



Universiteit
Leiden
The Netherlands

Pattern Recognition in High-Throughput Zebrafish Imaging

Nezhinsky, A.E.

Citation

Nezhinsky, A. E. (2013, November 21). *Pattern Recognition in High-Throughput Zebrafish Imaging*. Retrieved from <https://hdl.handle.net/1887/22286>

Version: Corrected Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/22286>

Note: To cite this publication please use the final published version (if applicable).

Cover Page



Universiteit Leiden



The handle <http://hdl.handle.net/1887/22286> holds various files of this Leiden University dissertation

Author: Nezhinsky, A.E.

Title: Pattern recognition in high-throughput zebrafish imaging

Issue Date: 2013-11-21



PATTERN RECOGNITION
IN HIGH-THROUGHPUT
ZEBRAFISH IMAGING

A. E. Nezhinsky

PATTERN RECOGNITION IN HIGH-THROUGHPUT ZEBRAFISH IMAGING

A.E. Nezhinsky

ISBN: 978-94-6191-966-3

PATTERN RECOGNITION IN HIGH-THROUGHPUT ZEBRAFISH IMAGING

Proefschrift

ter verkrijging van

de graad van Doctor aan de Universiteit Leiden,

op gezag van Rector Magnificus prof. mr. C.J.J.M. Stolker,

volgens besluit van het College voor Promoties

te verdedigen op donderdag 21 November 2013

klokke 10.00 uur

door

Alexander E. Nezhinsky

geboren te Leningrad, USSR in 1982

Promotiecommissie

Promotor:

Prof. Dr. J. N. Kok

Co-promotor:

Dr. Ir. F. J. Verbeek

Overige leden:

Prof. Dr. T. Bäck

Prof. Dr. W. Bitter

Prof. Dr. H. P. Spaink

Dr. W. A. Kusters

The studies described in this thesis were performed at the Leiden Institute of Advanced Computer Science and supported by the SmartMix Programme of the Netherlands Ministry of Economic Affairs and the Netherlands Ministry of Education, Culture and Science.

Contents

1	Introduction	9
1.1	Whole Mount High Throughput Screening	9
1.1.1	Non-Vertebrate Models	10
1.1.2	Vertebrate Models	11
1.2	Pattern Recognition	12
1.3	Outline of the Thesis	13
1.3.1	Image Processing and Segmentation	14
1.3.2	Analysis of Statistics of Patterns	14
2	Segmentation and Pattern Recognition in Image Space	17
2.1	Introduction	18
2.1.1	Related Work	18
2.1.2	Overview of Basic Approaches for Zebrafish Embryo Segmentation	20
2.1.3	Deformable Template Matching	24
2.2	Deformable Template Matching (Approach 1)	24
2.2.1	Method	24
2.2.2	Optimization by a Genetic Algorithm	26
2.2.3	Multiple Object Recognition	29
2.2.4	Experiments	30
2.2.5	Conclusions and Discussion	32
2.3	Deformable Template Matching (Approach 2)	33
2.3.1	Method	34
2.3.2	Application to the Zebrafish Larvae Images	38
2.3.3	Experiments and Results	40
2.3.4	Conclusions	42
3	Anchor Region Based Pattern Recognition in Image Space	45
3.1	Introduction	46
3.1.1	Related Work	46
3.1.2	Algorithm Overview	48
3.1.3	Pre-processing	49
3.2	Pattern Recognition: The Anchor Region Based Method	51
3.2.1	Prototype Template	51
3.2.2	Conversion from Image to Graph	53
3.2.3	Bayesian Formulation	61
3.2.4	Finding an Optimal Solution	63

3.3	Case Study 1: Zebrafish infection analysis	64
3.3.1	Zebrafish Orientation	65
3.3.2	Zebrafish Region Annotation	66
3.3.3	Data	66
3.3.4	Analysis and Results	66
3.4	Case Study 2: Other Model Organisms	68
3.5	Conclusions	70
4	Software Development and Evaluation for High Throughput Zebrafish Analysis	73
4.1	Introduction	74
4.2	High Throughput Experiments	74
4.2.1	Image Acquisition	74
4.2.2	Data Storage	75
4.2.3	User Analysis	76
4.3	Our Approach	76
4.3.1	Workflow	76
4.3.2	Components	78
4.4	Evaluations and Results	82
4.4.1	User Evaluations	82
4.4.2	Results	83
4.4.3	Performance Evaluations and Results	83
4.5	Conclusions and Discussion	87
5	Data and Pattern Analysis in Infection Studies in Zebrafish	89
5.1	Introduction	90
5.2	Analysis of High Throughput Zebrafish Infections Screens Based on Approach 2	91
5.2.1	Brightfield Imaging: Shape Localization and Annotation	92
5.2.2	Fluorescent Imaging: Analysis	93
5.2.3	Output and Dataset Creation	93
5.2.4	Analysis and Results	94
5.2.5	Conclusions	99
5.3	Analysis of High Throughput Zebrafish Infections Screens Based on Approach 3	100
5.3.1	Analysis and Results	102
5.4	In Depth Analysis of High Throughput Zebrafish Infections Screens	107
5.4.1	Experiments	107
5.4.2	Analysis and Results	107
5.4.3	Total amount of Infection	107
5.4.4	Amount of Infection separated into Regions	113
5.4.5	Infection Flow per Mutant Strain	118
5.4.6	Average Size of Clusters	120
5.5	Conclusions	124

6 Discussion	125
6.1 Pattern Recognition in Computer Vision	126
6.2 Software Development	127
6.3 Statistical Analysis	127
6.4 Conclusion	129
Bibliography	131
Dutch Summary	137
Russian Summary	141
Curriculum Vitae	143
List of publications	145
Acknowledgements	147

1 Introduction

Computers and computational methods have enabled experimental approaches in the life-sciences in which large amounts of specimens are processed in order to establish cause-effects relations or to detect rare events. These high volume experimental approaches are commonly referred to as High Throughput. In order to get the most out of such experiments, the aspects of experimental design and analysis need to be considered. Although computers enable High Throughput approaches, it is necessary to critically review the computational approaches that are addressed. Methods need to be adapted to the experimental design and sometimes, for several reasons, new methods need to be developed. High Throughput methods are applied in different fields of the life-sciences and the instrumentation for these High Throughput methods differs. In this thesis we consider High Throughput methods that are concerned with imaging, imaging with microscopes in particular. The scope of this thesis is to evaluate and develop methods for object recognition, extraction and measurement. In addition, given the output from a High Throughput experiment, the analysis of the measurement is addressed. The aim of this thesis is to find robust methods for object extraction and analysis. Throughout the thesis one particular application domain is used, infection studies in zebrafish for which High Throughput approaches are crucial. In order to understand the starting points for High Throughput methods, a survey of the literature is performed. For a review, two aspects have been particularly taken into consideration. The first being the relation between the experimental design and the aim of the experiment. The second being the computational aspects of the High Throughput experiment. In the next sections these two aspects are addressed. This introductory chapter closes with an overview of the contents of this thesis.

1.1 Whole Mount High Throughput Screening

A method called High Throughput screening (HTS) is used for scientific experimentation within different fields of modern biology. The method makes it possible to come to a high volume of experimental data. Results of such HTS experiments are used in setup for drug design and better understanding of various biological processes. There are different kinds of HTS methods, most often though they consist of typical basic steps which are applicable within different applications.

In order to prepare the experimental data a plastic micro-titer plate is used. Such a plate consists of a grid of wells (a multiple of 96). The wells are filled with the object of interest, which can be either an entire organism or a part of it. This is usually done by a robot in an automated fashion. The genetically altered object or organism is compared against a control group and is monitored through imaging. Resulting data is analyzed

1 Introduction

in order to gain further insights in certain conditions. Experiments that are done for genomes do not involve the time factor, while experiments for proteomics testing are usually done in a time-lapse fashion. Depending on the speed of development of the organism, the start, end time and the interval of a time-lapse experiment is adapted, in such a way that the time resolution is sufficient. Nowadays HTS is made possible by automated microscopes. In order to do the final analysis problem specific image analysis software is being developed.

In [36] High Throughput (HT) methods are distinguished in 3 groups: Phenotype, Transcript and Proteomic. In this thesis we will focus only on the HT for phenotype analysis, since phenotype is directly related to imaging.

In order to be able to do in depth disease and infection analysis by tissue screening multiple models are used to mimic its behavior in the human body. A cell-based model is fairly inexpensive, it is, however considered physiologically distant to human. Vertebrate models of mammals are more relevant, but expensive. A popular tradeoff between higher vertebrate models and cell based models are small organisms, like *C. elegans*, the fruitfly and the zebrafish. Because of their small size, these organisms can be imaged *whole mount*, which means the whole organism is treated and used in the imaging. We will discuss the HT possibilities of whole mount organism screening in more detail.

1.1.1 Non-Vertebrate Models

C. elegans (*Caenorhabditis elegans*) is an important non-vertebrate model organism that has a rapid development, is transparent and is easy to cultivate. The main advantage of the use of this model is the fact that *C. elegans* is the first organism of which the entire fate map from zygote to mature organism is available and it is the first organism with a completely sequenced genome. Fate maps provide a possibility to trace each cells predecessor through time during the development of the organism [56] (also known as cell lineage). The top of the fate map is the zygote cell followed at the next level by two daughter cells that at the next level divide into more daughter cells and in such a way a tree is formed and the nodes are indexed. This makes it very valuable in human disease exploration, since genetics of *C. elegans* are for about 70% similar to human. Computational aspects involve storing and traversing this cell lineage. AceDB (*AC. elegans DataBase*) is a database that is specifically developed for the *C. elegans* and can be used to browse the genomic data. Later AceDB was adapted to use with genomic databases of other organisms. Additionally *C. elegans* is widely used for its rapid development, transparency and ease of cultivation.

Rohde and Yanik [49] describe a HT approach for in vivo screening of *C. elegans* in a time-lapse fashion. *C. elegans* is immobilized by cooling it down. The proposed technology is used to perform subcellular-resolution femtosecond laser microsurgery of single neurons in vivo and to image the subsequent axonal (ALM/AWB) regeneration dynamics at sub-cellular resolution. Analysis of regeneration dynamics is done based on imaging and statistics.

1.1.2 Vertebrate Models

The Zebrafish (*danio rerio*) model is widely used as a vertebrate model. The layout of processes involved in human disease within the zebrafish organism often can be compared with their behavior in humans and therefore it can serve as a model for some human disease; an overview can be found in [2]. Zebrafish has a fast development and is easy to breed due to high fecundity rate: up to 200 offspring can be produced multiple times per week. It has a relatively small size which makes it easy to maintain large populations and produce large quantities for testing. The cell lineage of the zebrafish exists, but only in an early stage. The DNA of the zebrafish has a lot of homologs as compared to the humans, therefore the zebrafish model can be used to study the functional purpose of the genes [25] and is useful in the fields of developmental, genomics and drug research. It makes a good organism to use in research that is conducted in a HT fashion. Computational methods are needed to store genomic data and to browse it. In order to facilitate for HT data processing (of whole organism), computational approaches are needed for phenotype analysis, tracking and feature extraction.

Infection patterns can be tracked throughout the body due to transparency in early stages and especially when used with a fluorescent agent as a specific tissue promoter. Phenotypic development and infection pattern locations can be studied due to the external development of the zebrafish. In addition, evolving behavior of the zebrafish can be studied [20]. An early review of the HT approaches within the zebrafish research is given in [36].

Common carp (*Cyprinus carpio*) is also an important model organism due to it being a popular aqua cultivation species and its high reproduction rate. Most of the HT methods that are based on the carp model are about genomics and will not be considered in this thesis. However, nowadays carp is being considered in HT phenotype analysis.

Most HTS phenotype-based analysis methods are using brightfield or fluorescence microscopy.

Light microscopy based methods are used in order to do measurements on a whole mount organism phenotype for the aim of object localization and pattern recognition. A prerequisite would be analysis without the need to manually position the model organisms. One of the image based methods is Cognition Network Technology (CNT). CNT is a semantic and context-based segmentation approach that is applied in the detection of biological structures. In [62] CNT is used to detect transgenic fluorescent zebrafish embryos. The zebrafish embryos must be located in well plates and be presented in a lateral view. The shapes and regions can then be recognized by an artificial intelligence based algorithm that makes use of a predefined ruleset that use graded responses for area, length, and shape.

Another method for phenotype recognition for the zebrafish is presented by [34]. The recognition is based on a Support Vector Machine model that uses image descriptors (color histogram, representative color, and color layout) as input. This method does not perform detailed regional analysis but is defined as a three class classification problem. It is used to determine between three zebrafish embryonic conditions: hatched, unhatched, and dead.

1 Introduction

Cellular processes within model organisms can be analyzed and modeled by making use of tissue-specific fluorescent reporter genes. Regions of interest can then be localized by image based methods that are able to retrieve fluorescent regions.

One of the methods for in vivo time lapse imaging is described in [3]. The authors describe protocols building a light-emitting diode fluorescence macroscope in order to screen adult fluorescent zebrafish. The macroscope is multi-spectral and imaging is being done simultaneously on multiple fluorescent channels. This method allows for the zebrafish to swim around. The method is able to distinguish up to five fluorescent proteins that are located within the fish and simultaneously image up to 30 adult animals that transgenically express a fluorescent protein. While the authors focus on the imaging part, the segmentation of the produced images is based on the different webcam inputs and thus limited to webcam capturing software (AMCap) and manual analysis. Aim of the study was to use this HT method for imaging transgenic animals, screening for tumor engraftment, and tagging individual fish for long-term analysis.

Walker et al. [64] describe a simple whole-organism HT method for quantifying changes in reporter intensity in individual zebrafish over time termed, Automated Reporter Quantification in vivo (ARQiv). ARQiv differs from current “high-content” (e.g., confocal imaging-based) whole-organism screening technologies by providing a purely quantitative data acquisition approach. Data processing was based on the comparison of intensity values within image regions. The values separated into the groups *background*, *signal* and *total signal* and comparison could be done across time in percentage terms.

Complex Object Parametric Analyzer and Sorter or COPAS is a combination of an instrument and a software package developed by *Union Biometrica* in order to analyze and sort biological objects that range from 100 to 1500 microns in diameter. Instrument is based on the principles of flow cytometry (cf. [35, 8]), but accommodates for larger objects and uses a pneumatic sorting mechanism. Originally COPAS was intended for the use with nematodes and later it was adapted for the use with zebrafish embryos. COPAS is used for HTS applications. In depth analysis of the produced images is done by the COPAS Profiler II software package. Based on fluorescence levels an optical profile is created. This facilitates in quantitative detection of multicolor fluorescent labels and determination of labeled object distributions such as tumor cells or granuloma in the zebrafish.

Saji et al. [19] describe a HT screening approach to assist in hazard ranking of engineered nanomaterials (ENM). This is done by tracking cellular injury in zebrafish embryos through epifluorescence microscopy. The analysis of the response parameters of nanomaterials is performed by the creation of heat maps by statistical approaches and the use of self-organizing maps (SOMs).

1.2 Pattern Recognition

The HT methods previously described all have significant ingredients of computer science. An overview of the described methods and their relationship to computer science (CS) is provided in Table 1.1.

ref.	exp. type	main CS component	v/t	output
[49]	f	regeneration dynamics by intensity, statistics	v	images
[62]	b	segmentation by ruleset, graded response	t	images
[34]	b	classification by SVM, image descriptors	v	image descriptors
[3]	f	segmentation by imaging of different channels	v	images
[64]	f	comparison by region based intensity, statistics	v	regional report
[8]	f	optical profile by intensity	v	images
[19]	f	response parameters analysis by SOM	t	heat maps

Table 1.1: An overview of some HTS methods. Experiment type (exp. type) is either fluorescent (f) or brightfield (b) and performed in vivo (v) or in vitro (t).

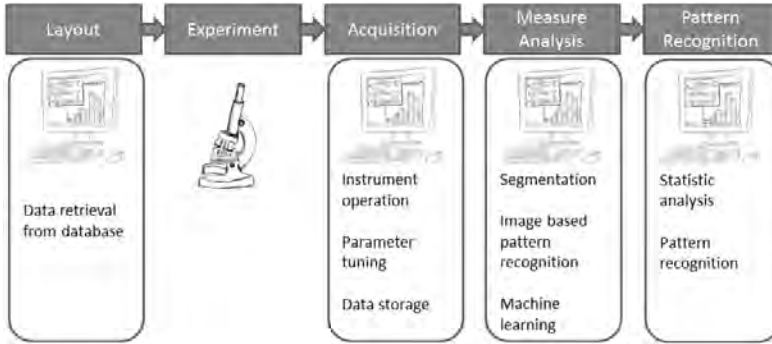


Figure 1.1: Schematic relations between a High Throughput Screening pipeline and Computer Science.

There are no standard methods which would be suitable for all applications and thus application dependent methods need to be developed. A schematic HT pipeline and its dependency on CS is shown in Figure 1.1.

Pattern recognition can be seen in two contexts: object recognition and data mining. Pattern recognition in terms of phenotypic object recognition plays an important role in almost every HT analysis process. Object recognition relates to looking for a pattern in the *image space*. In the case of separation of the objects of interest from the background we refer to it as segmentation. Data mining relates to looking for patterns in data *feature space*. Data mining depends on the data we use as input and can be used for finding global features followed by refining.

1.3 Outline of the Thesis

In this thesis we will in particular focus on two aspects of HTS: the image analysis step followed by pattern recognition of the resulting data. We have developed an image anal-

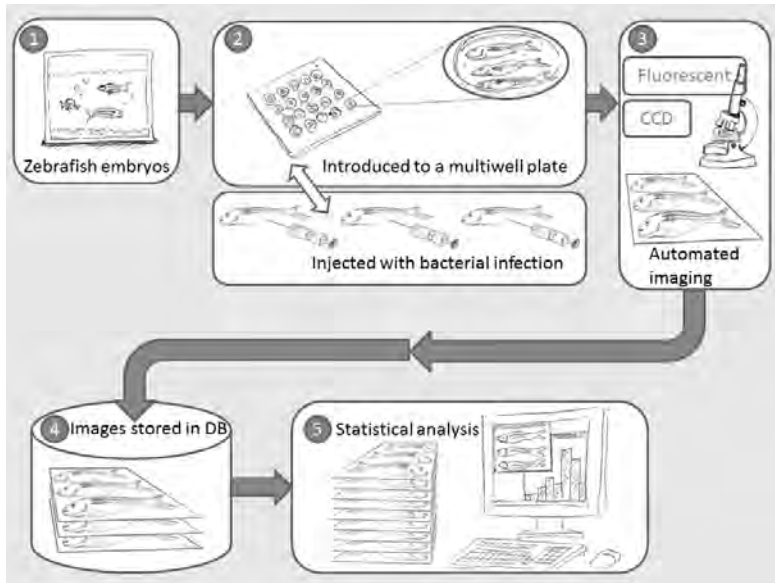


Figure 1.2: Example of a High Throughput Screening pipeline applied to zebrafish phenotype recognition.

ysis framework that is able to automatically recognize the zebrafish shape and retrieve the infection spread in a HT fashion. This pattern recognition framework allows for in depth analysis of the spread of infection within the zebrafish organism.

1.3.1 Image Processing and Segmentation

In the case study that is presented in this thesis we will be focusing on the zebrafish embryo as the model organism. We have developed a segmentation approach that is suitable for HT analysis and is able to retrieve the fish shape from images. An example of HTS is illustrated in Figure 1.2.

In Chapter 2 we describe our approach for retrieving the zebrafish shape from the input images. The approaches presented here are used not only to separate the foreground from the background but also to automatically recognize and annotate the zebrafish embryo shapes. We present three deformable template based approaches we have developed for this. In Chapter 4 we describe the graphical interface and the software that was created based on the algorithms described in Chapter 2 and Chapter 3.

1.3.2 Analysis of Statistics of Patterns

After retrieval of the zebrafish shapes and separating the input brightfield images into foreground and background the next step is to do measurements on the foreground area. In our case study the measurements are done on the response of the immune

system to the infection as well to the bacterial infection spread. In our research we aim to contribute to the understanding of the genes/proteins that are involved in the infection of the zebrafish with the *Mycobacterium marinum*, which is closely related to *Mycobacterium tuberculosis*, the causative agent of tuberculosis in humans. In chapter 5 we do a number of statistical assessments of the data that was collected by our HT approach. In Chapter 6 we discuss the results of the approach.

2 Segmentation and Pattern Recognition in Image Space

Based on:

A. Nezhinsky & F.J. Verbeek: Pattern Recognition for High Throughput Zebrafish Imaging using Genetic Algorithm Optimization. In: 5th IAPR Conference on Pattern Recognition in BioInformatics (PRIB 2010), Lecture Notes in Bioinformatics 6282: 302–312, Springer (2010)

and

A. Nezhinsky & F.J. Verbeek: Efficient and Robust Shape Retrieval from Deformable Templates. In: Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies (ISoLA 2012), Lecture Notes in Computer Science 7610: 42–55, Springer (2012)

2.1 Introduction

Retrieving location and contour of a shape is crucial for the analysis of large image datasets in many fields, such as optical character recognition (OCR) and bio-imaging. Segmentation is the decomposition of an image into foreground objects and background. Foreground objects are preferably the (biological) structures of interest, while other objects and noise are the background. In most cases objects of interest are homogenous regions and therefore most segmentation methods are based on homogeneity. Homogeneity of an object can be based on different characteristics, this can be color, texture or other similarity measures.

2.1.1 Related Work

Most segmentation techniques use a characteristic object feature in order to separate objects from the background. These features can be based on object shape or on color or texture and are attempting to mimic human visual system behavior. A successful segmentation turns a grayscale or color image into a binary representation in which one value represents the object of interest and the other the background.

The first step in many applications is to segment a meaningful biological structure from images and remove the background, despite its position, rotation and scale. Images in biology are acquired under different conditions which affects quality, brightness, contrast, lighting conditions and the amount of noise. Consequently, extraction as well as processing of shapes from the images might be hampered.

We can separate the segmentation techniques into two groups: bottom up segmentation and top down segmentation [4]. Bottom up segmentation uses image criteria in order to define image regions that are likely to be the structure of interest. In this case no machine learning or pattern recognition is involved.

The top-down approach on the other hand uses pattern recognition techniques that use predefined object representation in order to approximate the object location. Top-down methods are suggested to mimic human vision behavior and are becoming more successful in retrieving an object based on its predefined shape. On the other hand bottom-up techniques can get more precise local image boundaries [31]. Recent work [31, 4, 68, 22] stresses the importance of combining both methods exploiting their advantages.

Bottom-up Segmentation We first give a short overview of some bottom-up segmentation methods. Color and texture based approaches are intuitive and most common during low level segmentation.

Color based segmentation can be based either on gray scale images or on color. The common approach based on color value is *thresholding* an image with a single value. The thresholding method converts a gray scale image into a binary representation by a threshold value. The value is chosen either manually by the scientist or automatically [46]. During thresholding, pixels that have a grayscale value above the threshold value are marked as foreground. All other pixels are then marked as background. A number of variations exist on this approach. Thresholding can be done in an inverted manner,

such that pixel values below the threshold value are treated as foreground, the rest as background. Additionally multi thresholding methods exist, where a pixel is set as foreground if its value is in between two threshold values.

For color images these values first must be converted to gray scale or separated into different channels (RGB or HSV [21]) and then thresholded. This approach works well under the assumption that the object of interest is of a characteristic color that is significantly different from the color of the background or other objects in the image.

Another approach is to look for region similarity by applying region growing techniques within the *region-based* methods. A state of the art color region based method is Mean Shift discontinuity preserving altering [59]. It first determines the clusters in the image by merging regions iteratively based on color similarity, then it assigns class labels to formed clusters. If the Mean Shift is used with a generalized kernel it can be seen as a nonparametric algorithm. Another widely used and adapted region-based method is the watershed segmentation [12].

Besides color characteristics objects can be characterized by their edges. An edge location within each image is efficiently represented by the magnitude of the gradient. For each (pixel) position (x, y) in the image we have $|\nabla f(x, y)|$ as the magnitude of the gradient by usage of Sobel, Prewitt or Roberts filter [21] on the edge map. A binary representation can be also found directly by applying a Canny edge detector. An edge map represents shape boundaries. Alternatively edges can be found by considering zero-crossings of the Laplacian-of-Gaussian [53]. After the edges are identified the image can be separated into objects by using connected components labeling [50] combined with thresholding. These edge-based methods can only perform well if edges are clearly present in the image and no objects are overlapping one another.

A more elaborate list of (low level) segmentation techniques is given in [38].

Top-down Segmentation (Pattern Recognition) An important problem in image analysis field is the content retrieval of objects from images. While the bottom up segmentation techniques can be used to separate foreground from the background, it is not enough for annotation of objects within the foreground. To accomplish that certain pattern recognition based on a shape model are required.

Most widely used methods are: active shape models (ASM) [9], active snakes (or contours) [26], the Hough Transform [52], deformable templates [65] and level-set based methods (LSM) [51]. Most model based segmentation methods are concerned with the retrieval of certain predefined shapes from the images.

A large number of context based image retrieval systems have been described [70]. These can be divided into the *free-form* and the *parametric* approaches [24].

Free Form Free-form class models have no global shape limitation, but focus mainly on attraction towards certain image features [24]. Free form models need a correct global initialization inside an image and optimize the local shape. A popular *free-form* approach for shape retrieval is the Active Contour (or active snake) [26]. Shape edges need to be connected together for an active snake to perform correctly.

Parametric In papers dealing with recognition of objects with known shapes, [18, 44, 16] the authors stress the use of deformable templates. Deformable templates are part of the *parametric class* models as they make use of a predefined set of parameters. The representation of the parameters might differ, but often a template is used, which consists of a set of contour points to which we approximate the basic shape outline. A deformable template [24] approach gives the possibility to represent an object that can be subject to certain deformations in a compact manner. In that case object localization should be performed by a process of matching the deformable template to the object shape in the input image. In [9, 27, 45] a deformable template representation of a shape is a set of points approximating the outline as obtained from a priori knowledge.

Existing template matching systems are known for handling grayscale or binary images. Some systems work directly on the grayscale input image. In this case often shape descriptors like SIFT [37] are used to describe shape or texture characteristic. Other systems assume a *bottom-up* segmentation step like thresholding [21, 46] already converted the image to a binary representation. In [33] the authors propose a deformable template based region merging system. As input their method uses both *over segmentation* with region color and edge detection and is based on splitting and merging regions based on certain perceptual criteria.

2.1.2 Overview of Basic Approaches for Zebrafish Embryo Segmentation

We have tried to apply different bottom up segmentation methods in order to separate the zebrafish shapes from the background. Note, that this concerns solely segmentation, not recognition. This means the separation of foreground (the space taken up by zebrafish embryos) and the background of an image.

The motivation to apply color or grayscale based segmentation is based on the suggestion that the zebrafish embryos have a different color than the background. In order to determine the color that defines a zebrafish embryo automatically, we first attempt to determine the color of the background. This can be done under the assumption that a zebrafish embryo is never located on or intersecting the edge of the image. This means that all the pixels on the edge of the image are part of the background. All image objects that are not of that color are then treated as zebrafish.

First we attempt to do the segmentation based on grayscale thresholding. The most basic approach would be global thresholding. This means that image pixels with an intensity value below a threshold value t are labeled as foreground and the rest of image space is labeled as background. A global threshold method does not work for most of images at hand. This was due to the fact that because of uneven illumination the background intensity differed on different locations in the image and therefore there was no t value that could be set to separate foreground from background. An example of failing global thresholding is shown in Figure 2.1.

We also attempted to interpolate the grayscale background values from the edges to the center of the image. By this we tended to predict the background values as they should be at the location of the image and then subtract them from the original image. A commonly used way of doing it would be using a large Gaussian filter. However, since



Figure 2.1: From left to right: Original image (converted to grayscale) , thresholded grayscale image.

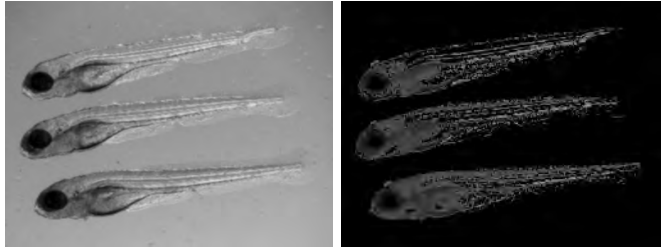


Figure 2.2: From left to right: original image, difference of interpolated image and original.

the zebrafish objects are very large this approach would not work. We applied a *bi-linear interpolation* method to be able to predict the background values of each pixel within the image. Instead of pixels we separated the image into small $a \times a$ rectangular areas and calculated their average intensity. If the difference between this intensity and the interpolated image was larger than a certain threshold t , area was treated as foreground, otherwise as background. Foreground pixels were multiplied with the original image, background area intensity was set to 0. In Figure 2.2 we show some results of this algorithm. While the results seem good there is an amount of variables (t, a) that still need to be set by hand per image before running the algorithm .

Following the grayscale threshold we have tried to separate the image based on its values in the color spaces. Again, we assumed that the edge of the image contains only background values. We have separated the image into HSI space, which is a more standard way to do segmentation on color images than separating into RGB channels. We have followed a typical HSI segmentation method as is presented in [21](page 331). The method is based on thresholding the saturation image S . Then the retrieved mask S_b was multiplied with the hue image H . The saturation threshold value is set based on the edge background values. An example of this method applied to a typical zebrafish image is shown in Figure 2.3. The spatial location of non black pixels in $S_b \times H$ represent the regions of interest. This segmentation is not accurate, since a lot of pixels within the zebrafish and on its border are treated as background.



Figure 2.3: From left to right: original image (converted to grayscale), its saturation component (S), thresholded saturation component multiplied by hue ($S_b \times H$).

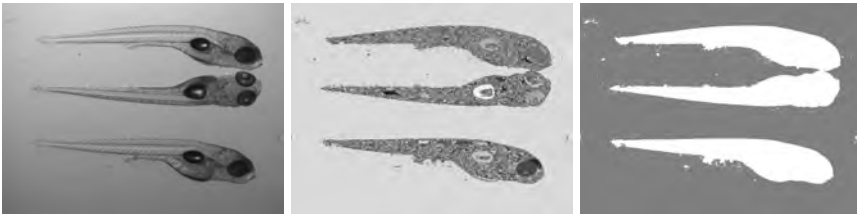


Figure 2.4: From left to right: original image, regions retrieved by mean-shift segmentation, retrieved background region in gray.

Another method based on color is Mean Shift discontinuity preserving altering [59]. The method progresses throughout the image by looking for color similarities. Then the image is labeled into regions. We have applied the method to the zebrafish images by assuming the entire background will be selected as one region. This region can then be identified since we know the background is present at the edge of the image. The result of such a segmentation is shown in Figure 2.4. As can be seen this method gives more accurate results.

In our test case the image background represents a liquid and therefore it is fairly smooth without high local color changes. The zebrafish objects however usually do have strong boundaries. This suggests the use of edge based methods. We have tested our data with several straightforward gradient operators. An operator that directly gives a binary image as output is the Canny edge detector. Its output is shown in Figure 2.5. As can be seen in the figure edge based methods are able to remove the background. However, since the zebrafish embryos have a lot of strong edges present within the fish it still might be difficult to automatically select the zebrafish outline.

Some of the approaches presented here were used as a preprocessing step during the actual pattern recognition process. The choices of these approaches are described in more detail in the following sections. A short summary on the methods as applied to zebrafish embryos is given in Table 2.1.

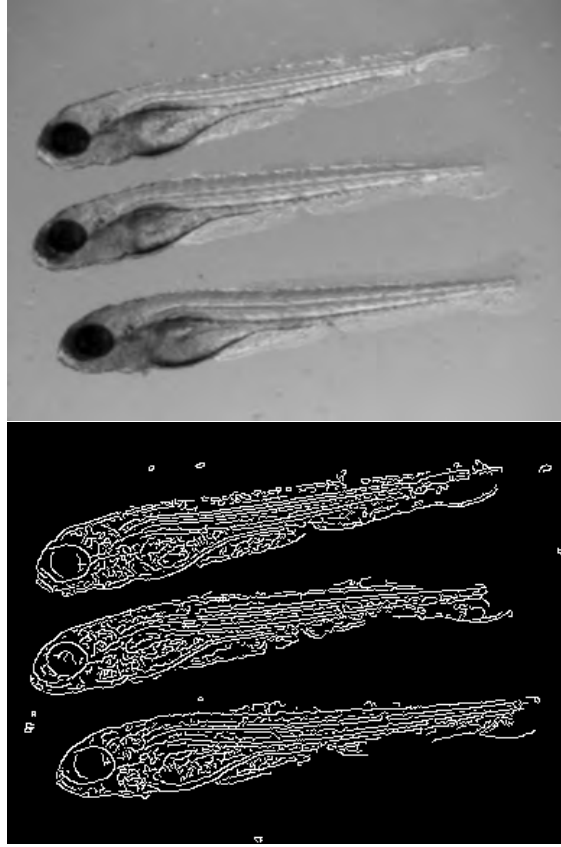


Figure 2.5: From left to right: Original image converted to grayscale, right canny edge detector result.

Segmentation technique	Discussion
binary thresholding	fails at uneven illumination
background interpolation	good, but variables need to be set per case
HSI space based	good, if zebrafish intensity differs from background
mean-shift	good results
edge based	good results, but needs post-processing

Table 2.1: Comparison of different bottom up segmentation methods with respect to zebrafish embryo retrieval.

2.1.3 Deformable Template Matching

The problem at hand is the retrieval of a known basic shape: the basic shape is known but can slightly vary. Besides separating foreground from background also content retrieval is important: we want to know how a shape is located, how many shapes are in the image and where which part of the shape is. This calls for a top down approach. A candidate would be a parametric approach. Template matching is a process of finding a predefined template in an image. The way template similarity as compared to a part of the original image is calculated differs per application [32].

2.2 Deformable Template Matching (Approach 1)

In this section we introduce a slice representation model (SR) of a deformable template. Instead of considering the outline of a shape, we simplify the shape representation by considering only certain characteristic horizontal slices and use these for template matching. The proposed SR model is made advantageous for efficient optimization with a Genetic Algorithm (GA). In [66, 48] authors also propose a GA for low parameter shape templates (circle and ellipse detection).

2.2.1 Method

We propose a template representation which captures both boundary and interior of the object and thereby describes the average structure of a predefined shape. The grid of the search space is determined by the discrete pixels in a $K \times L$ image matrix M . We take a binary image as starting point. The binary image is obtained by thresholding. Thereby we assume a background pixel has the value 0 and a foreground pixel value 1. In the following subsections we will describe the template representation in more detail as well as the deformations that a template can undergo.

Slice Representation Model

The prototype template T_0 we propose is represented by the following vector:

$$T_0 = (s_0, s_1, \dots, s_n, d), \quad (2.2.1)$$

where n is the number of slices; d is the distance between two slices in the horizontal direction. Initially T_0 is represented in a $X(\text{horizontal})$ – $Y(\text{vertical})$ space in the horizontal direction, with slices being parallel to the Y -axis. This is done for ease in the representation. A template slice itself is then represented by the vector:

$$s_i = (w_i, b_i, \varsigma_i), \quad (2.2.2)$$

where w_i is the fixed width in pixels in the horizontal direction, preferably $w_i = 1$; b_i is the fixed width in pixels in the vertical direction of the image, preferably $b_i = 0$ and located below and above the area of w_i . The b_i are required to define that the area surrounding the shape is preferably empty. And ς_i is the seed point of a slice; ς_i is

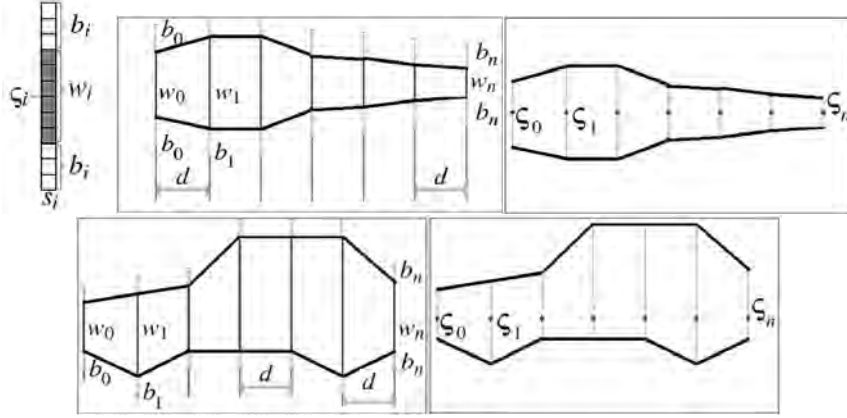


Figure 2.6: a) A single slice. The slice is always parallel to the Y-axis b) Template for a fish shape c) Location of ς_i in a fish shape template d) Template for a car shape e) Location of ς_i in a car shape template

needed to be able to define non symmetrical shapes as input. This point will be used as a reference for allowed slice shifts. In Figure 2.6a an example slice is shown; a slice is a sample of the template always perpendicular to the length axis.

In Figure 2.6b, 2.6c, 2.6d two examples of templates are shown. The template length is fixed according to $\ell(v) = n * d$. For templates with horizontal symmetry axis, ς_i is simply chosen as the center of each slice (cf. Figure 2.6c). For templates having no horizontal symmetry axis ς_i are located on the same horizontal line (cf. Figure 2.6e).

Deformations

This representation was chosen as we assumed the slices will be able to move vertically along the template to match a deformed (slightly shifted) shape. At template shift or slight rotation the global shape is still captured within the template. The total length of the template is fixed in the proposed representation.

A deformed template T is derived from the prototype template T_0 and is represented as $T(T_0, I)$. A deformation I will then be encoded by a state-sequence as follows:

$$I = (x, y + \delta_0, y + \delta_1, \dots, y + \delta_n), \quad (2.2.3)$$

where x is the shift in the X-axis direction of ς_0 ; $y + \delta_i$ is the translation measured in the Y-axis direction of the ς_i of slice i .

We assume the maximal vertical deformation between two slices is 45 degrees, cf. Figure 2.7a. As a result of this constraint $|\delta_{i-1} - d| < |\delta_i| < |\delta_{i-1} + d|$ applies for two starting points of consecutive slices i and $i + 1$. Then, each ς_i can be shifted vertically within the following boundaries:

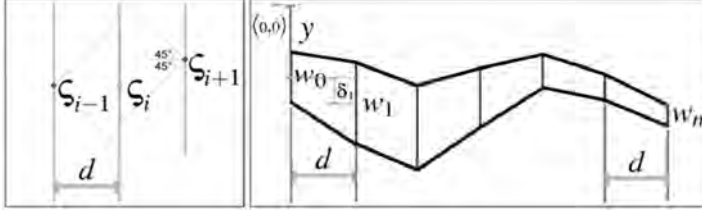


Figure 2.7: a) Vertical shift margins allowed for slice starting point ς_i , relative to ς_{i-1} and ς_{i+1} . b) A possible deformation for a fish template.

$$\max(\varsigma_{i-1} - d, \varsigma_{i+1} - d) < \varsigma_i < \min(\varsigma_{i-1} + d, \varsigma_{i+1} + d). \quad (2.2.4)$$

In Figure 2.7b we demonstrate a possible deformation of a template representing a fish.

Energy Function

The energy function Φ is a fitness function that specifies to what extent an identified shape in the image matches the deformed template. In the literature the probability of a deformed template being located over a shape is referred to as the likelihood [24].

Let us now introduce the details of the computation of Φ . Consider the fitness ϕ_i of a single slice i . In order to find the optimum we wish to maximize matching values (0 or 1) between pixels covered by the slice, i.e., pixels with value 1 contained in w_i and pixels (at top and bottom of the slice) with value 0 in b_i . We then have: $S_i[j]$ represents the j -th element (pixel) of slice S_i , as counted from the top of the slice.

$$\phi_i = \frac{1}{w_i + 2b_i} \sum_{j=0}^{w_i+2b_i} H_j, \quad H_j = \begin{cases} 1, & \text{if } M(x, j + y + \delta_i) = S_i[j] \\ 0, & \text{otherwise} \end{cases} \quad (2.2.5)$$

In Figure 2.8 an example of a matching is given. The location of the proposed matching is denoted in pattern. For this example:

$$\phi_i = \frac{1 + 0 + 0 + 1 + 1 + 1 + 1 + 1 + 0 + 1 + 1 + 1}{12} = 0.75$$

The deformation I consists of multiple slice shifts. The total fitness of all n slices is given by:

$$\Phi = \frac{1}{n+1} \sum_{i=0}^n \phi_i. \quad (2.2.6)$$

2.2.2 Optimization by a Genetic Algorithm

In this section we discuss the major components of a Genetic Algorithm (GA), i.e., population, evaluation, selection, crossover and mutation respectively. In a GA, a population

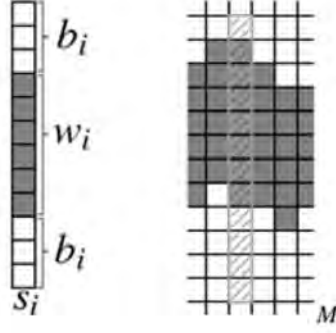


Figure 2.8: Example of matching a slice S_i on to a location in matrix M

P (of size m) of candidate solutions is evolved toward better solutions by introducing computer analogues for recombination, mutation and selection [1].

A candidate solution is also referred to as an *individual*. The outline of a generic GA pseudo code reads:

1. $t = 0$
2. Initialize $P(t)$
3. Evaluate $P(t)$
4. **while** not terminate **do**
5. $P'(t) = \text{SelectMates } (P(t))$
6. $P''(t) = \text{Crossover } (P'(t))$
7. $P'''(t) = \text{Mutate } (P''(t))$
8. Evaluate $P(t)$
9. $P(t+1) = P'''(t)$
10. $t = t + 1$
11. **end while**

The termination criterion can differ, depending on the problem at hand. The details of the operators are dealt with in more detail in the following subsections.

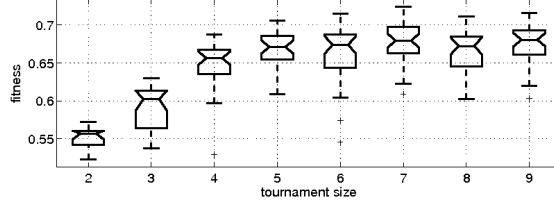


Figure 2.9: Fitness comparison for different tournament values in 30 runs

Representation of Individuals

A shape I based on the template T_0 , also referred to as *individual*, is represented in the image as, and consists of the following genomes:

$$I = (x, y + \delta_0, y + \delta_1, \dots, y + \delta_n). \quad (2.2.7)$$

The individuals are initialized with randomly valued genomes. According to common practice in the GA research, a population is generated with random genome values, covering the entire range of possible solutions. For finding shapes in images, this means random shapes are initialized on random locations in the target image with a random deformation according to Eq. 2.2.7.

Evaluation

The fitness function determines the quality of an individual and depends on the problem at hand. Function Φ (cf. Eq. 2.2.6) will serve as a fitness function for the genome values of an individual; Φ is then used to evaluate the candidate solutions in the selection step.

Selection

For the selection operation, the so-called tournament selection scheme [39] is used in order to prevent premature convergence. Tournament size, i.e., k_{size} , was determined through empirical testing. Consequently, comparison of high and low k_{size} is given in Figure 2.9. We have established $k_{size} = 7$ to be a good value for our experiments.

Crossover

A crossover is applied with single crossover point. A random point $p \in (0, n)$ is chosen where n denotes the length of the genome. After crossover of two individuals I_J and I_K the resulting individual I_L has the following form:

$$I_L = (x_K, y_K, \delta_{0_K}, \dots, \delta_{p_K}, \delta_{(p+1)_J} + a, \dots, \delta_{n_J} + a). \quad (2.2.8)$$

Variable a is needed to make sure Eq. 2.2.8 holds and is determined by:



Figure 2.10: Graphical representation of crossover function as derived from the data. Typical example in zebrafish imaging.

$$a = \begin{cases} \delta_{p_K} - d, & \text{if } (\delta_{p_K} - \delta_{(p+1)_J} > d) \\ \delta_{p_K} + d, & \text{if } (\delta_{p_K} - \delta_{(p+1)_J} < -d) \\ 0, & \text{otherwise} \end{cases} \quad (2.2.9)$$

Mutation

For the mutation operator we use standard settings commonly used for GAs. That is a uniform mutation, where each genome g has the probability to mutate: $1/n$ [1], where n is the genome length. For each genome:

$$g \rightarrow U(\bar{g}, g) \quad (2.2.10)$$

To make sure a uniform mutation is used we allow every slice center to mutate anywhere within the image space. Since the constrain $\max(\varsigma_{i-1} - d, \varsigma_{i+1} - d) < \varsigma_i < \min(\varsigma_{i-1} + d, \varsigma_{i+1} + d)$ holds, subsequent slice centers are not allowed to be more separated than distance d .

2.2.3 Multiple Object Recognition

The shape under study can have multiple slightly different (deformed) instances in one and the same image. The complete search space in this image is denoted by M_0 . First the best matching shape S_0 (with highest fitness) is localized. To decrease the search space for finding the next shape instance, we set all the elements contained within $S_0 \in M_0$ to 0 and name the resulting image M_1 .

This process is repeated by iteration and in that manner M_i is reduced for each next identified shape i . No shapes are found if the fitness of the found optimum $F(S_i)$ drops drastically, under a predefined threshold value r . The value of r can be determined empirically from a test on similar images (acquired under the same conditions). An example of such drastic fitness drop in fitness growth is depicted in Figure 2.11. This is a fitness plot for an image containing 3 fish shapes (cf. Figure 2.15a).

The pseudo code for finding multiple shapes in an image can be written as:

1. $i = 0$
2. $S_0 = GA(M_0)$
3. **while** $F(S_i) > r$ **do**
4. save S_i as found shape

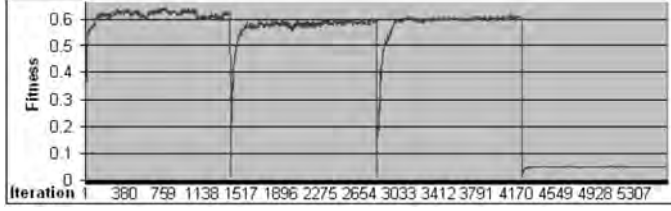


Figure 2.11: Fitness plot for an image containing three zebrafish shapes.

5. $M_i = M_{i-1} \setminus S_i$
6. $i = i + 1$
7. $S_i = GA(M_i)$
8. $t = t + 1$
9. end while

2.2.4 Experiments

To evaluate the performance of our template representation and GA optimization we have designed the task of finding objects based on predefined slice templates in images with different type of content. An experiment was performed with templates i.e. in synthetic as well as microscope images. With a simple interface the user selects both input images as well as the template shape for analysis.

Testing with Synthetic Images

We have used 20 synthetic binary images of 388×291 pixels. We have generated the images with different shapes located at different locations in images, slightly rotated, skewed and missing pixel data. Random noise (drawing debris) is introduced. The images randomly contain up to 3 instances of an object.

The first template used for the synthetic shapes is a template of an animal figure presented in Figure 2.12. We have created a synthetic binary image containing one deformed animal shape. The result of the application of the GA optimization is depicted in Figure 2.12.

The second template used for the synthetic shapes is a template of a house figure presented in Figure 2.13.

We have created a synthetic binary image containing two figures that have a deformed house shape. The result of the algorithm (implementation done in Delphi) is shown in Figure 2.13.



Figure 2.12: Very simple template of an animal shape (head, legs and body) and the result of shape localization in a synthetic image containing a simple animal shape instance.



Figure 2.13: Very simple template of a house shape and the result of shape localization in a synthetic image containing two simple house shapes.

Application to the Zebrafish Larvae Images

To evaluate the performance of our algorithm in a real-world imaging application we have chosen the task of finding zebrafish embryo shapes. The task at hand concerned using a High Throughput (HT) segmentation technique for retrieving the location and number of zebrafish embryo objects within images [54]. An additional requirement for this application is the need for automatic recognition of head, body and tail of each embryo. A typical binary image as presented for localization is shown in Figure 2.15a. This image was converted from color scale images to binary in a preprocessing step. We use a straightforward gradient operator (Sobel) followed by an iso-data threshold. In that way strong edge pixels were extracted for a binary representation [21].

Each image contains multiple zebrafish embryo shapes. The number of shapes in an image is not known in advance, the shapes might be overlapping. The shapes in all the images are assumed to be located approximately horizontal with a maximum angle of 45 degrees; so our approach could be used. We have applied the algorithm for a database of 100 images with the same settings for the GA ($m = 200$, $k_{size} = 7$, $r = 0.5$). For 87 images the amount of embryos in the image and the approximation of their shape could be retrieved correctly. In the cases where the algorithm failed, it was mostly due to large occlusion overlap or shapes being much longer or shorter than the proposed template. For the small and medium occlusions and overlap the algorithm performed correctly (cf. Figure 2.15). In Figure 2.14b the template of a zebrafish used in these results is

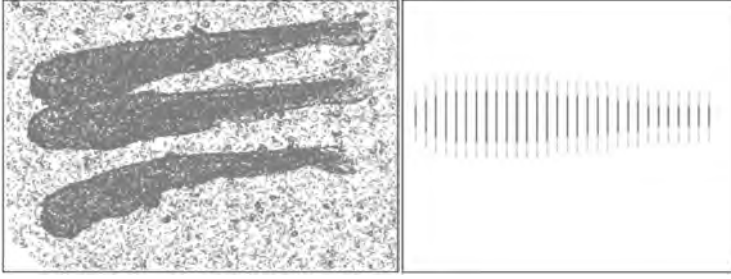


Figure 2.14: a) Typical binary image of zebrafish embryos in a resolution of 388×291 pixels. b) The template T_0 in a graphical representation that has been used on 100 tested images. Dark gray lines represent w_i and light gray lines represent b_i within slices. Some results are shown in Figure 2.15.

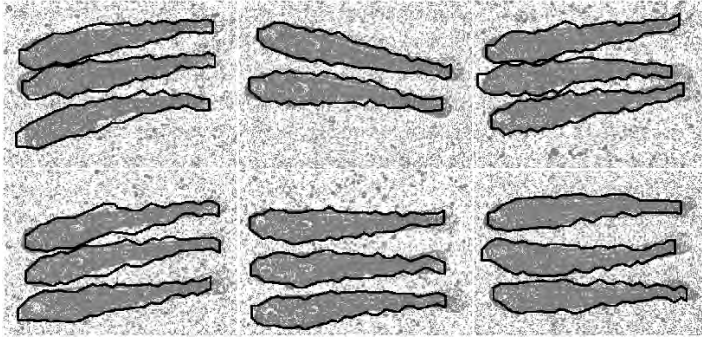


Figure 2.15: Graphical representation of template matching results for 6 images containing zebrafish larvae shapes.

shown in a graphical representation.

2.2.5 Conclusions and Discussion

In this section we have illustrated an application of a GA to optimize a deformable template approach. This approach can be used in different fields as the template can represent different shapes. Our approach was designed for an application in the HT screening of zebrafish embryos [54]. The optimization of template parameters is done through a Genetic Algorithm, which provides the possibility to search for an optimal solution in large search spaces. Our approach also allows to retrieve multiple instances of a certain object in a single image. Results indicate that this approach has a low error rate while computational performance is manageable and fast, which is, of course, very suitable for HT applications. Future work is directed towards a further generalization of the approach and making the template representation scalable.

2.3 Deformable Template Matching (Approach 2)

In the previous section we have illustrated a deformable template approach for the representation of a biological shape in a binary representation. We take this approach as a basis and further investigate the possibilities of a deformable template approach. We consider a representation on the basis silhouettes or boundary representations of prototype templates.

In research dealing with recognition of known shapes [18, 44, 16] the use of deformable templates is emphasized. Deformable templates typically are parametric class models; they start from a set of predefined parameters. The representation of the parameters might differ, but often a template is used, consisting of a set of contour points to which the basic shape outline is approximated. So, if the basic shape that is looked for is known, it still needs to be localized in the images. Therefore the prior knowledge can be exploited by choosing the parametric deformable shape template matching method as a basic approach [24, 70, 7]. Such an approach is used in the segmentation of cells in microscope images [18]. In this application the Hough transform approach is reformulated to be used as a deformable template. However, if the shapes are more complex than a circular shape like object, it is difficult to adapt to this approach. An example of a more complex shape is the segmentation of the masseter on the basis of a predefined template [44]; locally deformed instances of the template can be successfully extracted from input images. On the basis of this approach we propose a further generalization with which it is also possible to deal with multiple objects in one image as well as with global deformations; e.g., bending of the entire object. Based on silhouettes or boundary representations of prototype templates a considerable amount of research has been completed [16]. Usually the silhouettes are defined by contour points and make up the template. These templates can then be deformed by a set of parametric transformations, including both local and global transformations [70]. We have taken this traditional representation as a starting point; however, we have replaced the contour points in the silhouette by a contour area (cf. Figure 2.17). The reason for using this representation is that we would like to allow multiple overlapping instances of the object in one input image and therefore we have to accommodate for missing contour points.

In addition to the template matching, we also address the problem of shape normalization; in particular for applications of biological objects. The combination of shape localization and normalization has been successfully applied for the roundworm, i.e., *C. elegans* [47]. It is known as the BDB+ method. On the basis of the object boundary a straightening is applied so as to ease the further analysis of the objects. In our application, however, this method cannot be used; it starts from a predefined shape and then straightens the shape assuming the boundary has already been extracted. We want to investigate the recognition and straightening of more complex elongated shapes and, as indicated, account for the presence of multiple instances in one image.

The framework that we have elaborated consists of two steps. First, a preprocessing step including a segmentation of an input image in order to separate the object(s) of interest from the background is applied. Segmentation alone, however, does not give satisfactory results, as we are not only interested in separating background from fore-

ground, but we also want to recognize position and best possible representation of the object. This is realized in the second step consisting of a matching of a deformable template to the segmented image. This step is the main focus of this chapter. Finally, a post-processing step includes shape normalization through straightening of the extracted shape. Such is possible from contextual information about the object in the image that we have gained. Deformations are known and therefore deformations can be normalized according to the template. The framework was implemented in C++ using the *OpenCV* graphics library (<http://opencv.willowgarage.com>).

2.3.1 Method

The starting point of our algorithm is variation; i.e. a shape has variation, it can be inflicted with noise and it can be deformed or partially occluded. Our framework detects deformed instances of a predefined structure by means of deformable template matching and these instances are subsequently extracted from an input image. In Figure 2.16, an overview of the process is presented. The approach consists of two steps: the preprocessing step and the template matching step. First, during the preprocessing step, the input image is converted to a strict binary representation. In the main process the deformable template matching is applied in the binary image obtained from the preprocessing. This entails looking for the best match of a prototype template in the image. If a match is found the result is annotated according to the prototype template and henceforth, straightened.

Pre-processing

The first step in the analysis is retrieving foreground and background: i.e., an operation that converts an input image to a binary representation by marking the pixels which belong to foreground objects 1 and the background pixels 0. Different binarization methods are described in the literature; i.e., based on the usage of global or adaptive threshold methods, color or edge based segmentation. The choice of the method depends on the input image at hand, its properties and quality [21]. In the cases where prior spatial information is known this given can be exploited and the threshold value set can be based on this given.

After separation of foreground and background in the image, the contextual information still needs to be retrieved in order to recognize the objects of interest.

Prototype Template

The initial contour sketch of the object of interest is defined by the prototype template T_0 . The construction of T_0 is based on prior knowledge and is an approximate representation of how a typical object contour should look like and represents a contour location area. A few examples are shown in Figure 2.17.

A contour location area represents the region of interest within which the local template contour can exist and change. By doing so, the local deformations become limited

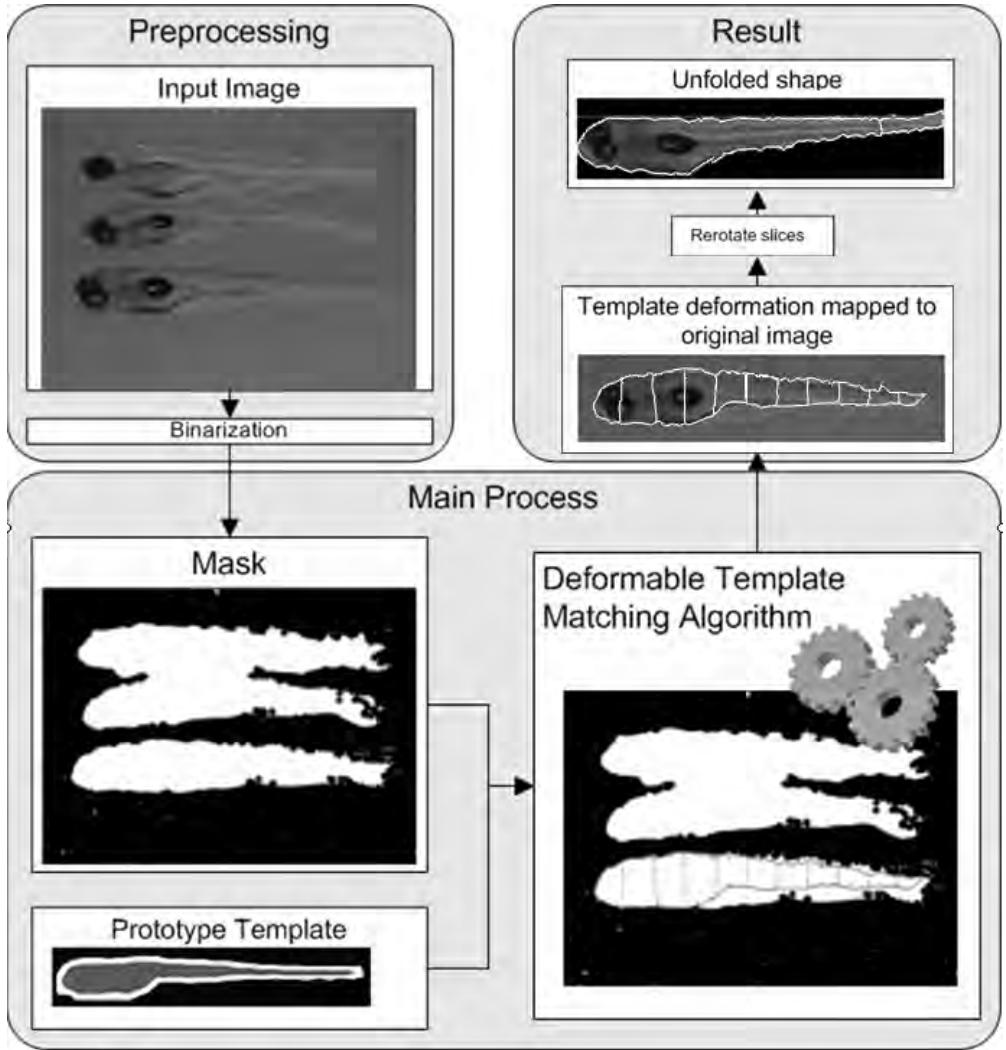


Figure 2.16: Proposed framework for automatic shape retrieval and straightening.



Figure 2.17: Some examples of prototype templates of different objects. Gray area represents the space where connected component boundaries might be located.

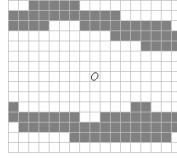


Figure 2.18: An example of matrix representing a template slice $t(i, j)$. Fields with value 1 are marked with gray color. Other fields have the value 0.



Figure 2.19: A prototype template as a chain of slices and a deformed instance. All shape boundaries that fit in the grey area fit the template. Black lines represent two example shapes that fit these templates.

by looking only at the pixels that are located within the contour location area (depicted in gray in Figure 2.17). Limiting the template boundary location of the deformed template is necessary to predict image boundary location in cases where it is missing, incomplete or overlapping.

As a result of this representation only global deformations are left, which will be described in the next subsection.

Parametric Transformation

Biological shape instances are often bended and rotated. To cope with these global deformations T_0 is distributed into n smaller sub-templates, so called slices t_i :

$$T_0 = t_0, t_1, \dots, t_n \quad (2.3.1)$$

A single slice can be seen as a rectangular matrix $t(i, j)$, consisting of binary values. This is shown in Figure 2.20.

The horizontal medial axis of the slice $O_{horizontal}$, is defined at $[i/2, *]$, and the slice origin as O at $[i/2, j/2]$.

Within the whole template the slices can rotate around their origin O to allow matching against rotated shape, but the origins are linked together as a chain (Figure 2.19); at any deformation of the total shape the distance between sequential slice origins remains the same.

A deformed template T' is derived from T_0 and is represented as $T' = (T_0, \underline{\xi})$. A deformation $\underline{\xi}$ of the slice chain is encoded by the following state-sequence:

$$\underline{\xi} = (x, y, \alpha_0, \alpha_1, \dots, \alpha_n) \quad (2.3.2)$$

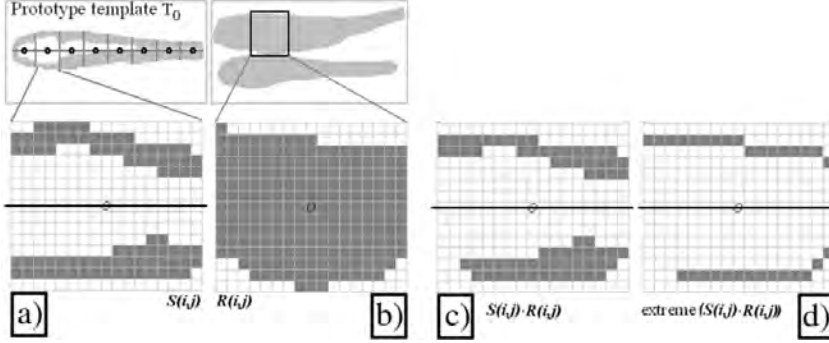


Figure 2.20: An example of template matching of a template slice $S(i, j)$ and a region $R(i, j)$.

Here x is the shift in the X-axis, y the shift in the Y-axis direction of the first slice t_0 and the angle of rotation α_i of each slice t_i . Due to the proposed slice based representation our deformable template approach is very suitable for use with elongated shapes.

Objective Function

The fitness of a template matching of an input image is measured by an objective function [24]. In [47] a parametric representation is used where the algorithm marches along the backbone of *C. elegans* to calculate the objective function.

A similar approach is applied, by comparing simultaneously the binary slice matrix $S(i, j)$ to a selected image region $R(i, j)$ of the same size (cf. Figure 2.20a,b). First the matrix is considered where both the template and the image region have overlapping foreground pixels, which are the result of $R(i, j) \cdot S(i, j)$ (cf. Figure 2.20c).

Since during this step the shape is a filled binary object, matches that are farthest from the slice center, yet still in the silhouette contour area are assumed to define the object. Then to get the actual border the algorithm marches along the horizontal medial axis of $R(i, j) \cdot S(i, j)$ iterating over 0 till j . Each orthogonal image plane pixel column p_i (with iterating from 0 to i) is compared to the template. It is assumed that the fish silhouette has only one silhouette pixel in the top and the bottom of each column. Therefore, per column the two *extreme* points that are of value 1 (as measured from the horizontal medial axis) remain 1, all other values are set to 0 (Figure 2.20d). The value of $extreme(S(i, j) \cdot R(i, j))$ is then the total amount of overlapping outer shape pixels between $S(i, j)$ and $R(i, j)$. This is treated as intermediate result. The quality (the objective function) of this result is then measured by the length of the retrieved border. The objective function is 1 if all pixels of the silhouette have been retrieved. Therefore the objective function for a slice $S(i, j)$ is:

$$\phi(S(i, j)) = \frac{0.5}{j} extreme(S(i, j) \cdot R(i, j)) \quad (2.3.3)$$

Since a template consists of n slices of the same size, $F(T, \underline{\xi})$ depends on the fitness function of each slice k :

$$F(T, \underline{\xi}) = \frac{1}{n} \sum_{k=0}^n \phi_k(S(i, j)) \quad (2.3.4)$$

Matching the Template to the Input Image

To check all possible occurrences T must be transformed, rotated and deformed using all possible parameters. To find the best solution there is a need to retrieve the global maximum of the fitness function for the input image. That is, to compare each $S(i, j)$ to all possible regions $R(i, j)$ within the image. A global search is computationally complex [28], especially when the image search space is large. In the literature Genetic Algorithms and dynamic programming approaches [33, 58] have been used for optimization.

Post-processing: Straightening the Template

After a sequence of slices is found, the shape can be straightened by rotating each found slice back. Since each deformed template T has a deformation defined by $\underline{\xi} = (x, y, \alpha_0, \alpha_1, \dots, \alpha_n)$ each of its slices is rotated back by the slice angle $-\alpha$. In that way the global deformation of the deformed template is reverted to the prototype template T_0 .

2.3.2 Application to the Zebrafish Larvae Images

Pre-processing

Because of uneven illumination, global threshold methods applied to a gray-scale version of the zebrafish images will not produce satisfactory results. We, therefore, employ an edge map based method to the input image. Edges define the boundaries between objects and background without strong dependency on flaws in the illumination. There exist several algorithms to create an edge map. After creation of the edge map a threshold must be set to select the strong edges.

Determining the threshold value of an edge map can be a cumbersome task. To set the threshold automatically without prior spatial relationship knowledge of the image, Otsu's method [46] might be applied.

However, we have prior spatial relationship knowledge of our particular data set and exploit this for our own border based method: the objects in the image are always located at some minimal distance d from image border. We try to exploit this. Thus we can assume that a layer of thickness d on the outside of an image contains only background pixels with some incidental noise. Of this layer we retrieve a number of local maximal pixel values. Of all the collected values we take the median value to determine the threshold value for the edge map.

To select the best preprocessing approach we compare the performance of different edge detectors (Sobel, Roberts) [21] in combination with Otsu's [46] method and our

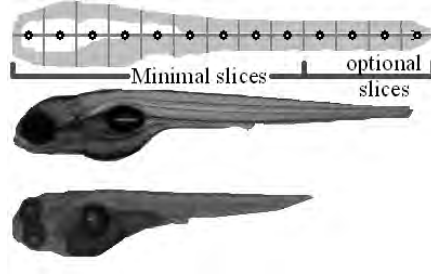


Figure 2.21: Minimal slices and optional slices in a prototype template. The template is compared to an example of a short and a long larvae shape.

border based method on 233 images. We count the number of objects in the segmented image. If the number of objects equals the number of objects in the original image we mark the prediction as correct. Both gradient methods in combination with our border method outperformed thresholding based on Otsu’s method. Out of the images incorrectly segmented in 17 cases the zebrafish shapes were touching each other and were thus connected. In all of these cases both edge detectors predicted 1 or 2 fish instead of 3 due to this connection.

After basic segmentation is realized, mathematical morphology operations are applied to get rid of noise and close up unwanted gaps: first the closing operator is applied to interconnect small regions and close holes, then connected components labeling (with filling up holes) is applied to obtain the closed shapes. In order to eliminate remaining noise we use the fact that we know the minimal area covered by a zebrafish. This area size can be automatically retrieved from the template size. Thus we remove all objects smaller then this minimal area. We do not remove objects that are larger than the maximal area, since larger object might be intersecting zebrafish shapes.

Main Process: Deformable Template Matching Prototype Template

Our initial zebrafish template in Figure 2.21 is created from averaging a test set of training shapes [10]. This template is created from averaging a set of 20 zebrafish larvae shapes.

Zebrafish larvae tend to have different lengths. Therefore, the length of the template is not fixed, but can vary between some minimal (min) and maximal (max) number of slices t_{min} and t_{max} . If the number of found slices is smaller than min or larger than max we assume the shape is not found. All the slices t_x with $min < x < max$ are thus optional slices. This is depicted in Figure 2.21. The max and min are set based on the length of the longest and shortest encountered zebrafish.

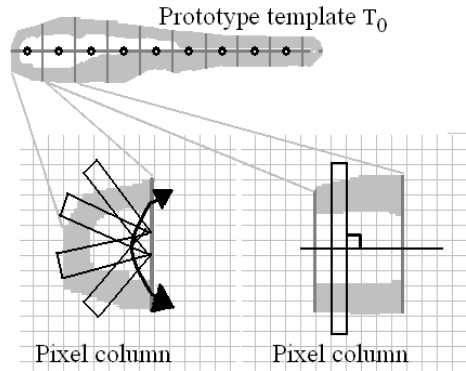


Figure 2.22: Marching direction for the first slice depicted in the image pixel matrix.

Objective Function for the First Slice.

The described objective function is applicable for slices in which the important information is located above and below the slice center. While most of the zebrafish larvae template applies to this condition the very first (head) slice does not. The reason for it is that its shape is close to half circular. This is depicted in Figure 2.22.

To cope with this case, instead of retrieving extreme values above and below the median axis (described in Section 2.3.1) extreme values are retrieved in a circular way as shown in Figure 2.22.

Matching the Template to the Shape

A *top-down* dynamic programming approach is applied with a hash table for intermediate result saving, to obtain a global optimum.

In our test case the larvae shapes are located in an approximately horizontal position in the input image. This given is used to reduce the search space. This is done by assuming that each slice can rotate between -45 to 45 degrees as measured from the image horizontal axis. Additionally a discrete set of deformation angles for each slice is used.

To further reduce the search space a *Multi resolution* algorithm [30] is used. First the solution is located on a low resolution template and a low resolution input image. Then, the solution is used for initialization in a higher resolution.

2.3.3 Experiments and Results

An evaluation of our algorithm is performed on a data set consisting of 233 images. The data set was directly obtained from a running experiment. Out of these images 177 (76%) contained multiple (3) zebrafish larvae, 33 (14%) contained 2 larvae and 23 (10%) 1 larva.

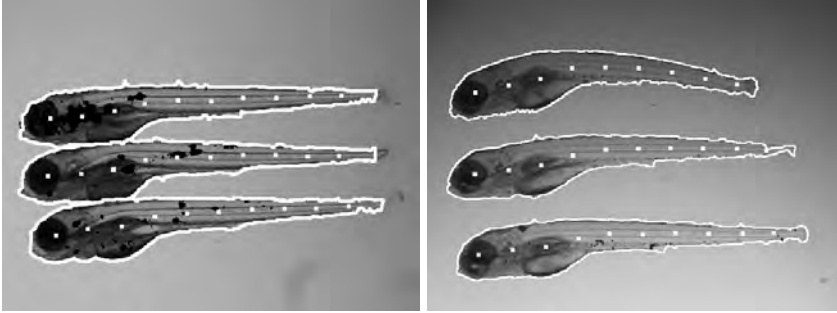


Figure 2.23: Automatic localization of the zebrafish shapes using deformable template matching. The white line defines the shapes as found by the algorithm. White dots represent the found slice centers. In the left image the shapes are touching each other, which makes recognition more difficult. These images were created using the same template (these images were used to determine bacterial infection, black regions, in each larva separately).

A first basic test is to check for how many of the tested images the number of larvae is predicted correctly. The result was that for all images (100%) the number of the larvae shapes (1, 2 or 3 larvae, even if overlapping) within each image was correctly retrieved by our framework.

The automated prediction of the number of shapes in an image looks promising, but only retrieving the number of shapes is not sufficient for a proper analysis of the results. To that end we also test the accuracy of the algorithm, that is, how precise the shapes were retrieved. In Figure 2.23 representative results of retrieved shapes are depicted. We show shapes that are deformed in different ways as well as shapes touching each other. The template in Figure 2.21 was used for the creation of all these images.

We are not aware of other methods that can be used for automatically retrieve predefined shapes from images without an initialization, and therefore we have compared the resulting shapes of each retrieved zebrafish larva shape against ground truth images of the same shapes annotated by experts. In order to have manageable proportions in the evaluation, we reduced our test set to a total of 104 zebrafish shapes (distributed among 35 images, containing up to 3 shapes per image).

Four experts (test-person T1, T2, T3, T4) were asked to delineate the correct outline of the zebrafish larva. Drawing the shapes was realized with a LCD-tablet (*Wacom*) using TDR software [61].

Next, the precision of our approach is compared by applying it to the same input data (algorithm output A). A comparison of human to automatic retrieval was also used in [47] for validation of the results. The accuracy of our shape retrieval algorithm is measured by the equation proposed and explained in [44]:

$$accuracy = 2 \times \left(\frac{N(M_{area} \cap S_{area})}{N(M_{area}) + N(S_{area})} \right) \times 100\% \quad (2.3.5)$$

	A	T1	T2	T3	T4
A	-	96.85	97.19	96.29	96.47
T1	96.85	-	97.61	97.21	96.81
T2	97.19	97.61	-	97.17	97.46
T3	96.29	97.21	97.17	-	96.68
T4	96.47	96.81	97.46	96.68	-

Table 2.2: Accuracy comparison of our algorithm A and the test subjects T1, T2, T3, T4. The matrix is symmetrical, yet we have shown all the values for viewing convenience.

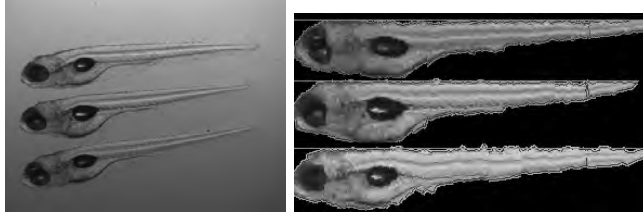


Figure 2.24: Automatic straightening results of zebrafish larvae. Image (left) is the input image. Image (right) is the straightening result.

We have compared the accuracy of our algorithm to T1, T2, T3, T4. The average accuracy was established as 96.71% ($\sigma = 1.27$). Note that this accuracy could not be achieved only by the segmentation step, as 35 (of the 104) shapes used for this test were touching each other and their boundary was derived through the template matching.

Table 2.2 depicts the results of the comparison of the accuracy of our algorithm A with the test persons. In addition the inter-observer variation is analyzed.

As can be seen from the table the accuracy between our algorithm and each test person is as close to the accuracy of the test persons to each other. This indicated that the algorithm retrieves shapes as good as or comparable to manual retrieval.

In the last part of the experiment the objects, i.e., zebrafish, are straightened. This is accomplished using a template with a straight top border in order to align the found slices with their top to a horizontal line. The results are shown in Figure 2.24 and Figure 2.25. To retrieve and straighten a single zebrafish shape from a 2592×1944 image our application needed about 35s of CPU time on an Intel Dual Core 2.66Ghz, 1.00Gb.

2.3.4 Conclusions

In this section we further investigate the possibilities of a deformable template approach. The prototype template proposed in this section is represented as a bitmap and can easily be adapted to the needs of the application while the same algorithm is used. Results can then be compared by shape normalization. The algorithm we propose does not rely on initial localization of the shape and therefore does not require any manual intervention

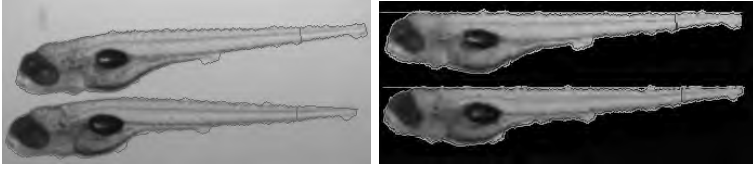


Figure 2.25: Automated straightening results of zebrafish larvae. Image (left) shows the found shapes projected on the input image. Image (right) shows the automatic straightening result.

or analysis. The framework was applied in an experimental set up for HT screening with a readout in images. In the application to zebrafish screening an average accuracy of about 96 percent has been achieved. The framework can be easily adapted to work with other shapes, e.g., in the life sciences or in other fields that require accurate and robust shape retrieval. Further analysis of the object straightening precision is part of the future work.

3 Anchor Region Based Pattern Recognition in Image Space

Under preparation for publication and based on:

A. Nezhinsky & F.J. Verbeek: Anchor Region Based Pattern Recognition in Image Space

3.1 Introduction

We have designed and developed a generic algorithm for use in High Throughput applications in the life sciences. We want to automatically analyze images; in particular we aim at the detection and annotation of shapes of biological interest.

Shapes often seem similar to the human eye; this is a perceptual generalization as each individual (instance) is different. Size, position and a lot of other aspects differ. Global proportions and image magnification are the same. An image can contain multiple shapes and some of them are clustered together which complicates automatic analysis. Without proper localization and annotation of the regions in the shapes, the measurement of features within each instance is hampered. Additionally, each shape should be annotated in such a way that it can be compared to other retrieved shapes. For our analysis we want to establish an uniform way to overcome these problems.

The analysis is part of a High Throughput setup and therefore the starting point for our approach is, that in at least 95% of the cases, retrieved shapes should be detected and annotated correctly.

As a case study we have considered High Throughput analysis of zebrafish as it requires automated analysis of thousands of zebrafish larvae [54]. For each zebrafish larva, a brightfield image is acquired as shown in Figure 3.1. The brightfield image is used for the localization and annotation of the zebrafish shape. Until recently, these images were manually analyzed. The analysis included localization of the zebrafish shape in the brightfield images.

This chapter is structured as follows:

- In Section 3.1.2 we describe the development of the algorithm.
- In Section 3.3 we continue with the application of the algorithm to zebrafish larvae.
- In Section 3.4 we extend the application of the algorithm to other shapes.
- In Section 3.3.4 a typical case study with statistical processing of the result is presented to show the virtue of the current application.
- We close this chapter with a discussion of all results presented (Section 3.5).

3.1.1 Related Work

In order to make our approach more generic we do not design an algorithm specific for our case study, but rather design an algorithm capable of success when applied to other problems. The general starting point therefore, is to find the objects that can be globally defined by their shape. In addition, we investigate the effect of the presence of more than one instance. As prior knowledge we adapt the fact that the global shape is known but the position for each shape is not known. We do not, however, consider active snakes or other *free form* methods as a possibility.

We assume that in an image only the shapes of interest and some accidental noise are present. Therefore, if we can separate the foreground from the background the

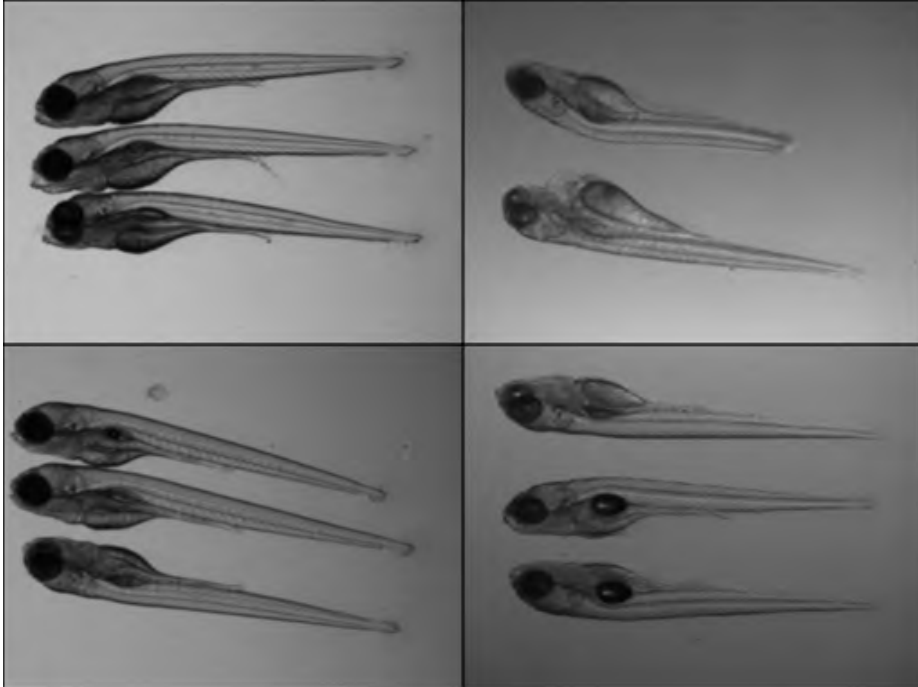


Figure 3.1: Examples of brightfield images (converted to grayscale) containing zebrafish embryos. Note the differences in illumination resulting in shades in the background and variations in intensity.

foreground object there will most likely contain the shape of interest and/or some accidental noise. An example of an approach combining both *bottom-up* and *top-down* methods is given in [22]. The authors first exploit *bottom-up* image cues to create an over-segmented representation of an image and then merge the segments by assigning labels that correspond to the object category.

By the binarization (*bottom-up*) approach we are converting the problem from the RGB domain to the binary domain, [21]. The choice of approach is problem related and is strongly dependent on the data at hand. In the cases where prior spatial information is known this characteristic can be exploited and used for setting the threshold value. We have made a small inventory of different approaches for binarization and compared the results.

Following the binarization we require a *top-down* approach for the pattern recognition step, that can simply use a sketch or a logic representation of the object we are looking for. A deformable template approach can be used for this purpose. Binary images are successfully used for template and polygon matching [17, 29].

When binary objects are unintentionally interconnected it is good practice to split up the binary image into a collection of binary regions based on some geometric assumptions about the data. In the literature the rules to split up large objects are the *minima rule* and the *short cut rule* [33] suggesting splitting an object at its concavities. Finding concavities is preceded by finding the convex hull of an object [6]; this method can be very effective. In such problems the choice of the algorithm that is used to split the binary objects based on their concavities is very problem dependent. The recognition step still must be performed on the resulting binary elements. We want to investigate if a more globally operating method can be found.

A brute force template matching algorithm that uses straightforward matching of a template element to every possible deformation, transition and rotation would be too computationally intensive [28]. This problem can be overcome by reducing the search space through application of predefined rules or a method should be used which is not dependent on pixel by pixel comparison. Predefined rules can be retrieved from *a priori* knowledge about the image. We therefore want to investigate a new direction in template matching approaches as addition to existing techniques. To that end we developed an algorithm capable of retrieving deformed shapes based on a prototype model.

3.1.2 Algorithm Overview

We propose a complete framework for analysis and recognition of biological shapes. It consists of multiple steps and several scenarios. The analysis pipeline is shown in Figure 3.2. As input a color or grayscale image (*Input image*) is used and in addition a *Prototype Template* of the shape that is looked for.

First, a *Preprocessing* step is applied in order to separate the object(s) from the background and by this the Input Image is converted to a *Binary image*. Segmentation alone, however, does not give satisfactory results, as we are not only interested in separating background from foreground, but we also want to recognize the particular shapes and their position and best possible representation in the image.

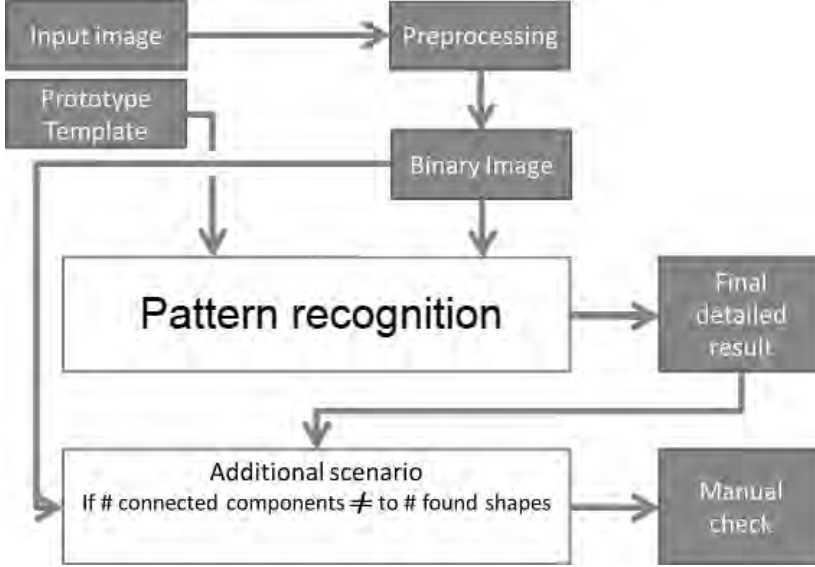


Figure 3.2: Proposed framework for the main scenario of shape localization and annotation.

The second step, a *Pattern Recognition* step, is applied in order to recognize the shapes from the binary image and annotate them. This is done by retrieving the best representation of the prototype template in the binary image. The resulting shape (*Final detailed result*) can be used for further measurements.

In order to make this model fit for High Throughput (HT) applications we propose a scenario so that the results produced by the main scenario need to be checked by a human for consistency. This scenario is an addition to the fully automated scenario in the following way. From the *Preprocessing* step the number of *Connected components* that are extracted as foreground is retrieved. In cases in which the shapes are not overlapping each other and no other objects are present in the image, the final result should present the number of objects that equals the number of connected components. Therefore we can assume that in most cases the number of connected components from the preprocessing step equals the number of objects present in the image. If the number of connected components does not equal the number of retrieved shapes from the final result, we mark this result as something that needs to be evaluated by a human observer for correctness. The complete process is shown in Figure 3.2.

3.1.3 Pre-processing

For the proposed approach we need a bottom up technique that can easily separate foreground and background. The production of a binary image I_b from a grayscale or a RGB color image differs per problem at hand and will therefore not be the focus here.

However, we will in short describe the binarization algorithm for a specific shape, since we will use it as an example throughout this chapter.

Thresholding of the typical images containing biological objects is not a trivial process. This is due to the fact that images often suffer from uneven illumination as result of microscopy, contain noise and have different color values representing the similar structures. As an example consider the images and their differences in Figure 3.1.

We have evaluated the performance of different thresholding methods for the images as used in this case study [43]. For the segmentation we did not use a texture based descriptor, since it was reported [69] that this method produced false positives when a background region contained texture similar to the object of interest. A detailed comparison of binarization methods for the images containing zebrafish is given in Chapter 2.

For the binarization we have probed a state of the art method, mean shift segmentation [59]. This approach distributes the image into connected regions (or clusters) based on color similarity. Since the mean-shift can run with different parameter settings these should be derived from the data. To accomplish this we run the algorithm with different settings.

To determine the correct settings for the mean-shift segmentation algorithm for the case study we use the assumption that the shape is never located at the edge of the image (1) and that the objects of interest are smooth shapes (2):

1. We assume the objects of interest are never located at the edge of the image. Therefore we can assume that the image border consists for the most part or solely of the background: all pixels that are part of the same cluster as the border pixels are also background. As a result of this assumption the background can be easily subtracted from the image, and in this way only the objects of interest and noise remain.
2. A smooth object will tend to have a lower compactness (Eq. 3.1.1) then edgy ones, so we can use this measurement to determine if an object is present in the image. For each collection of clusters resulting from mean-shift segmentation we start with checking the compactness of the objects (or clusters). We subtract the background clusters from the collection and of the remaining ones we calculate the average compactness. The settings that provide the highest compactness are chosen as best settings. In order to calculate compactness area and perimeter of each binary object o_i are considered:

$$compactness = perimeter(o_i)^2 / area \quad (3.1.1)$$

As an example throughout this section we use the binary image I_b as shown in Figure 3.3. This image was derived from the grayscale image through mean-shift segmentation. The result of the preprocessing is a binary image without annotation of the shapes. Moreover non specific shapes, noise and debris are still present in the image and need not be taken into account for our measurements. Therefore, additional segmentation is needed.



Figure 3.3: An example of grayscale and binary image containing zebrafish embryos and some artificial debris.

3.2 Pattern Recognition: The Anchor Region Based Method

Now that we have removed the background with the pre-processing step we need to recognize and annotate the shapes that are segmented from the binary image. As *Pattern Recognition* step we have developed a Deformable Template Matching method: the anchor region based algorithm. In Figure 3.4 an overview is shown.

We investigate the feasibility of a new Deformable Template Matching approach compared to solutions presented in Chapter 2.

The template matching takes a binary image I_b as obtained from the preprocessing step. Binary image I_b consists of two types of regions: foreground regions R_0, \dots, R_i and the background.

Foreground regions R are defined as a collection of interconnected pixels with value 1. Background regions B are defined as a collection of interconnected pixels with value 0.

3.2.1 Prototype Template

The prototype template consists of template elements (t_0, t_1, \dots, t_n) . The sequence of elements in the tuple defines the order of these objects. Each element t is defined by its size r and a length limitation l :

- r is the radius of a circle that inscribes the object to be determined
- l is the distance between the inscribed circle centers of t

Graphically this is depicted in Figure 3.5.

Since we want to focus on biological shapes the size and length may vary substantially. Therefore we represent r and l as variables instead of constants. We assume, however, that the ratio of different template elements as relative to each other is the same, while the object itself might be scaled. In this way we can define a collection of shapes with variable width and length that all match the shape of interest.

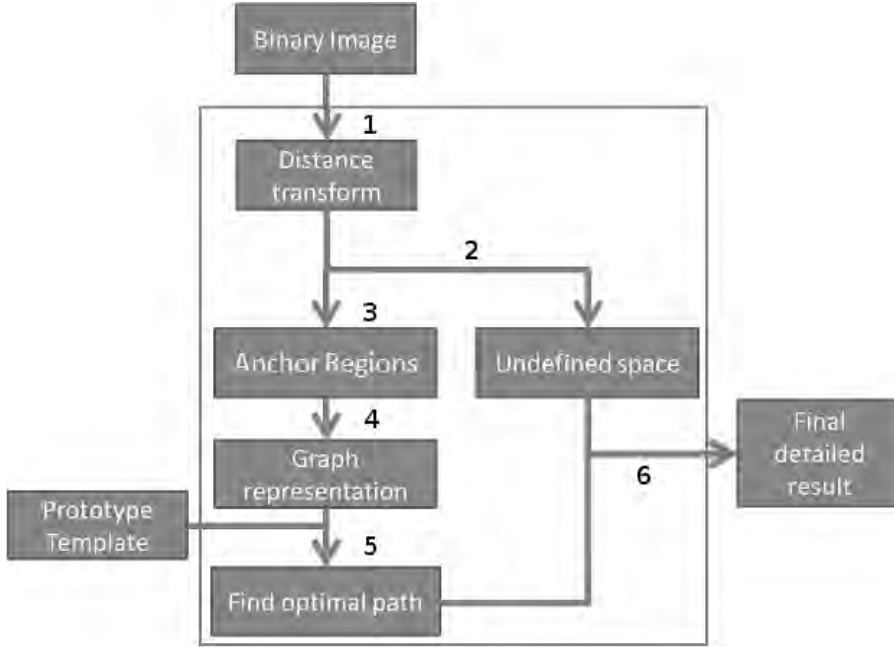


Figure 3.4: Overview of the proposed framework for Object Recognition. The first input is a binary image that consists of fore- and background regions that are obtained from the pre-processing. Second input, a prototype template, represents the ideal shape of the object that we would like to retrieve. As output detailed results are provided in the form of object instance locations. The numbers $(1, \dots, 6)$ refer to the explanatory text.

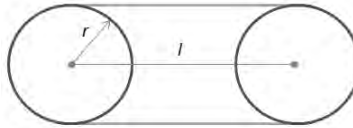


Figure 3.5: A template element $t(r, l)$, defined by the inscribed circle radius r and inner length l (extension within which this circle can shift).

Example Template

For example, if we want to localize all elements that look like a zebrafish larvae as depicted in Figure 3.3 we can define the prototype template as a head object t_0 followed by a body object t_1 followed by a tail object t_2 . The prototype template shape sequence is then $[t_0 t_1 t_2]$. Now we construct the template as follows.

For the head region instead of a constant value for r we define $r_0 \geq 0, r_0 < r_1$. This means that the head element must have a radius of inscribed circle smaller than for the body region.

The radius of the tail region at its largest should be smaller than the body region, with a minimum of 70%, thus, $0 < r_2 < 0.7r_1$.

We assume that the length of the head and body region together should be less than half of the entire shape, thus, $l_0 + l_1 < 0.5 \cdot (l_0 + l_1 + l_2)$.

The zebrafish larvae shape can be represented as:

$$T_0 = \begin{cases} [t_0 t_1 t_2] & \text{sequence} \\ r_0 \geq 0, r_0 < r_1, 0 < r_2 < 0.7r_1 & \text{radius relationship} \\ r_{min} < r_i < r_{max} & \text{radius limits} \\ (l_0 + l_1) < 0.5 \cdot (l_0 + l_1 + l_2) & \text{length relationship} \\ l_{min} < (l_0 + l_1 + l_2) < l_{max} & \text{length limits} \end{cases} \quad (3.2.1)$$

The values of $r_{min}, r_{max}, l_{min}, l_{max}$ are here to restrict the allowed object size or length for parts of the zebrafish object. If we set $r_{min} = 0, r_{max} = \infty, l_{min} = 0, l_{max} = \infty$, the representation becomes fully scale independent.

3.2.2 Conversion from Image to Graph

In this section we elaborate on the steps 1, 2, 3 and 4 from Figure 3.4. The template matching step consists of searching for characteristic regions in I_b as they are defined in the prototype template T_0 . In order to do so we need to convert the search space into a representation that can be used with the template T . To that end we need an approach that converts the representation of the objects in the image from binary shapes to a graph that can be traversed and evaluated. The representation must be reversible, that is, upon localization of the template elements in the graph we must be able to convert graph nodes to the original shape.

The concept of anchor points [57] is very helpful to obtain the template elements. Anchor points are defined as a collection of non-removable pixels during the thinning process. In [57] a set of anchor points (RCDM) based on a reduced set of *center of maximal disks* (CMDs) is retrieved for reversible skeletonization.

Our approach is based on retrieval of a set of CMDs followed by the reduction of the set in such a way that the global characteristics of the shape still can be described while the search space is drastically reduced. We adapt the basic method to our needs.



Figure 3.6: *Euclidean Distance transform D* of the binary image from Figure 3.3. An inverted lookup table is used, therefore the lighter pixels correspond to low DT function values (closer to the background) and dark pixels correspond to high values (further from the background).

Compute Distance Transform

The Euclidean Distance Transform is a tool common in shape matching and doing measurements related to distance and CMD retrieval [67, 5, 13] and therefore it lends itself perfectly for the problem at hand. The Euclidean Distance Transform (EDT, or DT for short here) is a generally used operator which stands for the calculation of the smallest Euclidean distance from each pixel to a region of interest (cf. [15, 5, 14]).

As input image we use the binary image mask I_b . As output we construct an image that will represent a DT map which will denote the radius of each region (step 1 in Figure 3.4). The resulting Distance Transform has the same dimensions as I_b and is denoted as D . The value of each pixel in the DT image corresponds to the distance to the background d . The result of a DT applied to the grayscale image in Figure 3.3 is shown as in Figure 3.6. In this image the locations of a larger inscribed circle for I_b have lighter color.

Convert Distance Transform to Anchor Regions

Now from the DT the Anchor points are obtained (step 3 in Figure 3.4); the DT is reduced to a set of anchor points through repetitive dilation of region in D . This object will be called D_{best} .

We set the starting $d_i = r_{min}$ (smallest radius of the object we are looking for) and increase by $d_{i+1} = d_i + \delta$ while $d_i \leq r_{max}$ with stepsize $\delta = 1$. We choose $\delta = 1$ in order to simplify the calculation, as a consequence the resulting anchor points may have a thickness of more than one pixel and might not be accurately reversible. In our case, the representation is used for recognition of the general outline of the object during the template matching step and complete reversibility is not required. Note that the anchor points in this case will be represented as clusters of non-removable pixels rather than one-pixel points. Therefore we prefer the term *anchor regions* instead.

For each iteration the following is done. First we threshold D with distance value d_i . The resulting image $D[i]$ will contain only the regions that were created with element width r_i . For each pixel at $D(x, y)$ the following is done:

$$D[i](x, y) = \begin{cases} 1 & D(x, y) = d_i \\ 0 & otherwise \end{cases} \quad (3.2.2)$$

$$D_{best}(x, y) = \max(D[i](x, y), D_{best}(x, y)) \quad (3.2.3)$$

Now, D_{best} contains only the largest radius regions.

To retrieve the approximation of the shape that is covered by the retrieved radius region we perform backtracking, therefore $D[i]$ is subsequently dilated with a circular structuring element E_i with a radius d_i . The result is subtracted from D in order to prevent evaluating at the same area in the next iteration:

$$D' = D - (D[i] \oplus E(d_i)) \quad (3.2.4)$$

Each result is the input to the next iteration. At completion of iterations (stop criterion defined as $d_i \leq r_{max}$) D_{best} will contain only the global anchor regions a_0, a_1, \dots, a_n . In Figure 3.7 the resulting D_{best} is depicted.

In order to define each anchor region a from D_{best} in such a way that it can be used as a node in a graph, an anchor region a is represented in the following manner: $a(d, F, c)$, where:

- d is the radius of the circular structuring element belonging to the anchor region. For each a we store the $d(a) = d_i$ that was used for its creation.
- F is the shape representing the anchor region. It is a collection of pixels that contribute to the anchor region shape.
- $c(c_x, c_y)$ is the anchor point, which also is the center of mass of the shape.

The overview of the entire process is given in the following pseudo code fragment:

1. $d = r_{min}$
2. $i = 0$
3. **while** $d \leq r_{max}$ **do**

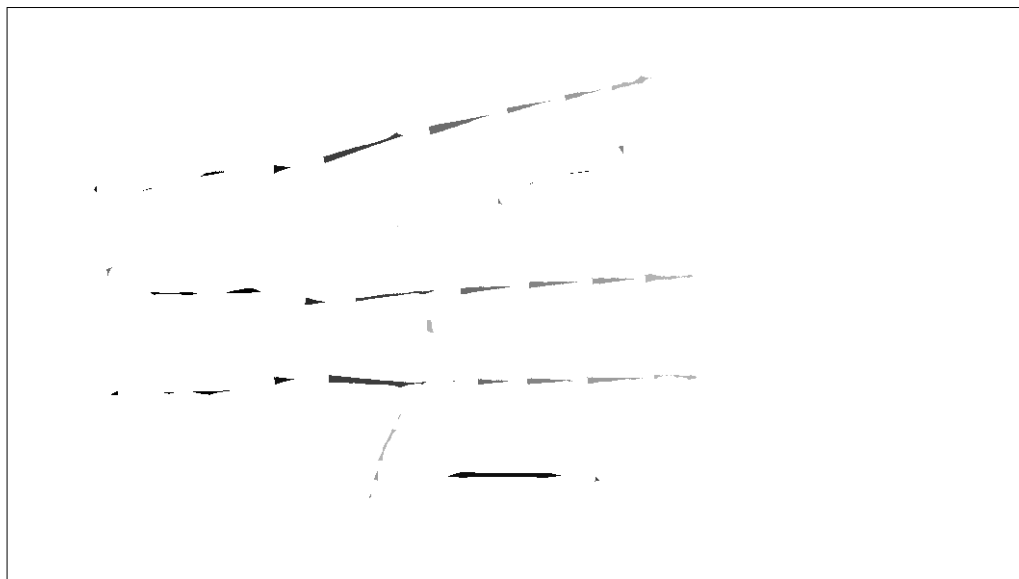


Figure 3.7: D_{best} : a collection of anchor regions a_0, \dots, a_n selected from the Distance Transform of example in Figure 3.3. d for each a is encoded as a grayscale value. Image saturation contrast has been improved for visual purposes.

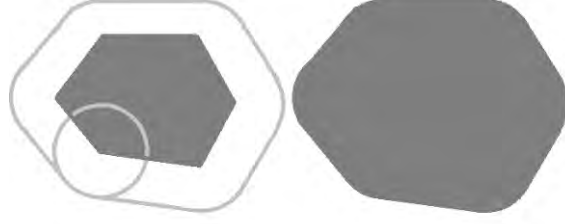


Figure 3.8: Example of a dilation process of an anchor region by a circular structuring element. Dark area to the left is the anchor region being dilated, resulting in the dark area to the right.

```

4.    $i = i + 1$ 
5.    $d = d + 1$ 
6.   for  $x = 0$  to  $ImageWidth$  do
7.     for  $y = 0$  to  $ImageHeight$  do
8.       if  $D(x, y) = d$ 
9.          $D[i](x, y) = 1$ 
10.      else
11.         $D[i](x, y) = 0$ 
12.      end for
13.    end for
14.     $D_{best}(x, y) = \max(D[i](x, y), D_{best}(x, y))$ 
15.     $D = D - (D[i] \oplus E(d))$ 
16.  end while

```

Determine Undefined Space

Now that we have determined all anchor regions in D_{best} we can consider the area it represents. A consequent dilation of every anchor region a_k will result in such area. A graphical example of a dilation process of an anchor region is shown in Figure 3.8.

Taking Figure 3.7 as an example, if we perform a dilation of each a_k with a circular structuring element with radius $d(a_k)$ this will result in a configuration as depicted in Figure 3.9:

$$F(a_k) \oplus E(d(a_k)) \quad (3.2.5)$$



Figure 3.9: Dilation of every anchor region shape in D_{best} from Figure 3.7.

The result obtained by application of Eq. 3.2.5 captures the original shape in a global manner but is missing out on the small details; more specifically, objects that have $d < r_{min}$ are discarded.

This error is introduced through the usage of non linear filters. For retrieving the global shape of the objects this does not give any problem.

However, since we need to reason about the retrieved regions, we save all these regions as undefined regions U (step 2 in Figure 3.4). The region U can be determined by a subtraction of the dilated D_{best} from the original binary image:

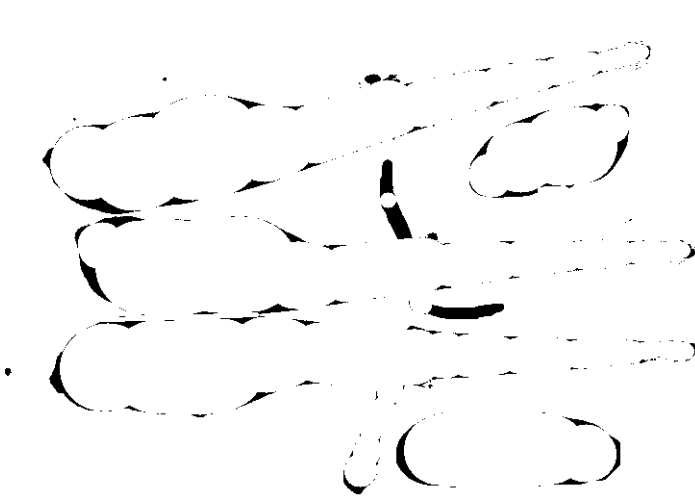
$$U = I_b - (D[i] \oplus E(d_i)) \quad (3.2.6)$$

for all i . For the example in Figure 3.10 the resulting undefined region image is depicted.

Connecting Nodes

The next step consists of creation of an undirected graph G connecting the anchor regions (step 4 in Figure 3.4). This is accomplished by connecting all neighboring anchor regions through their centers of mass.

We define a_m to be a neighbor of a_n if the elements of the line from the center of mass of a_m to the center of mass of a_n ($c(a_m)$ to $c(a_n)$) are entirely within regions belonging to the dilated sequences of these anchor regions (or nodes). An example of this neighboring definition is shown in Figure 3.11. This definition is comparable to the definition of Voronoi (or Delaunay) neighbors: if two Voronoi regions share a boundary,


 Figure 3.10: Undefined space U .

the nodes of these regions are connected with an edge. In our case the regions themselves can not be seen as Voronoi regions, however the neighboring property is the same.

In order to handle the undefined space the previous rule is extended as follows. We define a_m to be a neighbor of a_n if the elements of the line from $c(a_m)$ to $c(a_n)$ are entirely within regions belonging to the dilated sequences of these anchor regions (or nodes) or to undefined regions, cf. Eq. 3.2.7. An example is shown in Figure 3.12.

$$\overline{c(a_m)c(a_n)} \in F(t_m) \oplus E(d(a_n)) \cap F(a_n) \oplus E(d(a_n)) \cap U \quad (3.2.7)$$

In Figure 3.13 we show G as it is created for the example of the zebrafish larvae.

The rule as described in Eq. 3.2.7 is always applicable for neighbor definition except in pathological cases, for example due to poor image quality.

Convert Anchor Regions to Template Elements

Since a path in G is a sequence of anchor regions these need to be converted to template elements. One template shape element t_i can consist of multiple anchor regions. To that end a collection of anchor regions $[a_0..a_n] \in t_i$ if:

- All anchor regions are of the same radius as the template element:
 - for all i with $0 \leq i \leq n$, $d(a_i) = r_i$
- The length between the start and the end of the anchor region chain fits the template element length definition:

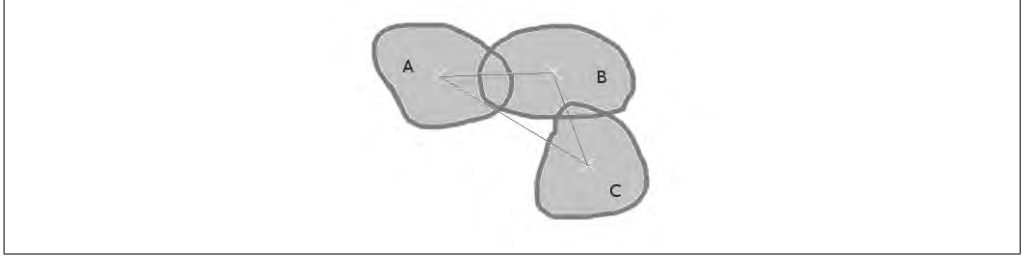


Figure 3.11: Consider the areas A , B and C . Elements of the line from $c(A)$ to $c(B)$ are entirely within regions A and B . This makes A and B neighbors. Elements of the line from $c(B)$ to $c(C)$ are entirely within regions B and C . This makes B and C neighbors. Some elements of the line from $c(A)$ to $c(C)$ are not within regions A and C . A and C are therefore not neighbors.

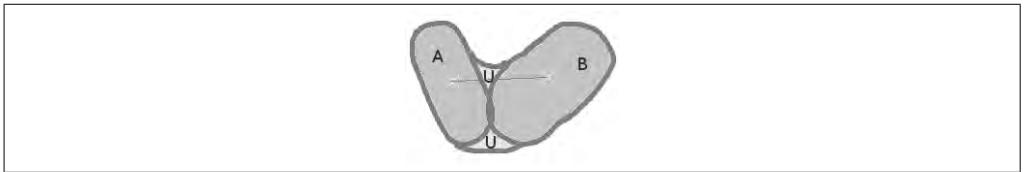


Figure 3.12: Consider the areas A , B and U . U represents the undefined regions. Elements of the line from $c(A)$ to $c(B)$ are not entirely within regions A and B . However, elements of the line from $c(A)$ to $c(B)$ are all within regions A , B and U . A and B are therefore treated as neighbors.

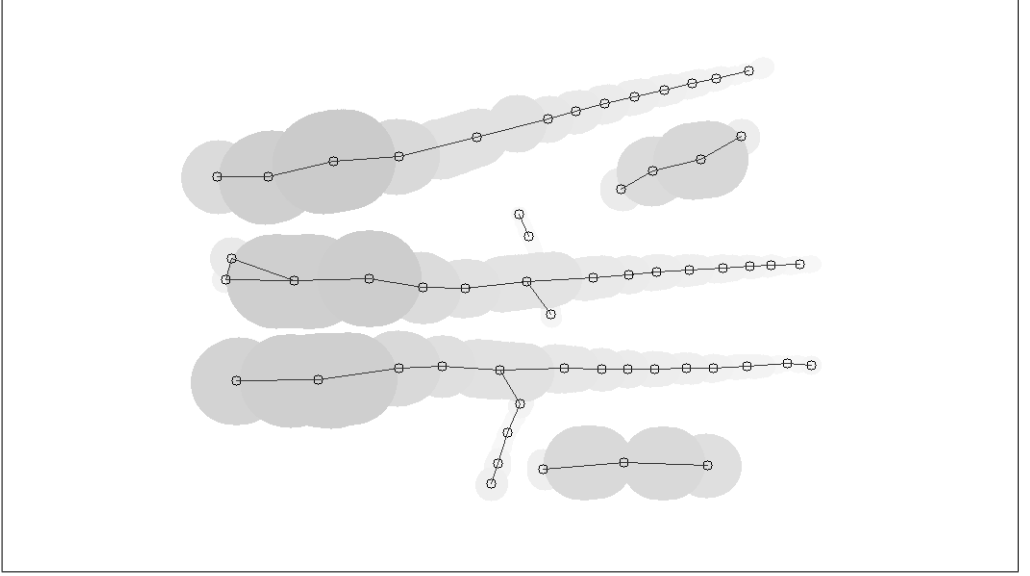


Figure 3.13: Graph G depicted on image containing the dilation of D_{best} .

$$-l_{min}(i) \leq \overline{c(a_0)c(a_1)} + \overline{c(a_1)c(a_2)} + \dots + \overline{c(a_{n-1})c(a_n)} \leq l_{max}(i)$$

Now that we have a conversion approach for anchor regions to template elements we can assume that graph G consists of template elements t .

3.2.3 Bayesian Formulation

In the graph G , representing the input image, we will search for the presence of the prototype template. Therefore G is traversed. An optimal solution determines the location of the best matching shape.

An optimal solution is defined as one with the highest probability being a deformed instance of the prototype template. The probability of a shape T_i being present in an image is expressed as: $P(T_i|G)$:

$$P(T_i|G) = \frac{P(G|T_i)P(T_i)}{P(G)} \quad (3.2.8)$$

Let δ be any deformation of the prototype template T_0 . Then, according to [60] $P(G|T_i)$ can be expressed as marginal probability sum of all probabilities for a deformation to happen joint with the probability for its validity $P(G, \delta_i|T_i)$. This sum is approximated by $\int P(G, \delta_i|T_i)d\delta$ and can be rewritten in the following way:

$$P(G|T_i) = \int P(G, \delta_i|T_i)d\delta = \int P(G|\delta_i, T_i)P(\delta_i|T_i)d\delta \quad (3.2.9)$$

Then, in order to maximize the probability that given graph G we can identify our template T in it can be related to the maximum of the following expression:

$$P(G|\delta_i, T_i)P(\delta_i|T_i) \quad (3.2.10)$$

$P(G|\delta_i, T_i)$ represents the *likelihood*. $P(\delta_i|T_i)d\delta$ is the *prior probability* of a deformation. We can reduce the problem to minimizing external and internal energy [60], related respectively to likelihood and prior probability:

$$E_{ext} = -\log P(G|\delta_i, T_i) \quad (3.2.11)$$

$$E_{int} = -\log P(\delta_i|T_i) \quad (3.2.12)$$

This yields a total energy function that needs to be minimized being $E = E_{ext} + E_{int}$.

Likelihood

The likelihood is a measurement of the similarity between an underlying object in G and the deformed template T_i in the image [24].

A deformed object of interest is a path in graph G that contains all the template shape elements that are present in T_0 (and these elements are encountered in the same sequence).

A solution s is an ordered list of template elements $[t_0, \dots, t_k]$. Cases where $s = [t_0, \dots, t_k] \notin T_0$ are not taken into consideration as a valid solution, therefore

$$P(G|\delta_i, T_i) = \begin{cases} 0, & \text{if } s \notin T_0 \\ 1, & \text{otherwise} \end{cases} \quad (3.2.13)$$

For the zebrafish test case we assume that its basic structure should always be the same. Therefore we do not introduce any fuzzy classifier for the external energy and only consider the cases where $P(G|\delta_i, T_i) = 1$. Then energy function that needs to be minimized becomes:

$$E = E_{int} \text{ if } s \notin T_0 \quad (3.2.14)$$

Prior Probability

Prior probability [24] is model driven and is the probability of a deformed T_i matching the original T_0 . We provide a prior that is applicable to the zebrafish larva, being the main test case of this chapter, based on the following assumptions:

- The template, as described in Eq. 3.2.1 assumes a regular (healthy) zebrafish is straight. Bended instances are allowed, but the straight ones should be preferred.

- The template, as described in Eq. 3.2.1, is represented in such a way, that shorter instances could be (mis)interpreted as correct matches, while being a subset of the correct match. For example a zebrafish match with a short tail can be considered a solution, while the tail actually is longer. In order to prevent this we need to prioritize the solutions that represent longer shapes.

As a result of these assumptions we construct the following priors:

- We favor the less deformed shapes over the heavily deformed ones. For two sequential nodes t_a and t_{a+1} we check the angle $\theta_{a-1,a+1}$ between nodes as compared to their predecessor node t_{a-1} :

$$\theta_{a-1,a+1} = \arctan(c(t_{a-1})_x - c(t_a)_x, c(t_{a-1})_y - c(t_a)_y)(180/\pi) - \arctan(c(t_a)_x - c(t_{a+1})_x, c(t_a)_y - c(t_{a+1})_y)(180/\pi) \quad (3.2.15)$$

In order to consider the less deformed solutions we look at:

$$\theta_{largest} = \max(\theta_0 \dots \theta_{k-1}) \quad (3.2.16)$$

We prioritize the candidates with the lowest $\theta_{largest}$. This means the prior probability increases as $\theta_{largest}$ decreases.

- Additionally solutions that represent longer shapes need to be prioritized. Length of a solution l is represented as the sum of distances from anchor regions t_a to t_b mass centers:

$$l = \sum \overline{c(t_a)c(t_b)} \quad (3.2.17)$$

We prioritize the candidates with the highest l . Therefore prior probability increases when the solution path length increases.

To that end we set the internal energy to:

$$E_{int} = e^{-ml} \cdot e^{-k(1-\theta_{largest})} \quad (3.2.18)$$

with m, k normalizing constants. In Eq. 3.2.14 we state to consider only cases where likelihood equals 1 therefore $E = E_{int}$.

3.2.4 Finding an Optimal Solution

The Bayesian approach leads to a list L_{all} that contains all solutions in the dataset. The following rules are considered while looking for optimal solutions:

- Create a list L_{best} that will contain the optimal solutions
- Iteratively select the solution s with lowest E from L_{all} . To prevent cycles, solutions with multiple instances of the same node are not considered.
 - Save s to L_{best}
 - Remove s from L_{all}

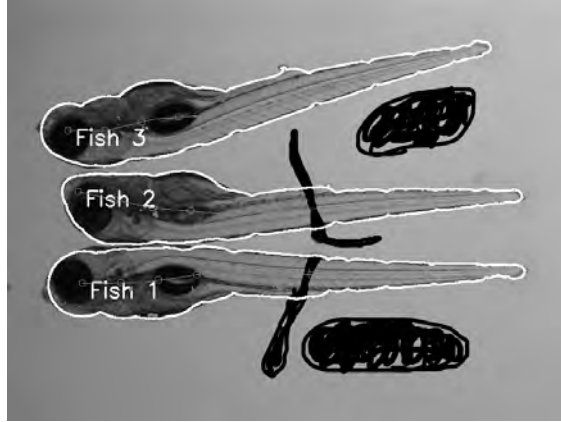


Figure 3.14: A visual representation of retrieved zebrafish larvae. Detected shapes are delineated with white.

- Remove all solutions that share at least one node with s from L_{all}
- Continue looking for solutions, while L_{all} is not empty (contains solutions)

The global shape of the objects can now be selected from the solutions in L_{best} . The shape is then retrieved by applying a dilation to the anchor regions.

The shape is not exact as it was an approximation derived from a distance transform and dilation. In order to accurately retrieve our objects we miss out on the small objects in the image at the locations where the opened regions are overlapping each other. These small objects appear because opening does preserve the main geometric features, but excludes features that are smaller than the structuring element radius. These small objects (or artifacts) do remain in the undefined space U . Therefore we add elements from U to the solution under evaluation. In order to know which elements to add we select all elements that are the neighbors (cf. Eq. 3.2.7) of the retrieved regions (step 5 in Figure 3.4).

In our example the optimal result with three best solutions is shown in Figure 3.14. Note that the algorithm correctly annotated the three zebrafish shapes, while the artifacts were not included.

3.3 Case Study 1: Zebrafish infection analysis

We have applied our approach for shape retrieval and annotation in a case study regarding the spread of induced bacterial infection within the zebrafish larvae. In Chapter 5 a more detailed description is given.

To better understand the infection progression large quantities of zebrafish larvae images needed are infected and after a few days of infection images are taken. All these images need to be analyzed. Until recently, these images were analyzed manually. Of

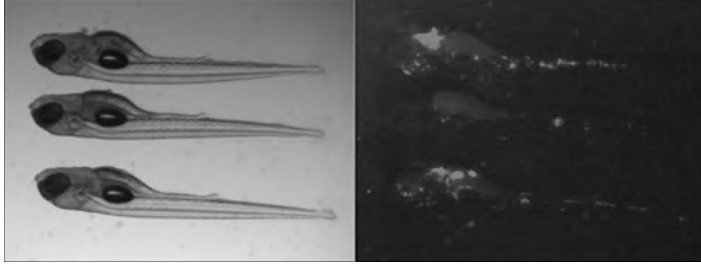


Figure 3.15: A brightfield image (in grayscale representation) and a matching fluorescent image (in grayscale representation). Images were taken with Leica DC500 microscope.

each location containing the zebrafish larvae two images were taken: one brightfield image in order to establish the location of the larvae, and one fluorescent image containing only location of the infection. In Figure 3.15 example of such an image pair is shown.

The analysis includes localization of the zebrafish shape in the brightfield image and quantification of the granuloma cluster size and spread from the fluorescent images. This enables the necessary HT automated approach.

For this case study we use a comparison of infection pattern of the wild-type of the *Mycobacterium marinum* (MM) that made the zebrafish sick with a mutant strain (714M) that induced less infection [42].

A brightfield image is used for finding and annotating the zebrafish shape. Additionally, locations of subareas can be estimated. We annotate the zebrafish larvae shape based on the prototype template representation as shown in Eq. 3.2.1. By using this template as prototype the algorithm estimates the location of the head, body and tail region of the zebrafish larvae.

3.3.1 Zebrafish Orientation

The template we have used for the zebrafish does not discriminate between the top and the bottom of the zebrafish. Therefore an additional step is needed to retrieve the orientation of the zebrafish larvae shape; this is necessary for a good assessment of the infection.

In order to be able to determine that the following approach is suggested. We localize all the convex deficiency locations. The bottom of a zebrafish shape will usually have larger convexity defects than the smooth top. If the largest convexity defect is then located below the shape path we assume the fish is on its belly, otherwise its on its back.

The annotation allows us to retrieve some other characteristic locations within a shape. For example the heart of the larva and the point at which usually injections are performed can be not automatically retrieved. This could be done by defining the locations of these points of interest in the prototype template. For example, the heart is located between the head and the body nodes at the bottom of a larva. The injection point is located

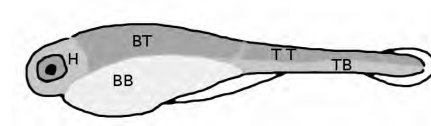


Figure 3.16: Graphical representation of the separation of a zebrafish into regions.

approximately halfway the tail area. An automatic annotation of these locations can then be performed as we know the locations of the head, body and tail regions and we have obtained the shape orientation from our analysis.

3.3.2 Zebrafish Region Annotation

Now for the assessment of the infection, head (H), body bottom (BB), body top (BT), tail bottom (TB) and tail top (TT) are defined. From the position assessment the orientation of the zebrafish could be easily retrieved. This made it possible to divide areas into the *top* and *bottom* part. Separation of these regions was based on [63].

The estimated location of these areas is shown in Figure 3.16. After annotation the found regions were masked with the fluorescent image of the same fish (cf. Chapter 4) in order to define infection location and amount.

3.3.3 Data

For this case study 189 zebrafish larval shapes were retrieved and analyzed. The images are the input for the analysis framework which consisted of two steps. First the zebrafish are localized by our algorithm and then used as a mask for the fluorescent images. Infection spread and size is saved to a comma separated file. In Figure 3.17 and Figure 3.18 an example of the result is shown. Based on this annotation the infection patterns were established and described in more detail in Chapter 4

3.3.4 Analysis and Results

In our case study we set out to analyze the relationship between mutant and wild-type (MM) in the amount of clusters, spatial distribution and the cluster size. The results presented here followed from an in depth statistical analysis that is described in Chapter 5.

In all regions of the zebrafish the wild-type gives a higher area of infection compared to the 714M; overall the infection area is 5 times larger. In the head region the same percentage of the granulomas in both MM and 714M are located. In the body bottom region a higher percentage of the granulomas of the MM then 714M are located. In the tail bottom region a higher percentage of the granulomas of the 714M then MM are located. A graphical representation of these results is shown in Figure 3.19. In all regions of the zebrafish the MM gives a higher amount of clusters then the 714M; overall the amount of clusters is 3 times higher. The spread of the clusters is not significantly



Figure 3.17: Output for a single image as created by the our algorithm. The found shapes are denoted with red line. The purple area indicates the presence of granuloma formation at the current location. Optimal graph G is printed within each shape. Blue lines represent node connections that were classified as tail area, green lines as body area and red as head. This image is created in an automated fashion.

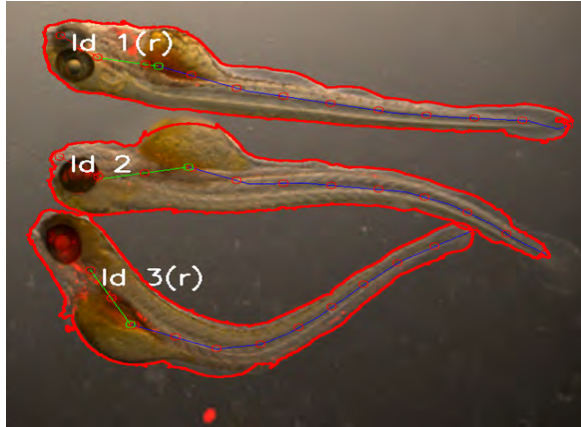


Figure 3.18: Output for a single image as created by our algorithm. The found shapes are denoted with red line. Note that the shapes in this example are heavily deformed, yet the shapes were found and annotated correctly. This image is created in an automated fashion.

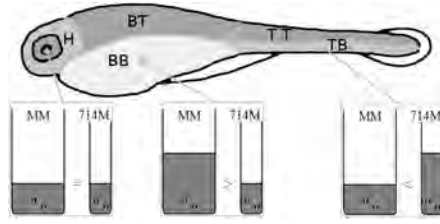


Figure 3.19: Graphical representation of the amount of infection (see Chapter 5 for more information on these infographics). While there is always more MM infection than 714M, the percentages of their presence in certain regions are different.

different for MM and 714M. In this manner we can derive statements on the behavior of mutant strains.

3.4 Case Study 2: Other Model Organisms

For an additional test of the algorithm we test the possibility to separate different shapes/organisms in the same image. An experiment was set up with two different shapes. We have collected images containing both *Xenopus laevis* and *Danio rerio*. In early stages (20-30) the *Xenopus* has approximately a length in the same order of magnitude as a zebrafish embryo in larval stage. However, the shape of the organisms is different: the tail of the *Xenopus* has the same width as its body, which is different from the zebrafish. In Figure 3.20 we present an image containing the zebrafish in larva stage and the *Xenopus* in stage 26.

We have used Figure 3.20 as input for our algorithm with a prototype template $T_{0_{zebrafish}}$ as presented earlier in this paper. As result the zebrafish larva was found while the *Xenopus* shape was rejected as a solution. The result is shown in Figure 3.21. As can be seen the non zebrafish shapes were not selected, while zebrafish shapes were annotated.

In order to be able to find the *Xenopus* we have adapted the prototype template to the following:

That is, the r of the tail of the *Xenopus* should in general be close or equal to its body thickness (biggest part of the tail should be or at least 70% of it):

$$T_{0_{xenoropus}} = \begin{cases} [t_0 t_1 t_2] & \text{sequence} \\ r_0 \geq 0, r_0 < r_1, r_2 > 0, r_2[\text{biggest}] > 0.7r_1 & \text{radius relationship} \\ r_{min} < r_i < r_{max} & \text{radius limits} \\ (l_0 + l_1) < 0.5 * (l_0 + l_1 + l_2) & \text{length relationship} \\ l_{min} < (l_0 + l_1 + l_2) < l_{max} & \text{length limits} \end{cases} \quad (3.4.1)$$



Figure 3.20: Brightfield image of a zebrafish in larval stage (on top of the image) as well as the *Xenopus* in stage 26. Image was provided by N.Bardine.



Figure 3.21: Results of applying or algorithm to an image of a zebrafish larva and the *Xenopus* larva with a prototype template $T_{0_{zebrafish}}$.



Figure 3.22: Results of applying or algorithm to an image of a zebrafish larva and the Xenopus larva with a prototype template $T_{0_{xenopus}}$.

Results of the segmentation are shown in Figure 3.22. As can be seen with this template the Xenopus shape is annotated, while the zebrafish shape was ignored.

To show that our algorithm is independent in scale and size and only shape based we also run the algorithm for an image that is containing the zebrafish larva and the Xenopus in a later stage of development (see Figure 3.23). In stage 45 Xenopus shape is very similar to the zebrafish and its shape has approximately the same characteristics. The difference, however is the size, as the Xenopus in this stage is several magnitudes bigger then the zebrafish.

Results of the segmentation are shown in Figure 3.24. As can be seen both the Xenopus as the zebrafish are annotated. This is due to the fact that their shapes are similar and thus both fit the prototype template of the zebrafish larva regardless of their size.

3.5 Conclusions

The algorithm is robust; it can deal with variation in position and orientation which makes it a perfect candidate to use for the complex shapes and scenes as apparent in life sciences: if no special instrumentation is used the location and orientation will not be known.

We have successfully used the algorithm in a case study for HT of automatic recognition of zebrafish larva, specifically for in depth analysis of granuloma cluster spread. From statistical analysis of the output data we gained insight on the distribution pattern



Figure 3.23: Brightfield image of a zebrafish in larval stage (on top of the image) as well as the *Xenopus* in stage 45. Image was provided by N.Bardine.



Figure 3.24: Results of applying or algorithm to an image of a zebrafish larva and the *Xenopus* larva with a prototype template $T_{0_{zebrafish}}$.

3 Anchor Region Based Pattern Recognition in Image Space

of the wild-type strain and how it was different from mutant 714 strain.

4 Software Development and Evaluation for High Throughput Zebrafish Analysis

Under preparation for publication and partially based on:

E.J. Stoop, T. Schipper, Huber S.K. Rosendahl, A.E. Nezhinsky, F.J. Verbeek, S.S. Gurucha, G.S. Besra, C.M. Vandenbroucke-Grauls, W. Bitter & A.M. van der Sar: Zebrafish embryo screen for mycobacterial genes involved in the initiation of granuloma formation reveals a newly identified ESX-1 component, Disease Model Mechanisms: 526–536 (2011)

A.E. Nezhinsky & F.J. Verbeek: Pattern recognition for high throughput zebrafish imaging using genetic algorithm optimization. In: 5th IAPR Conference on Pattern Recognition in BioInformatics (PRIB 2010), Lecture Notes in Bioinformatics 6282: 302–312, Springer (2010)

4.1 Introduction

In the previous chapters we have elaborated on the segmentation of instances in an image. From the introduction it was clear that the field of application is High Throughput imaging for zebrafish. In the previous chapters the zebrafish has already been used as a test object; in this chapter we further elaborate on the successful segmentation algorithms to embed these in software that can be used in a High-Throughput context to derive measurements.

4.2 High Throughput Experiments

In recent research the *Mycobacterium marinum* (MM) infection model in the zebrafish larvae is exploited in order to gain more insight on the development and mechanisms behind granuloma formation (cf. Chapter 5) in zebrafish as part of tuberculosis research.

We present a High Throughput (HT) screening setup that is used to obtain insight into spread of bacteria by investigating different conditions, e.g., different mutants of MM (cf. Chapter 5) or different infection volumes. Features are extracted to compare a control condition to an experimental condition (wild-type versus mutant).

Screening was performed *in vivo* on larvae that were anesthetized in egg water with 0.02% (w/v) ethyl-3-aminobenzoate methanesulfonate salt and images were taken with a CCD camera mounted on a fluorescence stereo microscope. Until recently the analysis of the images was done manually, which made true HT screening impossible. Therefore suitable HT analysis software is needed.

4.2.1 Image Acquisition

For the purpose of localization and quantification of the infection within the larvae for each subject instance a multimodal 2 channel image is acquired: one channel is containing a brightfield and one channel is containing a fluorescent image. Within each instance of the screen up to 3 zebrafish larvae were placed in a single well. Intentionally, the orientation of the larvae is head-left; however deviations from the preferred orientation do frequently occur. Both brightfield and fluorescent images were acquired using the Leica MZ16FA light microscope and captured with a Leica DC500 (DFC420C) CCD camera 24 bit color with an image size of 2592×1944 pixels (about 5 Mpixels).

Besides the larvae no other objects are present in the brightfield image with the exception of incidental noise and debris. An example of a brightfield and a matching fluorescent image is shown in Figure 3.15. One experiment contains multiple subject instances that are created with the same microscope settings. For the automation, the brightfield images are used in order to determine the position and location of the larvae.

The fluorescent images contain the signal from the fluorescent agent that is present in the larvae, more specifically in the bacteria. The fluorescent agent used is *DsRed*, which indicates the presence of bacteria and is visualized in the microscope from the *red* channel with quite strong signal (good signal/noise ratio). Besides infection present, the color channels can contain values above 0, possibly the result of auto-fluorescence

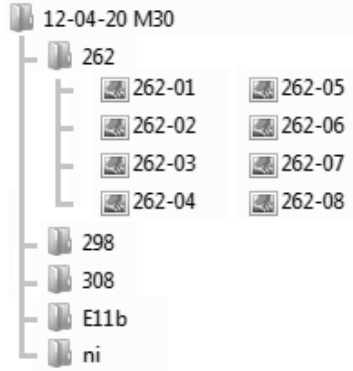


Figure 4.1: An example of data organizations as currently done for an array of experiments.

of components in the objects or medium [40]; this should therefore be considered as background noise. Fluorescent images are used to derive the amount and the location of infection per object.

The analysis is done in time, all the images that are taken during the experiments are stored for analysis.

4.2.2 Data Storage

Images from the experiments are stored on a hard disk in an organized semi-structured manner. This organization is file system based. Images (pairs of brightfield and fluorescent images) resulting from the same experiment are grouped together in folders that are named after the experiments. Folders of experiments are then grouped in a parent folder which is a group of experiments that are done at a specific date or in specific conditions. The organization is *as is* taken from the lab.

As an example consider Figure 4.1. Note, that in this example the directory name contains the the date of the experiment, but this is not regular practice and therefore can not be considered as a fixed rule. In this example all experiments that are performed on *20 April 2012* are grouped together in the folder *12-04-20 M30* that contains sub folders of experiments for mutant strains that are numbered the same as the folder names: *262*, *298* and *308*. The *E11b* folder contains the experiments that were done on zebrafish infected with the wild-type strains, i.e., the control group. The *ni* folder contains images of zebrafish that are not infected by any bacteria. So, each of the folders (in the example folder of mutant *262* is unfolded) contains images that were taken during the experiments.

The images are stored either as 24 bit color lossless (*png* or *tif/tiff*) encoding. An example of the content of such a directory is shown in Figure 4.2.

The type of image (either brightfield or fluorescent) can not be retrieved from the filename. However, the relation between a fluorescent image and a brightfield that



Figure 4.2: Containment of a windows directory that represents the output for a single experiment number 262.

contains the same object is encoded in the filename. The numerical value of the last 2 digits of a fluorescent image is always one higher as compared to the brightfield image from the same experiment. This is the only filename encoding rule that is applied by different users. Note, that even and odd number are not fixed for brightfield/fluorescent images and they can not be identified as such. In Figure 4.2 we therefore can conclude that image *262-04.tif* is taken of same specimen as *262-03.tif*. However to confirm the connection a preprocessing step still is needed in order to determine which images are fluorescent, for example to prevent the pairing of files *262-03.tif* and *262-02.tif* to each other.

4.2.3 User Analysis

We identify the following characteristics of the users that will be working with the software:

- The users are researchers in the life sciences.
- They work on a PC for analysis and texting applications.
- They are used to doing High Throughput analysis.
- They are used to storing the data in a semi-structured way.

4.3 Our Approach

We have worked out the idea to automate the workflow as much as possible and present the user with an interface that to a certain extent matches users workflow. First we will describe the workflow as it exists now. Then we will describe the components out of which the software is built up and subsequently, we will discuss each of the components in more detail. The prototype name of the software is *ZFA* (Zebrafish Analysis).

4.3.1 Workflow

A starting point for the workflow was the data storage as described in 4.2.2. An overview of the workflow is presented in Figure 4.3.

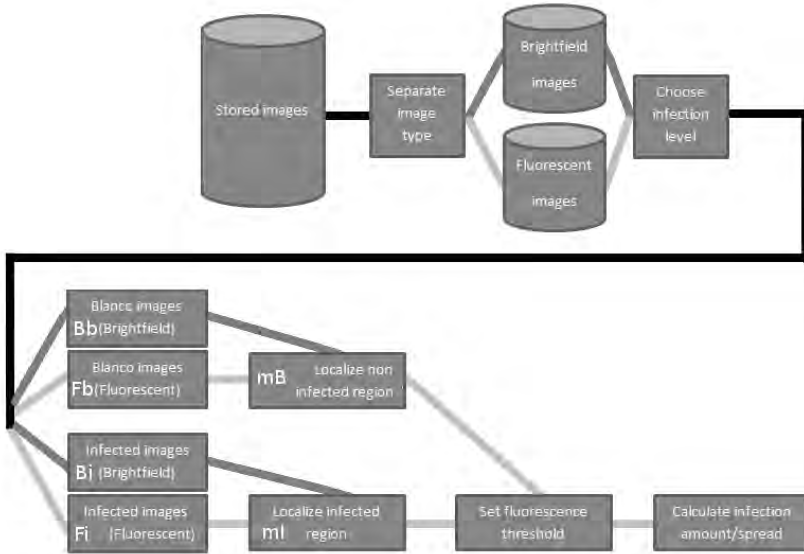


Figure 4.3: The original workflow of moving from stored images to measurements.

The images that are stored are either fluorescent or brightfield and are located in the same directory. The first step in the parsing is a labeling of the image content (brightfield or fluorescent).

Each experiment (contents of a single directory) is considered to be *blanco* or an *infected* image group. *Blanco* images are images that contain embryos that are not infected. Each blanco image pair consists of a brightfield and a fluorescent image. Consequently, from these images, the average fluorescent background value can be established per experimental set. *Infected* images are images that contain embryos that are infected. Each infected image pair consists of a brightfield and a fluorescent image. The user determines which images should be treated as infected and which as blanco. The user can retrieve this knowledge from the directory names (cf. Section 4.2.2).

Each brightfield blanco image Bb is used to get information on the area that is occupied by the not infected larvae. This area is then used as a mask for the fluorescent blanco image Fb and produces masked result mB .

Each brightfield infected image Bi is used to determine the area that is occupied by the infected larvae. This area is then used as a mask for the fluorescent infected image Fi and produces a masked result mI .

The average intensity of area mB is used as a threshold for mI . This process is described in more detail in Section 4.3.2.

The amount of infection and its position is then analyzed in the thresholded mI . The

output of an experiment should be represented as measurements of the infection within each zebrafish embryo.

4.3.2 Components

From the workflow several processes emerge and these processes are included in the software as components, i.e., the *preprocessing*, the *interface*, the *segmentation* and the *data processing* components. Next, each of the components is described in more detail.

We decided to build software along principles of evolutionary prototyping.

Physical Implementation

The software is written in the C++ programming language in combination with the OpenCV library and the interface was created using QtDesigner. The software shell is chained in a pipeline as two executables. One contains the interface and the pre-processing component. The other one contains the pattern recognition and the data processing component. The first program automatically feeds the data to the second one whenever the user runs an analysis.

Upon execution of the segmentation algorithm runs the interface is not locked, as an advantage this means it continues to be interactive: the user can select and inspect images, while the segmentation is ongoing.

Image Preprocessing

Both the fluorescent and the brightfield images are located in the same directory. A pre-processing step is needed to separate them into the two groups. Therefore an automated approach is developed to accomplish this separation.

In order to find a way to separate the images we have made the following assessment of 3000 fluorescent images. First we examine the average intensity of the images. The histogram of the images was analyzed and an average intensity was calculated. An example histogram of a fluorescent image is shown in Figure 4.4. The average intensity $\sum f(x, y) / n \times m$ (sum of pixel intensities divided by the total amount of pixels) always remained under one third of total intensity spectrum. Therefore, if the average grayscale value is lower then $MaxIntensity/3$ the image is considered to be a fluorescent image.

The filename of the fluorescent images is always linked to the matching brightfield by a number in the filename and therefore the corresponding brightfield image is easily retrieved.

User Interface

The interface component directs the user to select the working directories for both input and output and subsequently presents the user with the previews of the images in selected directories and allows to start the analysis and adjust different settings. Throughout the development multiple versions of the interface have been proposed and evaluated with the users. Version *#jan2013* of the interface is shown in Figure 4.5. We will describe the

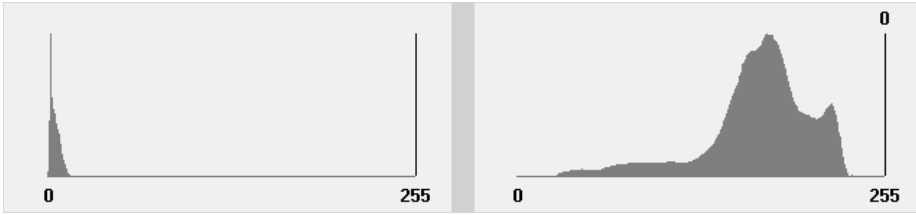


Figure 4.4: Left: histogram of a sample fluorescent image. As can be seen from the histogram all intensity values are located close to 0. Right: histogram of a sample brightfield image (converted to grayscale). As can be seen the values are more scattered through the entire interval.

sub components in more detail. For each description the corresponding sub component number is given in Figure 4.5:

1. A drop down *File* menu. Options of the menu are: *Open*, *Run*, *Set output folder* and *Exit*.
 - *Open*: if option is selected an open directory dialog appears. The user selects directories that contain the *blanco* images and the directories that need to be analyzed. The selected data is then automatically separated into brightfield and fluorescent images and a list of image file names is presented within the interface (sub component 8). For each directory the user can choose if it either is a blanco, should be analyzed or both.
 - *Run*: the images in the selected directories are transferred to the segmentation program and subsequently image processing is performed. The result is written to a *csv* file.
 - *Set output folder*: sets the directory where the csv file and the output images will be written to. Last working directory is always kept if the program is restarted.
 - *Exit*: exists the interface.
2. The directory that contains the output csv file and the output images (its value can be modified from the File menu).
3. The output filename can be directly changed from the main interface window. Its value is saved upon exiting the entire program and is reloaded at start up.
4. The ongoing current task of the *ZFA* is shown here. From this task an indication is presented on the current image analysis task — including the current image.
5. Progress bar. At the onset of the analysis the bar is empty. It progresses while the images are analyzed, with 100% of images analyzed represented by a full bar.



Figure 4.5: Final version of the interface component.

6. An option to additionally look at specific locations: here the *heart* and the *injection point* area. If selected, *ZFA* will annotate these areas and do additional measurements at those locations. More information of the location of these points can be found in Chapter 5.
7. (and 9.) In this window a point-click on a filename activates a preview of the fluorescent image and the matching brightfield image. Note, that the colors for display are complementary to the original ones for visibility purposes.
8. The image preview window shows a list of images that are contained in the selected directory. The user selects, by ticking a checkbox, which of the loaded directories should be analyzed and starts the image analysis pipeline. The user can make a selection between the type of images in the selected directory. If the user selects *Blanco* the images will be treated as blanco images (see Section 4.3.2). If *Analyze* is selected the images will be analyzed and the results will be written. If no options are selected (no checkboxes are selected), then the directory is not considered at all.

Image Analysis

In order to consider only the fluorescent signal that is resulting from infection the background is removed from the fluorescent images. We consider only the red channel of the fluorescent image as that is the only channel that contains meaningful data. Then, all

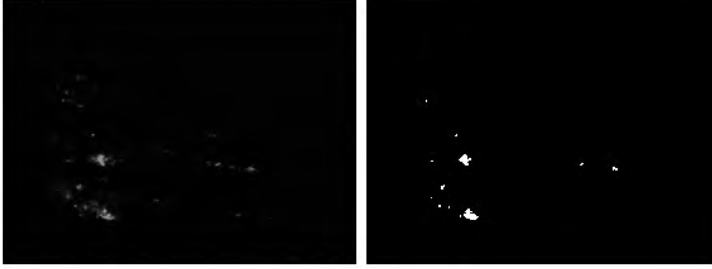


Figure 4.6: Within the mask a threshold value is set (left: fluorescent image, right: binary image as a result of thresholded left image).

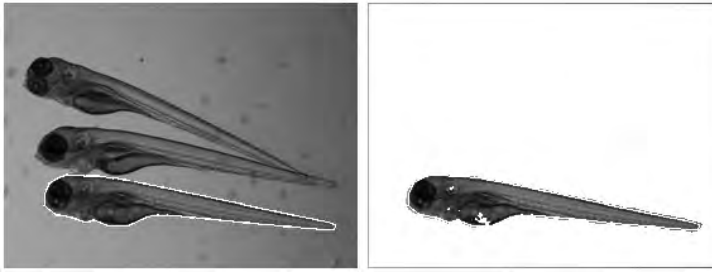


Figure 4.7: Each zebrafish embryo mask is multiplied with the corresponding thresholded fluorescent image (resulting from Figure 4.6) .

signal lower than a certain threshold value b is set to 0. The value of b is retrieved from the blanco images. We assume blanco images contain no infection and all the signal in it is considered background; we take the value of the brightest pixel of the blanco and treat it as b . If more then one blanco image is present in the set an average value of the brightest pixel of each blanco image is used instead.

All blanco images should be from the same set as the images we are analyzing and the amount of selected blanco images should be > 0 (the user must select at least one directory containing *blanco* images), otherwise b can not be computed. The resulting thresholded image is shown in Figure 4.6.

The brightfield images are used to provide the mask of the shape under investigation. The shape of the zebrafish embryos is localized by a model based segmentation algorithm as described in Chapter 3. Each corresponding mask is then used for the measurements in the fluorescent image. Besides only localizing the shape we also label different areas as being specific regions of interest within the region mask (cf. Chapter 3).

For the fluorescent images only infection within the mask is used for the resulting feature extraction. An example of this process is shown in Figure 4.7. In this manner multiple instances of a shape in one image are processed one by one.

Data Processing

After the retrieval of the infection in each zebrafish different features of the infection are analyzed in more detail. An infection area is characterized by clusters of immune cells (aka granuloma, cf. [54]). The measurements that can be done include measurements of the shape and the amount of the cluster areas. This includes counting the amount of clusters, their size, their spatial distribution within specific regions of the specimen, the texture and the integrated intensity of each cluster. The extraction of the features is followed by a statistical analysis of the features in context of the experiment. This is elaborated in more detail in Chapter 5. After data is processed, this component writes the data to a *csv* file so that further statistical analysis and classification can be performed to the data by the user and other software. The process and the results of the analysis are described in more detail in Chapter 5. Next to the output *csv* file also output images are written. These images are the annotated representations of the extracted masks and include the granuloma presence locations. These images can be used for visual inspection of the results.

4.4 Evaluations and Results

In the context of the development of the software there are two kinds of results that should be evaluated. The first are the results of user evaluations of the interface and workflow. The second are the segmentation algorithm evaluations. That is referring to the measurement of performance and accuracy.

4.4.1 User Evaluations

The prototyping is combined with a series of evaluations. Requirements for *ZFA* were gathered through an elicitation process (derived from the workflow analysis). Users¹ were asked to evaluate the product in different development stages of development. Evaluations were performed on a functional prototype. Throughout different versions feature requests were added at different stages of the software but the core process was functional in all versions. For the evaluation the application prototype was installed on the PC of the user.

Before requirements can be analyzed, modeled, or specified they must be gathered through an elicitation process. During the elicitation phase we have retrieved the requirements from studying the workflow. We have used the evolutionary prototyping method with the user feedback regarding the interface design. The major requirements were known and the way they should be implemented in the interface needed to be iteratively adapted. The solutions were tested and adapted until interface interactions were expressed as being satisfactory. After receiving an improved version the users were interviewed about the usability of the system.

¹Biologists at the Vrije Universiteit (Amsterdam) and IBL (Leiden).

4.4.2 Results

The option for an output folder selection was introduced because the users desired to be able to store multiple output *csv* files in a single, yet user chosen directory. Directory location is shown directly in the interface, yet to change it the user must go through the *File* menu. This option is added, because the output directory is usually not changed by the users. However the option to change the *csv* output filename was integrated directly in the interface, since users regularly tend to change this.

The *current task* window for showing the current task was introduced to provide a better feedback to the user at the moment a problem occurs. This can be the case when a certain image has a problem and can not be segmented for some reason. The task window shows the filename of the last image that was analyzed. The user can inspect that particular image to check if the problem can be solved or perhaps exclude that particular image from the analysis.

The progress bar was introduced as the users wished to be informed on the time they need to wait for the program to finish.

Most comments of the users related to the organization of the way a directory structure containing the images was shown. In the early versions, directories that contained the images showed previews of the fluorescent images. This was done as the previews could indicate a rough measure of infection to the user. At first, the previews were shown in the original (red over black) colors. However, the users commented that it was hard to distinguish a weak red signal on a predominantly low intensity (seen as black color) image. For that reason the image colors in the previews have been inverted in the RGB model as shown in Figure 4.8. The inversion of the RGB channel image was done according to the logical complement using the following scheme: R(red) was converted to GB(green,blue); G was converted to RB; B was converted to RG. Since the original image contained a signal only in the red color resulting images contain only a GB (GB is also known as *cyan*) signal on a white background.

In an evaluation the users reported slow loading times of the interface. This especially occurred when the directories for the analysis contained a larger number of images and was due to preloading and processing (inverting the colors) of every image. In order to shorten the loading time in a newer version instead of a preview of all images, the previews are only displayed on a filename select from a list in the GUI. The resulting interface is shown in Figure 4.9.

4.4.3 Performance Evaluations and Results

It is important that the users can easily operate the software, but it is evenly important that the software provides correct measurement results. Therefore we have done performance and accuracy tests of the pattern recognition step. The interface, as described in this chapter, was also coupled to different implementations of the segmentation algorithm. The three different approaches for segmentation are described in more detail in Chapters 2 (Approach 1 and 2) and 3 (Approach 3).

To compare the performance of the approaches we consider measurements that are

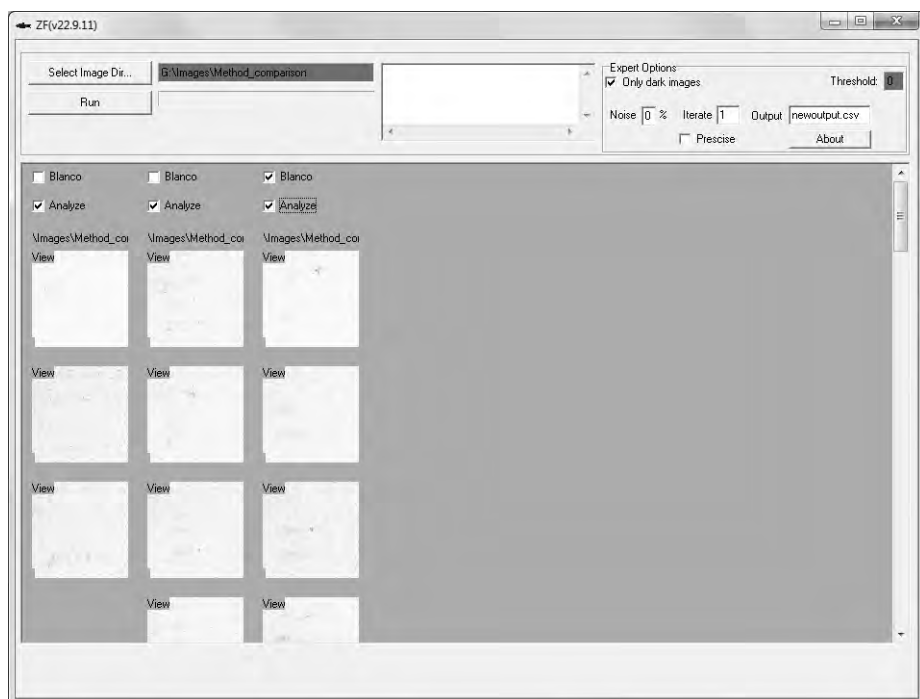


Figure 4.8: Interactive layout. Image previews are shown in complementary colors with a white background.

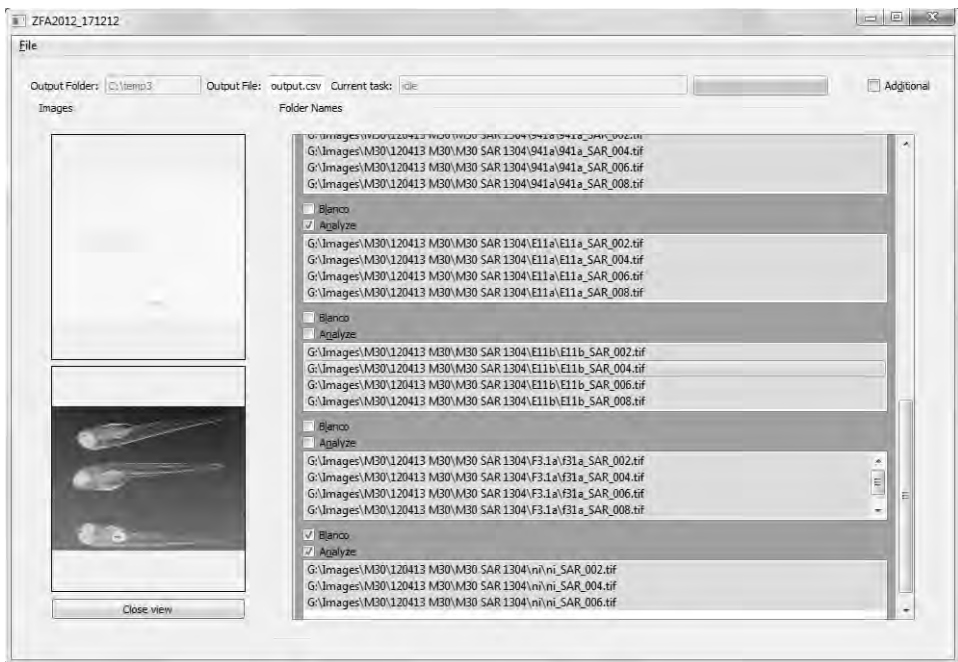


Figure 4.9: Interactive layout. Image previews are not shown. Only when an image is selected from the filename list the preview appears on the left.

Segmentation technique	Approach 1	Approach 2	Approach 3
true positive (tp)	100	115	120
false positive (fp)	12	5	0
false negative (fn)	23	8	3
Precision	0.89	0.96	1.00
Recall	0.81	0.93	0.98
F	0.95	0.98	0.99

Table 4.1: Comparison of the proposed approaches with respect to zebrafish embryo retrieval.

common for such pattern recognition problems, namely *precision* and *recall* (also known as *sensitivity*). Precision is a measurement that represents the fraction of retrieved shapes that are relevant to the objects under study. Recall is the fraction of relevant objects that are retrieved. High precision yields a low percentage of false positives (*fp*), while high recall yields a low percentage of false negatives (*fn*). This is given by:

$$precision = \frac{tp}{tp + fp} \quad (4.4.1)$$

$$recall = \frac{tp}{tp + fn} \quad (4.4.2)$$

In order to combine the two measurements the *F-score* is used: $F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. We compare the values of *precision*, *recall* and *F* for the three approaches in order to see improvement in implementation. In Table 4.1 we present an overview for the measurements for the three approaches and data it was tested on. The same test set was used for the test of all the three approaches. We compiled the dataset from different experiments with different background intensities. The dataset consisted of 46 images containing 3 (36 images), 2 (6 images) or 1 zebrafish shape (3 images) per image. So in total we tested with 123 zebrafish shapes. Of these images we labeled 25 as *hard*, since in those cases the zebrafish shapes were touching, not placed in a lateral view or showed severe deformations (e.g., bent tail). We assume a true positive has occurred if the retrieved shape when analyzed by a human observer can be confirmed as zebrafish. We assume a false positive occurs when a shape is found, yet it is incorrect (we establish this happens when more then 20% of the shape is outside of the real zebrafish shape, or at least 20% of the fish is not included). A false negative occurs when a shape is in the image but it is not annotated as such.

The first approach could not retrieve a lot of zebrafish shapes and it introduces a lot of false positives when only a part of the shape was retrieved. This happened mostly in the cases where the zebrafish shapes were close to each other or the tail was heavily deformed. All shapes were retrieved roughly as an approximation.

The second approach performed well, but still some shapes were not or not correctly retrieved. In cases of false positives, this happened when the zebrafish tails were heavily

deformed and the algorithm failed to detect the tail. The shapes were retrieved more accurate than by the use of the first approach, however the shapes were still not smooth. Time needed to do analysis on all the images was approximately the same as with the first approach.

The third approach performed very well on the dataset. The only three shapes that were not found by the third approach were all located in the same image. Its analysis failed because the preprocessing step (mean-shift segmentation) could not separate the foreground from the background due to low intensity difference. Failing to retrieve these three individuals thus had nothing to do with the actual pattern recognition step. A smooth shape was retrieved for every image. Time needed to do analysis on all the images was approximately 4 times shorter than with the first and the second approach.

Additional elaboration of results and comparisons can be found in Chapter 6.

From this it is clear that Approach 3 gives the best performance and is used in the latest version of ZFA software.

4.5 Conclusions and Discussion

An evolutionary prototyping procedure with evaluations helped to improve *ZFA* on the level of (G)UI as well as performance and features. Also the segmentation algorithm yields good results and can be used successfully in a High Throughput approach.

ZFA is successful at recognizing the zebrafish larva shape and analyzing the infection within the larvae. *ZFA* is robust as it can be adapted to recognizing other shapes by simply changing the template; therefore it can be suitable for other organisms and especially in HT.

In the current version the measurements that are performed on the infection are: localization, infection cluster size, infection cluster amount and average infection cluster intensity. More texture measurements are to be performed such as shape descriptors and in depth intensity analysis.

For now the directory structure is used to select image folders containing the data that needs to be analyzed. This makes the software easy to use in a different laboratory without the need for a special database. However future work will include an online database repository where uploading the images through a web based interface becomes part of the process. This will make local installation of the software obsolete. Currently the measurements are stored in an output file. The file is in a comma separated format. This is a feasible approach in a laboratory environment where the users are using different files for different experiments. However when the amount of images grows the output should be stored in a relational SQL database for easier indexation. The data can then be organized based on keywords such as experiment type, experiment date, mutant number and the id of the biologist that took the images. Links between experiments can then be retrieved based on SQL queries.

ZFA has proven its value as a software solution for the analysis in a HT screening pipeline.

5 Data and Pattern Analysis in Infection Studies in Zebrafish

Based on:

A. Nezhinsky, E.J. Stoop, A.M. van der Sar & F.J. Verbeek: Numerical Analysis of Image Based High Througput Zebrafish Infection Screens: /Matching Meaning with Data/. In: Bioinformatics 2012, Proceedings of the Int. Conf. on Bioinformatics Models, Methods and Algorithms: 257–262 (2012)

and

A. Nezhinsky, E.J. Stoop, A.A. Vasylevska, A.M. van der Sar & F.J. Verbeek: Spatial Analysis of Bacterial Infection Patterns in Zebrafish. In: Proceedings 21th Annual Belgian-Dutch Conference on Machine Learning (2012)

and

A. Nezhinsky, A.M. van der Sar & F.J. Verbeek: In Depth Analysis of High Throughput Zebrafish Infection Screens. In preparation (2013)

5.1 Introduction

Tuberculosis is a serious disease and a significant part of the world population is infected. Unfortunately, effective treatment is still difficult due to bacteria resistance. In order to elucidate which genes are responsible for infection, the behavior of the tuberculosis bacteria *Mycobacterium tuberculosis* needs to be analyzed. In our study this behavior is modeled by a close relative — the *Mycobacterium marinum* (*MM*). The *MM* is hosted in cold blooded animals. For our study the zebrafish is used as a host. Zebrafish makes a good model for analysis as its immune system is in many ways comparable to human. The zebrafish larvae can be obtained in large numbers and studied by HT imaging (cf. Chapters 2, 3).

Infection of the zebrafish with *MM* is characterized by the presence of granulomas. Granulomas are clusters of immune cells and bacteria indicating infection. They can be visualized with fluorescent agents.

In order to determine which genes of *MM* are involved in formation of granulomas we obtained a dataset of 1000 random mutants of the *MM* bacteria and screened for those mutants that were not able to efficiently infect zebrafish larvae [54]. In this manner 30 mutants have been identified that were unable to infect larvae (cf. [55]).

In order to gain more insight in the progression of *MM* infection it is required to analyze infection spread in the host, c.q. the zebrafish, over a certain period of time. This requires the following questions to be answered:

- (1) Is there a pattern in the organization of granuloma clusters in certain tissues?
- (2) Does appearance differ for certain bacterial mutants?

This analysis is accomplished with the HT imaging as described in Chapters 4. For each zebrafish a brightfield and fluorescence microscopy image is acquired. The analysis included localization of the zebrafish shape and qualitative estimation of the granuloma cluster size and spread. Consequently, no quantitative data could be retrieved. The results are presented by means of different types of custom designed infographics.

We have designed and implemented an automated framework for shape retrieval and cluster analysis [43]. This framework has been applied to large scale HT applications [54].

The basis of the software framework is an algorithm for shape retrieval to automatically find the zebrafish shape(s) in the image. The algorithm uses deformable template matching [23] and labels the regions for further analysis. This approach made it possible to analyze the infection amount per fish in an automated fashion (cf. Chapter 4).

Proof of Principle: Wild-type versus Mutant 714

As a proof of principle, a study for the detailed analysis was performed to find a strategy for analysis. As an initial test case mutant 714 was chosen; this is one of the 30 mutants which is not successful in infecting the fish. We are focusing on the question: can the size of the granuloma clusters be analyzed from the infected fish? In addition, we compare similarity in larvae infected with the wild-type *MM* and mutant 714.



Figure 5.1: A brightfield image may contain up to 3 shapes of the zebrafish larva.

In depth Analysis: Wild-type versus a List of Mutants

After the proof of principle from the first assessment we apply our algorithm for in depth analysis of a larger set of mutants. We compare the results of infection to the readout for the wild-type. The following mutants indicated by a number were part of our analysis: 262, 308, 414, 415, 421, 423, 431, 730, 748, 801, 817, 885, 941 and 943. In order to process this amount of data an objective test was needed.

Microscopy

Images were taken in batches of 30 wells. Each batch contained sibling zebrafish larvae. Images were acquired with a Leica DC500 microscope. The general layout of an experiment takes 3 zebrafish larvae per well. As a result a single image can contain up to 3 individuals.

For each well, both brightfield and fluorescent images were acquired. The brightfield image contains the zebrafish shape, while the fluorescent image contains the signal at granuloma locations. An example of such images for a single well is presented in Figure 5.1 and 5.2 (cf. Chapters 2, 3). As can be seen the fluorescent images have very low and different intensity values. By inspecting the images, it is not difficult to conclude that consistent manual analysis therefore will be very difficult and imprecise.

5.2 Analysis of High Throughput Zebrafish Infections Screens Based on Approach 2

The images are the input for the analysis framework *ZFA* (cf. Chapter 4). As algorithm for the pattern recognition we used Approach 2 (cf. Section 2.3). First, the zebrafish are localized and the result is used as a mask for the fluorescent image. Within the mask

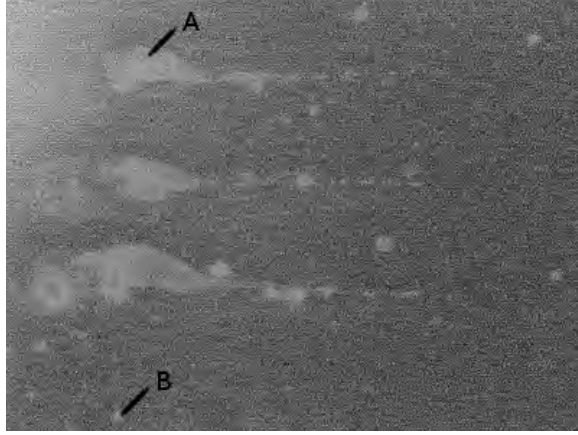


Figure 5.2: A fluorescent image containing the signal of granuloma spread. For visualization purposes the contrast is enhanced. The image contains specific (A) and non specific (B) staining as well as systemic noise.

a threshold value is determined. Finally the data is analyzed and written to a comma separated file.

The data set we have used for the analysis consisted of 189 infected zebrafish larvae. The larvae were divided into 3 groups: not infected larvae *NI* (5), infected with *Mycobacterium marinum* wild-type *MM* (67) and those infected with the 714 mutant *714M* (117).

For the infection approximately the same amount of bacteria was used; the volume was plated on 7H10 plates. At injection the zebrafish were 6 days old. The infection has progressed for 5 days, after this the imaging was performed.

5.2.1 Brightfield Imaging: Shape Localization and Annotation

The ZFA determines a region of interest (ROI) and provides annotation of the relevant areas. A deformable template is used for the retrieval of the zebrafish shapes. In order to be able to detect different regions of the fish, the template was divided into 11 regions (or *slices* in Section 2.3) counting from head to tail (numbered from 0 to 10). We have chosen for division into 11 parts, as it is empirically established that this amount of parts allows best for annotating the shape as well as doing spatial analysis. In Figure 5.3 the graphical representation of a prototype template used for our experiments is shown. Enumerations (i) with values 0 and 1 are considered as the head regions, 2 till 4 as the trunk regions and the remaining parts as the tail regions. This division of the larva in regions has been successfully used in other applications [63]. The injection point for infection is located at approximately region 5 [11] and this can be used for the analysis.

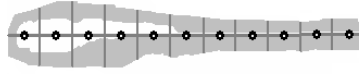


Figure 5.3: Graphical representation of the prototype template used for our experiments. The vertical bars divide the zebrafish into regions. The template is created from averaging a test set of 20 training shapes

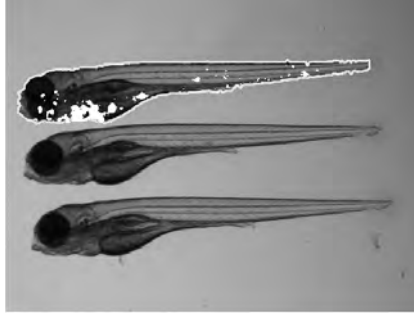


Figure 5.4: Graphical output of the framework overlaid on the original image for the top fish in the image. The light gray line denotes the shape mask contour and the white regions indicate the presence of granuloma formation. This image is created in an automated fashion.

5.2.2 Fluorescent Imaging: Analysis

For the actual measurement of clusters fluorescent images were used and related to the mask size, obtained from the brightfield images. The NI group is expected to have no infection at all and thus the level of their maximal fluorescent signal is considered as noise level n . No infection / granuloma formation is present in NI, therefore this group was only used to obtain reference for a noise level. In the other groups all signal below n is considered noise, while all signal above n represents granuloma presence. This signal is analyzed per fish and written to a *csv* file as shown in Table 5.1. The notions will be explained in the sequel.

5.2.3 Output and Dataset Creation

Output of the analysis software is created in the form of an overlay image containing the detected zebrafish and the infection as shown in Figure 5.4, together with a *csv* file (cf. Chapter 4).

Clusters of bacteria are labeled and for each of the clusters the surface area is determined. The infection is present in clusters:

Definition 1. *ClusterCount* is the amount of clusters present throughout the entire zebrafish larva, denoted by CC . A single cluster is defined as a connected collection of

Field name	Explanation
TotalArea	Total shape area
ClusterCount CC	Amount of clusters in the larva
ClusterSize CS	Area covered by all clusters
ClusterCountAt[i] $CC[i]$	Like CC but for a single template region
ClusterSize[i] $CS[i]$	Like CS but for a single template region

Table 5.1: Fields contained in the output csv per larva, i to the template region number (0 to 10) as described in Section 5.2.1.

pixels of which every pixel is covering an area that is classified as infection. The image pixels are classified as *infection* in an automated fashion as described in Section 4.3.2. Background pixels are automatically excluded from the analysis.

The total area of the spread is the sum over all clusters:

Definition 2. *ClusterSize* is the total area covered by the infection throughout the entire zebrafish larva, denoted by CS . Area is expressed in units that represent the amount of pixels that are classified as *infection*. This classification was performed in an automated fashion as described in Section 4.3.2. Background pixels are automatically excluded from the analysis.

From the template we define 11 regions and a cluster is always assigned to the region center with closest geometrical distance [43].

5.2.4 Analysis and Results

In this section an analysis of the results is presented. First the distribution of the clusters is discussed and second a relation to the amount of infection is derived from the data.

Distribution of Cluster Amount

In our study we set out to analyze the relationship between mutant and wild-type with respect to the amount of clusters and spatial distribution. To that end we compare the average number of clusters (feature Cluster Count, $CC[i]$) between MM and $714M$ (cf. Figure 5.5).

Definition 3. *Cluster Count[region]* is the amount of clusters present in a certain region (or slice), denoted by $CC[i]$. A cluster is defined as a connected collection of pixels of which every pixel is covering an area that is classified as infection.

The image pixels are classified as *infection* in an automated fashion as described in Section 4.3.2. Background pixels are automatically excluded from the analysis.

From the initial measurements this distribution does not seem to be conclusive. This is due to the fact that the mean was taken from a dataset with a certain scatter. This scatter is shown in Figure 5.6.

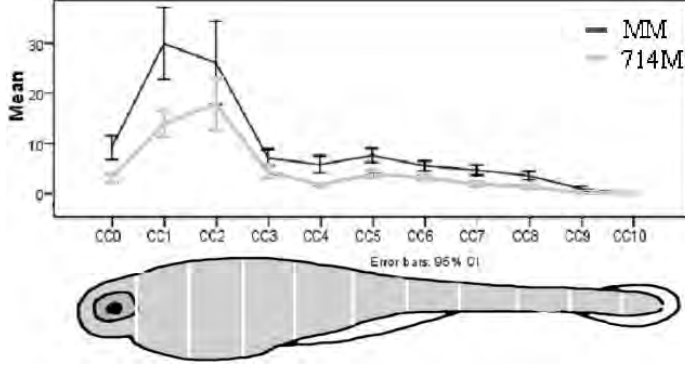


Figure 5.5: Spatial comparison of the average amount of clusters for MM and 714M in relation to the zebrafish template with a 95% confidence interval.



Figure 5.6: Scatter plot of the amount of clusters in each case of the test set used.

We are interested in the distribution of the granuloma clusters and in order to analyze the different batches in the same way we normalize the $CC[\#]$ over the total CC . The normalization is performed for each individual case and subsequently the mean is calculated. In Figure 5.7 the results are depicted.

Definition 4. *Normalized Cluster Count [region]* is the normalized $CC[i]$ feature, denoted by $CCn[i]$. Normalization is done over the total amount of clusters throughout the entire zebrafish larva region: $CCn[i] = CC[i]/CC$

From the graph we can observe that *MM* and *714M* have the same behavior. The mean and the 95% confidence interval suggest that the two distributions can be considered as similar. Additionally we assume, after inspection of a Q-Q plot, that the data for each variable has a normal distribution. Our null hypothesis H_0 states that, under assumption that the two groups are independent, their variances are equal. We therefore apply Levene's Test for Equality of Variances to the $CCn[i]$, $0 < i < 10$. The results are shown in Table 5.2.

From the Levene's test we obtain the value of p , that stands for significance. If $p > 0.05$ then the null hypothesis is accepted. For zebrafish regions 1, 3, 4, 5, 8, 10 the hypothesis is accepted, the corresponding variances are equal (marked with * in Table 5.2). For regions 0, 2, 6, 7 and 9 the variances significantly differ. Finally, we performed the independent samples t-test. Based on the results from Levene's test we know which variances significantly differ; in Table 5.3 only the correct assumptions are listed.

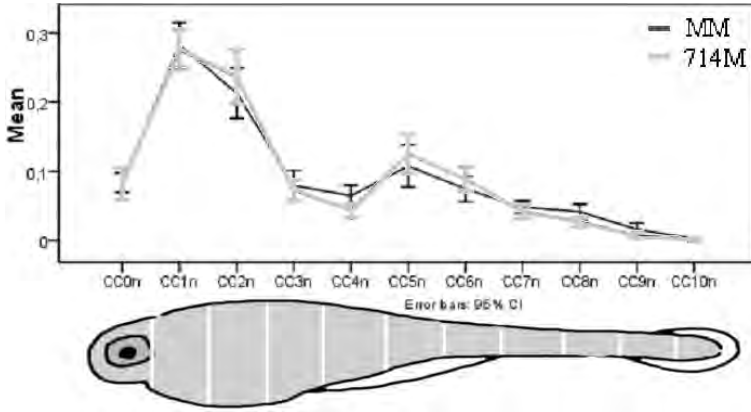


Figure 5.7: Spatial comparison of the normalized amount of clusters for MM and 714M with a 95% confidence interval as compared to the zebrafish template.

Region	H_0 Var	p value	
$CCn[0]$	=	0.017	
$CCn[1]$	=	0.218	*
$CCn[2]$	=	0.010	
$CCn[3]$	=	0.526	*
$CCn[4]$	=	0.512	*
$CCn[5]$	=	0.221	*
$CCn[6]$	=	0.011	
$CCn[7]$	=	0.042	
$CCn[8]$	=	0.488	*
$CCn[9]$	=	0.046	
$CCn[10]$	=	0.662	*

Table 5.2: Levene's Test for Equality of Variances for Cluster Count $CCn[i]$. Equal variances are marked with *.

Region	Var	t	Sig. (2t)	Mean Diff	Std. Err Diff	
$CCn[0]$	\neg	0.014	0.989	0.000	0.014	
$CCn[1]$	$=$	0.244	0.808	0.006	0.023	
$CCn[2]$	\neg	-0.909	0.365	-0.024	0.027	
$CCn[3]$	$=$	0.417	0.677	0.005	0.013	
$CCn[4]$	$=$	2.216	0.028	0.020	0.009	*
$CCn[5]$	$=$	-0.844	0.400	-0.018	0.022	
$CCn[6]$	\neg	-1.225	0.222	-0.015	0.013	
$CCn[7]$	\neg	0.815	0.416	0.005	0.007	
$CCn[8]$	$=$	2.093	0.038	0.013	0.006	*
$CCn[9]$	\neg	1.603	0.112	0.008	0.005	
$CCn[10]$	$=$	0.270	0.787	0.000	0.001	

Table 5.3: t-test for Equality of Means. Significant difference in the mean value is marked with *.

We observe that there is a significant difference (meaning significance < 0.05) in the mean value for regions 4 and 8 (marked with * in Table 5.3). Preliminary conclusions are as follows. In the head the highest proportion of clusters is found, followed by the injection site. Globally the distribution is the same for both *MM* and *714M*. Exceptions are region 4, adjacent to the injection site, and region 8 where a significant larger distribution of the clusters of *MM* bacteria is found compared to *714M*.

Distribution of the Amount of Infection

Next, we analyze the relation between mutant and wild-type in the area covered by granulomas:

Definition 5. *Cluster Size [region]* is the total area covered by the infection in region i , denoted by $CS[i]$. The area is expressed in units that represent the amount of pixels covering the region of interest within in the input images.

The image pixels are classified as *infection* in an automated fashion as described in Section 4.3.2. Background pixels are automatically excluded from the analysis.

In Figure 5.8 a graph, with 95% confidence, is depicted of the comparison of the average area of infection between *MM* and *714M*.

Again, as with the cluster count, we normalize the $CS[i]$ over the total CS for both *MM* and *714M* and compare the results in Figure 5.9.

Definition 6. *Normalized Cluster Size [region]* is the normalized $CS[i]$ feature, denoted by $CSn[i]$. Normalization is done over the total area covered by the infection throughout the entire zebrafish larva region: $CSn[i] = CS[i]/CS$.

From the graph in Figure 5.9 it seems that the mean and the distribution are similar for some regions and different for others; i.e., 1, 4, 5 seem to have a very different mean.

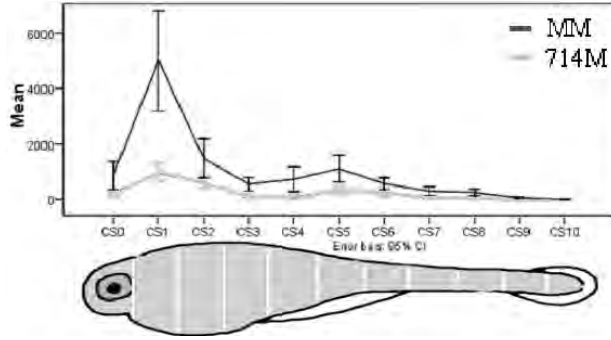


Figure 5.8: Spatial comparison of the average $CS[i]$ for MM and $714M$ in comparison to the zebrafish template.

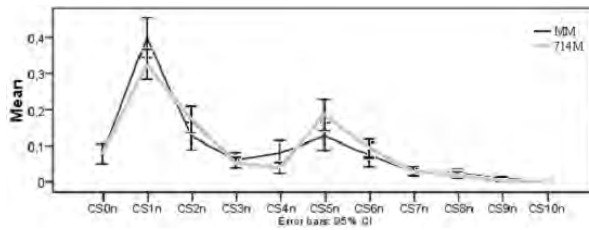


Figure 5.9: Spatial comparison of the normalized average total cluster size for MM and $714M$.

Region	H_0 Var	p value	
$CSn[0]$	=	0.266	*
$CSn[1]$	=	0.807	*
$CSn[2]$	=	0.025	
$CSn[3]$	=	0.690	*
$CSn[4]$	=	0.001	
$CSn[5]$	=	0.002	
$CSn[6]$	=	0.388	*
$CSn[7]$	=	0.255	*
$CSn[8]$	=	0.907	*
$CSn[9]$	=	0.036	
$CSn[10]$	=	0.274	*

Table 5.4: Levene’s test for equality of variances for total normalized Cluster Size $CSn[i]$, i is the region id.

Again, our null hypothesis states that, under the assumption that the two groups are independent, the variances are equal. We apply Levene’s Test for Equality of Variances to the $CS[i]$, $0 < i < 10$. The results are shown in Table 5.4.

For regions 0, 1, 3, 6, 7, 8 and 10 the significance is > 0.05 and thus our hypothesis is accepted. In Figure 5.9, it can be seen that the differences in the mean are considerable though the variances for MM and $714M$ remain the same.

To determine the difference of the mean values we use the knowledge gained from the Levene’s test to do the independent samples t-test. The results are shown in Table 5.5.

We observe there is a significant difference in the mean value for regions 1, 4 and 5 (marked with * in Table 5.5). This result is in correspondence with the initial observation of Figure 5.9 in which these means seemed rather different. The preliminary conclusions from these findings are the following. The larger part of the infection migrates towards the head of the zebrafish. The second largest part of infection, however, remains at the injection site. In wild-type infected fish (MM), a larger proportion of clusters is located in the head compared to the $714M$ mutant; i.e., significant difference of the mean while the variance is equal.

5.2.5 Conclusions

We have used a novel framework for automated granuloma cluster recognition in order to analyze the spatial distribution in zebrafish larvae. As a proof of concept we have analyzed the data for the zebrafish larva infected with the wild-type *Mycobacterium marinum* and a MM mutant $714M$.

From a statistical analysis of the data we can derive information on the spread of granulomas. In fish infected with the wild-type *Mycobacterium marinum* a higher amount of granuloma clusters is found. However, if we look at the normalized spread of infection it behaves approximately the same; it either stays at the site of the injection or it moves

Region	Var	t	Sig. (2t)	Mean Diff	Std. Err Diff	
<i>C</i> <i>Sn</i> [0]	=	0.016	0.987	0.000	0.021	
<i>C</i> <i>Sn</i> [1]	=	2.065	0.040	0.073	0.035	*
<i>C</i> <i>Sn</i> [2]	⊃	−1.739	0.084	−0.046	0.027	
<i>C</i> <i>Sn</i> [3]	=	0.517	0.606	0.006	0.012	
<i>C</i> <i>Sn</i> [4]	⊃	2.176	0.032	0.041	0.019	*
<i>C</i> <i>Sn</i> [5]	⊃	−2.102	0.037	−0.060	0.029	*
<i>C</i> <i>Sn</i> [6]	=	−0.850	0.397	−0.018	0.022	
<i>C</i> <i>Sn</i> [7]	=	−0.305	0.761	−0.003	0.009	
<i>C</i> <i>Sn</i> [8]	=	0.558	0.578	0.004	0.007	
<i>C</i> <i>Sn</i> [9]	⊃	1.443	0.151	0.004	0.003	
<i>C</i> <i>Sn</i> [10]	=	−0.528	0.598	0.000	0.001	

Table 5.5: t-test for Equality of Means for the normalized Cluster Size feature. Significant difference in the mean value is marked with *.

towards the head of the larva. For the wild-type *Mycobacterium marinum* it seems that the infection is more likely to migrate towards the head compared to the 714 mutant; in the 714 mutant it is established that the majority of the infection remains located at the injection site. The percentage of the amount of clusters per region is distributed approximately in the same way for both groups in this test.

In the following sections this approach is further elaborated with more mutants and a larger dataset. Moreover, other measurement parameters are to be considered in the analysis. Large volumes of data will allow to do predictions from the measurements using machine learning approaches.

5.3 Analysis of High Throughput Zebrafish Infections Screens Based on Approach 3

In the previous section we have analyzed the infection spread within the zebrafish larvae. In this section a pilot analysis is shown for Approach 3 that is described in Chapter 3. We repeated the experiment of Section 5.2. Again, we have chosen mutant 714 (*714M*), as it is reported as one of the 30 mutants which does not make the fish ill. As a result of this study we could, indeed, find certain characteristic patterns in granuloma cluster spread and size for both *MM* and *714M*. The acquisition is accomplished through imaging. For each zebrafish, a brightfield and fluorescence microscopy image is acquired (Figure 5.1 and 5.2). Until recently, these images were analyzed manually. The analysis included localization of the zebrafish shape in a brightfield image and qualitative estimation of the granuloma cluster size and spread from a paired fluorescent image. The fluorescent signal is often of low intensity and noisy and therefore manual analysis is difficult and will not result in objective evaluation.

We have designed and implemented a framework for automated shape retrieval and

cluster analysis [43] (cf. Chapter 2). The recognition algorithm was based on a deformable template matching [23, 16] approach that distributes the zebrafish shape into vertical sub regions (slices). Besides our own work this framework has also been applied in large scale applications [54]. However, this algorithm used for zebrafish detection had certain drawbacks: the algorithm was slow due to high complexity of template matching [23]; the slices were fixed-width vertical regions that did not discriminate between the top and the bottom of the shape.

The results obtained with Chapter 2 have resulted in the development of another algorithm based on Anchor Regions. This is described in Chapter 3. By the use of this algorithm it is possible to distribute the zebrafish shapes into certain characteristic (Figure 5.10) but not size fixed regions. This algorithm is more true to nature, it can better deal with the large biological variation in shape and size of the zebrafish larvae (Figure 3.1).

Brightfield Imaging: Shape Localization and Annotation

In our previous analysis of infection spread we used *Approach 2* as described in Chapter 2 to localize the zebrafish shape and divide it into 11 regions counting from head to tail. This allowed for annotating the shape as well as doing spatial analysis. For this study we have abandoned this approach and developed a new template matching algorithm that localizes the zebrafish shapes without the computationally complex sub-region search approach.

This novel algorithm is based on mathematical morphology and probability theory. The algorithm estimates the location of the head, body and tail region of the zebrafish larvae and its orientation. This is done by distributing the template into possible sub templates, that are representing these regions. A probability tree is created to find an optimal shape resembling a zebrafish, cf. Chapter 3. Due to the rotation invariance of the algorithm the zebrafish larvae do not need to be aligned in a certain way. In Figure 3.14 the example of an output of the novel algorithm is shown.

Fluorescent Imaging: Analysis

The fluorescent signal is analyzed per fish mask, as it was found in the previous step. No granuloma formation is present at the not infected group *NI*, therefore this group was only used to obtain an intensity value that represents the noise level threshold n . All signal below n is considered noise, while signal above n represents granuloma presence.

The larva is modeled in 5 regions, listed in Table 5.6. The division into regions is an estimation and is proposed based on possible regions of interest. The detection of the regions is done in an automated fashion by *Approach 3* (by choosing a prototype template that consists of head, body and tail regions).

This result is presented in an infographic in Figure 5.10. The use of this kind of infographics is to illustrate and summarize the data gained from the statistical analysis in a very clear way. This way of data visualization will be used throughout this chapter.

Area Name	Notation
HEAD	H
BODY TOP	BT
BODY BOTTOM	BB
TAIL TOP	TT
TAIL BOTTOM	TB

Table 5.6: Automatic region assignment.

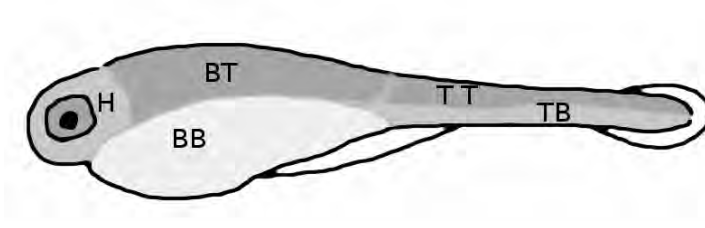


Figure 5.10: Graphical representation of the separation of a zebrafish into distinguished regions.

Output and Dataset Creation

Output is created in the form of an overlay image as in Figure 3.14, containing the retrieved zebrafish shape with the infection in the overlay and an entry in the *csv* file containing the granuloma spread in each of the regions; i.e., H , BT , BB , TT , TB , cf. Table 5.6. The granuloma spread for each region i is described by the amount of Clusters in a region, $CC[i]$, Definition 3, and the total amount of Infection in each region, $CS[i]$, Definition 5. Hence, we introduce the feature *Average Cluster Size*:

Definition 7. *Average Cluster Size [region]* is the average area covered by the infection in a single cluster at region i , denoted by $ACS[i]$. The area is expressed in units that represent the amount of pixels covering the region of interest within in the input images. The image pixels are classified as *infection* in an automated fashion as described in 4.3.2. Background pixels are automatically excluded from the analysis. Per region *Average Cluster Size [region]* is calculated by: $ACS[i] = CS[i]/CC[i]$.

5.3.1 Analysis and Results

In our study we set out to analyze the relationship between mutant and wild-type in the amount of clusters, spatial distribution and the cluster size. Note, that it is impossible to directly compare the amount of infection between different regions, since the regions are of different size. However it is possible to compare each region between the wild-type and the 714 mutant.

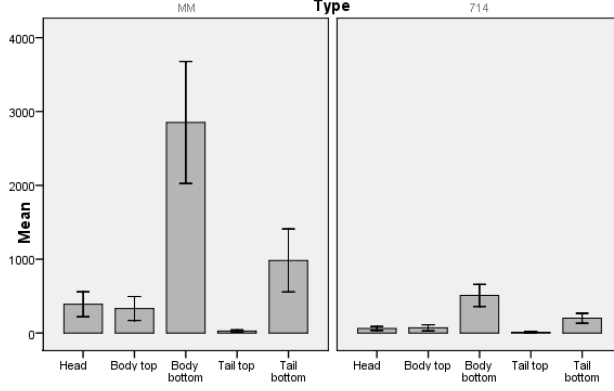


Figure 5.11: Spatial comparison of the mean amount of infection for MM and 714M in relation to the zebrafish regions.

Amount of Infection

First we compare the Cluster Size, $CS[i]$, Definition 5, measured as a numerical density in each region, compared between MM and $714M$ (cf. Figure 5.11).

At this point this distribution is not conclusive. This is due to the fact that MM make the fish more ill and thus overall produces more granulomas compared to $714M$. The mean amount of infection throughout the entire fish is 4558.24 (standard deviation $\sigma M = 569.51$) for MM and 825.69 ($\sigma M = 102.74$) for $714M$. The amount of infection in wild-type (MM) is roughly 5 times higher compared to $714M$.

In order to analyze the different batches in the same way we consider the normalized Cluster Size feature, $CSn[i]$, Definition 6. The normalization is done for each individual case and subsequently the mean is calculated. In Figure 5.12 the results are depicted.

From the graph we can observe that MM and $714M$ have approximately the same behavior. The mean and the 95% confidence interval suggest, that the two distributions in H , BT and TT can be considered similar. In BB (body bottom) and TB (tail bottom) the distribution behaves differently. Let us consider this in more detail. Our null hypothesis states that, under the assumption that the two groups are independent, their variances are equal. We therefore apply Levene's Test for Equality of Variances to the $CSn[i]$, $i \in \{H, BT, BB, TT, TB\}$, followed by the independent samples t-test. Based on the results from Levene's test we know which variances significantly differ. We observe that there is a significant difference for the total amount of infection (respectively $Sig.2t = 0.011$ and $Sig.2t = 0.018$) in the mean value for regions BB and TB (variance not equal).

Amount of Granuloma Clusters

We compare the average number of clusters, Definition 3, (variable $CC[]$) between MM and $714M$ (cf. Figure 5.13).

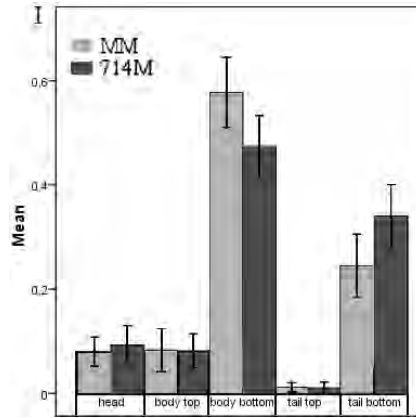


Figure 5.12: Comparison of spatial distribution of the normalized mean amount of infection ($CSn[i]$) for MM and $714M$ in relation to the zebrafish regions with a 95% confidence interval.

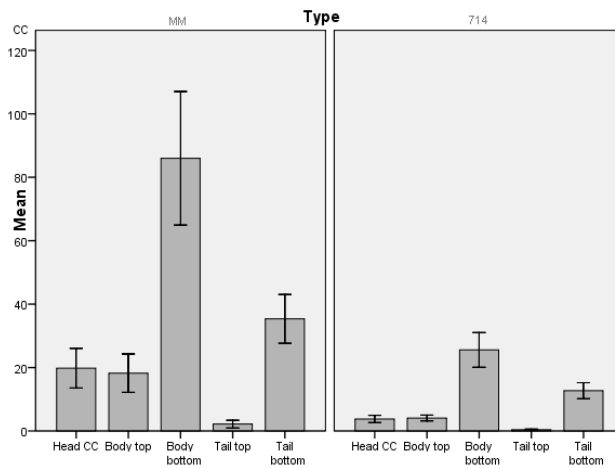


Figure 5.13: Comparison of spatial distribution of the mean amount of clusters for MM and $714M$ in relation to the zebrafish regions.

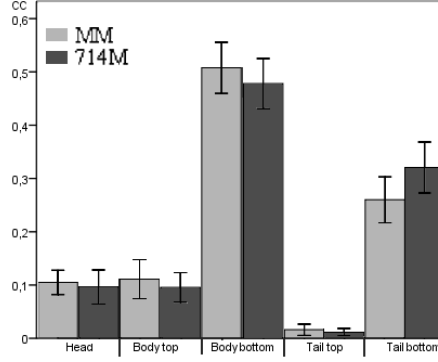


Figure 5.14: Comparison of spatial distribution of the normalized mean amount of clusters for *MM* and *714M* in relation to the zebrafish regions.

Throughout the entire fish for *MM* the $CC[i]$ is higher than for *714M*. *MM* has a mean amount of 161.94 ($\sigma M = 18.00$) throughout the entire fish and *714M* has a mean amount of 46.40 ($\sigma M = 4.13$). The amount of clusters in *MM* is about 3 times higher than in *714M*. This makes it difficult to compare the relative behavior of granuloma clusters. Therefore let us consider the normalized clusters amount $CCn[i]$, Definition 4, in Figure 5.14.

From the graph we can observe that *MM* and *714M* have approximately the same behavior. We analyze the graph in more detail by testing equality of variances and mean equality of each region, in the same way as described previously and find no significant difference in the mean values.

Average Size of Granuloma Clusters

We also look at the average size of the granuloma clusters, $ACS[i]$, Definition 7. Figure 5.15 shows the average cluster sizes for different regions of the zebrafish. We analyze the graph in more detail by testing equality of variances and mean equality of each region, in the same way as described previously. We observe that there is a significant difference (respectively $Sig.2t = 0.000$ and $Sig.2t = 0.009$) in the mean value for *BB* and *TB* regions (variance not equal).

Results

In all regions of the zebrafish for *MM* the amount of infection is higher than the *714M* with larger granulomas.

The same percentage of the granulomas in both *MM* and *714M* are located in the head region. A larger percentage of the granulomas of the *MM* than *714M* are located in the body bottom region.

A higher percentage of the granulomas of the *714M* than *MM* are located in the tail bottom region. In Figure 5.16 a graphical representation of the form of an infographic

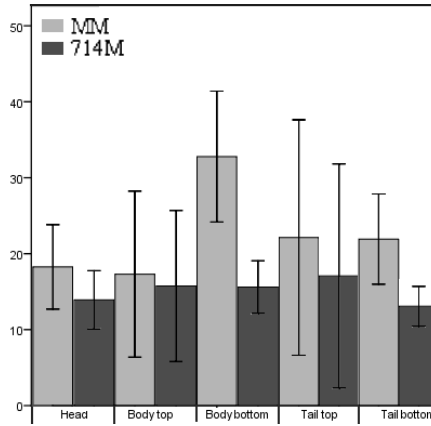


Figure 5.15: Comparison of spatial distribution of the average cluster sizes ($ACS[]$) for MM and $714M$ in relation to the zebrafish regions.

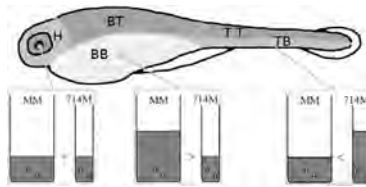


Figure 5.16: Graphical representation of the amount of infection. There is always more MM infection than $714M$; in the image this is illustrated by a wider bar. The percentages of their presence in certain regions are different; this trend is depicted through level height.

of these results is shown.

MM has a higher amount of clusters than $714M$ in all regions of the zebrafish (overall the amount of clusters is 3 times higher). There is no significant difference in cluster spread pattern for MM and $714M$.

In most regions the relative cluster sizes are not significantly different (though MM tends to have larger clusters). The exceptions are the body bottom and the tail bottom area, where MM gives significantly larger clusters than $714M$.

As a result of using *Approach 3* instead of *Approach 2* we could consider infection behavior in regions of interest. This resulted in more true to nature analysis that could be directly comprehended by the users.

5.4 In Depth Analysis of High Throughput Zebrafish Infections Screens

For a more in depth analysis of the data we consider a dataset that was created during 11 experiments performed on infection by different mutants of the *Mycobacterium marinum* strains visualized with fluorescence in *mCherry* red.

We analyze the data to find statistically relevant differences in behavior. We attempt to relate the result to biology. Analysis is performed on the same selection of mutants.

Mutants that were considered during the experiments were the following: 262, 308, 414, 415, 421, 423, 431, 730, 748, 801, 817, 885, 941, 943.

The following mutants were attenuated for early granuloma formation (infection levels were less than 50% of the E11 (wild-type) infection ([55, 64]): 262, 308, 421, 431, 730, 748 and 801 (cf. [55, 66]).

For mutants 748 and 801 it is known that both have a mutation in the genes from the ESX-1 region. Genes from the ESX-1 region are being knocked down in order to see which gene influence the virulence.

Mutant strains with knocked out genes that are involved in cell wall bio-synthesis (cf. [55], p64) are: 262, 431 and 730. Mutant strains 262 and 431 showed a substantial increased susceptibility to rifampicin (cf. [55, 65]). Other mutant strains are not specifically annotated and their specific functional profile is not known.

5.4.1 Experiments

The data that has been provided is separated into 11 experiments. Cases in each experiment are provided as images, i.e., a readout entails a brightfield image and a corresponding fluorescent image.

For retrieval, each experiment is identified by the user that took the images and a time stamp at the date the experiment was performed. The number of samples, i.e. zebrafish, that were exposed to the mutants differs per experiment.

5.4.2 Analysis and Results

We set out to analyze the difference in infectious behavior of the different mutants and attempt to discover distinguishable characteristics between the different mutants. We have established a more fine grained set of features that we measure; these are listed in Table 5.7.

5.4.3 Total amount of Infection

As a first comparison we consider the total area covered by the infection CS , Definition 5, measured for each mutant type.

Definition 8. *Cluster Size per mutant* is defined as average CS of all mutant m bacteria infected larva, denoted by $CSM[m]$.

Variable	Description
CS	Definition 5
CC	Definition 3
ClusterRegion	CS/CC (Average size of one cluster)
HClusterSize	Area covered by infection in the larva <i>Head</i> area
BClusterSize	Area covered by infection in the larva <i>Body</i> area
TClusterSize	Area covered by infection in the larva <i>Tail</i> area
HClusterSizeN	$HClusterSize/CS$
BClusterSizeN	$BClusterSize/CS$
TClusterSizeN	$TClusterSize/CS$
IPClusterSize	Area covered by infection in the larva <i>Injection Point</i> area
HRClusterSize	Area covered by infection in the larva <i>Heart</i> area
IPClusterSizeN	$IPClusterSize/CS$
HRClusterSizeN	$HRClusterSize/CS$

Table 5.7: Different variables that can be taken into account and their description.

It is preferred to compare each mutant infection pattern to the infection by the wild-type (*E11* or *MM*). We separate the data into groups, grouped by mutant id without considering the experiment number. The first step is to test each group for normality. Therefore we consider two well-known tests of normality, the Kolmogorov-Smirnov Test and the Shapiro-Wilk Test. The sample size for each mutant is sometimes low (< 50), therefore the Shapiro-Wilk Test seems more appropriate, as it is typically suitable for handling both small and large sample sizes. We have tested the distributions per mutant for all experiments for being normally distributed by applying a Shapiro-Wilk Test. The results show that none of the mutant data was normally distributed. In Figure 5.17 this can be seen through inspection of the mean and the standard deviation. Explanation for the lack of not normally distributed values is the following: the amount of injected infection and imaging conditions were different for each experiment. A solution is to normalize the values per mutant per experiment.

The next step would be to normalize the groups one by one and then combine them back into one whole. The infection amount is normalized per experiment over the mean amount of infection of the wild-type in that experiment:

Definition 9. *Normalized Cluster Size per mutant* is the normalized $CSM[m]$ feature per mutant m bacteria infected larva, denoted by $CSMn[m]$. Normalization is done over the wild-type infected fish of a single experiment: $CSMn[m] = CSM[m]/CSM[MM]$.

The resulting dataset, again, contained no normally distributed variables. In order to be able to compare each mutant to the wild-type we checked the (not normal) distributions for similarity by the nonparametric Mann-Whitney test. Distributions similar to normalized E11 that were found were: 423, 885, 943. Suggested results are the following:

- Mutant 423 shows *no significant difference* in amount of infection as compared to the wild-type (according to a combination of all experiments).

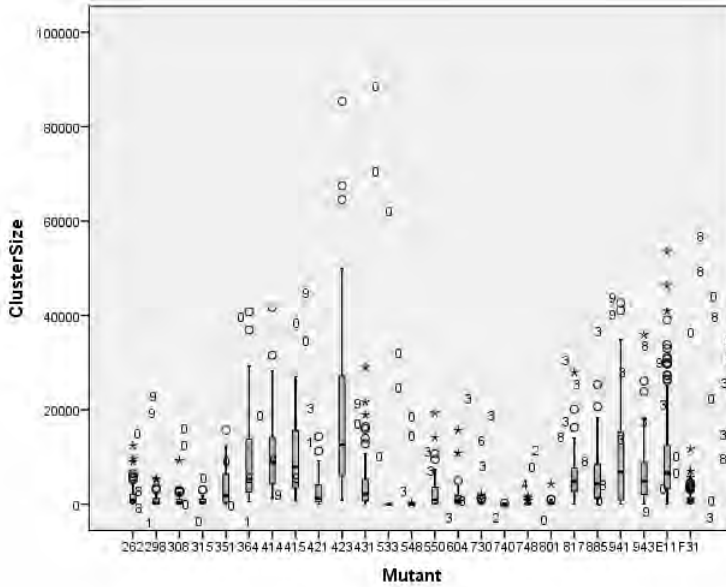


Figure 5.17: Graphical representation of the ClusterSize as it was measured for different mutants.

- Mutant 885 shows *no significant difference* in amount of infection as compared to the wild-type (according to a combination of all experiments).
- Mutant 943 shows *no significant difference* in amount of infection as compared to the wild-type (according to a combination of all experiments).

In order to be able to compare more mutants, we have attempted to separate the data into additional groups, namely clustered by mutant and experiment. To control if the data were normally distributed again, the Shapiro-Wilk Test was needed. Within the 11 experiments for the following mutants ClusterSize variables were normally distributed:

- Experiment 0: wild-type, 431, 748, 801, 941, 943
- Experiment 1: 423, 431
- Experiment 2: wild-type, 262, 941
- Experiment 3: wild-type, 421, 817, 943
- Experiment 5: wild-type, 421, 730, 817, 885, 941, 943
- Experiment 6: 262, 548, 730, 817, 885
- Experiment 7: wild-type, 943

- Experiment 8: wild-type, 423, 730, 817, 943
- Experiment 9: wild-type, 308, 414, 415
- Experiment 10: 298, 423, 941

In this manner the experiments where the infection pattern in wild-type is normally distributed can now be compared to the normally distributed mutant infection pattern. This was performed by Levene's test for variance followed by an Independent t-test for equality of means. With this test we could find the difference in infection and get an estimation of the difference.

For all the experiments where infection patterns were not normally distributed the Mann-Whitney U-test was used. This test could not be used to determine the actual difference of infection size, however it does indicate if the percentage of infection is significantly different. We use this test to support the results that were gained from experiments where the t-test could be used. The mutant strains which had no normally distributed data in any experiments are not considered. We also did not take into consideration those mutant infection patterns in which the amount of samples in an experiment was < 10 . The results of $CSMn[m]$ comparison are shown in Table 5.8.

It is difficult to extract direct conclusions from these results, since the conclusions presented here are only supported by limited experiments. However preliminary conclusions are shown in Table 5.9. For some mutant strains the results were not informative, as different experiments provided results that could not be integrated. Mutants that behaved in a similar way for different experiments were: 262, 308, 730, 748, 801 and 885.

Discussion

All considered, the mutant 885 behaves in the same way as the wild-type. No conclusive remarks can be stated for mutant 423 and 943, as in the current dataset the experiments provide contradictory results. A graphical representation of those results excluding the contradictions is shown in Table 5.10.

Mutant strains 748 and 801 are known to have a mutation in the genes from the ESX-1 region. For the ESX-1 it is known that the mutants are less virulent [55, 54].

In our results mutant strains 748 and 801 show a very large difference in infection percentage as compared to the wild-type (2% vs 100%). The low percentage of infection for both strains corroborates the known low virulence phenotype in zebrafish. We therefore assume that the results obtained with the other mutants can be used to determine their virulence phenotype.

Mutant strains 262 and 730 are gene knock-outs involved in cell wall bio-synthesis and show a similar behavior — that is a very large difference in infection percentage from the wild-type, respectively 25% and 6%. Since the knocked out genes are located at different locations in the pathway the difference in percentages suggests a different mechanism, yet similar effect.

	0	1	2	3	4	5	6	7	8	9	10
262											
308											
414											
415											
421											
423											
431											
730											
748											
801											
817											
885											
941											
943											

Table 5.8: Variance and mean comparison between *MM* and each of the mutants. In cases where the t-test could be performed we also show the ratio of infection mean *mutant*/*MM*. For experiments which had a not normal distribution Mann-Whitney (*) is used to indicate if the percentage of infection is significantly the same or different.

Mutant	sig. diff	according to experiment	avg. percentage of MM infection
262	yes	2	25%
308	yes	9	11%
414	yes	10	
	no	0, 9	
415	yes	1	
	no	0, 9	
421	yes	3, 5, 7	
	no	2, 6	
423	yes	0, 1	
	no	8	
431	yes	1, 9, 10	
	no	0	
730	yes	5, 8	6%
748	yes	0	2%
801	yes	0	2%
817	yes	3, 7	
	no	5, 6, 8	
885	no		
941	yes	2, 9, 10	
	no	0, 3, 4, 5	
943	yes	3, 7	
	no	5, 6, 8	

Table 5.9: Experiment results: difference in amount of infection as compared to the wild-type throughout the entire fish.





Mutant		Wild-type	
262			
308			
730			
748			
801			
885			

Table 5.10: Difference in amount of infection as compared to the wild-type throughout the entire fish, depicted as bar height.

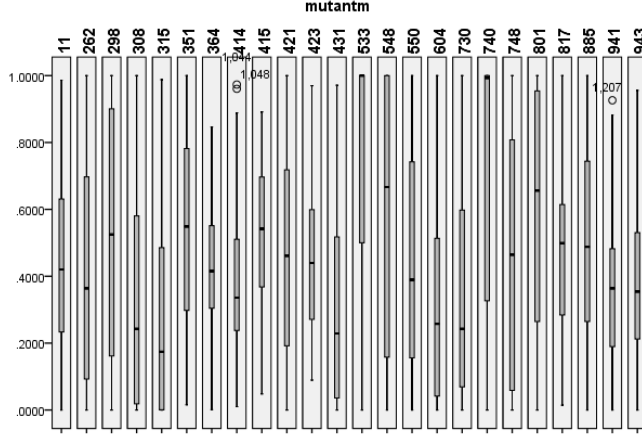


Figure 5.18: Box plot representation of the normalized cluster size CSn for the *Body* region of the wild-type and the different mutants. A lot of difference can be seen in the behavior of the mutant strains.

5.4.4 Amount of Infection separated into Regions

We want to compare the amount of infection as it is present in the characteristic regions (*Head, Body, Tail, Heart Region, Injection Point*). Note that initially we consider the comparison of the amount of infection per region for different mutants (e.g., does the percentage of infection in the *Head* region differ for mutants A and B), instead of comparing the infection spread within each mutant. All cases can be compared in a correct way by normalizing the amount of infection in each region over the total infection amount in each fish. In this way the problem of a different total amount of infection is solved. For all objects the entire infection in each case is set to 1.0 and normalizes the amount of infection in each region. A normalized distribution of infection in the *body* area the fish is shown per mutant in Figure 5.18. We see a lot of difference in the behavior of the mutant strains, yet it needs to be proven to be statistically significant. We attempt to test the significance by an independent sample t test.

In order to perform this test the data under comparison needs to be normally distributed. Therefore we apply a Shapiro-Wilk test to test for a normal distribution. We have analyzed the following features:

- $CSn(H)$ contained no normally distributed data for any of the mutants;
- $CSn(B)$ contained normally distributed data for mutants: 351, 364, 414, 415, 423, 817, 943;
- $CSn(T)$ contained normally distributed data for mutants: 351, 414;
- $CSn(HR)$ contained normally distributed data for mutant 414;

- $CSn(IP)$ contained no normally distributed data for any of the mutants.
- Subsequently only the relationship between 351, 364, 414, 415, 423, 817, 943 in $CSn(B)$ and subsequently for 351, 414 in $CSn(T)$ could be considered for parametric statistical testing.

Our null hypothesis states that, under the assumption that the two groups are independent, their variances are equal. Therefore we need to test the variances first. This is done with Levene's test for equality of variances. If the variances are equal an independent samples t-test is performed in order to determine the equality of means. In Table 5.11 the results are shown.

The results of these tests are interpreted in the following way:

- Mutant 351 had statistically significant larger percentage of total infection ($53\% \pm 35\%$ of total infection) present at the *body area* as opposed to mutant 943 ($38\% \pm 27\%$ of infection)
- Mutant 415 had statistically significant larger percentage of total infection ($52\% \pm 20\%$ of total infection) present at the *body area* as opposed to mutant 414 ($39\% \pm 23\%$ of infection)
- Mutant 414 had statistically significant larger percentage of total infection ($39\% \pm 23\%$ of total infection) present at the *body area* as opposed to mutant 943 ($38\% \pm 27\%$ of infection)
- Mutant 817 had statistically significant larger percentage of total infection ($47\% \pm 27\%$ of total infection) present at the *body area* as opposed to mutant 943 ($38\% \pm 27\%$ of infection)

None of the variables of the wild-type were normally distributed. In order to be able to accomplish a comparison of the infection spread of each mutant infection and the wildtype we have chosen a different approach.

As an alternative for the parametric t-test, the Mann-Whitney U-test and the 2-sample Kolmogorov-Smirnov test are considered. For this test the variables do not need to be normally distributed. However the variables do need to follow an approximately same distribution and there is a loss of information, since values are presented as ranks. Mean equality can therefore not be predicted. This makes it a less powerful test than the t-test. We compare the control group *MM* to all the mutants and look for similar distributions. The null hypothesis is that the distributions of both groups are equal. The following distributions were found as the same. We used the Mann-Whitney U-test, since the amount of samples was fairly low and our data distribution is skewed (a lot of 0 values). We state H_0 : there is no difference in ratio of infection between the wildtype and mutant x . The results that reject H_0 ($p \leq 0.05$ and $|z| > 1.96$) are given in a graphical representation in Table 5.12, matching z and $p(2t)$ values for this table are given in Table 5.13.

Compare	Measure	Var	Means	N	Mean	σ	$\bar{\sigma}$
351/364	(B) <i>CSn</i>	!equal	no sig. diff				
351/414	(B) <i>CSn</i>	equal	no sig. diff				
351/415	(B) <i>CSn</i>	equal	no sig. diff				
351/423	(B) <i>CSn</i>	equal	no sig. diff				
351/817	(B) <i>CSn</i>	equal	no sig. diff				
351/943*	(B) <i>CSn</i>	equal	sig. diff	21/49	.533/.375	.292/.235	.063/.033
364/414	(B) <i>CSn</i>	equal	no sig. diff				
364/415	(B) <i>CSn</i>	equal	no sig. diff				
364/423	(B) <i>CSn</i>	equal	no sig. diff				
364/817	(B) <i>CSn</i>	equal	no sig. diff				
364/943	(B) <i>CSn</i>	equal	no sig. diff				
414/415*	(B) <i>CSn</i>	equal	sig. diff	33/30	.393/.526	.253/.205	.044/.037
414/423	(B) <i>CSn</i>	equal	no sig. diff				
414/817	(B) <i>CSn</i>	equal	no sig. diff				
414/943*	(B) <i>CSn</i>	equal	sig. diff	30/49	.526/.375	.205/.235	.0374/.033
415/423	(B) <i>CSn</i>	equal	no sig. diff				
415/817	(B) <i>CSn</i>	equal	no sig. diff				
415/943	(B) <i>CSn</i>	equal	no sig. diff				
423/817	(B) <i>CSn</i>	equal	no sig. diff				
423/943	(B) <i>CSn</i>	equal	no sig. diff				
817/943*	(B) <i>CSn</i>	equal	sig. diff	51/49	.473/.375	.237/.235	.033/.033
351/414	(T) <i>CSn</i>	equal	no sig. diff				

Table 5.11: Levene's test for equality of variances followed by the t-test for equality of means. Mutant combination with significantly different means are marked with a *.






















































































Mut	Head	Body	Tail	Heart	Inj. Point
Location					
262					
298					
308					
315					
351					
415					
421					
431					
550					
604					
730					
740					
748					
801					
885					
941					

Table 5.12: Graphical representation of the percentage of infection measured in certain zebrafish regions. We compare each mutant (bars on the left of each comparison) to the wild-type (bars on the right of each comparison). Note, that not the amount of infection, but the percentages are compared.

Mut	Head	Body	Tail	Heart	Inj. Point
262	$z = -2.105, p = .035$			$z = -2.342, p = .019$	
298			$z = -4.261, p = .000$		$z = -5.488, p = .000$
308		$z = -3.457, p = .001$		$z = -4.525, p = .000$	$z = -3.327, p = .001$
315	$z = -4.630, p = .010$	$z = -2.570, p = .000$	$z = -3.694, p = .000$	$z = -3.981, p = .000$	
351	$z = -2.145, p = .032$				
415		$z = -2.067, p = .039$			
421	$z = -2.660, p = .008$			$z = -3.462, p = .001$	$z = -2.484, p = .013$
431		$z = -2.791, p = .005$		$z = -4.100, p = .000$	
548	$z = -3.124, p = .002$		$z = -2.667, p = .000$	$z = -4.132, p = .008$	$z = -5.117, p = .000$
550			$z = -2.282, p = .022$		$z = -2.197, p = .028$
604	$z = -2.947, p = .003$	$z = -2.097, p = .036$	$z = -2.996, p = .003$	$z = -3.861, p = .000$	
730	$z = -2.175, p = .030$	$z = -2.102, p = .036$		$z = -3.405, p = .001$	
740	$z = -4.627, p = .000$	$z = -3.162, p = .002$	$z = -3.533, p = .000$	$z = -5.595, p = .000$	$z = -5.184, p = .000$
748	$z = -3.575, p = .000$	$z = -2.059, p = .039$		$z = -4.704, p = .000$	$z = -4.193, p = .000$
801	$z = -3.407, p = .001$	$z = -2.545, p = .011$	$z = -2.920, p = .003$	$z = -7.323, p = .000$	$z = -5.439, p = .000$
885	$z = -3.519, p = .000$			$z = -4.394, p = .000$	
941		$z = -2.186, p = .029$	$z = -2.826, p = .005$		

Table 5.13: Reported z and $p(2t)$ values of the Mann-Whitney U-test. We compare each mutant to the wild-type. Statistically significant differences are given here.

Discussion

As we have found earlier (Table 5.10) some mutants give large differences in infection amount throughout the entire fish, as compared to the wild-type. If we in addition consider the infection amounts within the regions, as presented in Table 5.12, we observe new patterns. For example, ESX-1 mutant strains 748 and 801, also on regional level, show similar behavior, namely for both lower percentage of infection is concentrated in the tail, head, injection point and heart region as compared to the wild-type. Mutant strains 262, 431 and 730 (genes knock outs which are involved in cell wall bio-synthesis) show a similar behavior at the injection point and heart region. All of them have a higher percentage of infection than the wild-type present in the heart and the same ratio as the wild-type at the injection point. Yet, the percentages of infection as measured over the head/body and tail regions do differ.

According to these results the following mutant strains behave the same as the wild-type when measured for the HR/IP sites: 351, 415, 941.

The measurements presented here can not be interpreted to directly compare the infection amount between regions in a single mutant strain. For example, although mutant strain 262 has a lower amount of infection as compared to the wild-type in *HR* and the same amount of infection as the wild-type in *IP*, which does not mean that for mutant 262 there is more infection in *IP* than in *HR*. The reason for it is that the results presented here are relative to the wild-type and the distribution of infection within the wild-type should be considered to make that comparison.

5.4.5 Infection Flow per Mutant Strain

For each mutant strain we want to investigate the difference in the response of the innate immune system in a comparison of different regions in a zebrafish embryo. For example, for a certain mutant strain, does the *heart* contain more infection then the *injection point*.

We use the same normalized dataset from the previous subsection. The data is not normally distributed and thus we use the Wilcoxon Signed-Rank Test instead of the *paired t-test*. We do this test only for the injection point *IP* and heart *HR* since the area they cover are approximately the same (while the head/body/tail region differs per individual case). Therefore no additional normalization over the region size needs to be performed.

Additionally these two regions are indeed dependent, since per fish the amount of total infection is always 1.0 and thus there is a connection between *HR* and *IP*, namely $HR + IP + rest = 1.0$. We run the Wilcoxon test on the data. Our H_0 states that there is no difference between the ratio of infection in *IP* and *HR*.

As a result we found that for most mutants the spread did not differ and therefore H_0 is not rejected. Significant difference was only found for mutants 298, 415, 550 and the wild-type. If we consider the median values of these mutants the following can be concluded:

- Mutants 298, 415, 550 and the wild-type had a significantly higher percentage of

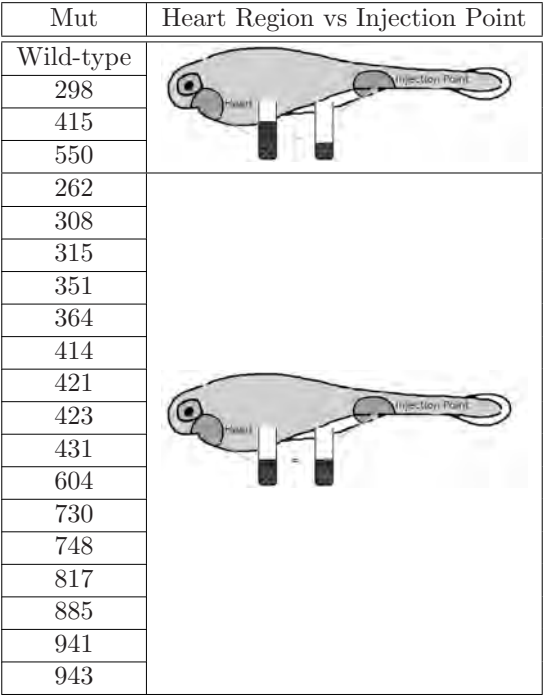


Table 5.14: Distribution of the percentage of infection within each individual, indicated by bar height. We consider the heart area and the injection point. The size of the areas is the same.

infection present in the *heart* as compared to the *injection point*.

- For other mutants there was no significant difference in the percentage of infection between the *heart* and the *injection point*.

A graphical representation of these results is shown in Table 5.14.

Discussion

Mutants 298, 415 and 550 seem to have the same flow of infection from the injection point to the heart. For all other mutant strains this mechanism works differently and the infection is evened out between the injection point and the heart.

We can see that the infection spread through the regions for the wild-type is: $HR > IP$ and therefore the results as they are presented earlier in Table 5.10 can indeed not be used for direct comparison of the distribution within a mutant strain.

In order to keep the statistics clear we have only compared regions of equal size to each other (HR and IP). Regions H , B , T have therefore not been analyzed in this experimental setup.

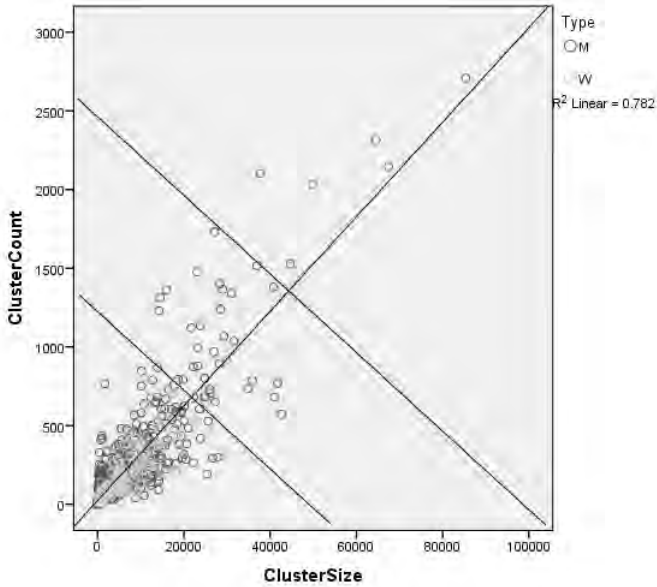


Figure 5.19: Scatter plot of the average cluster size. Spot location within each region does not provide locational information.

5.4.6 Average Size of Clusters

In our previous analysis we also took into consideration the normalized ClusterCount values. These values represent total infection area measured over all clusters at a certain location (cf. Section 5.2). Now we consider the average size of one cluster in a region i , $ACS[i]$ (Definition 7) separated over characteristic regions (*Head, Body, Tail, Heart Region, Injection Point*). Additionally we also consider the average size of one cluster within an entire fish.

Definition 10. *Average Cluster Size* is the average area covered by the infection in a single cluster, denoted by ACS . Area is expressed in units that represent the amount of pixels covering the entire object of interest within in the input images. The image pixels are classified as *infection* in an automated fashion as described in 4.3.2. Background pixels are automatically excluded from the analysis. *Average Cluster Size* is calculated by: $ACS = CS/CC$.

In Figure 5.19 we see that the correlation of the Cluster Size and Cluster Count shows a near linear increase. Pearson's p -value is 0.000, indicating that this relationship is significant. Pearson's correlation between the two groups is showing a significant positive correlation of 0.884, which illustrates that if the CS variable increases then CC also increases.

This also suggests that the interesting cases are the ones that do not follow this

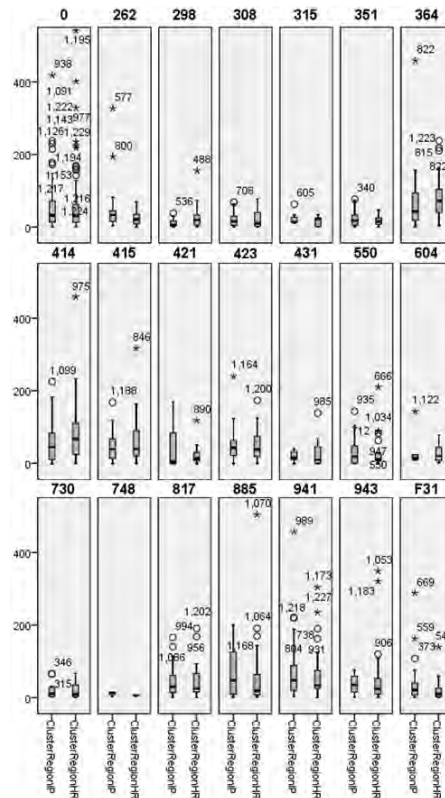


Figure 5.20: Box plot representation for the *ACS* is the injection point and the heart region for different mutants.

trend and are the outliers to the correlation regression (cf. Figure 5.19). A graphical rendition of the distribution for the injection point region and the heart region is shown in Figure 5.20.

We have performed normal distribution analysis on *ACS* for all the mutant strains and the regions *H*, *B*, *T*, *HR*, *IP*. From this approach we obtained the following outcome:

- ACS contained normally distributed data for mutants: 415, 885, 943;
- $ACS[H]$ contained normally distributed data for mutant 351, 415, 548.
- $ACS[B]$ contained normally distributed data for mutant 315, 604, 943;
- $ACS[T]$ contained normally distributed data for mutant 423;
- $ACS[HR]$ contained normally distributed data for mutants: 315, 351;
- $ACS[IP]$ contained normally distributed data for mutants: 351, 415, 748, 943.

Compare	Measure	Var	Means	N	Mean	σ	$\bar{\sigma}$
415/885	<i>ACS</i>	\neg	\neg				
415/943*	<i>ACS</i>	=	=	30/49	36/27.79	18.85/14.98	3.4/2.14
885/943*	<i>ACS</i>	\neg	\neg	58/49	40/27.79	27.35/14.98	3.59/2.14
351/415*	<i>ACS[H]</i>	\neg	\neg	14/28	13.68/29.19	7.70/22.01	2.06/4.16
351/548	<i>ACS[H]</i>	=	=				
415/548*	<i>ACS[H]</i>	\neg	\neg	28/6	29.19/8.29	22.01/6.89	4.16/2.81
315/604	<i>ACS[H]</i>	=	=				
315/943	<i>ACS[B]</i>	=	=				
604/943	<i>ACS[B]</i>	=	=				
315/351	<i>ACS[HR]</i>	=	=				
351/415	<i>ACS[IP]</i>	=	=				
351/748*	<i>ACS[IP]</i>	\neg	\neg	14/14	28.07/7.94	25.68/5.72	6.80/1.53
351/943	<i>ACS[IP]</i>	=	=				
415/748*	<i>ACS[IP]</i>	\neg	\neg	24/14	50.33/7.94	41.76/5.72	8.52/1.53
415/943	<i>ACS[IP]</i>	\neg	=				
748/943*	<i>ACS[IP]</i>	\neg	\neg	14/38	7.94/37.58	5.72/23.85	1.53/3.87

Table 5.15: Levene’s test for equality of variances followed by the t-test for equality of means. Mutant combinations with significantly different means are marked with a *.

Our null hypothesis, again, states that, under the assumption that the two groups are independent, their variances are equal. The results are shown in Table 5.15. Note, that the wild-type is not present in this table, since it contained no normally distributed data for the *ACS*.

The results of these tests are interpreted in the following way (Arbitrary Units (AU) were measured in pixels):

- Mutant 415 resulted in statistically significant larger clusters (36 ± 22 AU) throughout the entire larva as opposed to mutant 943 (28 ± 17 AU).
- Mutant 885 resulted in statistically significant larger clusters (40 ± 31 AU) throughout the entire larva as opposed to mutant 943 (28 ± 17 AU).
- Mutant 415 resulted in statistically significant larger clusters (29 ± 26 AU) in the head region as opposed to mutant 351 (14 ± 10 AU).
- Mutant 415 resulted in statistically significant larger clusters (29 ± 26 AU) in the head region point as opposed to mutant 548 (8 ± 10 AU).
- Mutant 351 resulted in statistically significant larger clusters (28 ± 32 AU) near the injection point as opposed to mutant 748 (8 ± 7 AU).
- Mutant 415 resulted in statistically significant larger clusters (50 ± 50 AU) near the injection point as opposed to mutant 748 (8 ± 7 AU).








Location		Average size of a granuloma
entire larva	*	
entire larva	*	
head	**	
head	**	
injection point		
injection point	***	
injection point		

Table 5.16: Graphical representation of the average granuloma cluster size. Only comparisons of mutant infection strains that were found to be significant are shown. In each image the light gray represents the region where comparison is done. The dark grey circle depicts the standard deviation of the largest and the inscribed white circle (if present) the smallest granuloma size. The black line inside the circle represents the average size. It is used for reference.

- Mutant 943 resulted in statistically significant larger clusters (38 ± 28 AU) near the injection point as opposed to mutant 748 (8 ± 7 AU).

No other statistically significant relationships were found. A graphical representation of these results is given in Table 5.16.

Discussion

We would prefer to compare the mutant strains to the wild-type rather than to another mutant strain. However, since the wild-type was not present in the results we are considering mutants that mimic the behavior of the wild-type and that are present in the results of our analysis.

As we have found earlier (cf. Table 5.8) mutant strain 415 had approximately the same amount of infection according to experiments 0 and 9 as the wild-type as measured throughout the entire fish. This also applies for mutant 885 (cf. Table 5.8). We therefore use 415 and 885 to predict that the wild-type will have slightly larger clusters throughout the entire body as mutant 943 (cf. * in Table 5.16).

Strain 415 shares the same distribution of the infection between the H region as the wild-type (cf. Table 5.12). We therefore use 415 to predict that the wild-type will have larger clusters in its head as compared to the mutants 351 and 548 (cf. ** in Table 5.16).

Strain 415 shares the same distribution of the infection between the *IP* and *HR* region as the wild-type (cf. Table 5.14 and Table 5.12). We therefore use 415 to predict that the wild-type will have larger clusters at the injection point as compared to mutant 748 (cf. *** in Table 5.16).

In order to be able to do other comparisons with not normally distributed data we need to use an alternative for the parametric t-test. However, since the mean equality can not be predicted it is more difficult to say something about it. In the case of *ACS* we have chosen not to perform non-parametric testing as it is less powerful and makes reasoning about the results harder. However this test will be included in future work.

5.5 Conclusions

In this chapter we have described a recipe for the analysis of the data that is retrieved by our algorithm from Chapter 3 in order to retrieve interesting patterns. We are attacking the problem at hand in the following way.

- First, the measurements are collected and arranged into usable data.
- Second, the data is reduced by statistical analysis. During this step tradeoffs must be chosen between parametric and non parametric testing, the type of normalization and data clustering.
- Then the data can be reduced by the use of an infographic. Our infographics contain a lot of data, yet show it in a compact, understandable way. The meaning of the data can be illustrated much faster than from written statistical data. All the infographics used here are supported by statistical analysis. In this way the data can be presented to the user in a very clear way.

6 Discussion

6.1 Pattern Recognition in Computer Vision

In Chapters 2 and 3 we have presented three different approaches to retrieve a predefined biological shape from an input image. All approaches were based on deformable templates. The approaches were designed in such a way so that they could be used within the HTS pipeline of zebrafish embryos. The methods needed to be able to retrieve shapes that are created as a result from different experiments, were subject to different light conditions and microscope settings. We have tried different representations of the zebrafish template.

First we have tried splitting the template into n vertical (to the zebrafish median axis) sub-templates that represent the boundary of the object. Subsequently the sub-templates were found one by one. As preprocessing step we have used an edge based representation. The edges were matched to each sub template. A mayor drawback was a high computational complexity due to a high amount of pixel comparisons within the image matrix space. This complexity could be reduced by applying a pyramid approach (lower resolution images) and introducing constraints to the location and orientation of the embryos within the image. One of the constraints that reduces computation time would be positioning the individuals with their head located left and their tail in the right part of the image in relation to the starting point. Since the zebrafish embryos differed in length, the size constraints of sub templates would complicate the analysis of differently shaped individuals.

Accordingly, we have tried using a variable number of sub templates, yet still with a fixed thickness. The prototype template that we proposed was represented as a bitmap that defines size limits for each sub template. It can easily be adapted to the needs of the application while the same algorithm is used.

Instead of looking at the edge data we considered filled contours. The same drawback as with the previous representation was that the shape was represented as a fixed maximal and minimal thickness at different shape locations, which excludes shapes that are slightly bigger or smaller from being matched.

We also introduced a shape normalization step. This was done by a straightening of the retrieved deformed shape based on the prototype template shape. This normalization step enabled comparison of differently deformed zebrafish embryos. However, due to the fixed thickness of sub-templates spatial analysis within individuals that differed in length were still difficult to compare.

In the third approach we represented the individuals based on their characteristic regions, rather than dividing into sub templates. This allowed for a better recognition of the individuals. The regions were found based on anchor points and during the search a higher order representation was used instead of pixel data. In this manner the computation time was reduced significantly and the problem of connected shapes was solved. Additionally, the constraint of locating embryos from left to right was lifted. Since the regions are recognized it is possible to do a more in depth analysis on certain regions of interest and compare the different shapes regardless of their deformation or length difference.

In order to localize a deformed template within the image space we examined a Genetic

Algorithm and a Dynamic Programming approach. First choice was the Genetic Algorithm due to its possibility to search for an optimal solution in large search spaces. This approach was feasible, yet in some cases algorithm could be trapped in a local optimum and was fairly slow due to high number of variables that were adjusted. In the third approach straightforward depth first graph searching could be used. This was motivated by the fact that the search graphs were relatively simple and thus no optimization was required. In cases where the graph is more complex a Dynamic Programming or another similar approach can be applied; such could be the case when images contain a large amount of different shapes.

The final version of the algorithm we propose does not rely on initial localization of the shape and therefore does not require any manual intervention or analysis. The framework can be easily adapted to work with other shapes, in the life sciences or in other fields that require accurate and robust shape retrieval. Further analysis of the object and the straightening thereof is part of future work.

A comparison of the result for the same image using the three algorithms is given in Figure 6.1.

6.2 Software Development

We have developed *ZFA*, a software solution that includes the interface and a processing unit, based on the algorithm that is presented in Chapter 3 (Approach 3).

Our solution is successful at recognizing the zebrafish larva shape and analyzing the infection within the larvae and is robust as it can be adapted to recognizing other shapes. The measurements that are performed on the infection are: localization, infection cluster size, infection cluster amount and average infection cluster intensity.

ZFA has proven its value as a software solution for the analysis in a HT screening pipeline.

6.3 Statistical Analysis

We have used the framework that we have developed for automated granuloma cluster recognition in order to analyze the spatial distribution in zebrafish larvae. As a proof of concept we have analyzed the data for the zebrafish larva infected with the wild-type *Mycobacterium marinum* (MM) and some of its mutants.

As a first assessment we compared the behavior of granuloma of the wild-type bacteria to the behavior of the mutant 714 strain. Mutant 714 was chosen as it was one of the mutants that did not make the fish ill and this assessment was used as a proof of concept for further investigation of other mutants. A statistical analysis on the spread of bacteria was performed and information on the spread of granuloma was derived. Since the MM tends to make the fish more sick it makes sense that overall more granuloma clusters are found in the wild-type infected fish. However, if we look at the normalized spread of infection it behaves approximately the same; it either stays at the site of injection or it moves towards the head of the larva. For the *Mycobacterium marinum* it seems that

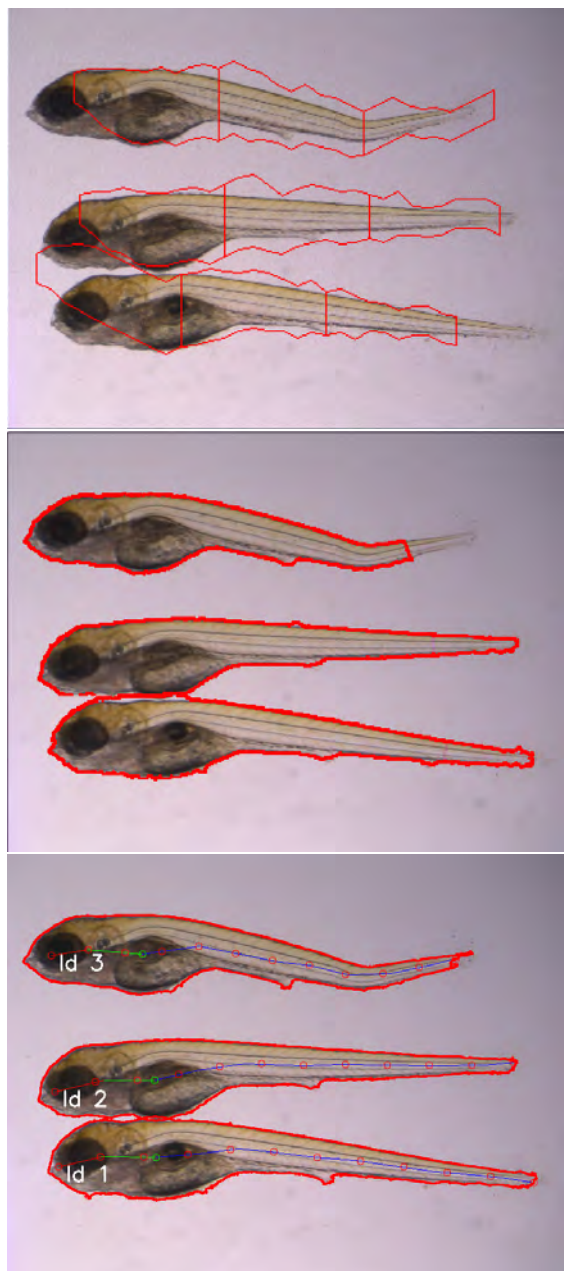


Figure 6.1: Shape retrieval results. The same input image is used for all three approaches (Approach 1 till 3 from top to bottom).

the infection is likely to migrate towards the head compared to the 714 mutant; in the 714 mutant it is established that the majority of the infection stays at the injection site.

In the second assessment we compared a large dataset of different mutant strains to the wild-type and to each other. Moreover, other measurement parameters could be considered in the analysis due to a more precise approach for retrieving the shape and its regions. Locations of interest such as the heart region and the injection point region could be retrieved. However, not all the mutants could be directly compared to the wild-type as they were not normally distributed. Those that were have been compared to each other and to the wild-type. Detailed results are provided in Section 5.4.

6.4 Conclusion

Here we present our final conclusions regarding our study of automation for High Throughput at the organismal level. We have applied our method to an important model organism, the zebrafish. With the development of our approach we made it possible to gain useful biological knowledge that used to be gathered manually in a High throughput fashion, which makes it a key activity within the field of bio-informatics.

We have shown and compared different deformable template based approaches that have been used for the pattern recognition step in the software. We have made it possible to automatically localize and annotate the shapes of the zebrafish larvae. The silhouettes that were identified serve as a mask to measure and analyze the fluorescent signal corresponding with the bacteria; if multiple fluorescent channels are used, other analysis can be accomplished with the same mask. In the case of zebrafish larvae, the current approach is sufficient for an analysis on spread through global regions in the zebrafish. However, in order to do an even more in depth analysis on infection spread, localization of certain organs within the fish or the determination of zebrafish age, we would require more characteristics than the silhouette. A silhouette representation does not suffice for that purpose since it is only describing the outside of the organism while inner information is left out. In this case an analysis of the brightfield signal within each mask can contribute with informative features. As an example we have experimented with a framework for the automated determination of the developmental stage of zebrafish embryos. A machine learning approach uses both silhouette and brightfield data from within the mask for feature extraction. In a preliminary setup we have set up a machine learning pipeline taking into account all signal in the brightfield image; the results are promising and indicate the feasibility of the approach (cf. Nezhinsky et al. [41], Figure 6.2).

We have discussed the construction and the evaluation of a workflow which we have embedded in a user interface to support the work with the algorithms described in this thesis in a high-throughput setting. So, during the development process prototypes of our software were already used for a real life problem. This evolutionary approach turned out to be successful. The reason for this successful design lies in the fact that evaluation and requirements were obtained from the intended user group which was involved in the development from the beginning and who were the intended end-users.

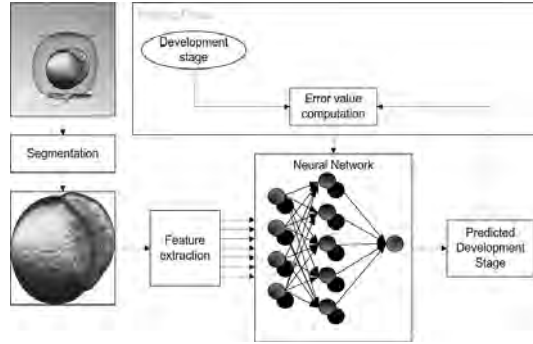


Figure 6.2: Overview of the proposed framework for automatic zebrafish embryo developmental stage determination.

The application of our software is described with a real life case study involving *Mycobacterium marinum* infection. The resulting data was analyzed for infection patterns. These patterns were related to the annotated areas within the zebrafish and therefore previously this kind of analysis was hampered due to the limitations given by manual analysis. With our algorithms the images could be processed in a High Throughput fashion and thus large amount of features could be generated from the input images. Using statistics on these features we were able to discover interesting differences in the organization and amounts of granuloma clusters for different bacterial mutants. By obtaining these results we have shown the need for such automation processes and smart software for these kind of screening applications.

Bibliography

- [1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford Univ. press, 1996.
- [2] B. A. Barut, L. I. Zon, Realizing the Potential of Zebrafish as a Model for Human Disease, *Physiological Genomics* 2 (2000) 49–51.
- [3] J. S. Blackburn, S. Liu, A. R. Raimondi, M. S. Ignatius, C. D. Salthouse, D. M. Langenau, High-throughput imaging of adult fluorescent zebrafish with an LED fluorescence macroscope, *Nature Protocols* 6 (2011) 229–241.
- [4] E. Borestein, E. Sharon, S. Ullman, Combining Top-down and Bottom-up Segmentation, *Presceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshop on Perceptual Organization in Computer Vision* (2004) 46.
- [5] G. Borgefors, Distance Transformations in Digital Images, *Computer Vision, Graphics, and Image Processing* 34/3 (1986) 344–371.
- [6] G. Borgefors, R. Strand, An Approximation of the Maximal Inscribed Convex Set of a Digital Image, *ICAIIP* (2005) 438–445.
- [7] P. J. H. Bronkorsta, M. J. T. Reinders, E. A. Hendriks, J. Grimbergen, R. M. Heethaar, G. J. Brakenhoff, On-line Detection of Red Blood Cell Shape using Deformable Templates, *Brakenhoff Pattern Recognition Letters* 21(5) (2000) 413–424.
- [8] R. Carvalho, J. de Sonnevile, O. W. Stockhammer, N. D. L. Savage, W. J. Veneman, T. H. M. Ottenhoff, R. P. Dirks, A. H. Meijer, H. P. Spaink, A High-Throughput Screen for Tuberculosis Progression, *PLoS One* 6(2).
- [9] T. F. Cootes, C. J. Taylor, D. H. Cooper, J. Graham, Active Shape Models - Their Training and Application, *Comput. Vision and Image Understanding* 61/1 (1995) 38–59.
- [10] T. F. Cootes, C. J. Taylor, A. Lanitis, Active Shape Models: Evaluation of a Multi-Resolution Method for improving Image Search, *Proceedings of the British Machine Vision Conference* 1 (1994) 327–336.
- [11] C. L. Cosma, K. Klein, R. Kim, D. Beery, L. Ramakrishnan, Mycobacterium Marium Erp is a Virulence Determinant required for Cell Wall Integrity and Intracellular Survival, *Infection and immunity* 74(6) (2006) 3125–3133.

- [12] J. Cousty, G. Bertrand, L. Najman, M. Couprie, Watershed Cuts: Minimum Spanning Forests and the Drop of Water Principle, *Transactions on Pattern Analysis and Machine Intelligence*. 31 (2009) 1362–1374.
- [13] O. Cuisenaire, B. Macq, Fast and Exact Signed Euclidean Distance Transformation with Linear Complexity, *IEEE International Conference on Acoustics, Speech and Signal Processing* 6 (1999) 3293–3296.
- [14] P. Danielsson, Euclidean distance mapping, *Comput. Graph. Image Process.* 14 (1980) 227–248.
- [15] R. Fabbri, L. D. F. Costa, J. C. Torelli, O. M. Bruno, 2D Euclidean Distance Transform Algorithms: A Comparative Survey, *ACM Computing Surveys* 40 (2008) 1–44.
- [16] P. F. Felzenszwalb, Representation and Detection of Shapes in Images, Ph.D. thesis, MIT, 2003.
- [17] N. Funobiki, M. Isogai, An Eye-Contour Extraction Algorithm from Face Image using Deformable Template Matching, *Mem. of the Fac. of Eng.* 40 (2006) 78–87.
- [18] A. Garrido, N. P. de la Blanca, Applying Deformable Templates for Cell Image Segmentation., *Pattern Recognition* 33/5 (2000) 821–832.
- [19] S. George, T. Xia, R. Rallo, Y. Zhao, Z. Ji, S. Lin, X. Wang, H. Zhang, B. France, D. Schoenfeld, R. Damoiseaux, R. Liu, S. Lin, K. A. Bradley, Y. Cohen, A. E. Nel, Use of a High-Throughput Screening Approach Coupled with In Vivo Zebrafish Embryo Screening To Develop Hazard Ranking for Engineered Nanomaterials, *ACS Nano* 5/3 (2011) 1805–1817.
- [20] R. Gerlai, High-Throughput Behavioral Screens: the First Step towards Finding Genes involved in Vertebrate Brain Function using Zebrafish, *Molecules* 15.4 (2010) 2609–2622.
- [21] R. Gonzales, R. Woods, *Digital Image Processing*, Addison-Wesley, London, 2nd edn., 2001.
- [22] X. He, R. S. Zemel, D. Ray, Learning and Incorporating Top-Down Cues in Image Segmentation, *Lecture Notes in Computer Science* 3951 (2006) 338–351.
- [23] A. K. Jain, Y. Zhong, M. P. Dubuisson-Jolly, Deformable Template Models: A Review, *Signal Processing* 71/2 (1998) 109–129.
- [24] A. K. Jain, Y. Zhong, S. Lakshmanan, Object Matching Using Deformable Templates, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18/3 (1996) 267–278.

- [25] C. A. Karlovich, R. M. John, L. Ramirez, D. Y. Stainier, R. M. Myers, Characterization of the Huntington's Disease (HD) Gene Homologue in the Zebrafish *Danio rerio*, *Gene* 217(1-2) (1998) 117–25.
- [26] M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active Contour Models, *International Journal of Computer Vision* 1/4 (1987) 321–331.
- [27] C. Kervrann, F. Heitz, A Hierarchical Static Framework for the Segmentation of Deformable Objects in Image Sequences, *IEEE Computer Vision Pattern Recognition* (1994) 724–728.
- [28] H. Y. Kim, S. A. de Araujo, Grayscale Template-Matching Invariant to Rotation, Scale, Translation, Brightness and Contrast, *IEEE Pacific-Rim Symposium on Image and Video Technology, Lecture Notes in Computer Science* 4872 (2007) 100–113.
- [29] F. Krolupper, J. Flusser, Polygonal Shape Detection for Recognition of Partially Occluded Objects, *Pattern Recognition Letters* 28 (2007) 1002–1011.
- [30] B. Leroy, I. Herlin, L. Cohen, Multi-resolution Algorithms for Active Contour Models, *ICAOS* (1996) 58–65.
- [31] A. Levin, Y. Weiss, Learning to Combine Bottom-Up and Top-Down Segmentation, *Lecture Notes in Computer Science* 3954 (2006) 581–594.
- [32] J. P. Lewis, Fast Template Matching, *Vision Interface* 95 (1995) 120–123.
- [33] L. Liu, S. Sclaroff, Region Segmentation via Deformable Model-Guided Split and Merge, *Boston University Computer Science Technical Report* 241 (2000) 98–104.
- [34] R. Liu, S. Lin, R. Rallo, Y. Zhao, R. Damoiseaux, T. Xia, S. Lin, A. Nel, Y. Cohen, Automated Phenotype Recognition for Zebrafish Embryo Based In Vivo High Throughput Toxicity Screening of Engineered Nano-Materials, *PLoS ONE* 7(4) (2012) 1.
- [35] M. R. Loken, *Immunofluorescence Techniques in Flow Cytometry and Sorting*, Wiley, 1990.
- [36] D. R. Love, F. B. Pichler, A. Dodd, B. R. Copp, D. R. Greenwood, Technology for High-throughput Screens: the Present and Future using Zebrafish, *Current opinion in biotechnology* 15/6 (2004) 564–571.
- [37] D. G. Lowe, Distinctive Image Features from Scale-invariant Keypoints, *International Journal of Computer Vision* 60(2) (2004) 91–110.
- [38] L. Lucchesse, S. K. Mitra, Color Image Segmentation: A State-of-the-Art Survey, *Proceedings of the Indian National Science Academy* 67(2) (2001) 207–221.
- [39] B. L. Miller, D. E. Goldberg, Genetic Algorithms, Tournament Selection, and the Effects of Noise, *Complex Systems* 9 (1995) 193–212.

- [40] M. Monici, Cell and Tissue Autofluorescence Research and Diagnostic Applications, *Biotechnol Annu.* 11 (2005) 227–56.
- [41] A. E. Nezhinsky, I. Martorelli, F. J. Verbeek., Detection of Developmental Stage in Zebrafish Embryos in a high throughput processing environment, *Proceedings 20th Annual Belgian-Dutch Conference on Machine Learning* (2011) 97/98.
- [42] A. E. Nezhinsky, E. J. M. Stoop, A. A. Vasylevska, A. van der Sar, F. J. Verbeek, Spatial Analysis of Bacterial Infection Patterns in Zebrafish, *Benelearn* 21.
- [43] A. E. Nezhinsky, F. J. Verbeek, Efficient and Robust Shape Retrieval from Deformable Templates, Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies, *Lecture Notes in Computer Science* 7610 (2012) 42–55.
- [44] H. P. Ng, S. H. Ong, P. S. Goh, K. W. C. Foong, W. L. Nowinski., Template-based Automatic Segmentation of Masseter Using Prior Knowledge, *IEEE Southwest Symposium on Image Analysis and Interpretation 2006* (2006) 208–212.
- [45] R. Nohre, Deformed Template Matching by the Viterbi Algorithm, *Pattern Recognition Letters* 17/14 (1996) 1423–1428.
- [46] N. Otsu, A Threshold Selection Method from Gray Level Histogram, *IEEE Transactions on Systems Man Cybernet* 9 (1975) 62–66.
- [47] H. Peng, F. Long, X. Liu, S. K. Kim, E. W. Myers, Straightening *Caenorhabditis elegans* Images, *Bioinformatics* 24/2 (2008) 234–242.
- [48] A. Ramirez, Circle Detection on Images using Genetic Algorithms, *Pattern Recognition Letters* 27/6 (2006) 652–657.
- [49] C. B. Rohde, M. F. Yanik, Subcellular In vivo Time-lapse Imaging and Optical Manipulation of *Caenorhabditis elegans* in Standard Multiwell Plates, *Nature Communications* 2 (2011) 271.
- [50] H. Samet, M. Tamminen, Efficient Component Labeling of Images of Arbitrary Dimension Represented by Linear Bintrees, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10/4 (1988) 579–586.
- [51] J. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics Computer Vision and Material Sciences.*, Cambridge University Press, 1999.
- [52] L. Shapiro, G. Stockman, *Computer Vision*, Prentice Hall, 2002.
- [53] G. E. Sotak, K. L. Boyer, The Laplacian-of-Gaussian Kernel: a Formal Analysis and Design Procedure for Fast, Accurate Convolution and Full-frame Output, *Computer Vision, Graphics, and Image Processing* 48/2 (1989) 147–189.

- [54] E. J. M. Stoop, T. Schipper, S. K. R. Huber, A. E. Nezhinsky, F. J. Verbeek, S. S. Gurcha, G. S. Besra, C. M. J. E. Vandenbroucke-Grauls, W. Bitter, A. M. van der Sar, Zebrafish Embryo Screen for Mycobacterial Genes Involved in Granuloma Formation reveals a Novel ESX-1 Component, *Disease Model Mechanisms* 4 (2011) 526–536.
- [55] E. J. M. Stoop, Bacterial Factors involved in Tuberculosis Infection, Ph.D. thesis, VU, 2013.
- [56] J. E. Sulston, E. Schierenberg, J. G. White, J. N. Thomson, The Embryonic Cell Lineage of the Nematode *Caenorhabditis Elegans*, *Developmental Biology* 100 (1983) 64–119.
- [57] S. Svensson, G. Borgefors, I. Nystrom, On Reversible Skeletonization Using Anchor-Points from Distance Transform, *Journal of Visual Communication and Image Representation* 10 (1999) 379–397.
- [58] H. D. Tagare, C. C. Jaffe, J. Duncan, Medical Image Databases a Content-based Retrieval Approach, *Journal of the American Medical Informatics Association* 4/3 (1997) 184–198.
- [59] W. Tao, H. Jin, Y. Zhang, Color Image Segmentation based on Mean Shift and normalized Cuts, *IEEE Transactions on Systems, Man, and Cybernetics* 37/5 (2007) 1382–1389.
- [60] E. Valveny, E. Marti, Deformable Template Matching within a Bayesian Framework for Handwritten Graphic Symbols Recognition, 3rd IAPR International Workshop on Graphics Recognition 1 (1999) 243–250.
- [61] F. J. Verbeek, P. J. Boon, High-resolution 3D Reconstruction from Serial Sections: Microscope Instrumentation, Software Design, and its Implementations, *International Symposium on Biomedical Optics. International Society for Optics and Photonics* 4621 (2002) 65–76.
- [62] A. Vogt, A. Cholewinski, X. Shen, S. G. Nelson, J. S. Lazo, M. Tsang, N. A. Hukriede, Automated Image-based Phenotypic Analysis in Zebrafish Embryos., *Developmental dynamics* 238/3 (2009) 656–663.
- [63] H. E. Voikman, H. Clay, D. Beery, J. C. Chang, D. R. Sherman, L. Ramakrishnan, Tuberculous Granuloma Formation is enhanced by a Mycobacterium Virulence Determinant, *PLoS biology* 2/11 (2004) 1946.
- [64] S. L. Walker, J. Ariga, J. R. Mathias, V. Coothankandaswamy, X. Xie, M. Distel, R. W. Koster, M. J. Parsons, K. N. Bhalla, M. T. Saxena, J. S. Mumm, Automated Reporter Quantification In Vivo: High-Throughput Screening Method for Reporter-Based Assays in Zebrafish, *PLoS ONE* 7(1) (2012) 1.

- [65] Y. N. Wu, Z. Si, C. Fleming, S. C. Zhu, Deformable Template as Active Basis, International Conference on Computer Vision 11 (2007) 1–8.
- [66] J. Yao, N. Khanna, P. Grogono, Fast Robust GA-Based Ellipse Detection, ICPR 2 (2004) 859–862.
- [67] J. You, E. Pissaloux, W. P. Shu, H. A. Cohen, Efficient Image Matching: A Hierarchical Chamfer Matching Scheme via Distributed Systems, Real Time Imaging 1 (1995) 245–259.
- [68] S. U. Yu, J. Shi, Object-Specific Figure-Ground Segregation, Computer Vision and Pattern Recognition (2003) 39–45.
- [69] H. Zhao, J. Zhou, A. Robles-Kelly, J. Lu, J. Yang, Automatic Detection of Defective Zebrafish Embryos via Shape Analysis, Digital Image Computing: Techniques and Applications (2009) 431–438.
- [70] Y. Zhong, A. K. Jain, Object Localization using Color, Texture and Shape, Pattern Recognition 33 (2000) 671–684.

Dutch Summary

In de levenswetenschappen wordt gebruik gemaakt van zogenaamde High Throughput methoden; deze bestaan uit experimenten waarbij een grote hoeveelheid data wordt gegenereerd om zo bepaalde eigenschappen te kunnen vaststellen. De uitvoering van deze methoden verschilt per toepassing. In dit proefschrift wordt met name aandacht besteed aan de High Throughput methoden waarin een effect wordt gemeten aan de hand van beelden van objecten, high-throughput betekent dat er grote hoeveelheden beelden worden gegenereerd en er dus veel objecten moeten worden herkend om uiteindelijk een uitspraak te kunnen doen over eigenschappen van deze objecten.

Een voorbeeld van een High Throughput proces is afgebeeld in Figuur 1.1. Het specifieke doel van het onderzoek beschreven in dit proefschrift is het vaststellen van robuuste methoden voor patroon herkenning binnen een High Throughput analyse, deze methoden te onderzoeken alsmede te valideren. Binnen het onderzoek beschreven in dit proefschrift wordt Patroon Herkenning gezien in zowel de context van object herkenning als ook in de context van data mining en beide aspecten worden in dit proefschrift aangehaald.

We hebben reeds aangegeven dat de High-Throughput context in dit onderzoek vooral veel beelden gegenereerd, deze beelden, microscoop beelden in het bijzonder, zijn dan de input voor analyse. Het is zaak in deze beelden de objecten kunnen herkennen en analyseren. In het veld van de beeldanalyse wordt daarvoor segmentatie gebruikt; dit is de verdeling van een beeld in interessante objecten en de achtergrond. Wij als mensen segmenteren doorlopend, dat wil zeggen scheiden van voor- en achtergrond, voor een computer is dat echter veel lastiger. Dit moet worden geleerd. Segmentatie technieken kunnen worden gescheiden in twee hoofdgroepen: bottom-up segmentatie versus top-down segmentatie. Bottom up segmentatie is gebaseerd op het vinden van gelijkwaardige eigenschappen van de objecten of de achtergrond om vervolgens deze van elkaar te scheiden. Echter, vaak is een scheiding van een beeld in de voor-en achtergrond regio's niet voldoende om het voorwerp van specifieke interesse te vinden. Daarom is daarnaast een aanpak vereist die gebaseerd is op patroon herkenning, een zogenaamde top-down benadering.

In hoofdstuk 2 presenteren we twee benaderingen voor de High-Throughput context die is bestudeerd in het kader van dit proefschrift: i.e. het segmenteren en annoteren van de objecten die onderwerp van het experiment zijn. Voor toepassing wordt gebruik gemaakt van de zebravis als model systeem, en daarom leggen we de nadruk op beelden die zebravis embryos bevatten. We onderscheiden twee stappen: een voorbewerkingstap waarbij de achtergrond wordt gescheiden van de voorgrond (bottom-up) en de Template-Matching stap, dat bestaat uit het zoeken naar een vervormbaar sjabloon in het voorbewerkte beeld (top-down), een zogenaamde Deformable Template Matching

methode. Voor deze Deformable Template Matching stap beschrijven we twee methoden die gebaseerd zijn op het silhouet van een gemiddeld object als een sjabloon. Vervolgens nemen we aan dat het beeld een of meerdere objecten bevat die een misvormde of verschoven afbeelding zijn van het sjabloon en we beschrijven een functie waarmee wordt aangegeven in hoeverre een geïdentificeerde vorm in het beeld overeenkomt met het vervormde sjabloon. Het zoeken naar een correcte deformatie is een complex proces, daarom passen we verschillende vormen van optimalisatie toe. Tevens bestuderen we de mogelijkheden om het objecten die als resultaat gevonden worden te normaliseren om zo een beter vergelijkbaar resultaat te verkrijgen. Het algoritme is geëvalueerd met zowel synthetische beelden als met beelden van zebravis embryos.

In hoofdstuk 3 presenteren we een methode van Deformable Template Matching die op een andere wijze verder is uitgewerkt. Het object waar we naar zoeken wordt nu, in plaats van een silhouet, gerepresenteerd door middel van zogenaamde ankerpunten. Dit is robuster en meer natuurgetrouw, omdat de vorm die gezocht wordt niet meer een van te voren vastgestelde grootte dient te hebben, maar als een schaal onafhankelijke beschrijving wordt gerepresenteerd. De voorgestelde methode kan volledig worden geautomatiseerd, en blijkt sneller en nauwkeuriger dan de methodes die in hoofdstuk twee zijn gepresenteerd. We hebben het algoritme geëvalueerd met beelden van zebravis en xenopus embryos.

In hoofdstuk 4 beschrijven we hoe de beschreven algoritmen zijn geïntegreerd een software pakket, i.e. ZFA. Dit pakket wordt gebruikt voor de analyse van High-Throughput data. We presenteren een interface die aansluit bij de workflow van de gebruikers. De gebruikers zijn in dit geval onderzoekers in de levenswetenschappen. De verschillende processen die vanuit de workflow ontstaan zijn, zijn in de software opgenomen als componenten: de voorbewerking, de interface, de segmentatie en de gegevensverwerking. Het High-Throughput toepassingsgebied waar de software wordt gebruikt is het onderzoek naar het doorgronden van infectie mechanismen die ten grondslag liggen aan Tuberculose. Tuberculose is een ernstige ziekte en een aanzienlijk deel van de wereldbevolking is geïnfecteerd. Het is echter nog steeds moeilijk om een effectieve behandeling te vinden vanwege de resistentie van bacteriën. Tuberculose wordt veroorzaakt door de bacterie - *Mycobacterium tuberculosis*. Voor het onderzoek naar infectie mechanismen wordt echter gebruik gemaakt van een nauw verwante soort - *Mycobacterium marinum*. Door gebruik te maken van een model organisme, de zebravis, kan het infectieproces worden bestudeerd in een groot aantal individuen. De zebravisjes moeten dan in beelden worden geïdentificeerd en vervolgens verder worden geanalyseerd. Onze software doet dat.

In hoofdstuk 5 worden de resultaten vergeleken die verkregen zijn uit het toepassen van ZFA in een lopend onderzoek betreffende de infectie van zebravissen met *Mycobacterium marinum*. Een dergelijke infectie wordt gekenmerkt door de aanwezigheid van zogenaamde granulomen. Granulomen zijn clusters van immuuncellen en bacteriën die wijzen op succesvolle infectie van de bacterie. In het onderzoek worden de bacteriën gevisualiseerd met fluorescent label. Om meer inzicht te krijgen op de progressie van de infectie is het nodig om de spreiding ervan te analyseren in de gastheer, de zebravis, gedurende een bepaalde periode. De resultaten worden onder meer gebruikt om vast te

stellen welke genen verantwoordelijk zijn voor infectie. Eerst wordt, via het algoritme uit hoofdstuk 3, de vorm van de zebravis embryo's gelokaliseerd. Vervolgens worden verschillende kenmerken van de infectie geanalyseerd door het deel van de software dat verantwoordelijk is voor de gegevensverwerking. Onze aanpak maakte het mogelijk om langs geautomatiseerde weg het gedrag van de infectie per zebravis te analyseren. Om te bepalen welke genen van *Mycobacterium marinum* zijn betrokken bij vorming van granulomen zijn 1000 willekeurige mutanten van deze bacteriën geanalyseerd en daarbij zijn de mutanten geselecteerd die zebravis larven niet ziek maakten. Op deze manier zijn 30 mutanten geïdentificeerd. We hebben onderzocht of er een patroon aanwezig is in de spreiding of grootte van granuloma clusters in bepaalde weefsels en hoe het gedrag verschilt per mutant ten opzichte van larven besmet met de wild-type bacteriën.

Tenslotte worden in hoofdstuk 6 resultaten van de voorgaande hoofdstukken samengevat. We beginnen met de segmentatie stap (Pattern Recognition in Images). We hebben drie verschillende methoden beschreven. We vergelijken de prestaties van de drie methoden. Alle methoden werden getest op dezelfde dataset. Resultaten laten een verbetering in de prestaties zien. Het Ankerpunt gebaseerde (hoofdstuk 3) methode gaf de beste resultaten. Dit algoritme zal daarom worden gebruikt voor de analyse en is deel van de software.

Russian Summary

В биологических исследованиях часто используют методику High Throughput, позволяющую с помощью большой серии экспериментов находить определенные характеристики биологических объектов. Конкретная реализация этой методики зависит от конкретных требований. В нашей работе мы применили методику High Throughput для измерения связанных с изображениями интересующих нас объектов.

Пример использования методики High Throughput приведён на рисунке 1.1.

Конкретной целью исследования в данной работе является создание ряда надежных методов Pattern Recognition в контексте методики High Throughput. Pattern Recognition можно рассматривать как распознавание объектов в изображении но также как Data Mining. Решения обеих задач детально описаны.

В нашем исследовании методом High Throughput обрабатывались изображения, полученные с помощью микроскопа. Изображения необходимо было и выделять (отделить от фона и случайных помех) и анализировать. Для человека разделение объектов переднего плана и фона (сегментация) несложно. Для компьютера же, - это нетривиальная задача. Методы сегментации могут быть разделены на две основные группы: снизу вверх (Bottom Up) методика основана на обнаружении эквивалентных свойств объектов и фона, что позволяет их разделить, но такое разделение не всегда достаточно, поэтому используется ещё и подход сверху вниз (Top Down), основанный на распознавании образов.

В главе 2 мы представляем два подхода к методике High Throughput: то есть, сегментацию и выделение объектов участвующих в эксперименте. В качестве реальных объектов были использованы рыбки данио (*Danio rerio*), т.е. осуществлялся поиск изображений эмбрионов рыбок. Мы выделяем два этапа: предварительная обработка, при которой фон отделяется от переднего плана (снизу вверх) и, второе, Template-Matching шаг, который заключается в нахождении деформируемого шаблона (сверху вниз). Для выбора Template-Matching шага были предложены два метода, опирающихся на выбор в качестве шаблона усреднённого силуэта объекта. При этом предполагалось, что изображение формируют один или несколько деформированных или смещенных объектов.

Поиск подходящих деформаций представляет из себя сложный вычислительный процесс, в котором используются различные виды оптимизаций. Также мы рассматриваем способы нормализации полученных результатов для возможности лучшего сравнения большего количества данных.

В главе 3 предложена дальнейшая разработка метода построения деформируемого шаблона. Поиск объект производится с помощью так называемых опорных точек вместо силуэта. Это способ более надежен и достоверен, т.к. исчезает необходимость

фиксировать размеры шаблона. Предложенный способ полностью автоматизирован, работает быстрее и точнее, чем методы предложенные ранее. Мы протестировали алгоритм, используя изображения эмбрионов рыбок данио и эмбрионов лягушки.

В главе 4 описан программный пакет ZFA, который может в контексте High Throughput анализировать изображения. Для разных пользователей представлен интерфейс соответствующих рабочих процессов.

Программный пакет включает: предварительную обработку, интерфейс, сегментацию и обработку данных. Предварительная обработка предусматривает выделение флуоресцентных и обычных изображений областей.

Интерфейс предоставляет возможность выбрать примеры, помещает изображения в выбранные папки. Форма эмбрионов данио локализована и доведена до алгоритма вычисления функции деформируемого шаблона. После определения формы предложен анализ характеристик различных инфекций. Требования к ZFA мы выбирали исходя из практических нужд. Известно, что туберкулез является одним из самых серьезных и распространённых заболеваний. Эффективное лечение его по-прежнему затруднено из-за его бактериальной устойчивости.

Поведение бактерии туберкулеза можно моделировать изучая поведение его близкого родственника - *Mycobacterium marinum*. При помощи использования данио как модель становится возможным отслеживать процесс инфекции. Для этого нужно локализовать рыбки данио и анализировать распределение инфекции в них. Наш программный пакет делает это возможным.

В главе 5 применяется ZFA в качестве программного обеспечения мониторинга исследования данио инфицированного бактериями *Mycobacterium marinum*. Такие инфекции характеризуется появлением гранул (скоплений иммунных клеток и бактерий). Гранулы легко визуализировать флуоресцентными агентами. Наш подход (использование алгоритма из 3 главы) позволяет автоматизировать анализ развития во времени инфекции в теле хозяина, данио. Для того, чтобы глубже разобраться в процессе развития инфекции необходимо выявить гены ответственные за развитие инфекции. Для этого были выбраны 1000 случайных мутантов этих бактерий, а из них выбраны те мутанты при которых рыбки не заболевали. Таким образом 30 мутантов были выделены для углубленного анализа. Мы изучали, кроме того, отличаются ли гранулы по размерам и распределениям в определенных тканях инфицированные генами *Mycobacterium marinum* от инфицированных туберкулёзом.

В главе 6 мы перечисляем и обсуждаем результаты исследований, проведённых в предыдущих главах.

Это и вопросы, связанные с сегментационным шагом, распознавания объектов в изображениях, особенности предложенных трех различных методов, мы сравниваем их производительность. Тест при сопоставлении разных методов мы проводим на одном и том же наборе данных. Исследования убедительно показали, что третий метод дает наилучшие результаты и поэтому используется в пакете ZFA.

Curriculum Vitae

Alexander E. Nezhinsky was born on May 7 1982 in Leningrad, the USSR. He graduated from the Stedelijk Gymnasium in Leiden in 2001. He studied Computer Science at Leiden University and gained his Master of Computer Science degree in 2007. Additionally he followed courses at the Academy of Arts (KAVBK) in The Hague.

In August 2008 he started as PhD candidate in the section Imaging and Bioinformatics under the supervision of Dr. Ir. F. J. Verbeek. In his PhD research he investigated different image analysis and pattern recognition techniques applied to the zebrafish model.

List of publications

- A. Nezhinsky, E.J. Stoop, A.M. van der Sar & F.J. Verbeek: Numerical Analysis of Image Based High-Throughput Zebrafish Infection Screens: /Matching Meaning with Data/. In: BIOINFORMATICS 2012, Proceedings of the Int. Conf. on Bioinformatics Models, Methods and Algorithms: 257-262 (2012)
- A. Nezhinsky, E.J. Stoop, A.A. Vasylevska, A.M. van der Sar & F.J. Verbeek: Spatial Analysis of Bacterial Infection Patterns in Zebrafish. In: Proceedings 21th Annual Belgian-Dutch Conference on Machine Learning: 27-31 (2012)
- A. Nezhinsky & F.J. Verbeek: Efficient and Robust Shape Retrieval from Deformable Templates. In: T. Margaria & B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies (ISoLA 2012), Lecture Notes in Computer Science 7610: 42-55, Springer (2012)
- A. Nezhinsky, I. Martorelli & F.J. Verbeek: Detection of Developmental Stage in Zebrafish Embryos in a high throughput processing environment. In: Proceedings 20th Annual Belgian-Dutch Conference on Machine Learning: 97-98 (2011)
- E.J. Stoop, T. Schipper, Huber S.K. Rosendahl, A. Nezhinsky, F.J. Verbeek, S.S. Gurcha, G.S. Besra, C.M. Vandenbroucke-Grauls, W. Bitter & A.M. van der Sar: Zebrafish embryo screen for mycobacterial genes involved in the initiation of granuloma formation reveals a newly identified ESX-1 component, Disease Model Mechanisms : 526-536 (2011)
- A. Nezhinsky, J.W. Kruisselbrink & F.J. Verbeek: Convex Shape Retrieval from Edge Maps by the use of an Evolutionary Algorithm. In: J. Filipe H. Gamboa A. Fred (Ed.), Proceedings 1st International Conference on Bioinformatics: 221-225 (2010)
- A. Nezhinsky & F.J. Verbeek: Pattern recognition for high throughput zebrafish imaging using genetic algorithm optimization. In: 5th IAPR Conference on Pattern Recognition in Bioinformatics (PRIB 2010), Lecture Notes in Bioinformatics 6282: 302-312, Springer (2010)

Acknowledgements

Here, I would like to explicitly acknowledge gratitude to the group members of Imaging&Bioinformatics at LIACS. Thank you all for the help, support and 'gezelligheid' during my PhD time, you are all great colleagues and friends. Dome and Willems, thank you for surviving in the same room and with me and making room 108 such a cool place to be. Thank you Raf and the rest of the coffee-gang for infinite amount of caffeine and very informative talks at the coffee corner.

I would also like to show my gratitude to Walter, Johannes and Michael, thank you for interesting discussions on various subjects and the help on evolutionary algorithms.

I would like to acknowledge all the biologists with whom I collaborated in this project, especially Esther and Astrid, it was very inspiring and interesting to work with you.

I am grateful to all my friends outside of LIACS for just being my friends! Especially to those that came to my defence, even flew in from Russia to visit me during this time, thank you very much for your support, this really means a lot to me! =) Thank you Sander for interest in my thesis.

I would like to thank my parents, for always believing in me, showing interest in my work, thinking about my future and always helping out if necessary.

And last but not least, Anastasia -Nastya- my wife who is always there for me and supports me in my decisions in both hard and easy times.

