Pattern Recognition in High-Throughput Zebrafish Imaging

Nezhinsky, A.E.

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. 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.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 florescent 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

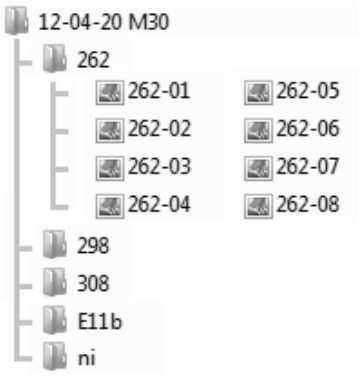

Figure 4.1: An example of data organizations as currently done for an array of experiments.

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.

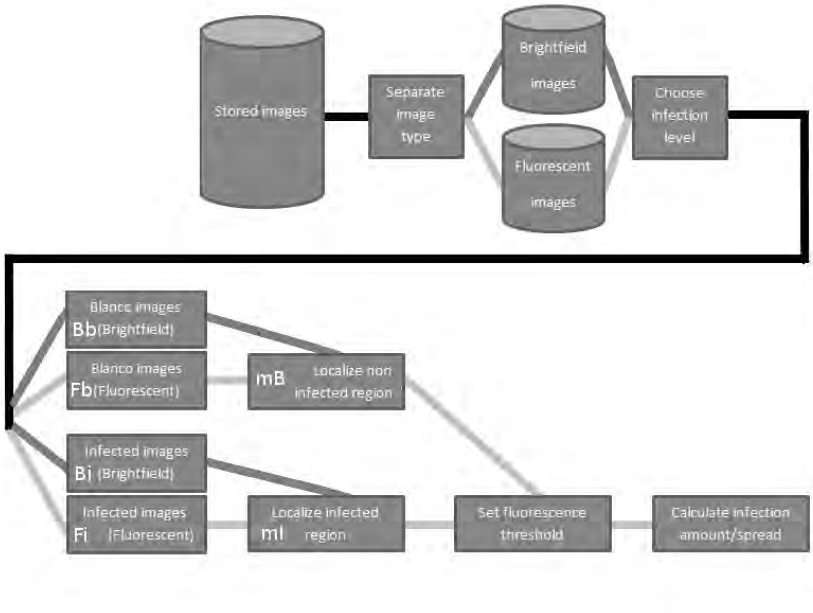

Figure 4.3: The original workflow of moving from stored images to measurements.

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 preprocessing 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 preprocessing 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

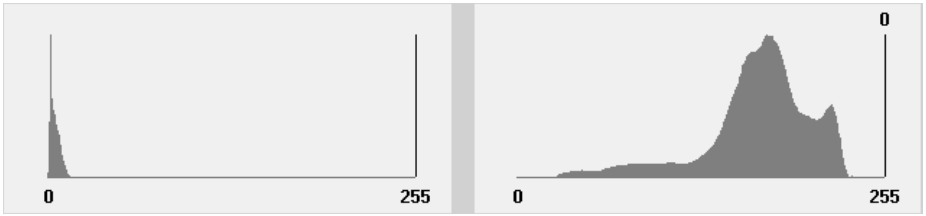

Figure 4.4: Left: histogram of a sample fluorescent image. As can be seen from the histogram all intensity values are located close to 0. Right: histogram of a sample brightfield image (converted to grayscale). As can be seen the values are more scattered through the entire interval.

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.

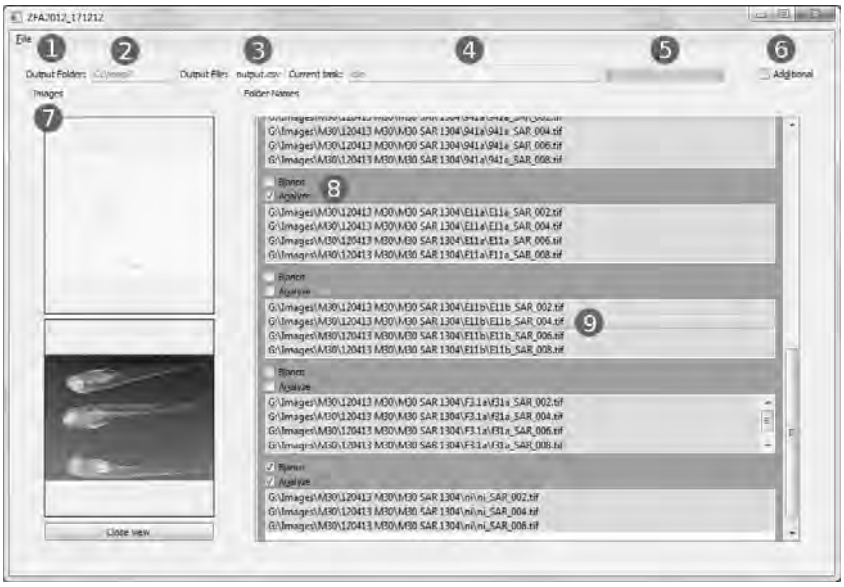

Figure 4.5: Final version of the interface component.

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

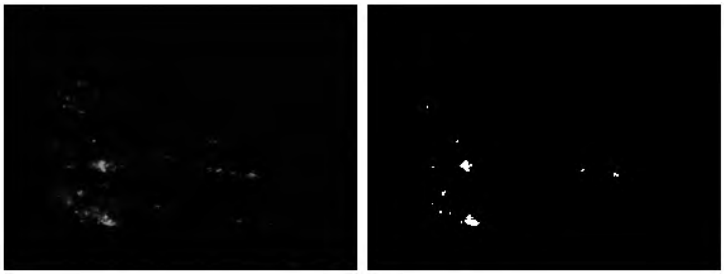

Figure 4.6: Within the mask a threshold value is set (left: fluorescent image, right: binary image as a result of thresholded left image).

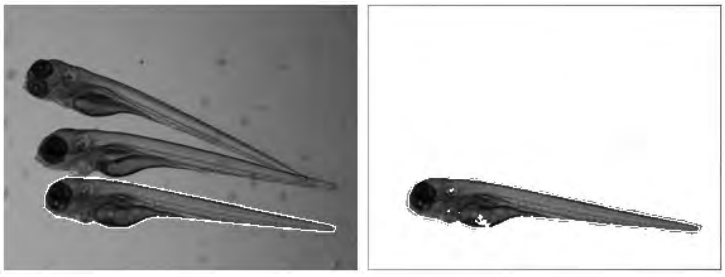

Figure 4.7: Each zebrafish embryo mask is multiplied with the corresponding thresholded fluorescent image (resulting from Figure 4.6) .

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). Users[1] 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.

[1] Biologists 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

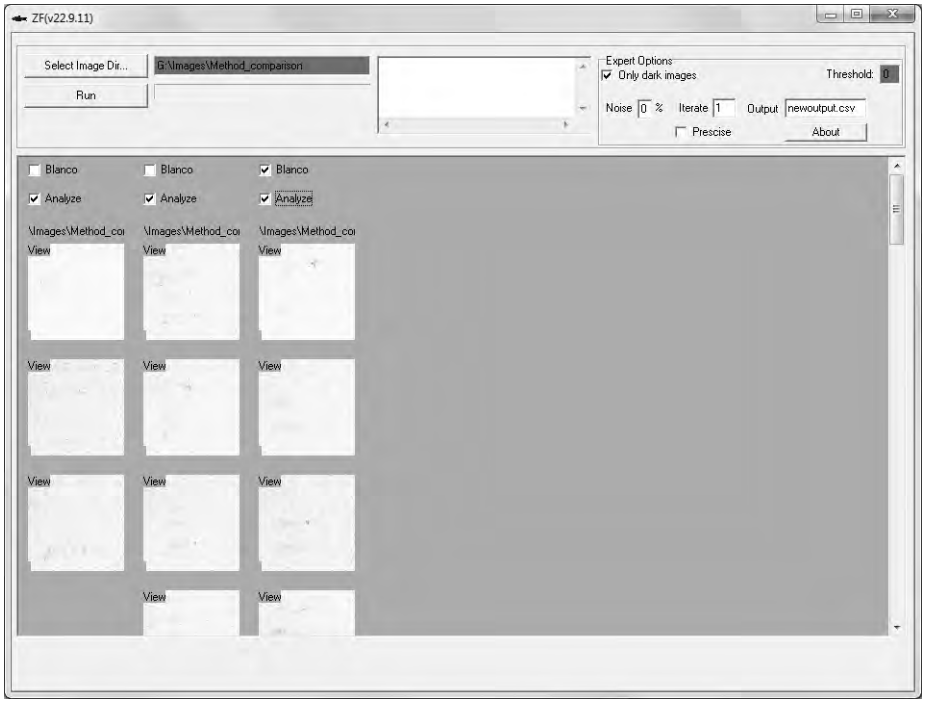

Figure 4.8: Interactive layout. Image previews are shown in complementary colors with a white background.

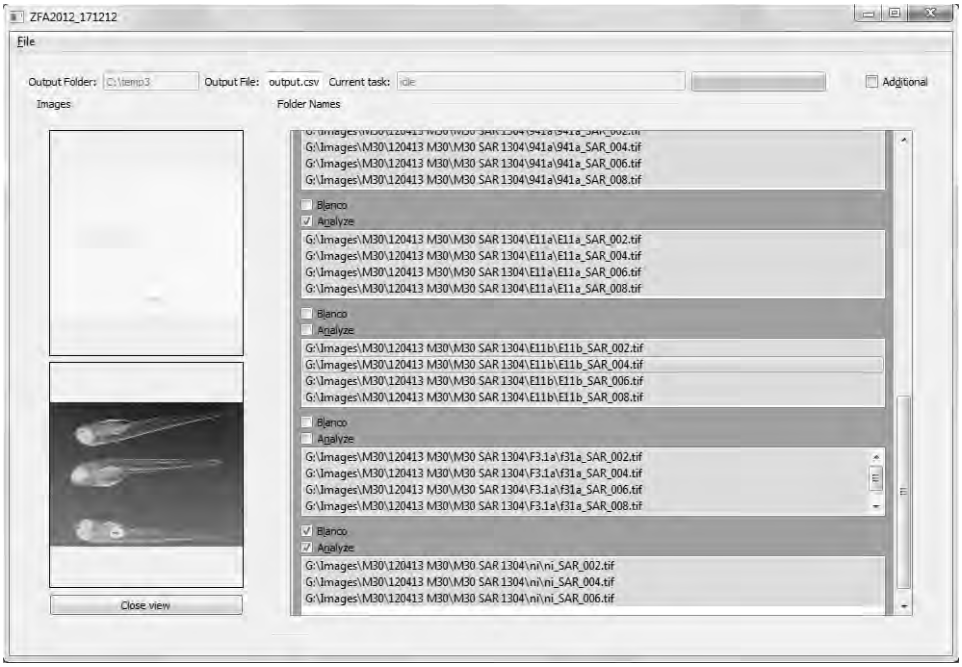

Figure 4.9: Interactive layout. Image previews are not shown. Only when an image is selected from the filename list the preview appears on the left.

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:

$$precision = \frac{tp}{tp + fp} \tag{4.4.1}$$

$$recall = \frac{tp}{tp + fn} \tag{4.4.2}$$

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.