translational reporter constructs with exons, introns and large regions beyond 5' and 3' preserved (Dolphin and Hope, 2006). Another approach is to use FISH or antibodies but, as with microarrays, these do not offer the same time-resolution.

As previously mentioned, the GFP measurement is on the protein level which is only indirectly related to gene expression. If the degradation is infinitely fast then the signal will disappear as soon as the gene is no longer expressed (and the mRNA has been degraded). The degradation can be sped up by integrating a PEST-motif into the promoter construct (Corish and Tyler-Smith, 1999). A complementary method is to model the

reaction kinetics and calculate backwards to find the expression. A very simplistic model for the expression of gene i in cell c is

$$\frac{\mathrm{d}P_{i,c}}{\mathrm{d}t} = -\alpha_{i,c}P_{i,c} + f_{i,c}(t)$$

where P is the protein concentration, α is the decay rate and f is the rate new GFP is created. The rate of protein production can then be calculated as:

$$f_{i,c}(t) = \frac{\mathrm{d}P_{i,c}}{\mathrm{d}t} + \alpha_{i,c}P_{i,c}$$

The problem is that the decay rate is unknown. It is possible to work around this by assuming that the source is always positive $f_{i,c}(t) > 0$, that there is no protein in the beginning of embryogenesis, $P_{first\ cell,c}(0) = 0$, and the protein concentration reaches steady state $\lim_{t\to\infty} P'_{i,c}(t) = 0$. The source f is then the level of translation. Either mRNA is considered to be fast-regulated, or the differential equation can be made second order to also model translation. The predicted result is however never better than the model; if the need of the modeling can be avoided then one source of error can be excluded. Therefore the modeling was excluded from paper III.

4.3 DISSECTING THE REGULATORY NETWORK

In this thesis we did not collect enough data to be able to hypothesize any regulatory network. *C. elegans* has about 1000 transcription factors (Reece-Hoyes et al., 2005). We covered about 7% with out analysis. The method does however only provide indirect evidence of interaction; what should be done once enough expression patterns have been acquired? One approach is to computationally infer the network. While there are many types of models to choose from, boolean models are simple and have been shown to be, in theory, flexible enough to model the *C. elegans* lineage (Larsson et al., 2011). This modeling method highlights some of the predictive limitations. Given a cell and its daughter cells and which genes are expressed, potential pathways can be inferred (Table 3).

Parent	Potential pathways
A, B	A → C
Daughter	B → C
A, B, C	A & B → C

Parent	Potential pathways
B, D	B → C
Daughter	D → C
B, C, D	B & D → C

Table 3: Genes (A..E) active in two cells cell and their daughter cells, and inferred pathways in each case.

For a given cell, there are many potential pathways with only 4 genes. The number of valid pathways is reduced by assuming that they are present in every cell. The simplest pathway predicted in this example is B → C, but this does not exclude other pathways, e.g., (A or D) → C. Hence, even when the expression patterns of all genes are known, further verification will be needed.

The easiest way of verifying a single pathway in *C. elegans* is by using RNAi and existing GFP expression reporters, though this only provides an epistatic model. For transcription factors in particular, ChIP-seq (Chromatin immunoprecipitation sequencing)

is a well established method for finding binding sites, and is able to verify several targets in a single experiment (Johnson et al., 2007). The SELEX (Systematic Evolution of Ligands by Exponential Enrichment) technique finds binding motifs *in vivo*, but can be scaled up to analyze many transcription factors in parallel (Jolma et al., 2010). Interactions other than protein-DNA lack similarly efficient techniques; hence there is plenty of research left to be done before all the regulatory pathways have been elucidated.

5 FUTURE PERSPECTIVES

5.1 NEW TECHNIQUES AROUND THE CORNER?

There are, by current estimates approximately 20 000 genes in *C. elegans*. In addition, it would be interesting to compare the patterns with related nematodes such as *C. briggsae*, *C. remanei* and maybe *P. pristiounchus*. If we wish to do 3 recordings of each gene (ignoring that some recordings will fail) then this means more than 200 years of microscopy. Moreover the strains have to be made and integrated. While it seems infeasible, in the context of the *C. elegans* research community, numerous laboratories can contribute so that such a goal is within reach. Nevertheless, embryonic recordings are unsuitable as a method for genome-wide screenings. Their value lies in the high time-resolution for a particular gene. Thus it mainly makes sense if there is a limited set of interesting genes, and for a limited number of perturbations, such as with RNAi.

An emerging way to examine gene expression at the single cell level is single-cell RNA sequencing (RNAseq). It is possible to multiplex the cells such that the throughput limitation is only by how many reads the sequencer can do in a single run (Islam et al., 2011). The problem with the technique is that it is currently difficult and slow to extract single cells from embryos. However, assuming there is a method to pull apart all the cells intact – then all genes can be screened at once, in all species, within a year, with a time resolution of every cell division. RNAseq has several limitations however, and does nevertheless not replace embryonic recordings. The signal-to-noise is low and the method would likely be slower for studying a single specific gene. The procedure to isolate cells may affect the expression pattern and studying expression dynamics within the lifetime of a cell will be difficult. Hence embryonic recordings will remain useful in the future. We wish to make our recording tools available to as many labs as possible, as the community can record many more expression patterns than if a single lab were to do it.

5.2 ADDING THE POST-EMBRYONIC GENE EXPRESSION PATTERNS

So far the attention has been limited to the embryos but development continues also after hatching. Since larvae are mobile and need to be fed, the current methods for embryonic recordings cannot be applied directly.

The localizome project uses the COPAS worm sorter, a flow cytometer also capable of scanning the expression along the body of larvae (Dupuy et al., 2007), to produce “chronograms”. One gene can be profiled in about an hour (own estimate) after worms with reporter constructs have been grown. The bottleneck is producing the strains. The technique has serious limitations: expression is given by location along the body instead of in a specific cell. This is an issue in particular for studying the neurons as they are densely located and the profiles might be unable to resolve nearby cells. It does however offer promise as a quick screening tool. This technique does not follow the expression of a single worm over time, but rather measures the population average over time. This does not pose a problem as long as there is no considerate individual variation.

Individual cells in the larvae can also be studied using microscopy (Long et al., 2009). Their analysis detects the worms, stretches them, and assigns names to the cells

based on a probabilistic model. A large percentage of the cells are detected accurately and the technique offers promise for further improvement. The quality of data is much higher than from the COPAS worm sorter since expression can be assigned to single named cells. Although the paper does not consider it, the technique can be extended to have time dynamics by tracking the worms on the plate. A faster alternative is to measure the population average the way the COPAS does it, by acquiring static snapshots of many individuals and correlating the length of the larvae to the age. A high throughput of worms can be obtained using microfluidics, which have recently been applied to *C. elegans*. This can be combined with sorting and microdissection experiments *(Chung et al., 2008; Guo et al., 2008)*.

5.3 THE FUTURE OF ENDROV

While Endrov of course has some caveats, I find that the design has turned out well. The choice of language, in particular, is very fortunate as Java is flourishing in the community. A missing piece has been how to write image processing code which is both fast and generic. Java implements generics (types over types, such as Set<Integer>) by reification (all generics type information is dropped after compilation), and it does not statically inline statements (a sort of copy-and-paste speeding up small functions). Oracle Java (http://openjdk.java.net) compensates for this by having optimistic optimizations. For example, it guesses the type and does run-time inlining. If the assumption was wrong then the optimization is undone and it falls back on a safe (but slow) method of execution. Still, it is hard for me as a programmer to predict when the optimizations will be performed and careful benchmarking is needed. The feature is also strongly dependent on which virtual machine is used – Google Android has its own virtual machine, Dalvik (http://source.android.com), doing other optimizations. Therefore the only way to predictably get good performance might be byte code rewriting, doing the inlining "manually". There is however another aspect to this and that is avoiding to reinvent the wheel. Another project, ImgLib (Preibisch and Saalfeld, 2010), based on generics with careful benchmarking, will be used by ImageJ2. Joining effort and integrating it in Endrov is a step in the right direction and avoids having two different libraries in the image processing community.

Flows have shown to be very useful for combining image processors. It remains to be seen if they are sufficient for most users needs, or if additional scripting languages are indeed needed. The complexity of programming increases by necessity once the task itself is more involved. I think that using Java instead of an intermediate scripting language is more beneficial in the long-term, but in the end it will be about user acceptance. Laziness is independent of the graphical representation and should now be proven to be helpful for handling large amounts of data, motivating the complexity of implementation.

Endrovs biggest problem at the moment is the limited number of users. Since the software was not promoted early on, out of fear that the journals would reject the article, other projects have now grasped a large share of the market. Additional promotion and integration with other programs, such as ImageJ2 (http://developer.imagej.net), will make our software more available. Creating a niche within the *C. elegans* community will be essential for the survival of Endrov. Otherwise it is best to let it be absorbed into other ongoing projects.

5.4 ACCESS TO MICROSCOPES

A big problem with accessing microscopes using open source software is that many vendors do not want to provide access to their hardware. Because microscopy hardware does not follow regular API:s they can restrict access by not giving out the specifications describing how to interact with the hardware. Usually communication is done by simple text commands sent using an RS232 serial interface. A typically stated reason is legal liability but I believe this is just an excuse; only systems integrators such as Zeiss, Nikon and Olympus are causing problems. The real reason is that they are focused on vertical integration – that is, they are selling total solutions, and this includes the software to run the system. Their actions are however holding back research, because their software imposes limits on what a customer can do. The community, as customers, have to insist that microscope companies stop with this practice.

The first and foremost solution is to require full specifications in the tenders for new microscopes. Luckily we can do more than so – we can reverse engineer the hardware and make the specifications ourselves, without the companies consent. The laws in the EU and the US are similar on this topic. This is an excerpt from the "Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs" (EU, 2009):

> Article 5 - Exceptions to the restricted acts
>
> …
>
> 3. The person having a right to use a copy of a computer program shall be entitled, without the authorisation of the rightholder, to observe, study or test the functioning of the program in order to determine the ideas and principles which underlie any element of the program if he does so while performing any of the acts of loading, displaying, running, transmitting or storing the program which he is entitled to do.
>
> Article 6 - Decompilation
>
> 1. The authorisation of the rightholder shall not be required where reproduction of the code and translation of its form within the meaning of points (a) and (b) of Article 4(1) are indispensable to obtain the information necessary to achieve the interoperability of an independently created computer program with other programs, provided that the following conditions are met:
>
> (a) those acts are performed by the licensee or by another person having a right to use a copy of a program, or on their behalf by a person authorised to do so;
>
> (b) the information necessary to achieve interoperability has not previously been readily available to the persons referred to in point (a); and
>
> (c) those acts are confined to the parts of the original program which are necessary in order to achieve interoperability.
>
> 2. The provisions of paragraph 1 shall not permit the information obtained through its application:
>
> (a) to be used for goals other than to achieve the interoperability of the independently created computer program;
>
> (b) to be given to others, except when necessary for the interoperability of the independently created computer program; or
>
> (c) to be used for the development, production or marketing of a computer program substantially similar in its expression, or for any other act which infringes copyright.

Hence it is possible to study the existing programs for these microscopes to figure out the specification. Because the protocols are so simple, all it takes is a serial port sniffer program (such as PortMon for Windows) and a little bit of dissecting. To make reverse engineering legally waterproof, some procedures should be followed. It is more than

sufficient if the commands can be figured out by the local users and sent to someone else, e.g., the Micro-manager team (Lindberg 2008), who can then implement a suitable driver.

5.5 DEVELOPING SOFTWARE IN A NATURAL SCIENCE ENVIRONMENT

With the massive amounts of data generated, and complicated cross-references between the records, biology is in dire need of a solid foundation for handling the data and conducting the computations. Ironically, most biologists I have met have had little interest in learning computer science theory, which is essential for designing and maintaining large computer systems. Likewise, most software engineers have little interest in applications such as biology. This is a serious problem getting worse every day. We will have to fight issues, such as there being multiple versions of the sequence file format FASTA, but no way to tell them apart, just because the designers did not create a file header, otherwise considered best practice. For many years biologists considered it bad to compress images (some probably still do) and struggled with hard disk limitations. Albeit anyone who has used a ZIP-file should know this need not be true. Data can only be used to its full potential if the software is adequate for the task, but it will not be, as long as no one cares.

Another problem is how software engineering is valued within the natural sciences. The publication of a program typically has little value beyond being “a tag” needed for accumulating citations, which is becoming increasingly important for funding. Moreover, there is seldom an article written beyond the first release. Where can new versions be published? Where can bug fixes that save hundreds of working hours be published? I think there is little value in such an article, and the current process of review would take too much effort. However, a system is needed to give credit to developers, and to boost their bibliometric ranking such that they can compete with other researchers. Currently software maintenance is hidden in biological research projects, but this gives the worst incentive: Do the minimum amount of work to get the software to work. Do not bother about engineering for the future, and by all means, do not work on features not needed for the project. No wonder why so many of the programs are old, ugly and buggy, and consequently similar applications are coded again and again just to address some issue.

For some software projects, the current funding model sometimes works. It is possible to secure funding for maintenance if the authors of the first publication can survive on the continued citations. This is however in contrast to how open source works, when people can join and be part of a project as they please. Except for the respect of the peers, in this system, a programmer has little to gain from maintaining the project of someone else.

The solution I propose is “micro-publications” – simply traceable elements without peer-review for acceptance, just stating what has been done. These can then be cited and measured bibliometrically. However, I would suggest the same drastic change for the rest of the biological sciences – individual experiments and hypotheses can become micro-publications. In such a system, there is little risk of scooping (data is not kept secret for several years) and the results would become available to others quicker. Articles would account less bibliometrically, and rather be reviews of micro-publications. Hopefully the articles would become fewer and be of higher quality, as there is no longer

an incentive for the “least publishable unit”. A challenge is how to ensure the quality of the data. Peer-review can be reinstated as a web of trust, whereby anyone can claim that a piece of data (or even another researcher) is trustworthy or not. The researcher carrying out the experiment should also state his/her opinion about the result. It is then possible to to infer probabilistically, from several researchers, and their credibility in turn, how much the data can be trusted. A possible scenario is shown in Figure 20. A web of trust model is already widely used for cryptography, see for example (Peterson and Davie, 2011).

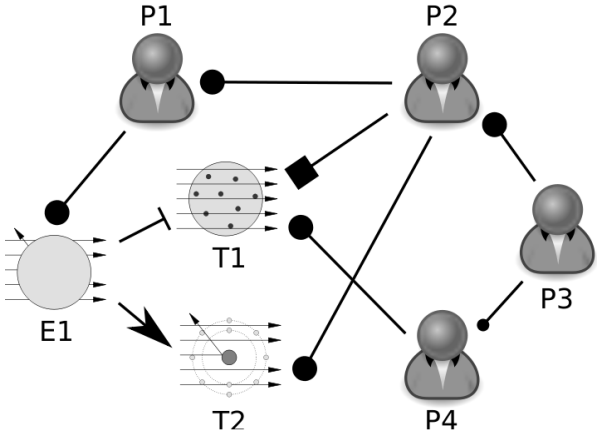

Figure 20: The web of trust in a micro-publishing system. An experiment E1 has been carried out by P1, who trusts the result (line with filled circle). His theoretical colleague P2 has analyzed the data and come up with a model T2. The model is favored if the experiment is correct (results depend on each other). An older model T1 is not favored, and P2 does not believe in it either. A competitor, P4, still believes in the old model though. A journalist P3 with no insight in the field wonders what is the most credible hypothesis (from her perspective). She trusts P2, who is a professor, more than P4, who is an amateur researcher. Thus, using the web of trust as a probability model with dependencies, she can find out how much she should trust the micro-publications and that T2 is indirectly the most credible theory to her. The web of trust can also be used to calculate a result impact factor, but then emphasizing on relations such as between E1, T1 and T2. An additional relation, “research inspired by” should then also be added.

Implementing micro-publications is less of a technical challenge, and more a political one. Without pressure from funding agencies, major journals could reject papers where data has already been dispersed as micro-publications. In addition, if only a handful scientists joins the initiative, other groups could use their results without contributing [tragedy of the commons, (Nisan et al., 2007)]. Whatever is the best way to proceed, I urge fellow scientists to think beyond their own projects and realize that there is a bigger picture. The system in which we are working is affecting our abilities to do good research. This has once been summarized:

> Geeks like to think that they can ignore politics; you can leave politics alone,
> but politics won't leave you alone.
>
> -- Richard Stallman

6 ACKNOWLEDGMENTS

To start off, I want to thank my parents for giving me the opportunity to follow my calling. Also great thanks to my friend and mentor Torbjörn Lundh – I might not have been doing research without you! There are plenty of great people down in Göteborg; Bernt Wennberg along with micket, tigge, antiirc & Jimmy.

I have appreciated the company of past and present *C. elegans* people. Peter and his group (Prasad, Gabi, Evgeni, Karin, Gilbert, Agata & Juan Carlos, Ninwa, Kristian and Debora) as well as Thomas and his group (Yong-Guang, Krishanu, Daniela, Kiyotaka "our ninja", Kedir, Deffiz, Konstantin, Dasha & Lois). Deffiz, if you're staying, maybe you should inherit my beige waistcoat? (ponytail wig can be arranged) Also, I think you're now promoted one step in the Novum beer brewing association :) My Sisyphean endeavor would have impossible without help. I hope Martin did not become addicted to coffee doing all the boring job. Last but not least, thanks to Jürgen and Iv(ana), both for the help and for making this a great time. Ivana, spoil Jürgen plenty!

I had the pleasure to supervise several of my own students; Kiarash, Ahmed, Adeel, Adam, Darvid (Arvid & David), Niklas, Javier, Christian, Stanislav and Kristina. I wish you all good luck with whatever path you choose to follow.

In particular, thanks to our choanozoa. I hope you've learned as much as I have!

Thanks to everyone at Novum, too many to mention; Andrew, Arttu and Brigitte among others, who hurt me on a weekly basis. Elisa, Lise-lotte, Pauliina & Tassos, Lena & Ylva, Helmi, Marco, Elo (who is probably trying to kill me with poisonous mushrooms). Mehrdad, John, and Jan-Peter lurking at the other end of the building, always with interesting problems. Sylvie who is protecting her beloved microscopes from me. David who is now located too far away. Our local IMDB-backup, Milica. Best of luck in the future Yongtao! Also regards to the poor souls who got left behind at Södertörn. I hope you make it out alive...

What would my PhD have been without some pastimes? Thanks to all those who I organized Nov2k™ together with; Anita, Lars, Sulaiman, Kristina$_0$, John and especially Olof. I hope we can do more entertaining things together! Kristina, did I ever tell you that you're crazy? Otherwise you have it stated here now in a peer-reviewed thesis. Thanks also for all the parties we organized! This also includes James, Aida & Rezin. If you're ever up for more, I'm in :)

I have had support of many friends outside work as well. Lollan (my personal shopper) and Freja (an ever good reason to visit Södertälje). Ricardo, if we both ever find some free time, we should do something but work together! My friends in Uppsala, Lois & Stefano (Live long and prosper) and Jan & Beatriz (thanks for the lessons, hope we can play together some time in the future). Not to forget the chaps up in Borlänge and even more rural areas.

I should thank Anna in particular. How would I have made it through without you? I hope I can return everything you have done for me one day!

This PhD was brought to completion under the supervision of Thomas Bürglin and Staffan Strömblad.

Finally, thanks to everyone else tirelessly hacking on free software, for fun or profit. It feels great to be part of this revolution!

The work in this thesis was sponsored by the Swedish Research Council (Vetenskapsrådet), the Swedish Foundation for Strategic Research and the Center for Biosciences. We are grateful for strains received from David Baillie through the Genome Canada collaboration, as well as to Ian Hope, Ding Xue, Peter Okkema, Peter Swoboda and the *Caenorhabditis* Genetics Center (funded by the NIH National Center for Research Resources), who provided additional strains.

7 REFERENCES

Andrews, G. R. (2000). Foundations of multithreaded, parallel, and distributed programming (Addison-Wesley).

Bay, H., Ess, A., Tuytelaars, T., and Vangool, L. (2008). Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding *110*, 346–359.

Billeter, M., Qian, Y. Q., Otting, G., Müller, M., Gehring, W., and Wüthrich, K. (1993). Determination of the nuclear magnetic resonance solution structure of an Antennapedia homeodomain-DNA complex. Journal of molecular biology *234*, 1084–1093.

Bürglin, T. R. (2000). A two-channel four-dimensional image recording and viewing system with automatic drift correction. Journal of Microscopy *200*, 75–80.

Bürglin, T. R. (2011). Homeodomain subtypes and functional diversity. Sub-cellular biochemistry *52*, 95–122.

Caswell-Chen, E. P., Chen, J., Lewis, E. E., Douhan, G. W., Nadler, S. A., and Carey, J. R. (2005). Revising the standard wisdom of C. elegans natural history: ecology of longevity. Science of aging knowledge environment : SAGE KE *2005*, pe30.

Chalfie, M., Tu, Y., Euskirchen, G., Ward, W. W., and Prasher, D. C. (1994). Green fluorescent protein as a marker for gene expression. Science (New York, N.Y.) *263*, 802–805.

Chung, K., Crane, M. M., and Lu, H. (2008). Automated on-chip rapid microscopy, phenotyping and sorting of C. elegans. Nature methods *5*, 637–643.

Corish, P., and Tyler-Smith, C. (1999). Attenuation of green fluorescent protein half-life in mammalian cells. Protein Engineering Design and Selection *12*, 1035–1040.

Cox, G. (2007). Optical imaging techniques in cell biology (CRC Press).

DICOM Standards Committee (2011). The 2011 medical DICOM standard. Available at: ftp://medical.nema.org/medical/dicom/2011/ [Accessed December 12, 2011].

Damas, L., and Milner, R. (1982). Principal type-schemes for functional programs. In Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages - POPL '82 (New York, New York, USA: ACM Press), pp. 207–212.

Darwin, C. (1859). Origin of Species. Library *138*, 30.

Data Structures and Algorithms in Java (2010). (Wiley).

Dolphin, C. T., and Hope, I. A. (2006). Caenorhabditis elegans reporter fusion genes generated by seamless modification of large genomic DNA clones. Nucleic acids research *34*, e72.

Dupuy, D., Bertin, N., Hidalgo, C. A., Venkatesan, K., Tu, D., Lee, D., Rosenberg, J., Svrzikapa, N., Blanc, A., Carnec, A., et al. (2007). Genome-scale analysis of in vivo spatiotemporal promoter activity in Caenorhabditis elegans. Nature biotechnology *25*, 663–668.

EU (2009). Directive 2009/24/EC of the European Parliament and of the Council of 23 April 2009 on the legal protection of computer programs.

Eberharter, A., and Becker, P. B. (2002). Histone acetylation: a switch between repressive and permissive chromatin. Second in review series on chromatin dynamics. EMBO reports *3*, 224–229.

Edelstein, A., Amodaj, N., Hoover, K., Vale, R., and Stuurman, N. (2010). Computer control of microscopes using μManager. Current protocols in molecular biology *Chapter 14*, Unit14.20.

Engel, K., Hadwiger, M., Kniss, J. M., Rezk-Salama, C., and Weiskopf, D. (2006). Real-time volume graphics (A K Peters, Ltd.).

Evans, T. C. (2010). Transformation and microinjection. WormBook, ed. The C. elegans Research Community, WormBook, doi/10.1895/wormbook.1.7.1, http://www.wormbook.org.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). Design patterns: elements of reusable object-oriented software (Addison-Wesley).

Goldstein, B., and Hird, S. (1996). Specification of the anteroposterior axis in Caenorhabditis elegans. Development *122*, 1467–1474.

Guo, S. X., Bourgeois, F., Chokshi, T., Durr, N. J., Hilliard, M. A., Chronis, N., and Ben-Yakar, A. (2008). Femtosecond laser nanoaxotomy lab-on-a-chip for in vivo nerve regeneration studies. Nature methods *5*, 531–533.

Hager, G., and Wellein, G. (2010). Introduction to High Performance Computing for Scientists and Engineers (CRC Press).

Hall, D., and Altun F, Z. (2008). C. elegans atlas (CSHL Press).

Hamahashi, S., Onami, S., and Kitano, H. (2005). Detection of nuclei in 4D Nomarski DIC microscope images of early Caenorhabditis elegans embryos using local image entropy and object tracking. BMC bioinformatics *6*, 125.

Hansen, P. C., Nagy, J. G., and O'Leary, D. P. (2006). Deblurring images: matrices, spectra, and filtering (SIAM).

Harvey, W. (1651). Exercitationes de generatione animalium.

Hayes, M. H. (1996). Statistical digital signal processing and modeling (John Wiley & Sons).

Hindley, R. (1969). The Principal Type-Scheme of an Object in Combinatory Logic. Transactions of the American Mathematical Society *146*, 29–60.

Hobert, O. (2002). PCR fusion-based approach to create reporter gene constructs for expression analysis in transgenic C. elegans. BioTechniques *32*, 728–730.

Hooke, R. (1665). Micrographia: or, Some physiological descriptions of minute bodies made by magnifying glasses (London).

Hope, I. A. ed. (1999). C. elegans - A practical approach (Oxford University Press).

Huang, B., Bates, M., and Zhuang, X. (2009). Super-resolution fluorescence microscopy. Annual review of biochemistry *78*, 993–1016.

Hunt-Newbury, R., Viveiros, R., Johnsen, R., Mah, A., Anastas, D., Fang, L., Halfnight, E., Lee, D., Lin, J., Lorch, A., et al. (2007). High-Throughput In Vivo Analysis of Gene Expression in Caenorhabditis elegans. PLoS Biology *5*, e237.

Ince, D. C., Hatton, L., and Graham-Cumming, J. (2012). The case for open computer programs. Nature *482*, 485–488.

Islam, S., Kjällquist, U., Moliner, A., Zajac, P., Fan, J.-B., Lönnerberg, P., and Linnarsson, S. (2011). Characterization of the single-cell transcriptional landscape by highly multiplex RNA-seq. Genome research *21*, 1160–1167.

Johnson, D. S., Mortazavi, A., Myers, R. M., and Wold, B. (2007). Genome-wide mapping of in vivo protein-DNA interactions. Science (New York, N.Y.) *316*, 1497–1502.

Jolma, A., Kivioja, T., Toivonen, J., Cheng, L., Wei, G., Enge, M., Taipale, M., Vaquerizas, J. M., Yan, J., Sillanpää, M. J., et al. (2010). Multiplexed massively parallel SELEX for characterization of human transcription factor binding specificities. Genome research *20*, 861–873.

Jones, S. L. P. (1992). Implementing lazy functional languages on stock hardware: the Spineless Tagless G-machine. Journal of Functional Programming *2*, 127–202.

Kamath, R. S., Fraser, A. G., Dong, Y., Poulin, G., Durbin, R., Gotta, M., Kanapin, A., Le Bot, N., Moreno, S., Sohrmann, M., et al. (2003). Systematic functional analysis of the Caenorhabditis elegans genome using RNAi. Nature *421*, 231–237.

Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., and Tanabe, M. (2011). KEGG for integration and interpretation of large-scale molecular data sets. Nucleic acids research *40*, D109–D114.

Kanehisa, M., and Goto, S. (2000). KEGG: kyoto encyclopedia of genes and genomes. Nucleic acids research *28*, 27–30.

Kimble, J., and Hirsh, D. (1979). The postembryonic cell lineages of the hermaphrodite and male gonads in Caenorhabditis elegans. Developmental Biology *70*, 396–417.

Kotsiantis, S. B. (2007). Supervised Machine Learning : A Review of Classification Techniques. Informatica *31*, 249–268.

Kurz, L., and Benteftifa Hafed, M. (1997). Analysis of variance in statistical image processing (Cambridge University Press).

Kvilekval, K., Fedorov, D., Obara, B., Singh, A., and Manjunath, B. S. (2010). Bisque: a platform for bioimage analysis and management. Bioinformatics (Oxford, England) *26*, 544–552.

Langer-Safer, P. R., Levine, M., and Ward, D. C. (1982). Immunological method for mapping genes on Drosophila polytene chromosomes. Proceedings of the National Academy of Sciences of the United States of America *79*, 4381–4385.

Lantuéjoul, C., and Beucher, S. (1979). Use of watersheds in contour detection. In International workshop on image processing, real-time edge and motion detection.

Larsson, J.-Å., Wadströmer, N., Hermanson, O., Lendahl, U., and Forchheimer, R. (2011). Modelling cell lineage using a meta-Boolean tree model with a relation to gene regulatory networks. Journal of theoretical biology *268*, 62–76.

van Leeuwenhoek, A. P. (1677). November letter.

Lenna (1972). Lenna. Playboy Magazine.

Lewis, E. B. (1978). A gene complex controlling segmentation in Drosophila. Nature *276*, 565–570.

Li, S. Z. (2009). Markov random field modeling in image analysis (Springer).

Lindeberg, T. (1991). Discrete Scale-Space Theory and the Scale-Space Primal Sketch.

Linkert, M., Rueden, C. T., Allan, C., Burel, J.-M., Moore, W., Patterson, A., Loranger, B., Moore, J., Neves, C., Macdonald, D., et al. (2010). Metadata matters: access to image data in the real world. The Journal of cell biology *189*, 777–782.

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

Lorensen, W. E., and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. ACM SIGGRAPH Computer Graphics *21*, 163–169.

Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence *22*, 888–905.

Matzke, M. A., and Birchler, J. A. (2005). RNAi-mediated pathways in the nucleus. Nature reviews. Genetics *6*, 24–35.

McGinnis, W., Garber, R. L., Wirz, J., Kuroiwa, A., and Gehring, W. J. (1984). A homologous protein-coding sequence in drosophila homeotic genes and its conservation in other metazoans. Cell *37*, 403–408.

McKusick, M. K., Bostic, K., Karels, M. J., and Quarterman, J. S. (1996). The Design and Implementation of the 4.4 BSD Operating System (Addison-Wesley; 2 edition).

Mello, C. C., Kramer, J. M., Stinchcomb, D., and Ambros, V. (1991). Efficient gene transfer in C.elegans: extrachromosomal maintenance and integration of transforming sequences. The EMBO journal *10*, 3959–3970.

Moore, S. W. (2009). Developmental genes and cancer in children. Pediatric blood & cancer *52*, 755–760.

Murray, J. I., Bao, Z., Boyle, T. J., Boeck, M. E., Mericle, B. L., Nicholas, T. J., Zhao, Z., Sandel, M. J., and Waterston, R. H. (2008). Automated analysis of embryonic gene expression with cellular resolution in C. elegans. Nature methods *5*, 703–709.

Nisan, N., Roughgarden, T., Tardos, É., and Vazirani, V. V. eds. (2007). AlgoriNisan, Noam Roughgarden, Tim Tardos, Éva Vazirani, Vijay V thmic game theory (Cambridge University Press).

Okabe, A., Boots, B., Kokichi, S., and Chiu, S. N. (2000). Spatial tessellations: concepts and applications of Voronoi diagrams (John Wiley & Sons).

Okasaki, C. (1999). Purely functional data structures (Cambridge University Press).

O'Rourke, J. (1998). Computational geometry in C (Cambridge University Press).

Peterson, L. L., and Davie, B. S. (2011). Computer Networks: A Systems Approach (Elsevier).

Preibisch, S., and Saalfeld, S. (2010). Into ImgLib - Generic Image Processing in Java. In ImageJ User and Developer Conference, pp. (1) 72–76.

Preza, C., Snyder, D. L., and Conchello, J.-A. (1996). Imaging models for three-dimensional transmitted-light DIC microscopy. Proceedings of SPIE *2655*, 245–256.

Qian, Y. Q., Billeter, M., Otting, G., Müller, M., Gehring, W. J., and Wüthrich, K. (1989). The structure of the Antennapedia homeodomain determined by NMR spectroscopy in solution: comparison with prokaryotic repressors. Cell *59*, 573–580.

Rabin, Y., and Podbilewicz, B. (2000). Temperature-controlled microscopy for imaging living cells: apparatus, thermal analysis and temperature dependency of embryonic elongation in Caenorhabditis elegans. Journal of microscopy *199*, 214–223.

Rasband, W. S. (1997). ImageJ. U. S. National Institutes of Health, Bethesda. Available at: http://imagej.nih.gov/ij/.

Reece-Hoyes, J. S., Deplancke, B., Shingles, J., Grove, C. A., Hope, I. A., and Walhout, A. J. M. (2005). A compendium of Caenorhabditis elegans regulatory transcription factors: a resource for mapping transcription regulatory networks. Genome biology *6*, R110.

Robert, V. J. P., Katic, I., and Bessereau, J.-L. (2009). Mos1 transposition as a tool to engineer the Caenorhabditis elegans genome by homologous recombination. Methods (San Diego, Calif.) *49*, 263–269.

Sayood, K. (2000). Introduction to data compression (Morgan Kaufmann).

Schnabel, R., Hutter, H., Moerman, D., and Schnabel, H. (1997). Assessing normal embryogenesis in Caenorhabditis elegans using a 4D microscope: variability of development and regional specification. Developmental biology *184*, 234–265.

Schrödinger LLC (2010). The PyMol Molecular Graphics System, Version 1.3.

Scott, M. P., Tamkun, J. W., and Hartzell III, G. W. (1989). The structure and function of the homeodomain. Biochimica et Biophysica Acta (BBA) - Reviews on Cancer *989*, 25–48.

Sethian, J. A. (1999). Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science (Cambridge University Press).

Shaner, N. C., Campbell, R. E., Steinbach, P. A., Giepmans, B. N. G., Palmer, A. E., and Tsien, R. Y. (2004). Improved monomeric red, orange and yellow fluorescent proteins

derived from Discosoma sp. red fluorescent protein. Nature biotechnology *22*, 1567–1572.

Shawe-Taylor, J., and Cristianini, N. (2004). Kernel methods for pattern analysis (Cambridge University Press).

Shreiner, D., Group, T. K. O. A. W., Licea-Kane, B., and Sellers, G. (2011). Opengl Programming Guide: The Official Guide to Learning Opengl, Versions 4.1 (Addison Wesley Professional).

Silberschatz, A., Korth, H. F., and Sudarshan, S. (2010). Database System Concepts (McGraw-Hill).

Spilianakis, C. G., Lalioti, M. D., Town, T., Lee, G. R., and Flavell, R. A. (2005). Interchromosomal associations between alternatively expressed loci. Nature *435*, 637–645.

Suloway, C., Pulokas, J., Fellmann, D., Cheng, A., Guerra, F., Quispe, J., Stagg, S., Potter, C. S., and Carragher, B. (2005). Automated molecular microscopy: the new Leginon system. Journal of structural biology *151*, 41–60.

Sulston, J. E., Schierenberg, E., White, J. G., and Thomson, J. N. (1983). The embryonic cell lineage of the nematode Caenorhabditis elegans. Developmental biology *100*, 64–119.

Sulston, J., and Horvitz, H. R. (1977). Post-embryonic cell lineages of the nematode, Caenorhabditis elegans. Developmental Biology *56*, 110–156.

Swedlow, J. R., Goldberg, I. G., and Eliceiri, K. W. (2009). Bioimage informatics for experimental biology. Annual review of biophysics *38*, 327–346.

Terry S. Yoo, M. J. A. (2002). Engineering and Algorithm Design for an Image Processing API: A Technical Report on ITK - the Insight Toolkit. In Proc. of Medicine Meets Virtual Reality, J. Westwood, ed., IOS Press Amsterdam, pp. 586–592.

Thomas, C. F., and White, J. G. (1998). Four-dimensional imaging: the exploration of space and time. Trends in biotechnology *16*, 175–182.

Thomas, C., DeVries, P., Hardin, J., and White, J. (1996). Four-dimensional imaging: computer visualization of 3D movements in living specimens. Science *273*, 603–607.

Toft, P. (1996). The Radon Transform - Theory and Implementation.

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

Wadsworth, C. P. (1971). Semantics and pragmatics of the lambda calculus.

Walter, T., Shattuck, D. W., Baldock, R., Bastin, M. E., Carpenter, A. E., Duce, S., Ellenberg, J., Fraser, A., Hamilton, N., Pieper, S., et al. (2010). Visualization of image data from cells to organisms. Nature methods *7*, S26–S41.

Wood, W. B. (1988). The Nematode Caenorhabditis Elegans (CSHL Press).

Wu, Y., Ghitani, A., Christensen, R., Santella, A., Du, Z., Rondeau, G., Bao, Z., Colón-Ramos, D., and Shroff, H. (2011). Inverted selective plane illumination microscopy (iSPIM) enables coupled cell identity lineaging and neurodevelopmental imaging in Caenorhabditis elegans. Proceedings of the National Academy of Sciences of the United States of America *108*, 17708–17713.

Yanai, I., and Hunter, C. P. (2009). Comparison of diverse developmental transcriptomes reveals that coexpression of gene neighbors is not evolutionarily conserved. Genome research *19*, 2214–2220.