Universiteit Leiden
The Netherlands

Bachelor Bioinformatics

Automating VAST imaging for high throughput and fluorescence

Bachelor Thesis

J. van Bijsterveld (s2902702) & S.W. van Benthum (s2922223)

Supervisors:
F.J. Verbeek & M. Lamers

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

03/08/2023

Abstract

Zebrafish are, because of their distinct features, easy to use and often used model organisms for biological research. To help decrease imaging time for zebrafish larvae, researchers have developed the Vertebrate Automated Screening Technology (abbreviated to *VAST*). This paper describes a method for zebrafish imaging aimed at high-throughput, high-quality imaging using the VAST.

The created method uses microscope control through an open source micromanager implementation and VAST control through Maarten Lamers' VAI-box and the VAST applications' external imaging functionality to create a two-camera setup for zebrafish imaging. The method is capable of efficient bright-field- and fluorescent signal image capture for multiple zebrafish, rotational angels, objectives and positions.

An implementation of the method was able to achieve 300 image captures in approximately seven minutes after setup through its user interface. Although, among others, ease of use could still be improved, the proposed method allows for efficient high-throughput, high-quality zebrafish imaging using the VAST system.

Contents

1 Introduction

This thesis reports on the BSc thesis research as part of the BSc Bioinformatics. The research has been done in the period March 2023 till July 2023.

1.1 Research questions

To accommodate zebrafish 3D model creation we propose a method for automation of zebrafish image capture using the VAST BioImager. We aim to create a method for high-throughput high-quality zebrafish image capture using the VAST BioImager as created and distributed by Union Biometrica [Uni23] (Section 2.2) and a LEICA microscope (Section 2.3).
We aim to automate alternating image capture between the VAST system's inbuilt camera (the VAST camera) and the LEICA microscope's Baumer Optronics camera (the microscope camera) at different zebrafish rotations. This allows for better quality and multi- channel image capture, which could in turn be used for the creation of the zebrafish models described in Section 1.3.
Therefore, this paper will focus on the improvement of zebrafish imaging using the VAST system. We aim to optimize quality, reproducibility and throughput.
These research aims are summarised in the following research questions:

RQ1: How can the two-camera setup for VAST imaging be optimized to enable high-throughput and high-quality imaging?
RQ2: How can the measurement of fluorescence in VAST imaging be optimized to achieve higher efficiency?
RQ3: How can a protocol for 3D reconstruction from the VAST using the microscope setup be developed?

The first research question (**RQ1**) deals with the creation of a two-camera setup. This term is used to describe a setup enabling the interplay between the VAST system's camera and a microscope camera. The concept of this arrangement entails seamless communication and coordination between the VAST system camera, the camera that is part of the VAST system and used for calibration and setup, and the microscope camera, used for high quality image capturing and fluorescence, allowing them to operate in tandem. This collaborative setup necessitates a robust communication framework that effectively integrates the functionalities of both the VAST system's camera and the microscope camera.

The second research question (**RQ2**) deals with multi-channel image acquisition. This research question adds additional requirements to the two-camera setup (the ability to change filter and exposure settings) and requests an efficient solution for fluorescent imaging using the VAST system. Enabling efficient capture of fluorescent data allows future researchers to normalise said data by using the surface area and volume calculated from the bright field 3D zebrafish model.

The third research question (**RQ3**) deals with combining the solutions to the previous questions into a protocol allowing for automated image capture using both the VAST system and the LEICA microscope. This allows for high-throughput high-quality multi-channel image capture. The images created by this process can then be used in the VAST pipeline for the creation of 3D

reconstructions.

1.2 Thesis overview

The remainder of this thesis is structured as follows: in the next part of the introduction (chapter 1) some project background information will be given, followed by a disclaimer (chapter 1.4). In the following chapter 2 important components for the creation of the method and the way they were used are described. In chapter 3 the design of the imaging method is discussed. In chapter 4 we will discus different implementations of the design, their results, and their impact on the zebrafish imaging workflow. Finally, we will present conclusions from our results and discus the limitations of this research, and propose possibilities for future work in chapter 5. Additional materials can be found in the appendix (chapter 8).

1.3 Background

Zebrafish (scientifically named *Danio Rerio*) is a commonly used model organism. The species has proven to be an invaluable, widely usable model organism that offers many benefits over other model organisms. One of the key benefits of zebrafish as a model organism is its high genetic similarity to humans. Around 70% of human genes have functional orthologues in zebrafish [HCTea13]. This similarity means that defects in the zebrafish genome often have the same consequences as in humans, allowing the study of human diseases (or diseases similar to human diseases) in zebrafish. A combination of easy reproduction (including external fertilisation), large offspring size [DS09], (relatively) easy insertion of compounds [Rub06] and translucence during early life stages further explains the choice for *Danio Rerio* as a model organism.

A few examples of recent scientific research making use of zebrafish as a model organism are given below:

1. Tuberculosis, a disease caused by bacteria, is a contagious disease which, if not properly treated, can be fatal for humans. Tuberculosis can be studied in zebrafish by infecting them with a tuberculosis inducing bacterium such as, for example, Mycobacterium Marinum. Zebrafish were used to study correlation between transcribed genes and a tuberculosis infection. During the study there was extra focus on genes related to the immune system. The results can be used to further study diseases that are induced by mycrobacteria [MVSV+05].

2. Heart attacks are a major cause of death in humans. They occur when a blood vessel supplying blood to the heart is blocked. When the person who suffered the heart attack survives, heart tissue is irreversibly damaged, which makes them more susceptible to future heart failure [GKDP15]. Finding a treatment to restore damaged heart cells is a major goal in medicine. It was discovered that adult zebrafish can, in contrast to humans, regenerate heart tissue. A protein called Neuregulin1 (abbreviated to *Nrg1*) plays a major role in this process [GKDP15]. Scientists are now investigating whether this protein could potentially repair mammalian (human) heart tissue.

3. Danio Rerio is also used as a model for human cancer research [Shi13]. Besides their role as model organisms, zebrafish also have a conserved cardiogenesis compared to humans, meaning

a wider range of experiments can be done in comparison to mammalian model organisms. This wider range of experiments in combination with more general advantages such as cost make the zebrafish an excellent model organism for cancer research.

The research mentioned above makes use of the fact that due to the translucence of zebrafish embryos, changes in phenotype (usually visualised through fluorescence) can easily be assessed. This process of monitoring phenotype changes is called whole-organism phenotype screening [NSTG20]. Phenotype-based screening, in contrast to target-based screening, allows researchers to examine a broader area than just the targeted cells. Since the whole organism is screened for changes, phenotype-based screening allows for the detection of off-target effects in addition to monitoring on-target effects.

Although whole organism phenotype screening has an advantage over target-based screening, it is a time-consuming ordeal. In the case of zebrafish, phenotype-based screening requires, among other things, the correct positioning and rotation of the zebrafish. Getting the zebrafish in the right position for imaging is not an easy task, since the used zebrafish organisms are very small.
To help decrease imaging time for zebrafish larvae, researchers have developed the Vertebrate Automated Screening Technology (abbreviated to *VAST*) [PMCK+10].

The VAST system has already been used for improvements in zebrafish visualisation. Using axial view images acquired by the VAST camera (explained in section 2.2) 3D models of zebrafish were created [GVSV17]. These images are used to calculate the surface area and volume of the zebrafish.

Although the VAST system was used, image acquisition was very time-consuming, since every image for the model had to be manually acquired. Each acceptable model requires a minimal amount of 42 axial views, with each view equally distant from the previous rotational position. Usually 84 up to 100 images are used for model creation to guarantee good model quality [GVSV17]. Taking the bare minimum (42) means that the creation of i zebrafish models requires at least $42 \times i$ images [GVSV17].

Model quality could further be improved by switching to a microscope for image acquisition. This would increase image quality and allow for multiple channels (like fluorescence) to be captured. However, usage of the microscope camera further increase the amount of images that have to be taken for each organism, since the required images would have to be acquired at multiple positions (at higher magnifications) with multiple channels. For i zebrafish, j positions and k channels, a minimum of $100 \times i \times j \times k$ axial view images have to be acquired. This is infeasible to do by hand.

1.4 Disclaimer

This thesis came to be as a cooperative effort between its two authors. Most work has been done in collaboration, with each author adding to the whole by contributing their distinctive line of thought on the same problem.

Sybe van Benthum mostly focused on designing the method and perfecting the implementation for microscope control. Jens van Bijsterveld mostly focused on implementing the designs and

understanding the different forms of VAST control. The order in which the author names appear on the title pages is alphabetical and does not indicate anything additional.

Overall, each author has evenly contributed to the execution and presentation of the research, and the creation of this paper. The synthesis of the efforts of both authors has resulted in a good design and a good approach for extending the usage of the VAST BioImager.

2 Materials and methods

This section deals with important components of the proposed imaging method. Each component is discussed together with the way it was used in this project.

2.1 Zebrafish models system

As briefly mentioned before (Section 1.3), research for creating 3D models by using the VAST system has already been done at Leiden University [GVSV17].
The 3D models were made by first capturing at least 42 images (each of a different angle, sampled systematically at uniform intervals) of a zebrafish by using the VAST camera, the camera that is part of the VAST system, used for basic image capture and control of the system. The images are then preprocessed, segmented, and fed into multiple mathematical equations that return a 3D representation of the zebrafish by using parameterisation. The surface area and volume of the zebrafish are also determined.
An example of a 3D zebrafish model can be seen in Figure 1.

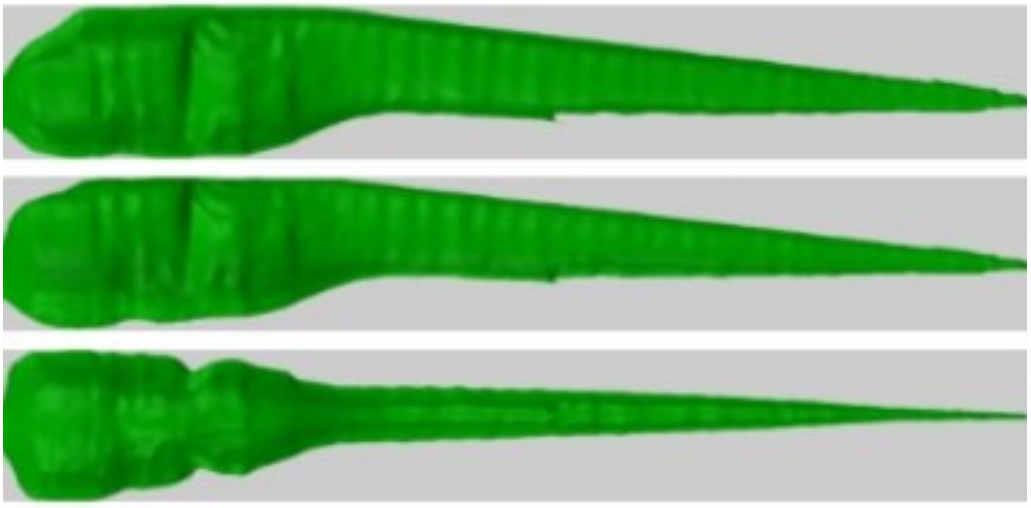


Figure 1: An example of a 3D zebrafish model created by Guo et al.[GVSV17]

The research by Guo et al. can be seen as the instigation for this paper. Based on their research a zebrafish pipeline was developed under the supervision of Fons Verbeek. This thesis provides one of the final parts of this pipeline. The zebrafish pipeline is further explained in Section 2.4.

2.2 VAST system

In this paper, when the term VAST system is used, it is used as a reference to the VAST BioImager as produced by Union Biometrica [Uni23].

The heart of the VAST system consists of a very thin glass tube called the capillary. The capillary is placed between two stepper motors that allow it to rotate. Both ends of the capillary are connected

to tubing, which in turn is connected to a pump. The tubing and pump allow fluids (containing zebrafish) to be pumped through the capillary.
The stepper motors hold the capillary in place in a small plastic container, which is filled with Milli-Q water during imaging. This submersion in Milli-Q is used to get a constant breaking index with respect to the glass. A tray light provides light for the laterally placed VAST camera, that captures the contents of the capillary through a prism.
A schematic illustration of the VAST system can be seen in Figure 2.

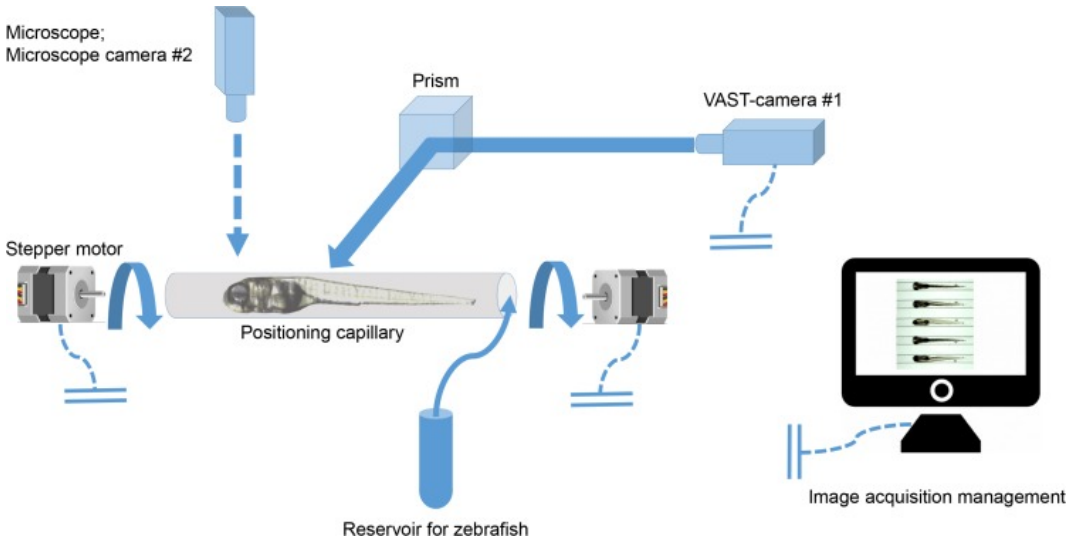


Figure 2: A schematic illustration of the VAST system as created by Union Biometrica. The microscope camera (Microscope camera #2) was not discussed yet in the text. The microscope camera for this project will be discussed in Section 2.3.

When using the VAST system zebrafish larvae are loaded into the capillary from a multiwell plate (LP-sampler), falcon tube (default), or petri dish (manual). The zebrafish larva can then be imaged from all different angles and positions by either moving (through the usage of the pump) or rotating (through the usage of the stepper motors) the zebrafish. This is controlled through the VAST application (Section 2.10) that runs on a connected personal computer.
Images of the correctly positioned zebrafish can be made by using the laterally placed VAST camera. An example of an image taken by the VAST camera can be seen in Figure 3.
The VAST system used for this project also comes equipped with the options *External Communications Package*. This means the VAST system also has TTL ports, which allow for communication with external devices by sending and receiving TTL triggers. What these triggers are for will be explained later.

2.3 LEICA microscope

The VAST system schematic (Figure 2) also shows a second, external, camera. In this project the VAST system is mounted to the stage of a LEICA DM6000 B microscope, equipped with a DFC 450 C (Baumer Optronics) CCD camera. The microscope is suitable for brightfield and fluorescence

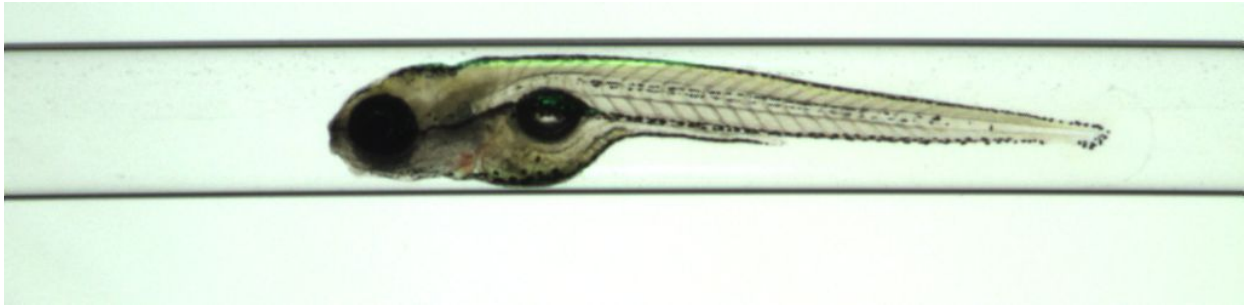


Figure 3: An example of images taken by the VAST camera.

microscopy. It is eqquiped with a Xeon lamp for providing the excitation wavelengths. For emission, two filters are in the filter block:

- N21, allowing visualisation of green fluorescence
- I3, allowing visualisation of red fluorescence

The microscope is equipped with three lenses. The automation process described in this paper has been created for usage with the lowest available magnification (2.5x), because a higher magnification would require repositioning of the zebrafish subject during the imaging process. The other two lenses, which might be used in future research improving on this thesis (outlined in Section 5.3), are lenses enabling 4x and 10x magnification.
The microscope is controlled by the CTR6000 controller, either through a touch operated panel with manual controls, or through either the proprietary LEICA Application Suite (Section 2.7) or through comparable software like μManager (Section 2.8).

2.4 VAST pipeline

The VAST pipeline is a term used to describe a long-term project by Fons Verbeek and others to create a high-throughput pipeline for 3D zebrafish model visualisation.

The pipeline starts with image acquisition through an interplay of the VAST system (Section 2.2), the LEICA microscope (Section 2.3) and the VAST PC (Section 2.5). That part of the pipeline, the last part to be finished, will be provided by the method proposed in this paper.
The acquired images are then processed in the way described by Guo et al. to create a 3D model. This model will then be uploaded to a user interface designed and implemented by *Rosa Zwart* [Zwa20] called the ZVizApp. The ZVizApp will provide users with a dashboard for visualising all processed zebrafish. The app supports both brightfield and fluorescence reconstructions. This is typically made for the Microscope camera output. Most reconstructions shown are based on output from the VAST camera. A visual representation of the VAST pipeline can be seen in Figure 4. Examples from the ZVizApp can be seen in Figures 5 and 6.

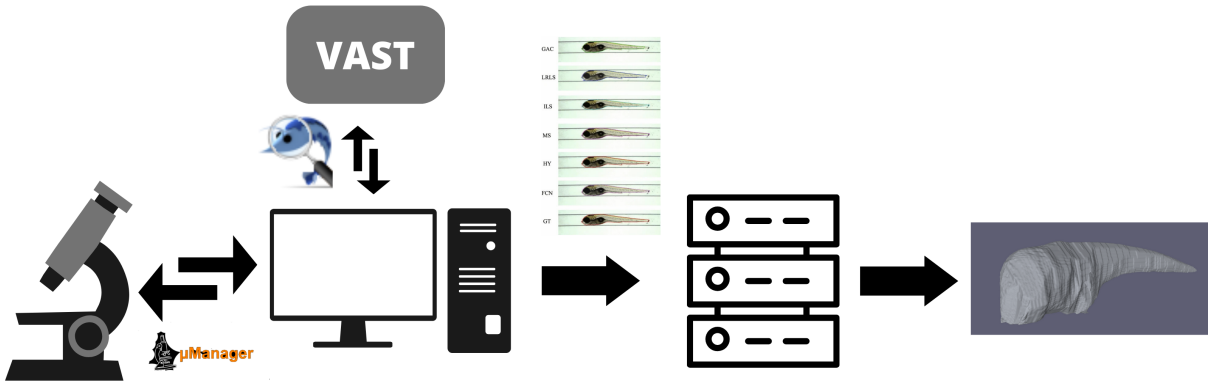


Figure 4: A visual representation of the VAST pipeline. Images acquired with the VAST system and the LEICA camera (both controlled through the VAST PC: Section 2.5) get sent to a server. The server creates 3D zebrafish models, which are displayed on the ZVizApp.

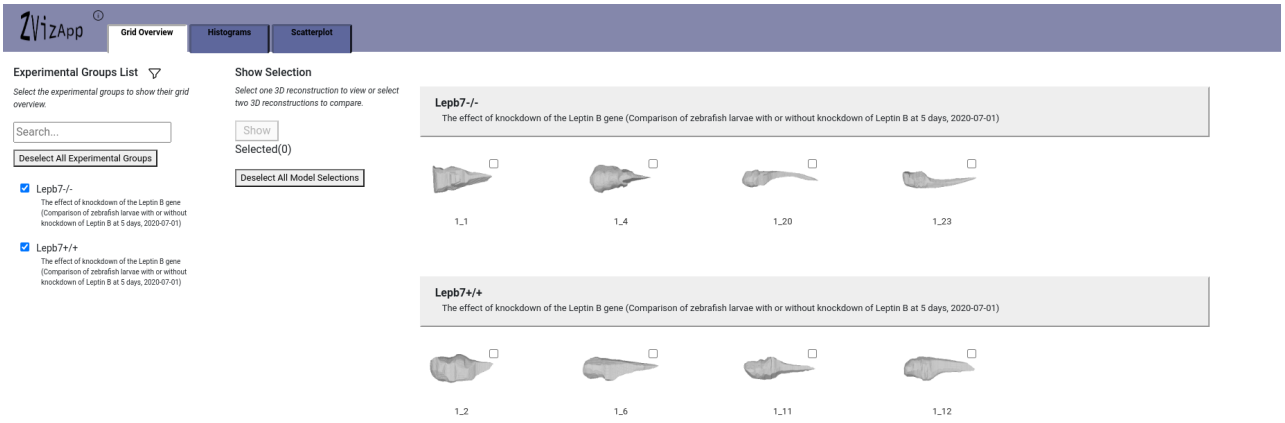


Figure 5: The ZVizApp homepage as it can be found at https://zebrafish.liacs.nl/#/gridTab. The homepage shows all currently available zebrafish models, and allows you to select one for additional information. This can be seen in Figure 6.

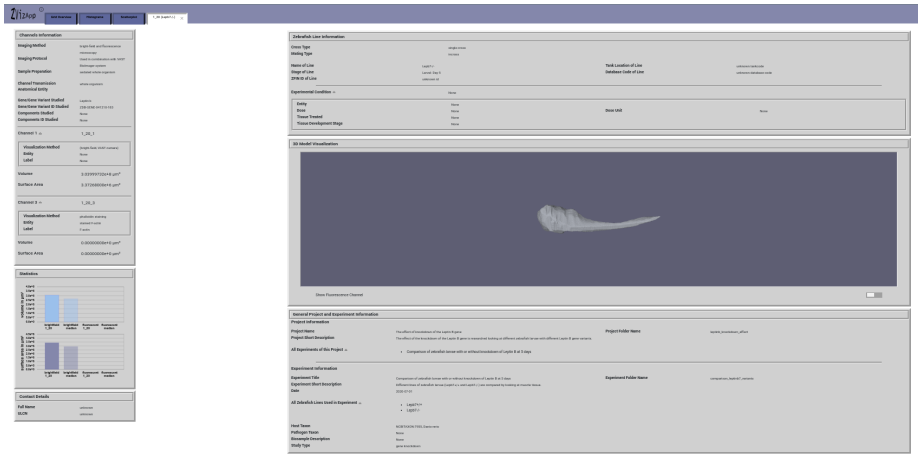


Figure 6: An example of an additional model information page in the ZVizApp. The shown page contains information about the zebrafish line, imaged channels, derived statistics, contact and general experiment information. The page also contains the zebrafish model in an interactable state. The information is derived from the `.json` metadata files as described by *Rosa Zwart* [**?**]. It is important to note that the interface accomodates both Bright Field and fluorescence reconstructions. Most reconstructions currently shown on the ZVizApp are made with the VAST camera and therefore lack this information. The VAST camera reconstructions are used as control reconstructions.

2.5 Computer hardware

As can be seen in the schematic representation of the VAST pipeline (Figure 4), the VAST system and LEICA microscope are both controlled by a computer. This computer, which we will call the VAST PC, is running Windows 10 and is equipped with two monitors.

The VAST PC is connected to all important hardware components. The main distinction is between the VAST camera and the microscope camera. The microscope camera is directly connected to the VAST PC, but the microscope itself is controlled through a CTR6000 controller. The nature of the connections is listed below.

- Firewire connection with the DFC 450c microscope camera
- Ethernet (GigE) connection with the VAST camera
- USB connection with the CTR6000 controller
- USB connection with the VAST BioImager
- USB connection with the VAI-box (explained in Section 2.12)

2.6 Programming

For the implementation of our design, we chose the programming language Python. Mainly because it offers a API to communicate with the microscope through μManager. Also, Python programs can

often intuitively be read, meaning they should (within reason) be accessible for scientists without extensive programming experience.
Python further allows easy installing and importing of modules, which makes usage of, for example, Keyboard and Mouse (Section 2.11) convenient.

2.7 Leica Application Suite

Leica Application Suite (or *LAS* for short) is the proprietary, closed-source, microscope control software provided by LEICA.
Although LAS allows for complete control over the LEICA microscope, it does not allow automation. When using LAS, every action (changing a filter, capturing an image, changing the objective) needs to be done by pressing the buttons in the user interface. This is, unfortunately, not high-throughput.

Throughout the project, LAS was used as a reference. It showed the different microscope functions that could be controlled through the CTR6000 controller and was used as a point of comparison for the images captured by our method.

2.8 μManager

μManager is an open source equivalent to LAS (discussed in Section 2.7). The μManager repositories, which are hosted on GitHub [MM23], contain a wide variety of implementations, ranging from completely user focused (with a graphical user interface) to completely developer focused (with an API).

For this project, we opted for the *pymmcore_plus* (short for Python MicroManager Core Plus) implementation of μManager, an extension of the *pymmcore* module, which allows access to the μManager C-core through a python API. The choice for a developer-oriented implementation enabled automated microscope control without the need for a traditional graphical user interface.

Since μManager can be used for many different types of microscopes, a configuration file is needed. This configuration file specifies the capabilities of a specific microscope, thereby telling micromanager what can be controlled through the CTR6000 controller.
The configuration file for this project's LEICA DM6000 B microscope was made by comparing options in LAS to different options in the μManager configuration file creation wizard in the μManager user interface. Configuration files can also be made by hand. The configuration file syntax can be found on the μManager website [Mic].

The configuration file used for this project (and by extension, for the method proposed in this paper) can be found in the Appendix (Section 8.1).

2.9 Wireshark

Wireshark is an open-source packet sniffer. A packet sniffer is a tool that is used to examine traffic over different computational communication channels (WiFi, USB connections, etc.).

Wireshark (together with similar tools like USBPCap and Device Monitoring Studio) was used in an attempt to reverse engineer communication between the VAST application (Section 2.10) and the VAST system (Section 2.2).

The reverse engineering process was ultimately unsuccessful because of a closed source (Cypress) driver. Some artifacts from the reverse engineering process like, for example, a list of packets and matching commands and a list of USB information, can be found in the appendix (Sections 8.2 and 8.3).

2.10 VAST application

The VAST application is an application for controlling the VAST system, provided by Union Biometrica. Like LAS (Section 2.7), the VAST application allows for full control of the VAST system but is unfortunately not scriptable. An example of the VAST application can be seen in Figure 7.

Central to the VAST application is the live VAST camera stream that displays the capillary. While looking at this stream buttons can be used to for example, position, rotate or image the capillary (and its contents).

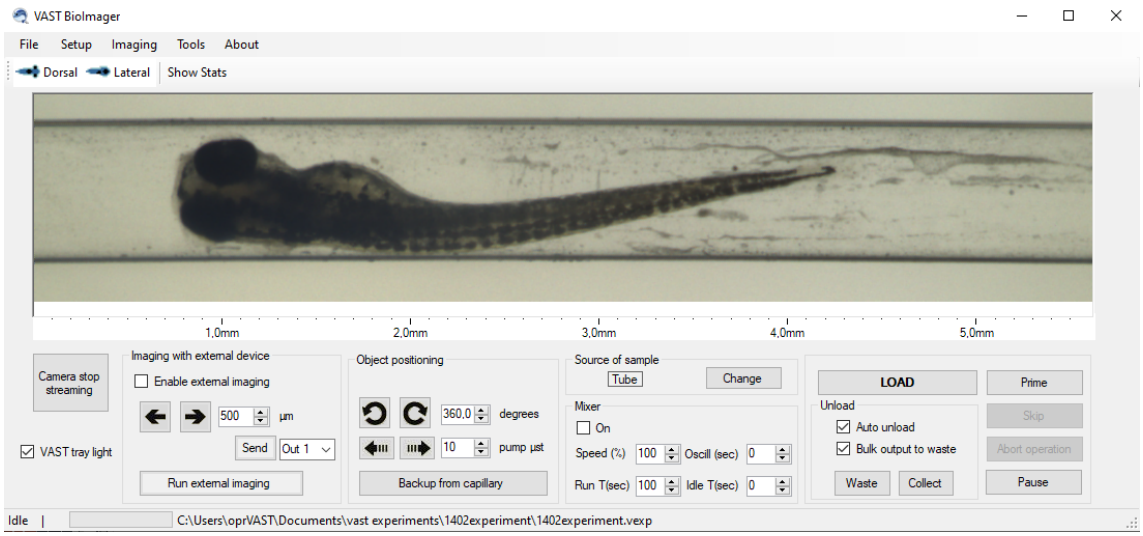


Figure 7: An image capture taken while using the VAST application as created by Union Biometrica. The VAST application displays the full range of control options for the VAST system, together with a livestream from the VAST camera.

The VAST application also has some more complicated functions, like, for example, zebrafish loading. Zebrafish loading is supposed to be an automated process, where the only user interaction is the button *LOAD*. The VAST application should be able to use predefined templates to recognise zebrafish, and thus make the distinction between air bubbles, debris, and imaging subjects.
However, during our usage of the VAST application, this zebrafish recognition function was far from perfect. It kept classifying zebrafish as bubbles, and thus discarding them. The only way to allow a zebrafish to remain in the capillary was to prematurely abort the loading operation while the

zebrafish was visible on the camera stream. We call this process manual loading. Manual loading is described in Section 4.3.

Another more complicated function of the VAST application is the *External Imaging* functionality. The external imaging menu allows users to define the actions that should be taken by the VAST system upon receiving TTL triggers.
An example of the external imaging menu can be seen in Figure 8.

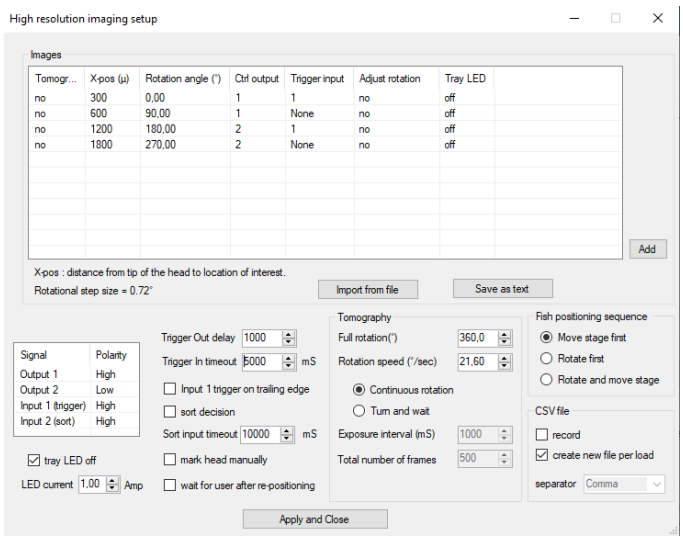


Figure 8: An example of the VAST applications external imaging menu, which can be found under `Imaging/Imaging with an external device` in the VAST application's main screen (Figure 7).

As can be seen in Figure 8, the external imaging menu consists of a table and some additional options. The table lists actions corresponding to received triggers from top to bottom. When external imaging is activated, the actions in the top row will be executed. When the first trigger is received, the second row will be executed. This goes on until all rows have been executed.

Although it was first attempted to completely circumvent VAST application usage in the final method (by reverse engineering VAST application commands using Wireshark, Section 2.9) both final implementations use the VAST application.

2.11 Keyboard and Mouse libraries

Keyboard and Mouse are two (suitably named) Python libraries for controlling a computer's keyboard and mouse. This allows programmers to spoof user interaction, thus automating user interface usage. Both libraries are open-source and hosted on GitHub [Bop19] [Bop21].

The Keyboard and Mouse libraries were used in the UI spoofing implementation (Section 4.1.3) to press buttons in the VAST application, thereby automating VAST control. The external imaging implementation does not use Keyboard and Mouse, since it does not need spoofing of user interaction

to work. The UI spoofing implementation, which does use Keyboard and Mouse, was used for fast prototyping.

2.12 VAI-box

The VAI-box (VAST Arduino Interface box) is an Arduino UNO created by Maarten Lamers.

Since the VAST PC does not have TTL ports, a way to convert TTL triggers to serial communication was needed. To accommodate this, the VAI-box acts as an interpreter between the VAST system's TTL ports and the VAST PC's USB port.
The VAI-box GPIO pins are connected to the in- and output TTL connectors on the VAST system. The VAI-box is also connected to the VAST PC by USB connection. Power is supplied through a power supply. A schematic view of the wiring can be found in Figure 9.

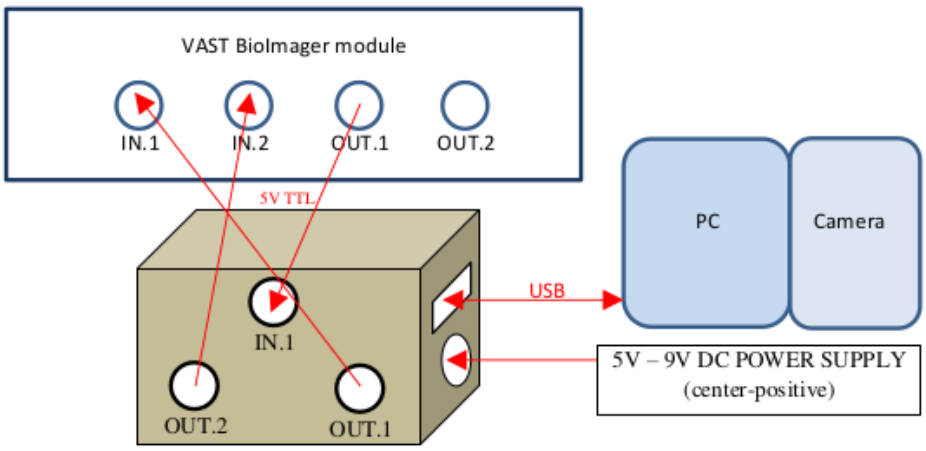


Figure 9: A schematic representation of wiring between the VAI-box, VAST PC, VAST system and the power grid. The VAI-box documentation can be found in the appendix (Section 8.4).

If the VAI-box receives a trigger from the VAST system (GPIO pin 2 becomes `HIGH`), the VAI-box sends a message (`R1`) to the VAST PC using serial communication. The VAI-box can also be used to send messages back to the VAST system. By sending `TT`, the VAI-box will send a trigger to the VAST system's TTL Input 1. By sending `TS`, the VAI-box will send a trigger to the VAST system's TTL Input 2.

The VAI-box was necessary for usage of the external imaging implementation 4.1.4 in order to facilitate communication between the VAST PC and the VAST system.

3 Design

We will now describe the process of designing a method that combines components from Section 2 into a workable approach for high-quality high-throughput image acquisition using the VAST BioImager.
In practice, designing the method required designing an overarching application for controlling both the VAST system and the LEICA microscope with as little user interaction as possible.

3.1 Requirements from the research questions

Before starting work on the design, it was necessary to take inventory of all requirements and restrictions we had already defined in the research questions.

As discussed, research question 1 mainly deals with effective communication between the VAST system and the LEICA camera. Effectiveness for imaging sets specific requirements for both microscope and VAST control.
Effective microscope control would mean an implementation that could autonomously capture images. This would mean control over light, objectives, table height and camera actions. Effective VAST control would mean complete control over relevant VAST actions for zebrafish positioning. These are the rotate, pump and move options.

Research question 2 extends upon the requirements from research questions 1 by adding multi-channel imaging (fluorescence imaging) to the requirements. This means an extension on the microscope features mentioned above. Control over diaphragms, filters and exposure time are added to the requirements.
VAST control requirements are also extended with control of the traylight. In theory, bright field LEICA images would be possible with and without the traylight, but an active traylight during fluorescence imaging would negate any fluorescent signal.
Research question 3 deals with combining the previous research questions into a single workable solution. This solution should first capture images with the VAST camera. These images will be used to create a control reconstruction. Afterwards, the microscope camera will be used for image capture. Since the microscope has better lenses and is able to capture fluorescence, these captures will be used for creating the desired reconstructions (with gene expression information).

3.2 Flowchart

To help in implementing the method, a flowchart was created. This flowchart contains a walk through the final approach. Blue boxes represent actions taken by the VAST system, while red boxes represent actions taken by the LEICA microscope. The coloured containers in the background are the states that need to be kept by the VAST PC.

The flowchart can be found in Figure 10.
Using the flowchart, an initial design of the resulting GUI could be made. This design shows what controls are required, and uses grouping and continuity derived from the flowchart to depict the expected flow. This GUI design derived from the flowchart can be seen in Figure 11.

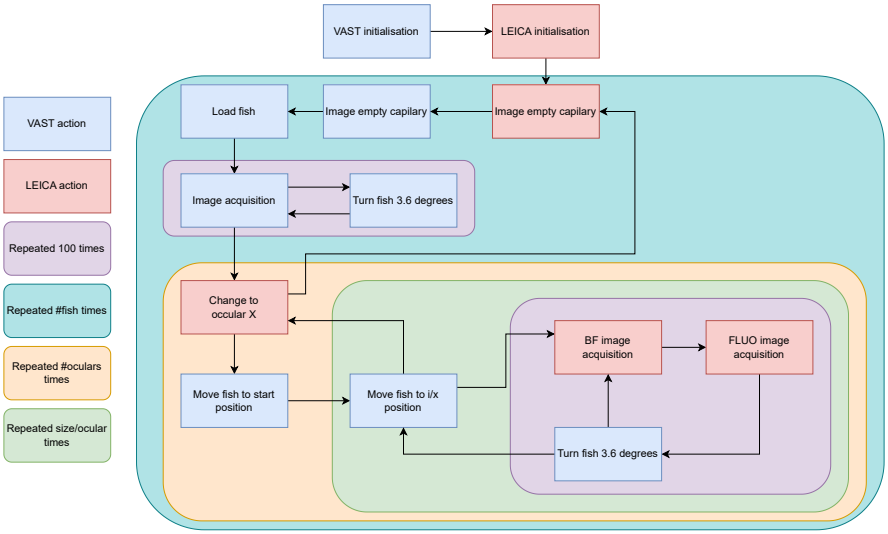


Figure 10: The flowchart depicting the final design of the imaging method proposed in this paper.

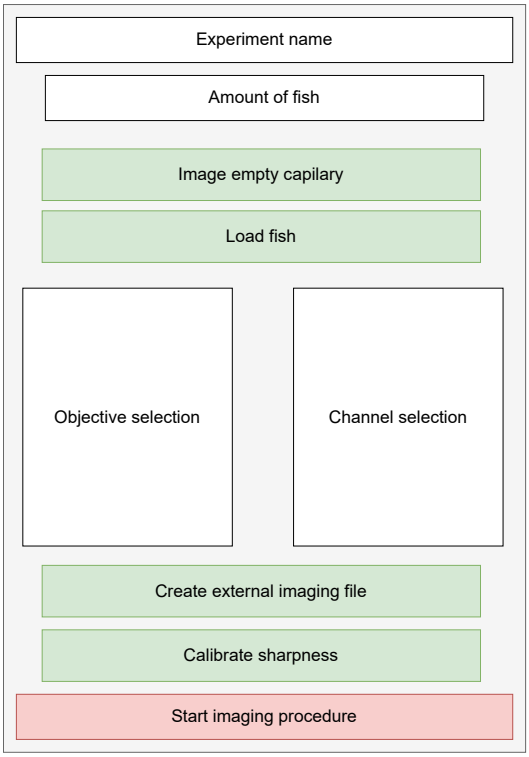


Figure 11: The initial design for the GUI accompanying the final implementation. It shows the grouping of buttons and the expected control flow. White buttons require information about the process as a whole, for example, the amount of fish that are to be imaged, or which channels to use for imaging. Green buttons require user interaction. These buttons are used for processes needed to prepare for autonomous imaging. The red button starts the automated imaging procedure.

3.3 Method design

Using the requirements set by the research questions, and the crude design shown in the flowchart (Figure 10), pseudo-code for the automated imaging process was created. This pseudo-code is shown below.

```
For f in fish  # Repeat procedures for each fish that needs to imaged
    For r in rotations  # For each rotation (between 42 and 240)
        Do rotate(r)  # Move the subject to the correct angle using the VAST
        Do VAST_imaging  # Image the subject using the VAST camera
    For o in objective  # For each objective we want to use
        Do LEICA_objective(o)  # Switch the current objective
        Do LEICA_height(o)  # Switch to the optimal height for objective o
        For p in positions  # For each position needed for that objective
            Do move(p)  # Move the subject to the correct position
            For r in rotations  # For each rotation (between 42 and 240)
                Do rotate(r)  # Move the subject to the correct angle using the VAST
                For c in channels  # For each imaging channel we want to use
                    Do LEICA_imaging(c)  # Image the subject using the LEICA camera
                Do create_metadata(f, o, p, r, channels)
Do save_metadata()  # Save metadata for these captures
```

From this pseudo-code design, it was concluded that for maximum coherence and minimum dependency, interaction between the VAST system and the LEICA microscope was not necessary. It is sufficient for both systems to communicate with the VAST PC, which will keep track of the current place in the program, and send commands to either system if an action is required.
This state machine approach was required, since our tests pointed out that direct VAST control through code would be nearly impossible, meaning the VAST application would need to be used as preparation for the imaging procedure. Prototyping (using the Keyboard and Mouse libraries: Section 2.11) pointed out which preparations a user should do before the automated process could take over.
Using the flowchart, GUI design and pseudo-code as a guide, two different implementations were made (Section 4.1). Each implementation uses the flow as described in the flowchart and pseudo-code, but a different kind of VAST control. The UI spoofing implementation (Section 4.1.3) was used for fast prototyping of the eventually created external imaging implementation (Section 4.1.4).

4 Results

4.1 Design implementation

Implementing the design found in the flowchart (Figure 10) was done in multiple steps. First of all, classes for executing microscope actions and VAST actions were created. Then, two overarching implementations were created. The UI spoofing implementation (Section 4.1.3) was used for prototyping. The external imaging implementation (Section 4.1.4) was made as an improvement on the UI spoofing implementation.

4.1.1 Microscope control

For implementing control of the LEICA microscope, we used the μManager implementation *pymmcore_plus* (Section 2.8) and the configuration file (Section 8.1). We implemented this combination in a specialised class (`MicroscopeManager`) containing functions specifically designed for controlling the LEICA DM60000B through the CTR6000 controller.
The `MicroscopeManager` class allows for easy control of the LEICA microscope without having to rely on complicated labels as found in the configuration file. The class was implemented in such a way that changing the class attributes (`brightness`, `light`, `lens`, `objectives`) directly results in changes in the underlying *pymmcore_plus* class, thus resulting in physical changes on the microscope.

The class functions (`switch_filter`, `switch_objective`, `snap_picture`) allow for easier implementation of microscope control in the main code by applying changes to multiple class attributes (like for example turning on the light, then snapping a picture), but also by checking if requested changes are valid. This makes the implementation more robust and allows for less mistakes during code execution in the overarching implementations.

The file containing the `MicroscopeManager` class can be found in the paper's GitHub repository [vBvB23a]. The class makes use of a configuration file with some information extracted through trial and error. This file can also be found in the GitHub repository [vBvB23b].

4.1.2 VAST control

Control of the VAST system was first attempted through reverse engineering VAST application (Section 2.10) commands through Wireshark (Section 2.9). Sending the found commands over the USB connection would, theoretically, allow for headless VAST control.
VAST control was eventually established through the VAI-box (Section 2.12). Communication with the Vai-box was implemented through the Python module *PySerial* in a class (`VastManager`).

The `VastManager` class connects to the VAI-box by scanning all available COM ports for a device with a matching description field (`Arduino UNO`). Once the correct COM port is found, a serial connection is established using a baudrate of 115200.

The `VastManager` class also has helper functions for, for example, waiting until a message is received by the VAST PC (`listen_for_response`, printing the current VAI-box settings (`get_settings`), and sending the "Send pulse to Input 1"" command to the VAI-box (`trigger_vast`).

The file containing the `VastManager` class can be found in the paper's GitHub repository [vBvB23c].

4.1.3 UI spoofing implementation

The first overarching implementation that was created was the UI spoofing implementation. This implementation combines microscope control through the `MicroscopeManager` class with automated VAST control by spoofing UI interaction through the Keyboard and Mouse Python modules (Section 2.11).

The UI spoofing implementation can be found in the GitHub repository [vBvB23d].

4.1.4 External imaging implementation

The UI spoofing implementation returned great results, but also had some downsides (described in Section 4.2). The second overarching implementation, the External imaging implementation, improved on the UI spoofing implementation by combining microscope control through the `MicroscopeManager` class with VAST control through the `VastManager` class and VAST application external imaging functionality (Section 2.10).

Usage of the external imaging functionality requires the ability to send and receive triggers, which was handled through `VastManager` and the VAI-box, and the ability to create a text file with actions for each trigger. The external imaging implementation manages the creation of the text file. This is done by creating a file with all required data entries based on the amount of objectives and fluorescent channels that should be used during imaging.

Since (like the UI spoofing implementation) the external imaging implementation still requires some form of user interaction, a small user interface was made for ease of use. This user interface does nothing more than enforcing the order of operations, and is not a final application, but more a proof of concept.

Both the external imaging implementation code and user interface can be found in this paper's GitHub repository [vBvB23e] [vBvB23f]. The complete workflow for using the external imaging implementation is described in Section 4.3.

4.2 Results

As mentioned in Section 4.1.3, one of the two final implementations (the UI spoofing implementation) does have some drawbacks. They are listed below.

First of all, the UI spoofing implementation has a fixed set of not automated prerequisites before execution. Before deploying the UI spoofing implementation the target directory for the VAST camera image acquisition should be empty. This is done by either emptying the default target directory or changing the target directory by saving a picture to a different directory before executing the python script. Another prerequisite is that the VAST system's traylight should be

turned on at the start of the process. Since the traylight is controlled by a toggle box, the python application does not know its state, so it assumes the traylight as on at the start of the script.
The last prerequisite to using the UI spoofing implementation is that the VAST application should be maximised in the leftmost screen. Since the `mouse` library uses coordinates to click certain buttons, the positions of relevant UI elements should be constant. By always having the VAST application maximised on the left-most screen, this is achieved.

The second disadvantage, after the prerequisites, is that the UI spoofing implementation is not as headless as we would have liked. This means that movement of the mouse at an inopportune moment can disrupt image acquisition. This means that, since the mouse and keyboard are used, the computer cannot be used during image acquisition.

The second functional implementation, the external imaging implementation, does not suffer from these drawbacks. However, it does require a little more user interaction than the UI spoofing implementation, since relevant steps in the VAST application, like for example loading the external imaging file, are not automated.

By using the external imaging application, images were acquired from a Fli GFP zebrafish (a zebrafish with green fluorescent proteins in the cardiovascular system). Figure 12 shows a compilation of 100 bright field images acquired through the VAST camera. Figures 13 and 14 show compilations of the same positioning taken with the LEICA microscope, using bright field and fluorescence (respectively). A single fluorescent channel capture can be seen in Figure 15. All images were taken in approximately 7 minutes, which would not have been possible by hand.
It can be seen that the LEICA captures (Figure 13) has a better resolution than the VAST camera (Figure 12). This is the reason for using the VAST camera reconstructions as a control for the LEICA camera reconstructions.

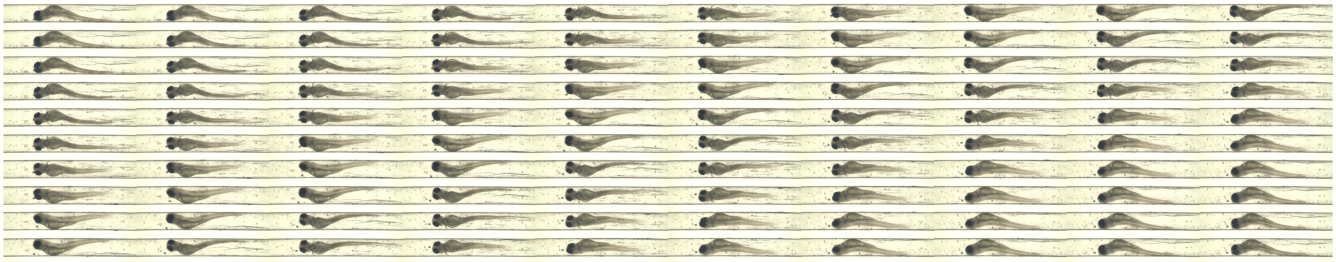


Figure 12: A compilation of 100 bright field images of a flea zebrafish acquired by the VAST camera during execution of the external imaging implementation.

4.3 Workflow

This section will describe the workflow that was used to acquire the images shown in the section above (Section 4.2).

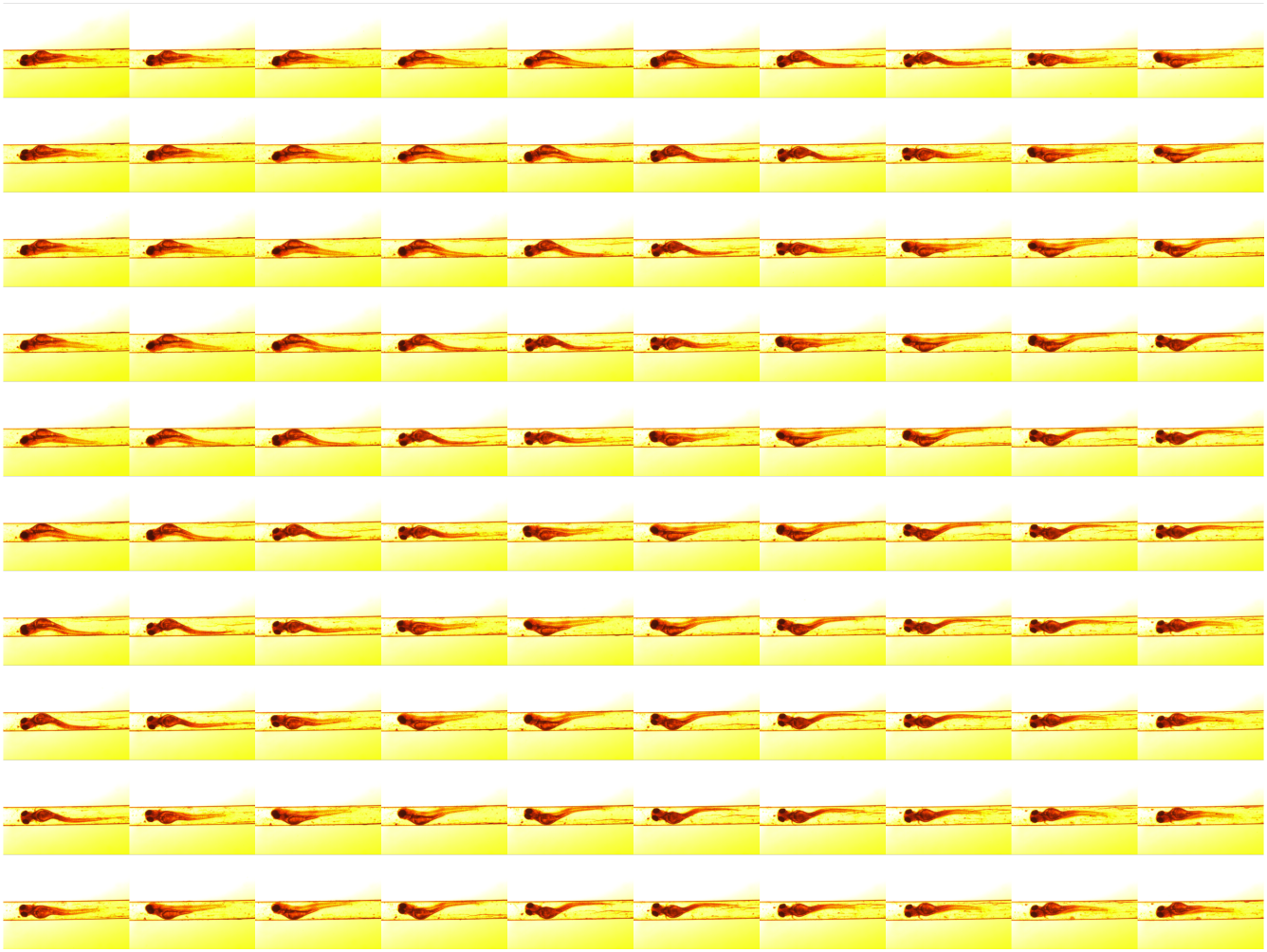


Figure 13: A compilation of 100 bright field images of a flea zebrafish acquired by the LEICA microscope during execution of the external imaging implementation. The images are little overexposed, but parameter tuning of light brightness and/or exposure can easily fix this.

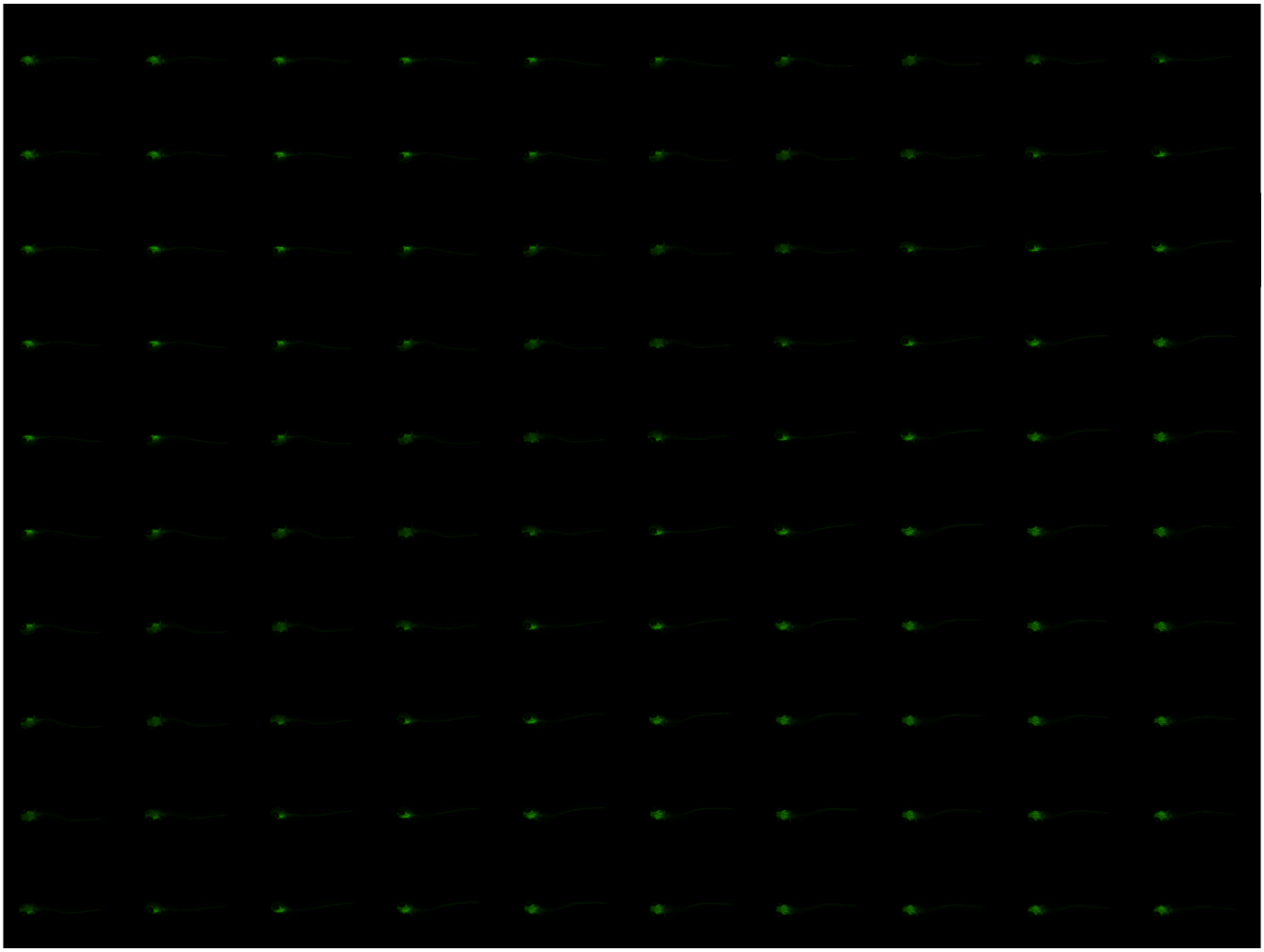


Figure 14: A compilation of 100 green fluorescence images of a Fli GFP zebrafish acquired by the LEICA microscope during execution of the external imaging implementation. A single fluorescent channel capture can be seen in Figure 15

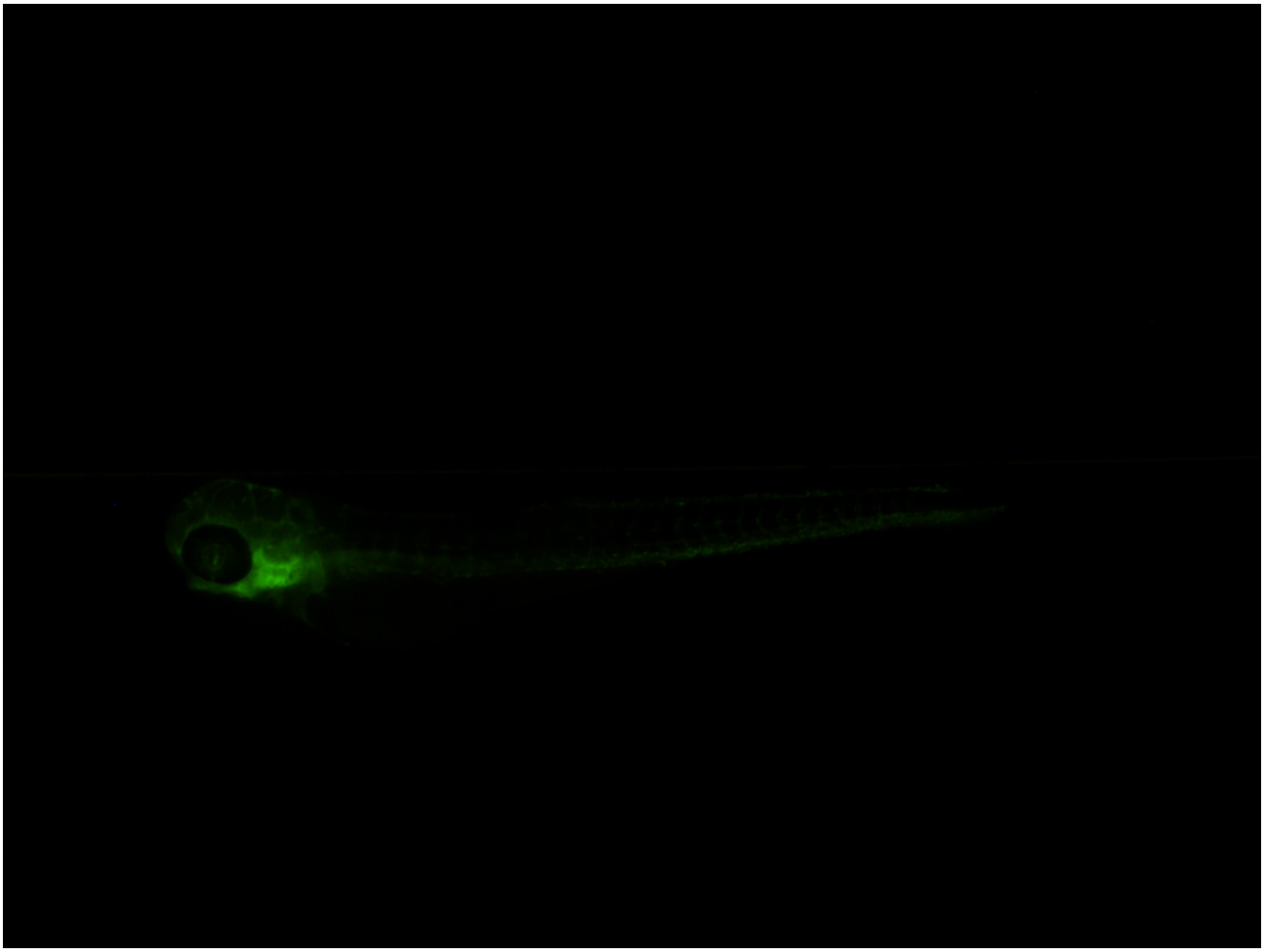


Figure 15: A single green fluorescent capture of a Fli GFP zebrafish acquired by the LEICA microscope during execution of the external imaging implementation.

As described in Section 2.10, zebrafish loading using the VAST is, at this moment, not without hiccups. To overcome this, a process we call manual loading was used in order to load zebrafish into the VAST system. The steps for manual zebrafish loading are presented below. These were derived from the notes on zebrafish loading made by Professor Fons Verbeek, which can be found in the appendix (Section 8.5) as a reference.

1. Wash all remaining zebrafish (parts) out of the VAST system using the `Prime` and `Waste` buttons
2. Place the loading tube into the petridish with zebrafish
3. Use manual pumping to remove any excess air bubbles from the loading tube
4. Use the `Load` button to initilise zebrafish loading
5. Using the loading tube, suck up a few zebrafish. Ensure there is space between the individual zebrafish to ensure they do not get stuck within the system
6. Once enough zebrafish have been sucked into the tube, pause loading using the `Pause` button
7. Move the loading tube to a falcon tube without zebrafish (just eggwater) and remove any excess air bubbles
8. Continue loading until you see a zebrafish, then press `Abort operation`
9. Use manual pumping

The *external imaging implementation* workflow is as follows:

1. Load the required amount of zebrafish into the VAST system using the manual loading process described above.
2. Run the external imaging implementation user interface. From now on, this will be refered to as *the UI*.
3. Use the UI to define the amount of zebrafish that are to be imaged and the name of the experiment.
4. Use the UI to select the required objectives and channels for image acquisition.
5. Press the *Image empty capillary* button in the UI. This button will walk you through imaging the empty capillary in the VAST application, and will then image the empty capillary using the LEICA microscope.
6. Press the *Create external imaging file* button in the UI. This button will create the external imaging text file and walk you through the process of loading it into the VAST application external imaging menu.
7. Press the *Start imaging procedure* button in the UI. This button will ask you to turn on the checkbox for external imaging in the VAST application and ask you to press the *Run external imaging* button in the VAST application.

8. From this point, all imaging steps are automated. You are now free to drink a cup of coffee while the *external imaging implementation* acquires zebrafish images from 100 different rotations.

During imaging, metadata is saved in a python dictionary. At the end of the imaging process, all metadata is saved as s json file. This json file contains information about the fish number, objective, position, angle and channels used for each captured image.

5 Conclusion and discussion

5.1 Conclusions

In this paper, we have discussed the components, design and implementation of a method allowing high-throughput high-quality image acquisition using the VAST system. We have seen that the output and performance of two types of implementations (UI implementation and external imaging implementation) are comparable, but that the external imaging application is the smoothest, most foolproof method for high-throughput high-quality multi-channel image acquisition.

We can conclude that the proposed method optimizes the interplay between the two different camera systems and optimizes acquiring fluorescent data from zebrafish larvae by decreasing the amount of user interactions needed for acquiring a 3D zebrafish model from a variable $42 \times i \times j \times k$ (where 42 is the absolute minimum, because usually 100 images are used for a reconstruction) to a constant value, decreasing hands-on time for researchers wishing to acquire 3D zebrafish models. We can also conclude that the proposed method can, in the future, be integrated into the VAST pipeline, through proper use of its inbuilt metadata and naming conventions.

5.2 Discussion

The research, although ultimately successful, was hampered by unfortunate hardware defects. The most unfortunate (and reoccurring) defect was the breaking of the capillary. Handling the capillary should always be done with extreme care, since its minimal size makes the glass very fragile, and therefore easy to break.

The switching from windows 7 to windows 10 of the VAST PC at the start of the project was another hiccup. Switching between operating systems might require an update or change in the drivers used to control either the VAST system or the LEICA microscope. A solution for the future could be to run the VAST application inside a virtual environment or container in order to ensure independence from the operating system.

It is unfortunate that, because of the proprietary cypress driver implicated in the VAST communication protocol, we were unable to create a truly headless implementation of the proposed method. The created methods all require actions to be taken in the VAST application.

A shortcoming of the implementations is the fact that both implementations do not factor in the magnification of different objectives. An objective with a higher magnification might need the zebrafish to shift positions in order to image it in its entirety. Although an option for position shifting is implemented in the code, manual measurements would need to be made in order for the implementation to know where the zebrafish needs to be positioned. This would also require extensive validation experiments before usage.

The UI design that was created for the external imaging is still a proof of concept, and does not yet resemble a user interface that is practical to use. The options in the UI are not properly grouped or ordered yet. As suggested by a member of the audience during the final presentation, it

would have been more logical to order the options in the user interface in the same order as the pseudo-code shown in Section 3.3.

5.3 Future work

The implemented solution greatly improves the workflow to use the VAST pipeline. Image capture time is greatly shortened, leading to scientists having more time to do other tasks. Despite that there is still a lot to improve on the solution presented here. The devised improvements can be divided in 3 categories: User interaction improvements, better support for high magnification and automasation.

One of the improvements that are aimed at enhancing the user experience will be to remove the VAST Bioimager software from the workflow. The software is now used in addition to the control software that was developed. The VAST software is needed to load fish and to set the script for the external imaging functionality. As using two pieces of software to control one process is not complimenting with the human computer interaction guidelines, the VAST bioimager software has to be cut out of the process. To do this, all functionality that is used must be replaced. As earlier mentioned, trying to reverse engineer the VAST software probably won't work. So the only option is to use an API that has not yet been provided by Union Biometrica.

The other enhancement would be to further develop the control software. The software that we presented here is more of a proof of concept. The software could be improved greatly. Some examples are: The layout could be improved, functionalities could be better explained, more functions could be added. Improving user interaction is beyond the scope of this study, so it will not be discussed further.

To support imaging with higher magnification, where the fish couldn't be completely captured in one shot, further enhancements have to be made. A system has to be in place that can automatically slide the table, snap a picture, slide, snap a picture and so on. This way the fish could be completely captured, with much more detail. A problem with this is that the capture will be much longer and the fish will probably move during that time, that spoils the capture.

Other enhancements would be to decrease the amount of times the researcher has to intervene, as to further automate the process. Now every fish has to be loaded manually. If you want to do statistically significant research, you need to image a lot of fish. So that's a lot of work if you have to load them manually. Automating this would save a lot of time.

Another task that has to be done for every fish is focusing the microscope. If an auto focus algorithm is developed or implemented, this wouldn't be a issue anymore.

When doing research to one type of fish the current solution and the improvements presented here above, will perform very good in theory, but when you want to research many different types of fish, it wouldn't work. This is because the tube with fish has to be changed manually. To solve this Union Biometrica has developed a device that uses a 96 well plate and robotic arm to automatically sample fish. This device could be incorporated into the workflow.

Further research could also focus on integrating the implementation described in this paper into the VAST pipeline. This integration would allow researchers to see 3D models of zebrafish in the ZVizApp (Section 2.4) in near real time. The integration into the pipeline would require an alteration to the metadata saved in the external imaging implementation in order to meet the requirements defined by *Rosa Zwart* [**?**].

It can be concluded that the research done so far is only the start of an easy, automated VAST system. The efforts done so far only scratch the surface of possibilities for future zebrafish imaging.

6 Acknowledgements

The authors want to thank the following people for their extensive help during the research.

First of all we want to thank our supervisor, Fons Verbeek, for his help with this project. He was always available to help us, even when on holiday, or late in the evening!

Furthermore we want to thank Gerda Lamers for her help with the project. During our many long hours in the lab, she was always nearby to help with, among others, unclogging the VAST system. Also, having someone nearby who knew where to get coffee was very nice.

Third of all we want to thank Maarten Lamers for his contributions to the project. His VAI arduino was essential to the succes of this project.

Last but not least, we want to thank the zebrafish caretakers, specifically Wanbin Hu, Gabriel and Guus, for their contributions to this thesis project. Without them, there would be no material to work with.

7 References

[Bop19] BoppreH. Keyboard, 2019. Accessed: 25 June 2023.

[Bop21] BoppreH. Mouse, 2021. Accessed: 25 June 2023.

[DS09] Allison D'costa and Iain T Shepherd. Zebrafish development and genetics: introducing undergraduates to developmental biology and genetics in a large introductory laboratory class. *Zebrafish*, 6(2):169–177, 2009.

[GKDP15] Matthew Gemberling, Ravi Karra, Amy L Dickson, and Kenneth D Poss. Nrg1 is an injury-induced cardiomyocyte mitogen for the endogenous heart regeneration program in zebrafish. *Elife*, 4:e05871, 2015.

[GVSV17] Yuanhao Guo, Wouter J Veneman, Herman P Spaink, and Fons J Verbeek. Three-dimensional reconstruction and measurements of zebrafish larvae from high-throughput axial-view in vivo imaging. *Biomedical optics express*, 8(5):2611–2634, 2017.

[HCTea13] Kerstin Howe, Matthew D Clark, Carlos F Torroja, and et al. The zebrafish reference genome sequence and its relationship to the human genome. *Nature*, 496(7446):498—503, April 2013.

[Mic] Micro-Manager Contributors. Micro-Manager.

[MM23] Micro-Manager. Micro-Manager GitHub Repositories. https://github.com/orgs/micro-manager/repositories, Accessed: 2023. Accessed on: June 19, 2023.

[MVSV+05] Annemarie H Meijer, Fons J Verbeek, Enrique Salas-Vidal, Maximiliano Corredor-Adámez, Jeroen Bussman, Astrid M van der Sar, Georg W Otto, Robert Geisler, and Herman P Spaink. Transcriptome profiling of adult zebrafish at the late stage of chronic tuberculosis due to mycobacterium marinum infection. *Molecular Immunology*, 42(10):1185–1203, 2005.

[NSTG20] Vandana S Nikam, Deeksha Singh, Rohan Takawale, and Minal R Ghante. Zebrafish: An emerging whole-organism screening tool in safety pharmacology. *Indian journal of pharmacology*, 52(6):505, 2020.

[PMCK+10] Carlos Pardo-Martin, Tsung-Yao Chang, Bryan Kyo Koo, Cody L Gilleland, Steven C Wasserman, and Mehmet Fatih Yanik. High-throughput in vivo vertebrate screening. *Nat. Methods*, 7(8):634–636, August 2010.

[Rub06] Amy L Rubinstein. Zebrafish assays for drug toxicity screening. *Expert opinion on drug metabolism & toxicology*, 2(2):231–240, 2006.

[Shi13] HR Shive. Zebrafish models for human cancer. *Veterinary pathology*, 50(3):468–482, 2013.

[Uni23] VAST BioImager platform overview. https://www.unionbio.com/vast/, 2023. Accessed: June 19, 2023.

[vBvB23a] Jens van Bijsterveld and Sybe van Benthum. VAST automation. https://github.com/JensVanBijs/vast_automation/blob/master/controllers/LEICA_control.py, Accessed: 2023. Accessed on: July 31, 2023.

[vBvB23b] Jens van Bijsterveld and Sybe van Benthum. VAST automation. https://github.com/JensVanBijs/vast_automation/blob/master/controllers/microscope_settings/microscope_configuration.json, Accessed: 2023. Accessed on: July 31, 2023.

[vBvB23c] Jens van Bijsterveld and Sybe van Benthum. VAST automation. https://github.com/JensVanBijs/vast_automation/blob/master/controllers/VAST_control.py, Accessed: 2023. Accessed on: July 31, 2023.

[vBvB23d] Jens van Bijsterveld and Sybe van Benthum. VAST automation. https://github.com/JensVanBijs/vast_automation/blob/master/ui_spoofing_implementation.py, Accessed: 2023. Accessed on: July 31, 2023.

[vBvB23e] Jens van Bijsterveld and Sybe van Benthum. VAST automation. https://github.com/JensVanBijs/vast_automation/blob/master/external_imaging_implementation.py, Accessed: 2023. Accessed on: July 31, 2023.

[vBvB23f] Jens van Bijsterveld and Sybe van Benthum. VAST automation. https://github.com/JensVanBijs/vast_automation/blob/master/external_imaging_ui.py, Accessed: 2023. Accessed on: July 31, 2023.

[Zwa20] Rosa Zwart. Comparative analysis of whole mount 3d imaging modalities for zebrafish research, 2020.

8 Appendix

The appendix contains additional information that might not be necessary to understand the paper, or the project it is based on, but could be of help for researchers wishing to continue the project.

8.1 μManager configuration file

```
# Reset
Property,Core,Initialize,0

# Devices
Device,COM1,SerialManager,COM1
Device,BaumerOptronic,BaumerOptronic,BaumerOptronic
Device,Scope,LeicaDMI,Scope
Device,IL-Turret,LeicaDMI,IL-Turret
Device,ObjectiveTurret,LeicaDMI,ObjectiveTurret
Device,TL-FieldDiaphragm,LeicaDMI,TL-FieldDiaphragm
Device,TL-ApertureDiaphragm,LeicaDMI,TL-ApertureDiaphragm
Device,IL-FieldDiaphragm,LeicaDMI,IL-FieldDiaphragm
Device,IL-ApertureDiaphragm,LeicaDMI,IL-ApertureDiaphragm
Device,FocusDrive,LeicaDMI,FocusDrive
Device,MagnificationChanger,LeicaDMI,MagnificationChanger
Device,IL-Shutter,LeicaDMI,IL-Shutter
Device,TL-Shutter,LeicaDMI,TL-Shutter
Device,XYStage,LeicaDMI,XYStage
Device,Transmitted Light,LeicaDMI,Transmitted Light

# Pre-init settings for devices
Property,Scope,AnswerTimeOut,250.0000
Property,Scope,Port,COM1

# Pre-init settings for COM ports
Property,COM1,AnswerTimeout,5000.0000
Property,COM1,BaudRate,19200
Property,COM1,DTR,Disable
Property,COM1,DataBits,8
Property,COM1,DelayBetweenCharsMs,0.0000
Property,COM1,Fast USB to Serial,Disable
Property,COM1,Handshaking,Off
Property,COM1,Parity,None
Property,COM1,StopBits,1
Property,COM1,Verbose,1

# Hub (parent) references
Parent,IL-Turret,Scope
```

```
Parent,ObjectiveTurret,Scope
Parent,TL-FieldDiaphragm,Scope
Parent,TL-ApertureDiaphragm,Scope
Parent,IL-FieldDiaphragm,Scope
Parent,IL-ApertureDiaphragm,Scope
Parent,FocusDrive,Scope
Parent,MagnificationChanger,Scope
Parent,IL-Shutter,Scope
Parent,TL-Shutter,Scope
Parent,XYStage,Scope
Parent,Transmitted Light,Scope

# Initialize
Property,Core,Initialize,1

# Focus directions
FocusDirection,FocusDrive,0

# Roles
Property,Core,Camera,BaumerOptronic
Property,Core,Shutter,Transmitted Light
Property,Core,Focus,FocusDrive
Property,Core,AutoShutter,1

# Labels
# IL-Turret
Label,IL-Turret,4,5-EMP
Label,IL-Turret,3,4-LGC
Label,IL-Turret,2,3-EMP
Label,IL-Turret,1,2-N21
Label,IL-Turret,0,1-I3
# ObjectiveTurret
Label,ObjectiveTurret,6,7-0x 0na
Label,ObjectiveTurret,5,6-0x 0na
Label,ObjectiveTurret,4,5-4x 0.1na
Label,ObjectiveTurret,3,4-63x 1.4na
Label,ObjectiveTurret,2,3-2x 0.07na
Label,ObjectiveTurret,1,2-20x 0.4na
Label,ObjectiveTurret,0,1-10x 0.3na

# Group: Camera
ConfigGroup, Camera, Default, Camera, Binning, 0
```

8.2 VAST command packets

All packets are sent to endpoint 0x01, from device BioSorter

Turn 3.6 degrees:

```
 01 16 13 00 00 00 00 02 31 31 61 4D 31 56 33 30   ........11aM1V30
 30 44 33 32 50 32 37 52 03 3B 00 00 00 00 00 00   0D32P27R.;......
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Pump 1 micro st

```
 01 05 05 00 00 00 00 2F 31 4E 31 0D 00 00 00 00   ......./1N1.....
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

01 05 04 00 00 00 00 2F 31 3F 0D 00 00 00 00 00    ......./1?......
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

 01 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00   .€..............
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

 01 05 0D 00 00 00 00 2F 31 4E 31 49 56 31 30 30   ......./1N1IV100
 44 31 52 0D 00 00 00 00 00 00 00 00 00 00 00 00   D1R.............
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Turn off traylight

```
 00 00 00 02 FF C0                                 ....ÿÀ
```

Turn on traylight

```
 00 00 00 02 CC 04                                 ....Ì.

 00 00 00 02 FE C0                                 ....À
```

500 micro meter right

```
 01 16 0D 00 00 00 00 02 31 31 61 4D 32 44 35 30   ........11aM2D50
 30 52 03 3C 00 00 00 00 00 00 00 00 00 00 00 00   0R.<............
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

500 micro meter left

```
 01 16 0D 00 00 00 00 02 31 31 61 4D 32 50 35 30   ........11aM2P50
 30 52 03 28 00 00 00 00 00 00 00 00 00 00 00 00   0R.(............
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
```

Turn off

```
 01 16 04 00 00 00 00 2F 32 54 0D 00 00 00 00 00   ......./2T......
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

 01 05 04 00 00 00 00 2F 31 54 0D 00 00 00 00 00   ......./1T......
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

 01 16 04 00 00 00 00 2F 31 54 0D 00 00 00 00 00   ......./1T......
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

 01 16 04 00 00 00 00 2F 31 54 0D 00 00 00 00 00   ......./1T......
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................
 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00   ................

 00 00 00 02 FF C0                                 ....ÿÀ
```

8.3 USB information

```
DEVICE ID 0bd2:2006 on Bus 002 Address 002 ================
 bLength                :   0x12 (18 bytes)
 bDescriptorType        :    0x1 Device
 bcdUSB                 :  0x200 USB 2.0
 bDeviceClass           :    0x0 Specified at interface
 bDeviceSubClass        :    0x0
 bDeviceProtocol        :    0x0
 bMaxPacketSize0        :   0x40 (64 bytes)
 idVendor               : 0x0bd2
 idProduct              : 0x2006
 bcdDevice              :  0x100 Device 1.0
 iManufacturer          :    0x1 Error Accessing String
 iProduct               :    0x2 Error Accessing String
 iSerialNumber          :    0x0
 bNumConfigurations     :    0x1
  CONFIGURATION 1: 160 mA ==================================
   bLength              :    0x9 (9 bytes)
   bDescriptorType      :    0x2 Configuration
   wTotalLength         :   0x35 (53 bytes)
   bNumInterfaces       :    0x1
   bConfigurationValue  :    0x1
   iConfiguration       :    0x0
   bmAttributes         :   0x80 Bus Powered
   bMaxPower            :   0x50 (160 mA)
    INTERFACE 0: Vendor Specific ============================
     bLength            :    0x9 (9 bytes)
     bDescriptorType    :    0x4 Interface
     bInterfaceNumber   :    0x0
     bAlternateSetting  :    0x0
     bNumEndpoints      :    0x5
     bInterfaceClass    :   0xff Vendor Specific
     bInterfaceSubClass :    0x0
     bInterfaceProtocol :    0x0
     iInterface         :    0x0
      ENDPOINT 0x1: Bulk OUT ================================
       bLength          :    0x7 (7 bytes)
       bDescriptorType  :    0x5 Endpoint
       bEndpointAddress :    0x1 OUT
       bmAttributes     :    0x2 Bulk
       wMaxPacketSize   :   0x40 (64 bytes)
       bInterval        :    0x0
      ENDPOINT 0x81: Bulk IN ================================
       bLength          :    0x7 (7 bytes)
```

```
 bDescriptorType  :    0x5 Endpoint
 bEndpointAddress :   0x81 IN
 bmAttributes     :    0x2 Bulk
 wMaxPacketSize   :   0x40 (64 bytes)
 bInterval        :    0x0
ENDPOINT 0x82: Bulk IN ================================
 bLength          :    0x7 (7 bytes)
 bDescriptorType  :    0x5 Endpoint
 bEndpointAddress :   0x82 IN
 bmAttributes     :    0x2 Bulk
 wMaxPacketSize   :  0x400 (1024 bytes)
 bInterval        :    0x0
ENDPOINT 0x86: Bulk IN ================================
 bLength          :    0x7 (7 bytes)
 bDescriptorType  :    0x5 Endpoint
 bEndpointAddress :   0x86 IN
 bmAttributes     :    0x2 Bulk
 wMaxPacketSize   :  0x200 (512 bytes)
 bInterval        :    0x0
ENDPOINT 0x8: Bulk OUT ================================
 bLength          :    0x7 (7 bytes)
 bDescriptorType  :    0x5 Endpoint
 bEndpointAddress :    0x8 OUT
 bmAttributes     :    0x2 Bulk
 wMaxPacketSize   :  0x200 (512 bytes)
 bInterval        :    0x0
```

8.4 VAI-box documentation

VAST BioImager Interface to External Device

18-7-2018 MHL; MINOR CHANGES 22-5-2023 MHL;

Overview

The “VAST BioImager” can communicate via pulses fed into its ports IN.1 and IN.2, or via pulses that it sends out of its ports OUT.1 and OUT.2. So, if you want to control the VAST from a PC, you must be able to generate and capture simple electrical pulses (TTL level, 5V).

However, modern PCs do not have general purpose I/O (GPIO) pins, so it is not possible to directly generate or capture simple pulses with a modern PC. Unless you use a simple device in-between the VAST and PC to do so. Such a device can be an Arduino. It can communicate with a PC via standard serial protocol (over a USB cable), while communicating with the VAST by generating and capturing pulses on its GPIO pins.

Serial communication between PC and Arduino enables the PC to send commands to the Arduino or receive information from it. Any PC programming environment will support serial communication, so this enables a wide range of PC programming tools to communicate with the VAST (via the Arduino).

Pulse-based communication between the Arduino and VAST is limited to whatever communication the VAST firmware allows. For designing the Arduino firmware, we used the VAST communication capabilities described in “VAST BioImager User’s Manual Rev. 1.06”.

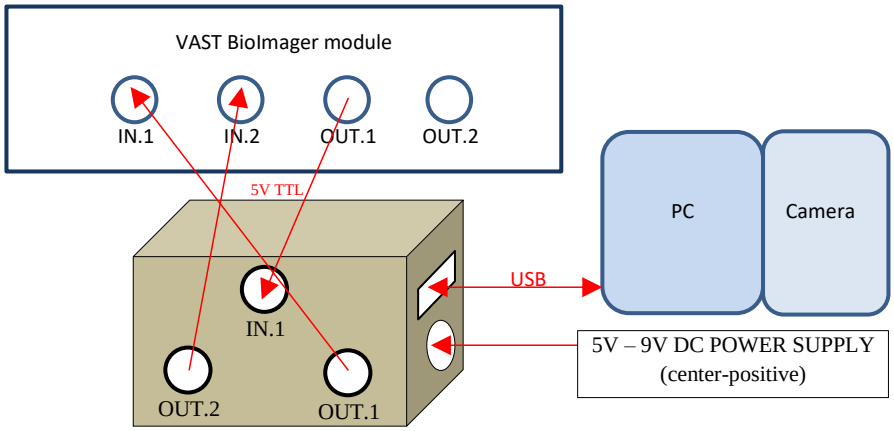
> **IN-1, IN-2, OUT-1, OUT-2**: TTL communication sites reserved for sending signals between VAST control module and an external microscope equipped camera. IN ports set to receive signals from outside source, OUT ports are set to send signals to an external device. See section External communications for more information. [PAGE 8]

The Arduino with support hardware was built into a self-contained box. We refer to this as the “grey box”. Updates of the Arduino firmware can be simply made using the standard Arduino development environment.

Basic Use

- Connect grey box via USB to the PC.
- Connect grey box to the VAST BioImager as shown below, using three BNC type co-axial cables.
- Connect grey box "POWER" to 5V – 9V DC power (center-positive).
- Make sure that the voltage stabilizer on top of the grey box is on (green LED should light up).
- Serial protocol over USB between PC and Arduino; default speed 115200 bps.
- Serial commands start with 2-char command identifier, followed by optional parameter, followed by a newline (char 10, a.k.a. '\n').

Connection Diagram



Connection notes:

VAST	"Grey Box"	Arduino	Description	Related serial command
IN.1	OUT.1	digital i/o pin 4	triggers VAST BioImager to proceed to next larva	`TT`
IN.2	OUT.2	digital i/o pin 5	triggers VAST BioImager to forward larva to output container, not to waste	`TS`
OUT.1	IN.1	digital i/o pin 2	indicates that larva is in place, ready for external imaging	`R1`
OUT.2	*n.c.*	digital i/o pin 3		

Serial Line Command List

Command	Parameters	Description
Commands sent by the Arduino over the serial line, to the PC		
R1	-	Received pulse from VAST port OUT.1
R2	-	Received pulse from VAST port OUT.2
OK	-	All is well at the Western front...
OK	<text>	Internal settings are returned in response to "X?" command
E0...E3	<text>	Error message
Commands sent to the Arduino over the serial line, from the PC		
TT	-	Transmit "trigger" pulse to VAST port IN.1
TS	-	Transmit "sort" pulse to VAST port IN.2
XH	-	Set received pulse detection from lines VAST OUT.1 and OUT2 to "high" (rising flank)
XL	-	Set received pulse detection from lines VAST OUT.1 and OUT2 to "low" (falling flank)
XP	200 (e.g.)	Set transmitted pulse width to 200ms
XD	100 (e.g.)	Set minimum separation of detected pulses to 100ms
XB	9600 (e.g.)	Set serial line baudrate. Calling command XB forces an automatic storage to EEPROM (similar to XS command). Accepted baudrates are 9600, 19200, 57600, and 115200. Requires Arduino restart to affectuate.
XS	-	Store current internal settings in EEPROM
XF	Y	Set internal settings to factory defaults (command "XFY")
X?	-	Print current value of all internal settings on serial line
XV	-	Print firmware version string on serial line

8.5 Manual zebrafish loading notes

The bubbles – Fish Loading (BK)

It is always important to use the purge (or wash?) option. That has to be done before you start to make sure all air-bubbles and old debris is cleared out of the tubing before you start. When you use the method where you take the tube out and load directly from a petri-dish there are a few tricks as well:

1) pause the loading in the software before you take the tube out, and make sure the loading is never running when the tube is not submerged to avoid sucking any air into the tube.

2) when the tube is submerged in the petri-dish with your larvae, push the reverse button a few times before you resume the loading. That way you can get rid of any tiny air bubbles at the very end of the tube.

3) when that is done, you load up your embryos from the petri-dish, with suitable distance between them. This is because sometimes one will be moving more freely than others through the tubing, and if one catches up to another one that increases the risk of them getting stuck.

4) when you have a number of larvae in the tube, you should pause the loading again before you take the tube out of the petridish. Pay attention as you reinsert the tube in the falcon tube with the little motor, if it gets caught or shaken a bit, a drop may come out at the very end and can then form an air bubble during the loading. If that happens, use the reverse function to push that air back out of the tube.

5) then resume the loading, and try to keep a count of how many larvae you have loaded, so you can better keep track of when it is necessary to load new ones again.

6) If you still have trouble after that, it is possible that a kink has formed in the tubing where the larvae get stuck. -So, try to check at the places where the tube bends to see if it's kinked and forms a clot. Otherwise, you can use the pump control options to increase the speed of the pumping, that should help by running a faster flow.

7) If you work with fixed larvae, you will find that they are very prone to get stuck. Then it is important to use some tween or another detergent to keep the larvae gliding. If you do this, you should also change the liquid in the feeding reservoir tank on the side of the machine. But bear in mind that that will increase the risk of air bubbles.

8) For fixed fish (Rico Bogarts) recommended to use 0,1 % pvp. This makes sticky fish glide better.