Bachelor Data Science and Artificial Intelligence

Zebrafish 3D reconstruction: optimization of the current pipeline for brightfield and fluorescent imaging

Maria Polityło

Supervisors:
Prof. dr. ir. Fons Verbeek & Ana Cristina Arcos Marin

BACHELOR THESIS

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

01/07/2026

Abstract

Automated image analysis is widely used in biological research because it is fast and reproducible. Zebrafish are a common model organism, and the VAST Bioimager can take axial images of zebrafish larvae at high throughput. These images can be used to build a 3D reconstruction, from which properties such as volume and surface area can be measured. The current reconstruction pipeline, however, is hard to use. It relies on hard-coded file paths, is spread across several locations, and has to be run from the terminal. On top of that, users have little control over important parameters. This thesis addresses these problems. A desktop application was developed to make the pipeline easier to use and to give users more control by letting them choose the input folder, the segmentation model, and the number of images. The app also supports batch processing and can run the pipeline from images that were segmented in a different way. In addition, experiments were carried out to find how many images are needed for an accurate reconstruction. Because the segmentation models did not generalize to the newer brightfield and fluorescent data, these experiments were run on the original brightfield datasets, using the reconstructed volume and surface area as measures of accuracy. The results show that more views did not always give a better reconstruction: the most accurate results came from around 20 to 50 views, while fewer views produced reconstructions that were too large (under-carving) and more views produced ones that were too small and sometimes broke into fragments (over-carving). The accuracy also depended on which views were selected, not only on how many, so a single firm minimum could not be established from the 100-image datasets alone. Larger datasets and improved segmentation models are needed to confirm these findings. Extending the pipeline to fluorescent imaging was explored, but a full 3D reconstruction could not yet be obtained.

Keywords: VAST Bioimager, axial-view microscopy, zebrafish imaging, 3d reconstruction, fluorescent imaging

Contents

1 Introduction

This section introduces the topic and motivation of the thesis. It explains the research context, states the problem with the current pipeline, presents the research question and sub-questions, and lists the main contributions. It ends with an overview of the rest of the thesis.

1.1 Research context

Biological research plays a crucial role in understanding mechanisms underlying development, disease, and treatments [oM19][MG17]. Imaging in particular can be a reliable source of information, allowing researchers to directly observe biological structures and processes in a non-invasive way. Techniques such as confocal microscopy and light-sheet fluorescence microscopy now enable imaging at high spatial and temporal resolution, even in living organisms [Com26][RKGS08]. These technologies generate large amounts of detailed data, however the key is extracting meaningful information from it.

To address this, image processing pipelines have been developed. Instead of manually annotating images, which is time-consuming and prone to variability, computational methods can process large datasets in a more efficient and reproducible way [ABB+25]. For example, segmentation algorithms can be used to identify specific structures within an image, while tracking methods allow researchers to follow cells or particles over time [ABB+25][TKL+13]. These approaches make it possible to process high volumes of data at once.

One commonly used model organism is the zebrafish (*Danio rerio*). One of their many advantages is their transparency during early stages of development, which makes them particularly suited for imaging [TZR+19]. It allows their internal structures to be observed without invasive procedures. Using fluorescent imaging, specific tissues can be labeled using markers such as green fluorescent protein (GFP), enabling the study of processes such as vascular development, organ formation, or neural activity in vivo [TZR+19] [GWK+17]. By applying image analysis methods, quantitative data can be extracted and provide insights that are relevant not only for zebrafish, but also for understanding similar biological processes in humans.

1.2 Problem statement

The focus of this thesis is the improvement of the existing 3D reconstruction pipeline for zebrafish. 3D imaging offers some advantages compared to traditional 2D imaging techniques. It captures the full spatial structure of the object, including the relative positions of its components, and enables accurate analysis of its shape, volume and surface area, that can be difficult using 2D imaging [GWK+17]. A confocal laser scanning microscope (CLSM) can be used to obtain 3D images of zebrafish [GVSV15]. However, on its own it does not support high-throughput screening. The VAST Bioimager was developed to address this limitation [Pul16][PMCK+10]. It enables automated loading of zebrafish specimens into a capillary that then rotates as the camera takes axial images of the specimen. The system can also be integrated with different imaging modalities, including brightfield microscopy, fluorescence microscopy, and CLSM [GVSV15][PMCK+10].

The axial images acquired by the VAST Bioimager can be used to reconstruct a 3D representation of the zebrafish, from which quantitative metrics such as volume and surface area can be derived. Such a reconstruction pipeline has already been developed by Guo et al at Leiden University [GVSV15]. The program has been improved over the years, however, several limitations remain in the current implementation. Firstly, the current pipeline is a Python script that may be overwhelming for users without programming experience, limiting its accessibility. Moreover, the implementation relies on hard coded file paths, which make it tedious to run and more prone to user errors. Lastly, so far, the pipeline has mainly been used on brightfield images. To fully use its potential, it should be extended to fluorescent imaging too. This would allow different metrics such as volume or surface area of different fluorescently labeled structures to be computed. These could then be easily compared between experimental groups, for instance to observe the changes in the volume of the vascular system under different conditions [SCE+00][KFS+22].

While the VAST Bioimager can take as many as 500 photos, the authors of the reconstruction pipeline found that as few as 20 axial views are sufficient to produce a good reconstruction and little improvement is achieved when increasing the number of views [GVSV15][GVSV17]. In brightfield imaging, increasing the number of views has a small impact on acquisition time. In contrast, fluorescence imaging is significantly more time-consuming, as each image requires a long exposure time, leading to much longer total imaging time per specimen compared to brightfield imaging. Taking 500 brightfield images can take as little as one minute, while the same number of fluorescent images can take up to an hour. For this reason, it is important to determine the minimal number of fluorescence images required to achieve a good reconstruction without compromising accuracy.

1.3 Research question

The main goal of this research is to make the reconstruction pipeline and the data flow more dynamic and efficient, and to expand it to fluorescence imaging and find the optimal settings. With this goal in mind, the following research question emerged:

How can the existing zebrafish 3D reconstruction pipeline be improved to enhance flexibility through parameterization to accommodate both brightfield and fluorescent imaging?

The research question can be further broken down into the following sub-questions:

1. How can the reconstruction script be redesigned to allow users without programming experience to operate it easily?
2. How can the pipeline be extended to support automated high-throughput processing of images from multiple modes?
3. What is the minimal number of images required to achieve accurate 3D reconstruction without compromising quality?

1.4 Contributions

The main deliverable of this thesis is a desktop app that would streamline the data processing and give users greater control over the 3D reconstruction. Another contribution is redesigning parts of the pipeline to make it more flexible and enable automated data flow to streamline the processing of images and ultimately be compatible with a database that could contain all the reconstructions produced. Lastly, research into how the accuracy of the reconstruction changes with varying numbers of images was conducted.

1.5 Thesis overview

This bachelor thesis, carried out at LIACS under the supervision of Prof. dr. ir. Fons Verbeek and Ana Cristina Arcos Marin, investigates the problems of the current reconstruction pipeline for zebrafish and explores how the number of input images affects reconstruction quality. The main goal is to design and evaluate a workflow that supports this process in a structured and reproducible way. The thesis is structured as follows. The current section introduces the problem, its relevance, and the research questions addressed in this work. Section 2 provides background information on the imaging setup, data structures and the reconstruction pipeline. Section 3 describes the methodology, including the data flow design, the application design, and the experiments carried out with different numbers of input images. Section 4 presents the results of the app implementation, image processing techniques that were used on the images, and the results of running the reconstructions on varying dataset sizes. Finally, Section 5 discusses the results and compares them to literature findings. Section 6 contains the conclusion and answers the research question, while Section 7 outlines directions for future work.

2 Background

This section gives the background needed to understand the rest of the thesis. It explains why zebrafish are used as a model organism, how the VAST Bioimager works, and how the current reconstruction pipeline turns axial view images into a 3D model.

2.1 Zebrafish as model organisms

Zebrafish (*Danio rerio*) are small tropical freshwater fish that originate from South Asia. They are among the most widely used model organisms in biological research due to their fast growth, ease of breeding, frequent spawning and transparency. Zebrafish larvae develop within 72 hours post-fertilization, and their bodies remain transparent during early stages of development [TZR+19]. In addition, around 70% of zebrafish genes have at least one human ortholog, which makes them a particularly valuable model for studying human biology and disease. [TZR+19][HJV+16]

Zebrafish have been used in research in a variety of fields [LC07]. They have become an established model in cancer research, where tumors can be induced and imaged in the living fish and used to screen for potential drugs [WRZ13]. They can also be used to study neurodegenerative disorders such as Parkinson's and Alzheimer's disease, as key features of these conditions, including protein aggregation and neuronal loss, can be reproduced in the fish nervous system [CKSP22]. Moreover, transparency and rapid development of zebrafish, together with their suitability for high-throughput screening, make them particularly good models for drug research [CAF+19]. Because humans and zebrafish share many biological pathways, findings from zebrafish frequently translate to human medicine, which makes them a powerful and cost-effective research tool.

2.2 VAST Bioimager

VAST stands for Vertebrate Automated Screening Technology. The VAST Bioimager is a system that has been developed specifically to enable high-throughput screening of zebrafish [Pul16]. It can automatically load a single zebrafish larva from a reservoir and position it within the capillary using stepper motors. The capillary is then rotated while the larva is kept in a fixed position and axial images of the specimen are taken. A major advantage of VAST is that not only does it offer its own camera, but it can also be mounted on top of other microscopes - such as brightfield or fluorescent microscopes. The VAST imaging system can be seen in Figure 1.

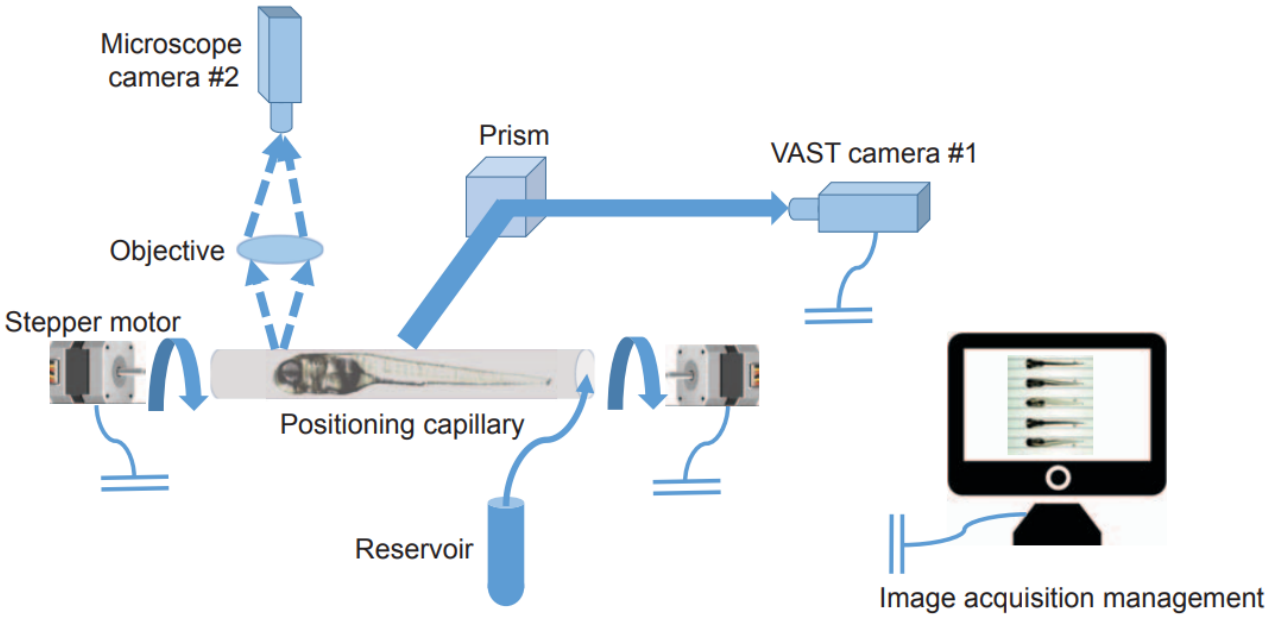


Figure 1: VAST imaging system [GVSV17]

2.3 Current reconstruction pipeline

The reconstruction pipeline consists of three main steps: image segmentation, camera parameter optimization, and the 3d reconstruction, which can be seen in Figure 2.

A folder with axial images of a single specimen is input, and the program outputs a number of files: folder of segmented images in .npz format, a JSON file pointing to the segmented images (an intermediate file that is used within the pipeline), a 3D mesh reconstruction in .ply format and a JSON file with important details such as the camera parameters and file paths, which will be discussed in later sections.

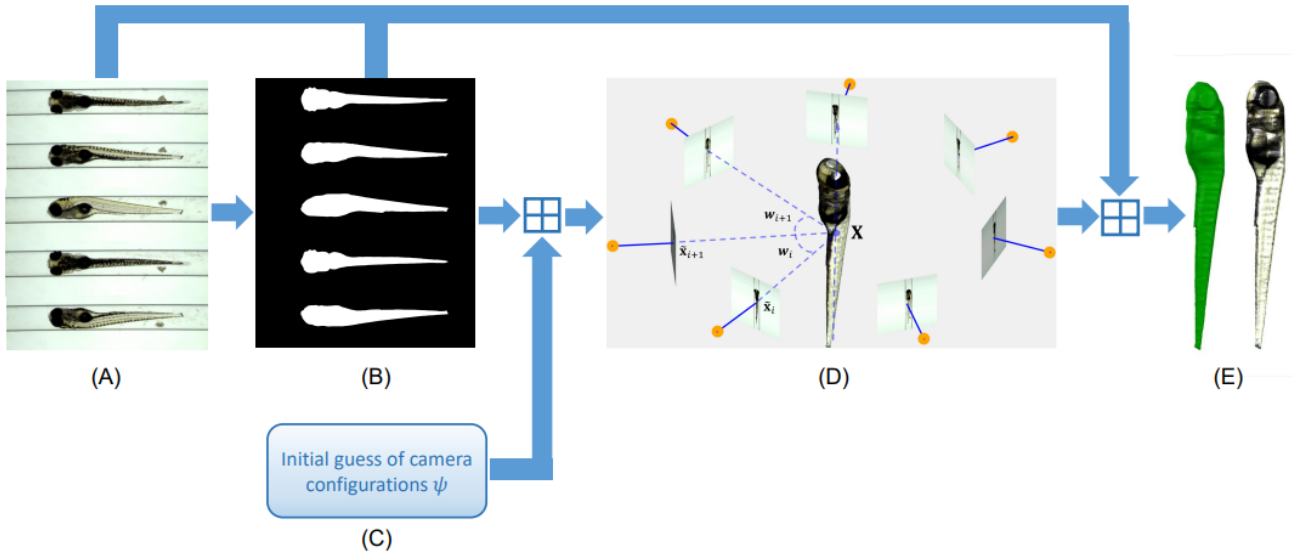


Figure 2: Zebrafish reconstruction pipeline. A) Input from the VAST Bioimager; B) Output of the segmentation step; C) Camera parameters optimization; D) 3D reconstruction; E) Reconstructed zebrafish [GVSV17]

2.3.1 Segmentation

The first step of the reconstruction pipeline is image segmentation, in which the outlines (silhouettes) of the zebrafish are extracted from each image. The raw images are converted into binary masks, as illustrated in Figure 2B.
This step can be performed using various methods, such as active contours [CKS95], mean shift algorithms [CM02], or convolutional neural networks (CNNs) [RFB15]. In this thesis, CNN-based methods, and specifically U-Net, are primarily used for segmentation, using models developed in previous years. However, it is important to note that segmentation can be imperfect, making it the most sensitive stage of the entire pipeline. Since all subsequent steps depend on the quality of the segmented images, accurate segmentation is crucial for achieving reliable 3D reconstruction.

2.3.2 Camera Parameters Optimization

After image segmentation, the camera parameters must be optimized (Figure 2C). These parameters describe how each 2D image is mapped into 3D space, and include intrinsic properties of the camera (such as focal length, scaling, and image center) as well as extrinsic properties (such as rotation and translation relative to the object). Together, they define the camera projection model [GVSV15]. During imaging, the capillary with the zebrafish rotates, meaning that each image is taken from a slightly different viewpoint. For reconstruction, this can be seen as the camera moving around the object (Figure 2D). To combine all views into a single 3D representation, these camera parameters need to be estimated accurately.

The initial parameters come from the imaging system, but they are only rough estimates. To improve them, the voxel residual volume (VRV) maximization algorithm is used [GVSV15]. This

method projects the segmented silhouettes from all views into a shared 3D voxel space using the current parameters. It then measures how well these projections match by looking at how much they overlap. The parameters are adjusted step by step to maximize this overlap, so that all silhouettes are as consistent as possible with one 3D shape. This optimization is done using an iterative method (Nelder–Mead) [GVSV15].

2.3.3 3D reconstruction

The final step of the pipeline is the 3D reconstruction (Figure 2 D). This is done using visual hull reconstruction, where the 3D shape of an object is estimated from the silhouettes seen from different perspectives [GVSV15]. Using the calibrated camera parameters, the silhouettes can be projected back onto a common 3D voxel space, forming a set of cone-shaped volumes. The reconstructed object is then obtained by taking the intersection of these volumes. In other words, only the voxels that are consistent with all silhouettes are kept, while the rest are removed.
This process is also performed through a number of iterations. Starting from an initial volume that fully contains the object, voxels are gradually removed if they do not agree with the silhouettes from each view. After processing all views, the remaining voxels form the final 3D shape, from which properties such as volume and surface area can be derived.

3 Methodology

This section describes how the work was carried out. It covers the imaging set-up and the file structures used to organize the data, the design of the desktop application, and the experimental design for testing how the number of images affects the reconstruction.

3.1 Imaging set-up and file structures

The purpose of this thesis is to streamline the processing of zebrafish images. Each zebrafish can be imaged in multiple ways. As a control, the VAST camera is used, while the specimens can also be imaged with brightfield and fluorescent microscopy. Moreover, different magnification lenses can be used. It is therefore essential to design file structures that would organize the images from all of these modalities. For every specimen, the images from every technique are saved in folders with corresponding names, all of which are saved in a single folder with the number of specimen imaged that day (ex. 001), which is saved in a folder with the imaging date in a YYYYMMDD format. An example file structure can be seen in Figure 3.

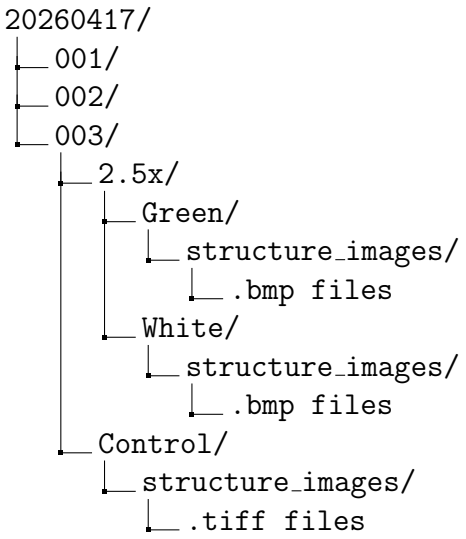


Figure 3: Example folder structure of dataset collected on 20260417. Images from microscope are in .bmp format, while those obtained with the VAST camera are .tif or .tiff.

After a folder is processed, the resulting folder structure can be seen in Figure 4.

The current pipeline only accepts a single folder that contains the images within a subfolder titled "structure_images". This is one thing that needs to be updated to make the implementation more flexible and robust. Moreover, to make the output more explicit, another folder will be added to the output. The segmentation step can be prone to error, however this can remain undetected until the very last part of the segmentation, when the 3D object is created. Therefore, to prevent this, an additional folder can be created in which the segmented images (saved as .npz) are also stored as PNGs. This way, the images could be viewed without any special program, in contrast to

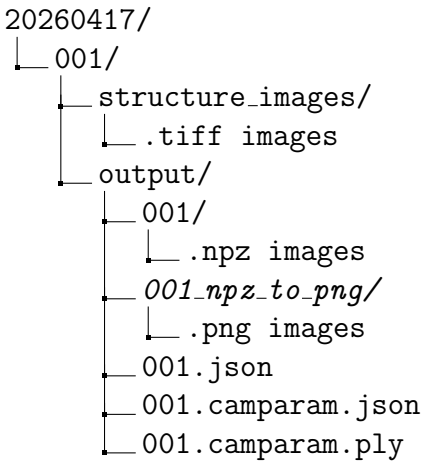


Figure 4: Example folder structure after going through the pipeline.

the npz files. This will allow the user to have even more control and rerun the pipeline with a different segmentation model if necessary. The suggested updated file structure can be seen in Figure 5.

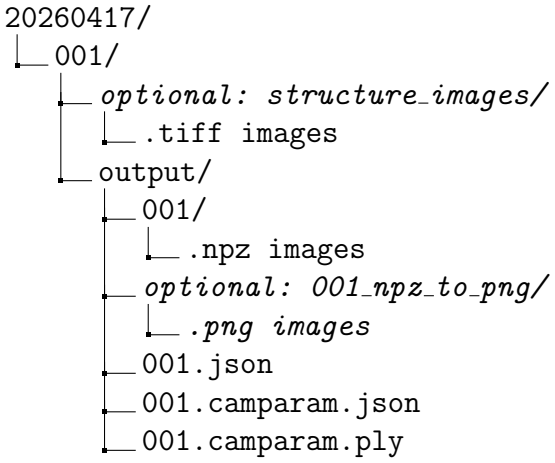


Figure 5: Proposed adapted folder structure after going through the pipeline. Changes to the current implementation are marked in italics.

Moreover, currently the two most important outputs of the pipeline are the 3D mesh (.ply file) and the JSON file containing the optimized camera parameters. However, this is not ideal for analysis of the results. Currently, to analyze the metrics such as volume and surface area, the user has to open the .ply file in a viewer (e.g. Meshlab), compute the metrics there, and then copy them to their desired location. This process is therefore hardly suitable for high-throughput imaging. Therefore, all relevant metrics will thus also be written to JSON file, allowing users to access all relevant information in one place. An outline of the current parameters within the JSON files can be seen below.

- `data_type`: format of the data resulting from the segmentation (usually npz, a compressed NumPy archive).

- `data_path`: filesystem path to the dataset of segmented images
- `f`: focal length of the microscope
- `alpha`: rotation around the X-axis
- `gamma`: rotation around the Y-axis
- `phi`: rotation around the Z-axis
- `w`: list of calculated angles per each image, spanning a full revolution (360 degrees)

These are the features that are present in the current JSON files. However, to enable processing of the results, the following will be added to the JSON files:

- `volume`: reconstructed 3D volume (in micrometers cubed) of the specimen body, obtained from the marching-cubes visual-hull pipeline.
- `surface_area`: surface area (in micrometers squared) of the reconstructed mesh.
- `n_vertices`, `n_faces`: describe the size and complexity of the reconstructed triangular mesh.
- `n_views`: number of camera views (silhouettes) actually used for the reconstruction.

By adding these parameters, the resulting JSON file should contain enough information to enable automated data analysis.

3.2 App design

3.2.1 Requirements and implementation choices

The app was developed in Python using CustomTkinter, due to its relatively simple interface design and suitability for users with limited experience in app development. In addition, several standard Python libraries were used for file handling and system operations, including `os`, `pathlib`, `shutil`, and `sys`. The `math` library was used for basic calculations, and `threading` was included to allow parts of the pipeline to run without blocking the user interface. The scikit-image (skimage) library was also used for image processing. The pipeline also depends on a number of external packages and environment-specific dependencies, which are required for segmentation and reconstruction. These can be found in the virtual environment setup.

3.2.2 Design

The main limitation of the current implementation is its lack of flexibility. Currently, users need to interact directly with the Python code and manually modify inputs, which makes the pipeline inaccessible and error-prone. In addition, user control over the pipeline parameters is limited.
To address this, an app was designed to provide a more user-friendly interface with greater control over the input parameters. The following options were implemented:

- Selecting an input folder through a file explorer instead of manually entering file paths

- Choosing between available segmentation models
- Setting the number of images to be used for reconstruction

Previously, users had to manually paste file paths into the terminal and adjust parameters directly in the code, which made the workflow inefficient and difficult to reproduce.

In addition, the original pipeline only allowed processing one folder at a time, meaning each dataset had to be handled manually in a separate run. To improve this, two operating modes were introduced: "single mode" and "batch mode". In single mode, the user can select one folder and run a single reconstruction, which is mainly intended for quick testing of individual datasets. Batch mode, on the other hand, is designed for automated processing. It allows the user to select a main directory, after which the program recursively searches for all relevant image folders. The user can then select which folders should be processed (for example, to skip already processed datasets) and define the aforementioned parameters for each selected folder. Lastly, as the segmentation models are not always reliable, the pipeline was adapted to enable running it from already segmented images. This provides more possibilities in case the initial segmentation does not work very well. It is also possible to convert the .npz files generated by the model to png inside the app. This makes the pipeline more reliable, as users are able to validate the segmentation output early on and decide whether or not it is good enough to proceed with the rest of the pipeline.

3.3 Experiments

After the app was implemented, experiments were run to find the minimum number of images needed for an accurate reconstruction. For each dataset downsampling was performed, such that only a percentage of the dataset was used for the reconstruction. The preliminary percentages to be tested are 5, 10, 20, 25, 50, 75 and 100%. However, these values may be adapted according to the experimental findings. While a maximum of 500 images can be taken for a single specimen using the VAST Bioimager, the quality of the images obtained in this way was found to be too poor for analysis. Thus, the datasets selected for analysis included: 14 datasets of 100 brightfield images, 11 datasets of 360 brightfield images and 10 datasets of fluorescent images (7 datasets of 101 red images, 2 datasets of 500 yellow images and 1 dataset of 500 blue images - from different fluorescent dyes). All images were of zebrafish three days post-fertilization (3dpf).

The reconstructions were evaluated by the accuracy of volume and surface area of the 3D models. The reconstructions from the different samples were then compared against findings from previous work [GVSV17]. Moreover, it has been shown that for visual hull reconstructions, more images tend to lead to better reconstructions [Lau94][SPM04]. Thus, the goal of the experiments is to see the minimal number of views required for an accurate reconstruction.

For analysis of the 3D reconstruction, the following Python libraries have been used: `pandas` for data collection and management, `matplotlib` for plotting the figures and `trimesh` for analysis of the 3D mesh.

4 Results

This section presents the results of the work. It first describes the implementation of the file handling and the adjustments made to the pipeline output. Then, it explains how the desktop application that was built and the image preprocessing steps that were needed for the new data, and finally the experiments on how the number of images affects the reconstruction.

4.1 Adjustments to file and folder structures

The first adjustment made to the pipeline was to the accepted file structures. As explained in Section 3.1, the pipeline only accepted a very strict naming convention. The input folder to the pipeline had to be called "`structure_images`". This was fixed to a more flexible version, where the pipeline accepts any folder, as long as it contains images immediately inside of it. This was done to prevent any recursive processing of other images potentially nested within the input folder.

Next, the JSON file was adjusted to contain more information about the reconstruction, including the volume, surface area, number of vertices and faces, and the number of views used for the reconstruction. Both volume and surface area in the JSON file are already given in micrometers cubed and squared, respectively. The original code already included a calibration of the voxel volumes such that the 3D mesh is automatically rescaled to represent volume and surface area in micrometers cubed and squared.

4.2 App implementation

4.2.1 Objects and classes

To make the implementation more robust, the classes `DatasetItem`, `SingleDataset` and `BatchDataset` were introduced. The implementation can be seen in Figure 6.

DatasetItem stores all relevant information for a single dataset, such as the folder path, selected model, number of images, and whether it is currently selected for processing. The image count is determined automatically when a folder is provided, and the number of selected images defaults to the total count. SingleDataset is a single DatasetItem, while BatchDataset is a collection of such items. BatchDataset loads all valid image folders from a root directory and creates a DatasetItem for each of them. These items are stored in a list, making it easy to iterate through them in batch processing and directly link them to the GUI list of datasets.
This structure was chosen to keep the data model consistent with the GUI, where users can select or unselect individual datasets. Because each DatasetItem stores its own state (such as whether it is selected), changes in the GUI, like unselecting a dataset in batch mode, are automatically updated.

```python
class DatasetItem:
    def __init__(self, folder_path="", model=""):
        self.folder_path = folder_path
        self.model = model

        if folder_path:
            self.image_count = count_images(Path(folder_path))
        else:
            self.image_count = 0

        self.selected_img = self.image_count

        self.selected = True # only important for batch mode

class SingleDataset:
    def __init__(self, folder_path=None):
        self.item = None

        if folder_path:
            self.load(folder_path)

    def load(self, folder_path):
        self.item = DatasetItem(folder_path=folder_path)

class BatchDataset:
    def __init__(self, root_folder=None):
        self.items = []

        if root_folder:
            self.load_from_root(root_folder)

    def load_from_root(self, root_folder):
        folders = find_image_folders(root_folder)

        for folder in folders:
            item = DatasetItem(folder_path=folder)
            self.items.append(item)
```

Figure 6: Python classes to create the dataset objects and store the information.

4.2.2 App layout

The app was designed to facilitate data analysis. Upon running the app, two modes are offered - single mode or batch mode (Figure 7). In case the user already has segmented images (i.e. the binary masks) obtained in a different way, they can also continue by pressing the last button. Single mode can be used to run a reconstruction from a single dataset of images (Figure 8).

Once a user selects a folder, the app automatically checks whether the folder contains images of the accepted format (.bmp, .tiff or .tif) and if so, how many. Then, the user is able to select the number of images to be used for reconstruction using the slider (Figure 9). The program then recalculates the number of images that will actually be used - for instance, if a user would like to only use 100 out of 360 images for the reconstruction, the program would need to take every 3.6th image, which is not possible. Therefore, the step size is rounded down to 3, resulting in image selection of 120 images, which is stated explicitly after user input to give the user a chance to change the number if needed. Then, the "Run reconstruction" button can be pressed to start the pipeline. Throughout the reconstruction, the outputs regarding the process from the original program can be seen in a box at the bottom of the app (Figure 10). This way the viewer can track the process of the reconstruction all in one place.

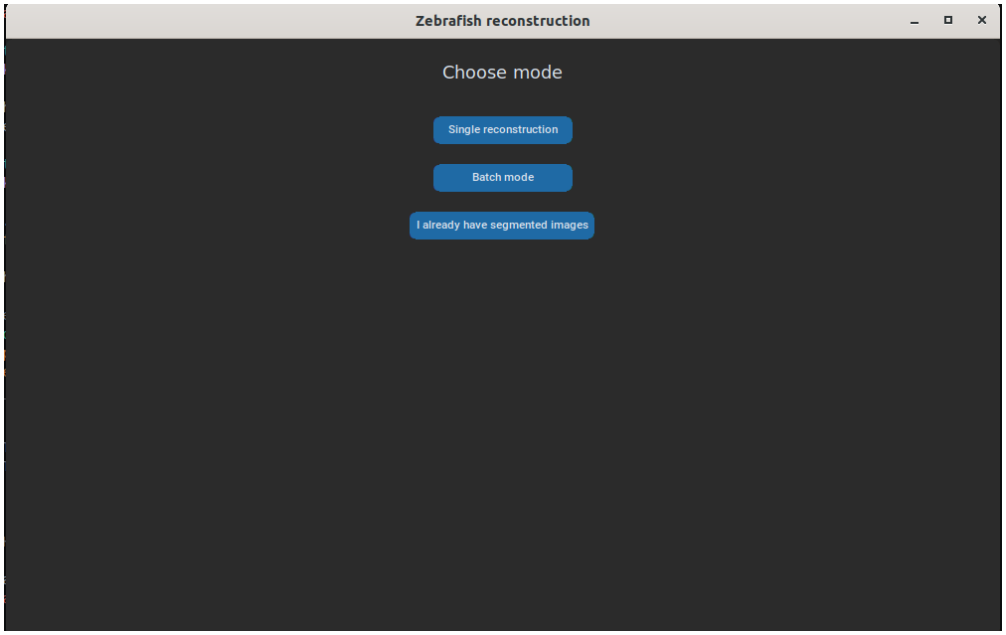


Figure 7: Home screen of the app, offering 2 modes to choose from and an option to continue with segmented images.

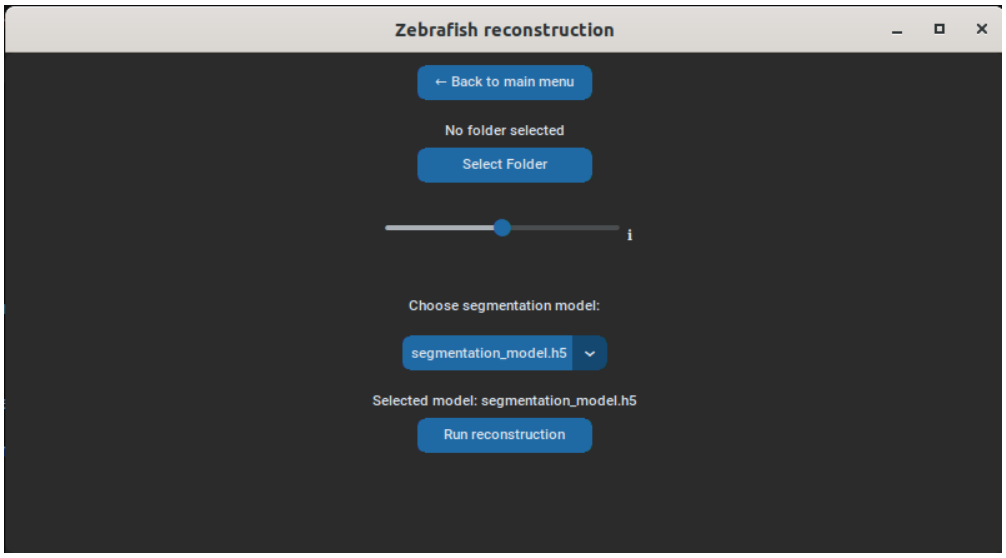


Figure 8: Initial state of the single mode screen.

The batch mode allows the user to choose a single parent directory (Figure 11). Then, the program recursively searches the subdirectories and outputs all the image folders found (Figure 12). These are then displayed and the user can select/unselect any folder. Other features are similar as for single mode - the user can select a segmentation model and number of images used for the reconstruction for every image. Once the run button is pressed, the reconstruction is performed for every image folder.

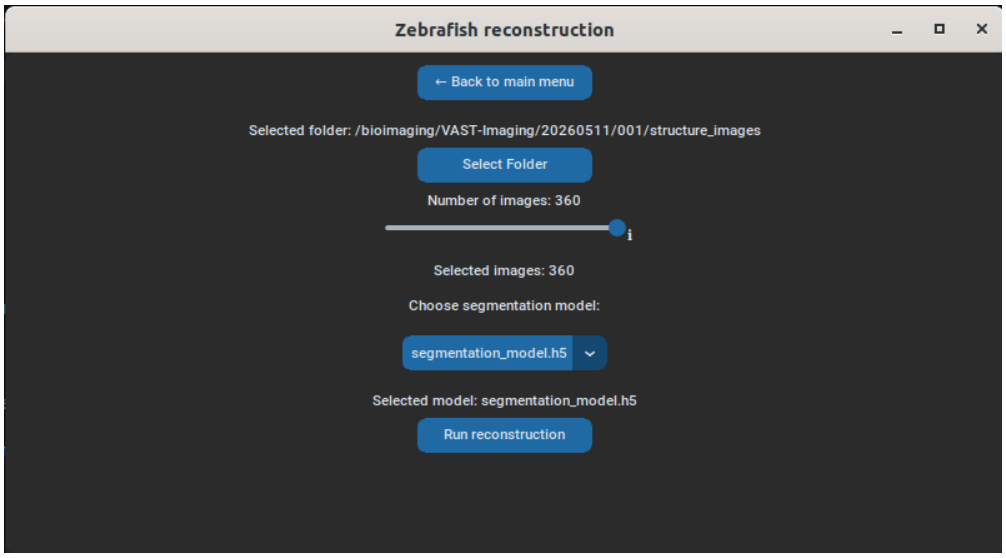


Figure 9: State of the single mode screen after the user has adjusted the parameters to their needs.

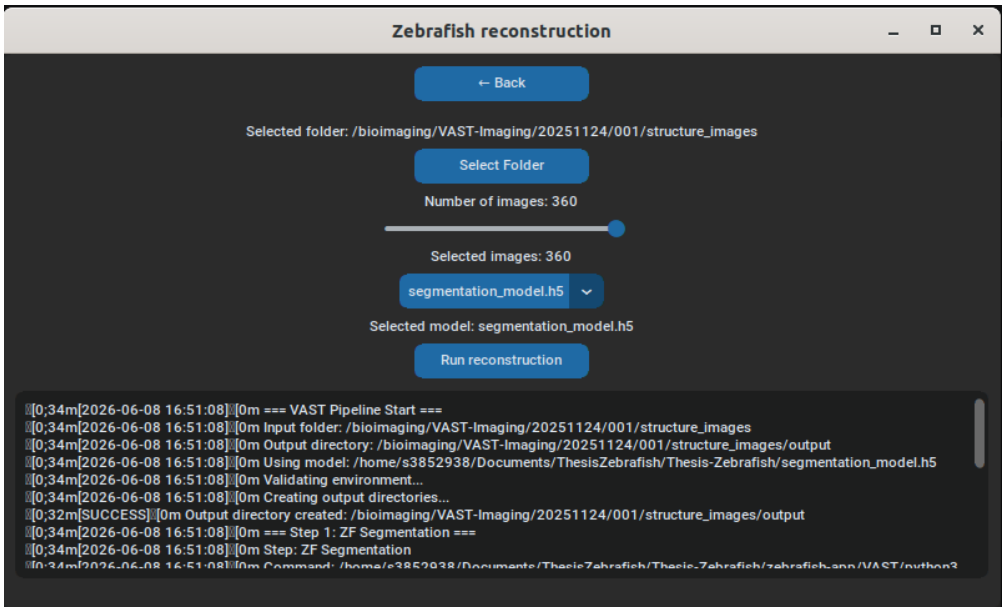


Figure 10: Output is displayed in single mode to allow the user to track the progress.

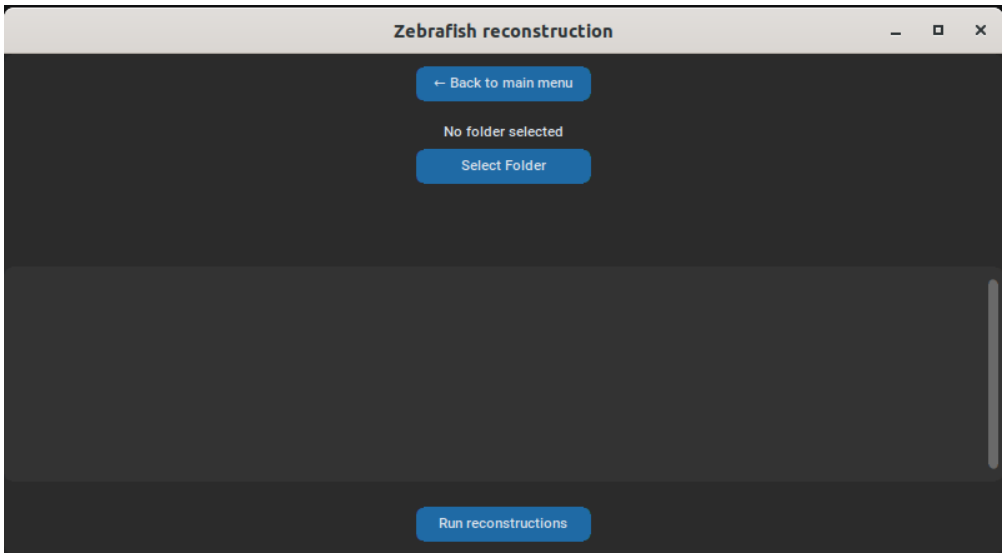


Figure 11: Start screen of the batch mode.

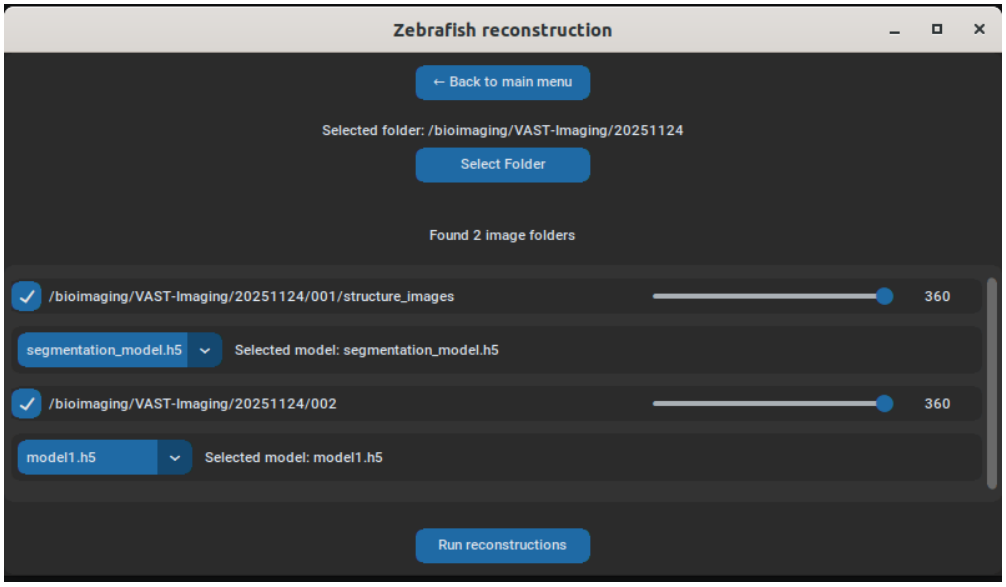


Figure 12: Batch mode after folder selection. All detected image folders are displayed and parameters can be chosen for each dataset.

4.2.3 Image selection

When a number of images is selected that, after adjustment, is not equal to the original number, a subset folder is created. The folder is made within the original input folder and contains the selection details in the name (e.g. "subset100of360"). The specified number of evenly spaced images from the original dataset is copied into that folder. By default, after the reconstruction is complete, the images are deleted from the subset folder to avoid storing the same images twice (the images are never deleted from the original input folder). If a user would like to confirm that the images were selected properly, this can also be changed by adjusting a variable `cleanup` in the app code.

Initially, there was a problem with the image selection, as the copied images were not evenly spaced. This was found to be due to the fact that the images were sorted alphabetically. When saved, each image has a format where the name is eventually followed by an underscore, three digit number and the file extension - e.g. *specimen_name_001.tiff*, *specimen_name_002.tiff* etc. However, when sorted alphabetically, the order of first few images turned out to be 1, 10, 100. This has been solved using a regular expression:

```
pattern = re.compile(r'_(\d+)\.(tif|tiff|bmp)$', re.IGNORECASE)
```

This expression looks for the ending (number and file extension) of the file name. Then the .group(1) function can be used to filter out only the three digit number, which can then be used to sort the images numerically. After that, every n^{th} image can be selected correctly.

4.2.4 Continue with segmented images

Segmentation is the most important step of the reconstruction pipeline as it determines the quality of the reconstructed mesh. The segmentation models are not always the most reliable, therefore, sometimes it is best to segment the images in a different way. It is also useful to, if using the segmentation models, be able to immediately verify whether the masks are appropriate. Therefore, upon choosing "I already have segmented images", the user is presented with two options as seen in Figure 13.

When the user chooses to continue with the reconstruction, they are presented with a screen similar to the one in single reconstruction mode (Figure 14). They can once again choose a folder and the number of images used for the reconstruction, and they can see the output in the box at the bottom of the app. This version does not support batch processing of multiple folders, as by design, the images should be run through the pipeline from start to finish. This option is only available to give the user more options, in case the segmentation models fail. However, in that case, ultimately the models should be retrained as usually segmenting the images with other methods requires human supervision, which makes the pipeline no longer automated.

When a user chooses to view the segmented images instead, they can click on "Show segmented images as png". This step can be performed independently or during a running reconstruction to spot inaccurate masks immediately. When an input folder is selected, the program then searches for npz and the user can convert them to png. When clicking the "See folder" button, the user is redirected to the output folder with png images. These steps can be seen in Figure 15.

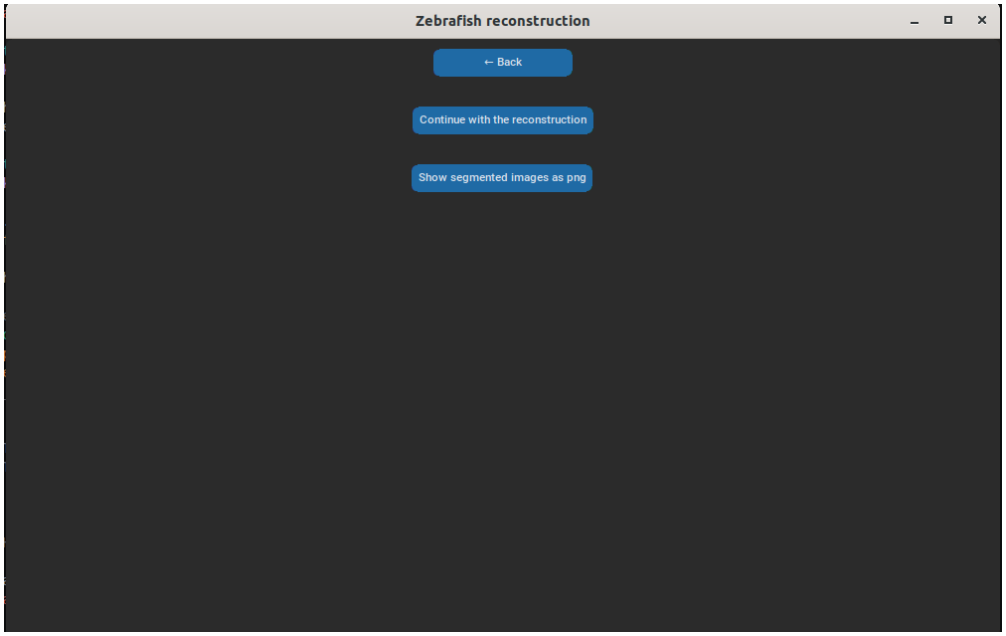


Figure 13: Upon clicking "I already have segmented images", the user can either continue with the reconstruction or view the masks by converting the files to png format.

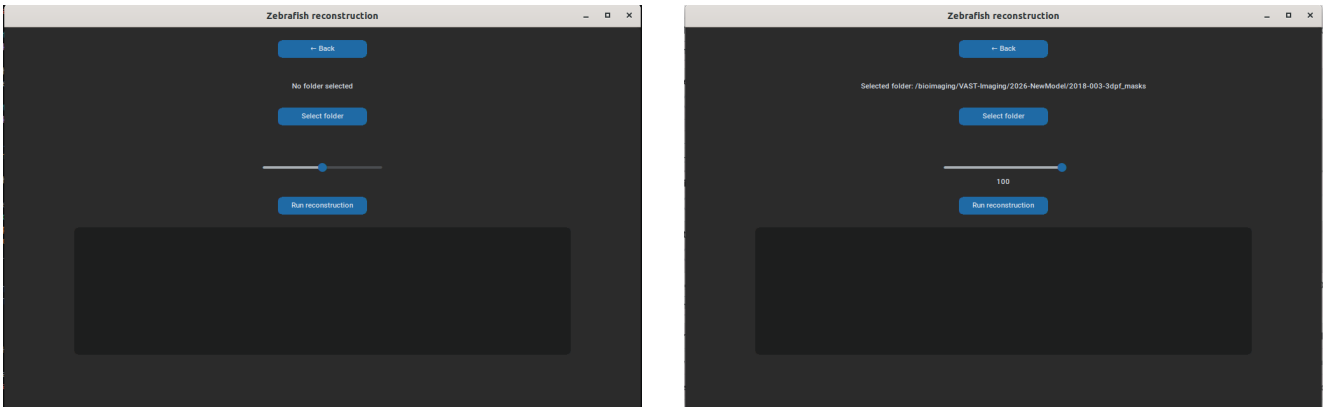


Figure 14: "Continue with the reconstruction" screen before and after choosing an input folder.

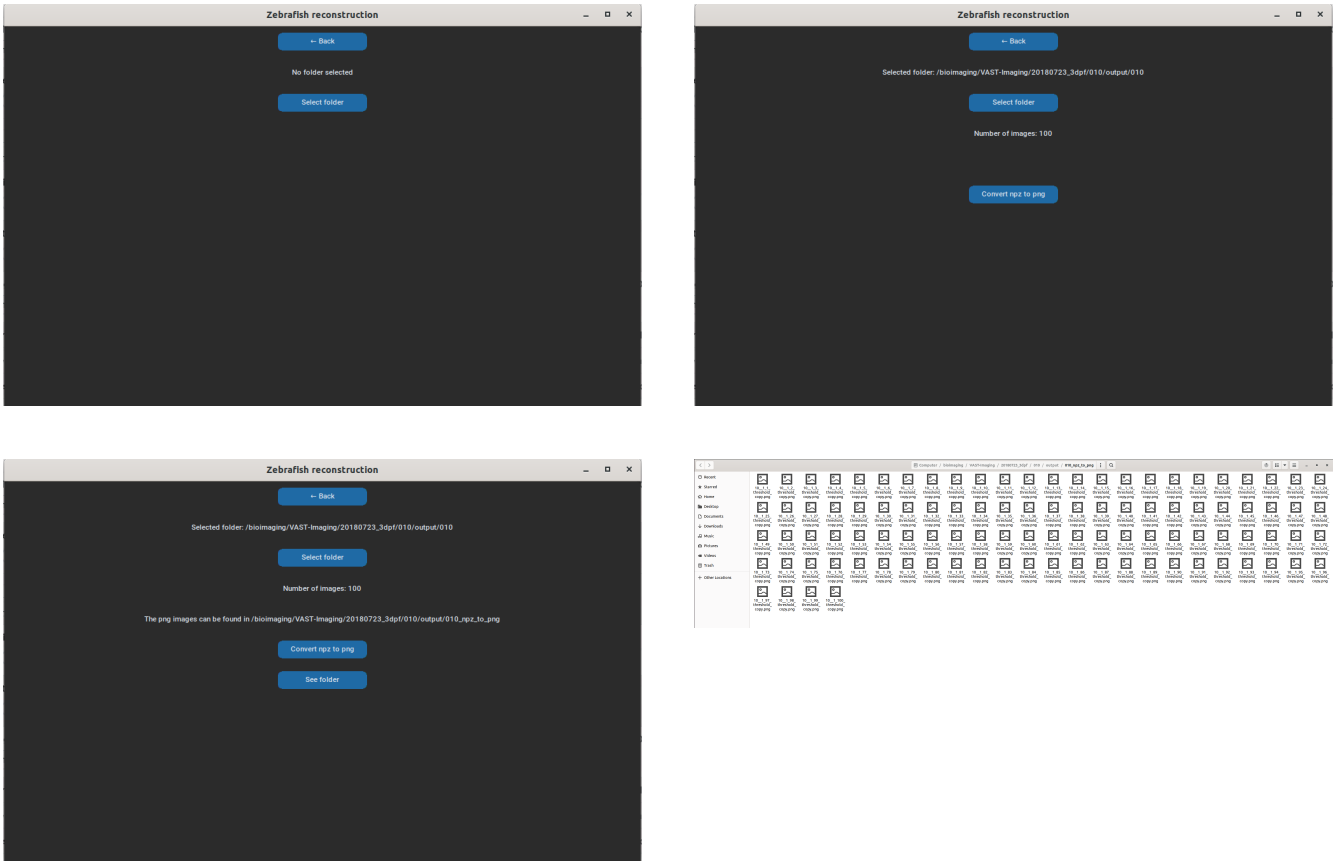


Figure 15: Example usage of the npz to png conversion. Upon selecting a folder with npz files and converting them to png, the user can be redirected to file manager to view the images.

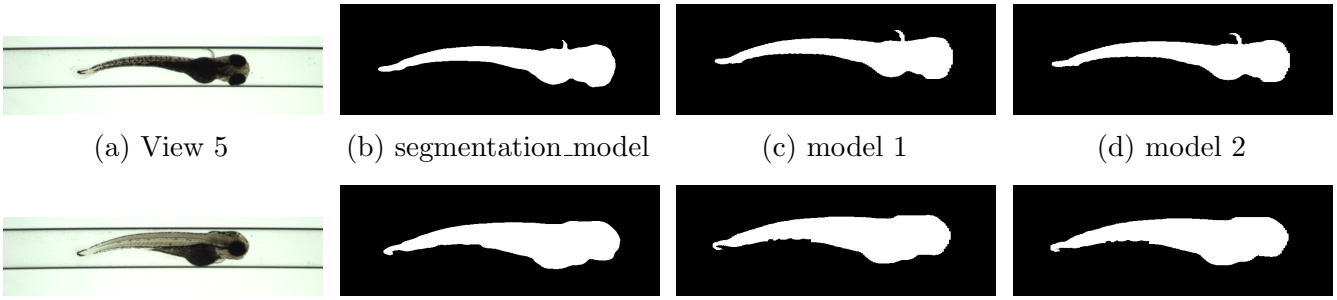
4.3 Generalization to new and fluorescent data

The segmentation models used in this pipeline were trained in previous years on a set of brightfield images that were all acquired in the same period and with the same microscope settings. The 14 datasets of 100 images that are used later in this thesis come from that same series, which is why they can be segmented reliably. When the models are applied to the newer brightfield datasets and to the fluorescent images, however, the segmentation is of much lower quality. This section first shows how the models perform on the old and the new data, then examines the differences between the two, and finally describes the techniques that were used to try to make the new data usable.

4.3.1 Segmentation of new brightfield data

To test how well the models generalized, 11 datasets of 360 brightfield images were selected. These were sampled to represent every n^{th} image, resulting in subsets of 360, 180, 120, 90, 72, 60, 51, 45, 40 and 36 images. However, when these datasets were first run through the pipeline, it became clear that the new images could not be segmented well by the existing models.

Figure 16 shows the binary masks produced by the available models for two views from one of the original datasets. While there are small differences between the models, the outline of the fish is clearly visible in all cases, so the masks can be used for the rest of the reconstruction. Figure 17 shows the same models applied to the newer data. Here, the segmentation is of very poor quality and the fish can no longer be separated from the background.



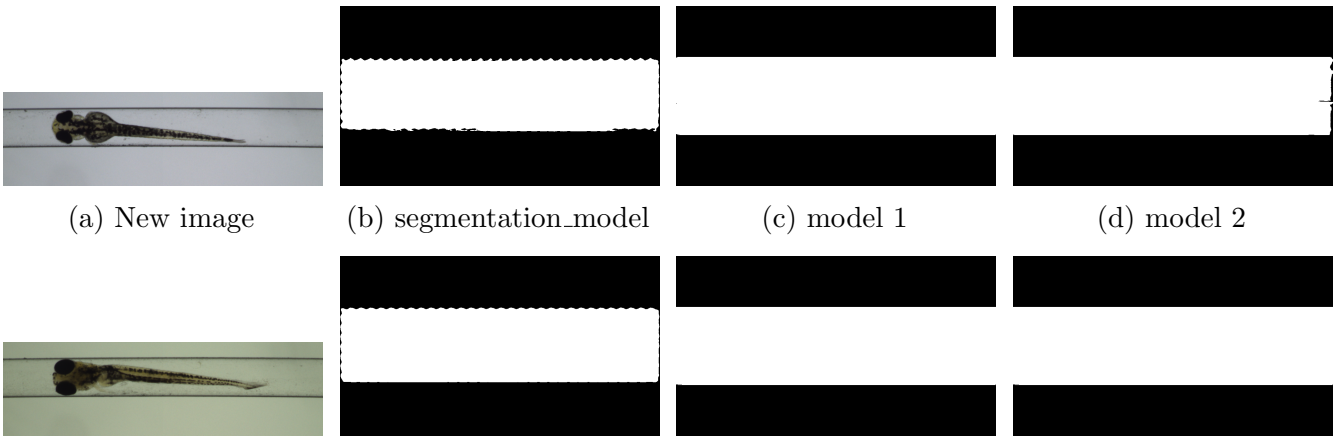
(a) View 5 (b) segmentation_model (c) model 1 (d) model 2

(e) View 27 (f) segmentation_model (g) model 1 (h) model 2

Figure 16: Comparison of the binary masks created from two views from one of the original datasets by the available segmentation models

This is likely due to a domain shift between the training data and the new data. The main difference seems to be the color balance of the images: the original images look noticeably greener and brighter than the newer ones. In an initial attempt to fix this, a new series of images was acquired with different microscope settings to make them appear greener (Figure 17e). However, as Figure 17 shows, this did not improve the masks. As retraining the models was not immediately feasible, the differences between the old and new images had to be examined more closely so that image processing could be used to make the new data compatible with the existing models.

To confirm that the color balance was the main difference, color histograms were plotted for the original and the new images (Figure 18). A digital image can be represented as an array of shape



(a) New image (b) segmentation_model (c) model 1 (d) model 2

(e) Greener image (f) segmentation_model (g) model 1 (h) model 2

Figure 17: Comparison of the binary masks created from images from two new datasets by the available segmentation models

`(height, width, channel)`, where the first two values give the position of a pixel and the third stores its intensity (from 0 to 255) per channel (red, green, blue) [GW18]. A color histogram shows how these intensities are distributed across all pixels in each channel [GW18].

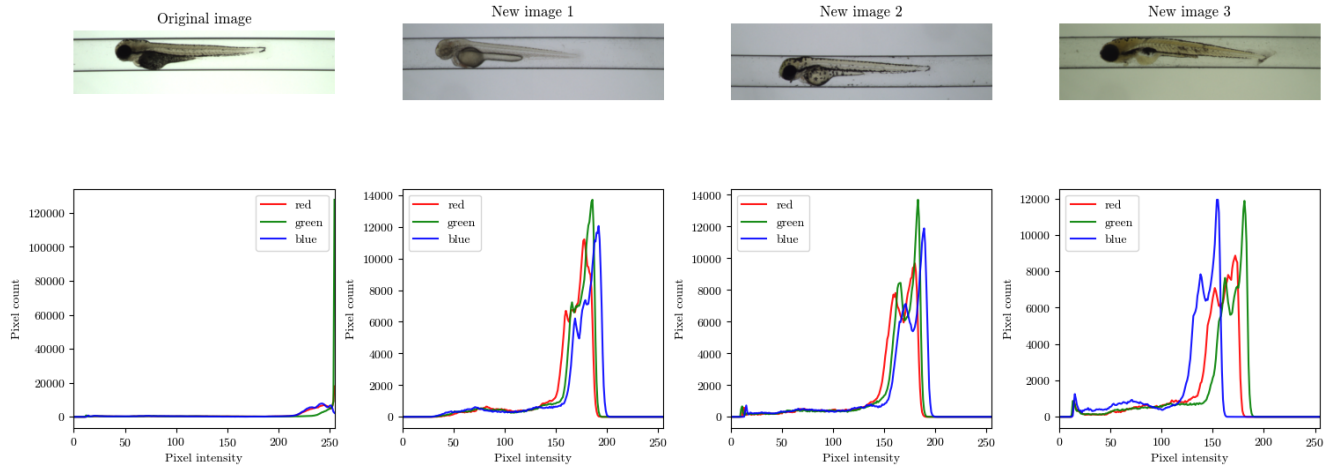


Figure 18: Color histograms of the original image and images from newer datasets.

The results show that the original images are highly saturated: in all channels the intensities are at the top of the scale, and the green channel is at its maximum for the large majority of pixels. The new images are much more balanced, with less intense colors and more similar distributions across the three channels. The greener series on the far right does show a shift of the green peak towards higher values, but the peak is still at a lower intensity than in the original images, and the other two channels remain relatively high. To correct for these differences, color transfer techniques were applied, which are described in the following sections.

4.3.2 Histogram equalization

The first color transfer technique tested was histogram equalization, also called histogram matching, which maps the color distribution of one image onto another [GW18]. It is based on the cumulative distribution function (CDF) of the pixel intensities per channel. For a single channel, the CDF at intensity i is the cumulative number of pixels with intensity at most i. When normalized, the CDF gives the proportion of such pixels, which forms a percentile curve. This curve is used to shift the intensities of the new image so that they match those of a reference image. For example, if 70% of the pixels in the reference image have a green value of 200 or less, while in the new image the same proportion corresponds to a value of 100, the intensities of the new image are shifted so that its CDF matches that of the reference.

The technique was applied in three ways. The first two used the `match_histograms` function from the scikit-image (skimage) library, which matches an input image to a single reference image [vdWSNI+14]. First, each new image was matched to a single representative image from the original dataset (Figure 19). Then, the images were matched to an "average" image, obtained by averaging the pixel values of several hundred original images (Figure 20). The idea was to create a reference that did not depend on the choice of a single frame. In the third approach, instead of using one image, the pixels of all reference images were pooled together to build a more representative CDF. Pooling the pixels of many images and taking their distribution is equivalent to averaging the histograms of the individual images. Unlike the averaged image, this keeps the original range of color intensities and reflects the distribution of a typical training image, while still being independent of the frame.

Figures 19 and 20 show the results of the first two approaches. In both cases the new image looks much more like the original images. The image matched to a single reference looks slightly sharper, while the one matched to the averaged image looks more blurry. This is likely because averaging reduces the variance of the color intensities and brings them closer together. To keep the distances between the intensities, the CDF from the pooled pixels was used instead. Figure 21 compares the color histograms of each method. The histogram of the pooled pixels closely resembles that of a single reference image, while the histogram of the averaged image looks much more chaotic.

To evaluate these methods, segmentation was applied to the adjusted images. There was little difference between the three histogram equalization methods, and at first glance the masks produced from all adjusted images look good (Figure 22). However, two problems appeared during the rest of the reconstruction (Figures 23 and 24). First, because histogram equalization shifts all pixel values, some local contrasts that were barely visible in the original images became visible and were then picked up by the segmentation model (Figure 23). Second, part of the zebrafish tail would sometimes disappear in one image while still being present in neighboring views (Figure 24). When this happens, the reconstruction cannot build a consistent 3D shape, and so no 3D model could be produced from these masks.

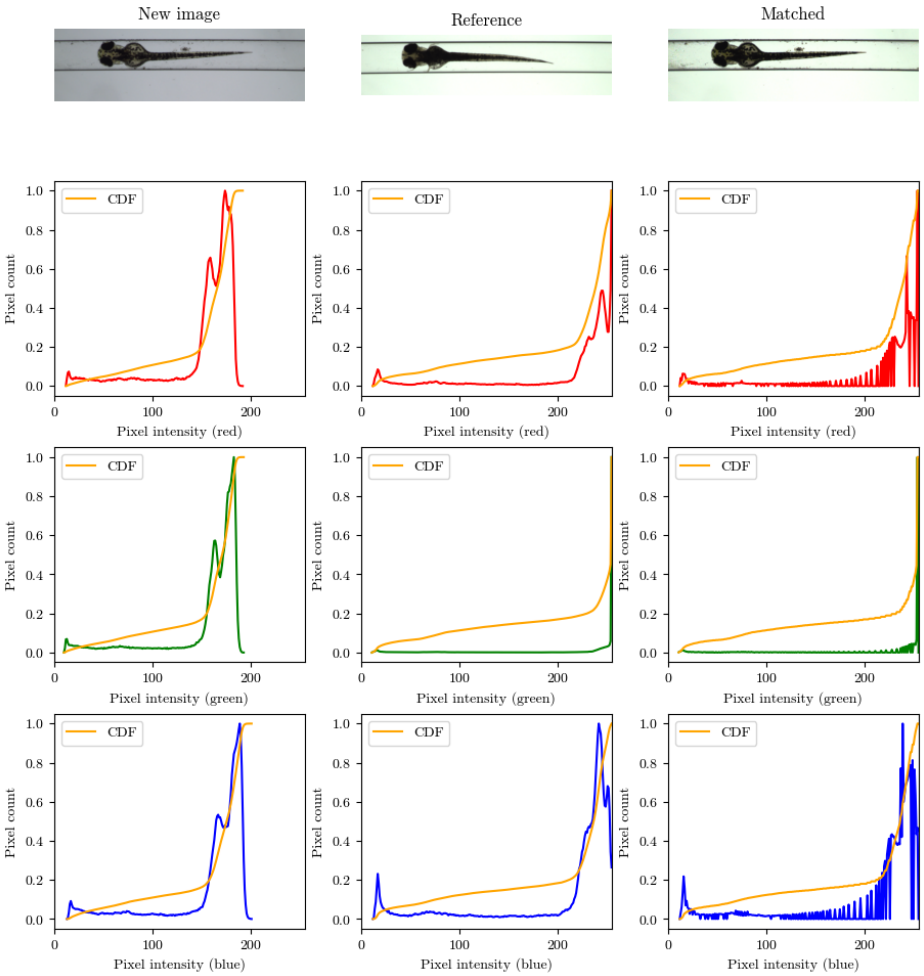


Figure 19: Histogram equalization applied based on the color distribution from a single reference image. Plots show the CDF and the distribution of pixel intensities in the new image, reference image and the result of the equalization.

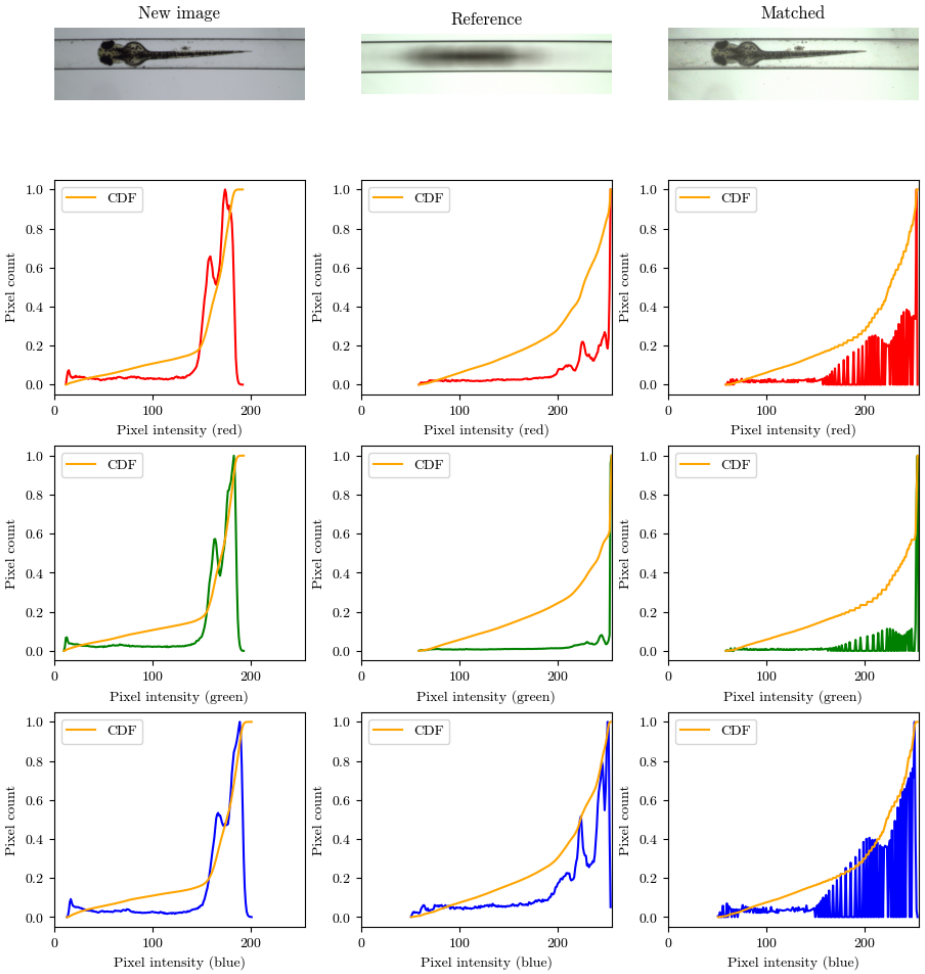


Figure 20: Histogram equalization applied based on the color distribution from averaged reference image. Plots show the CDF and the distribution of pixel intensities in the new image, the average of reference images and the result of the equalization.

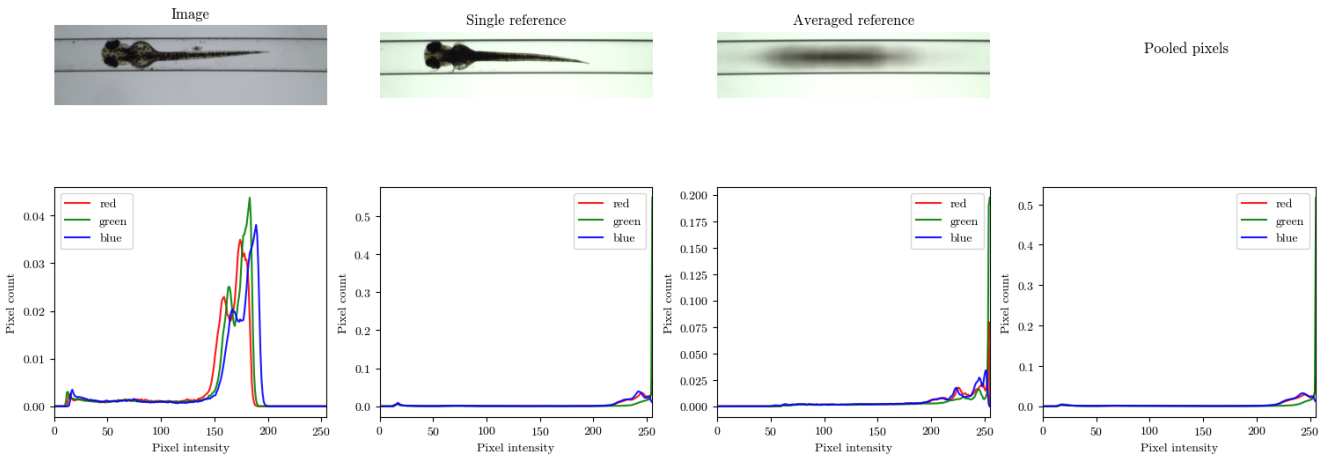


Figure 21: Normalized distribution of pixel intensities across channels for new image and each technique used.

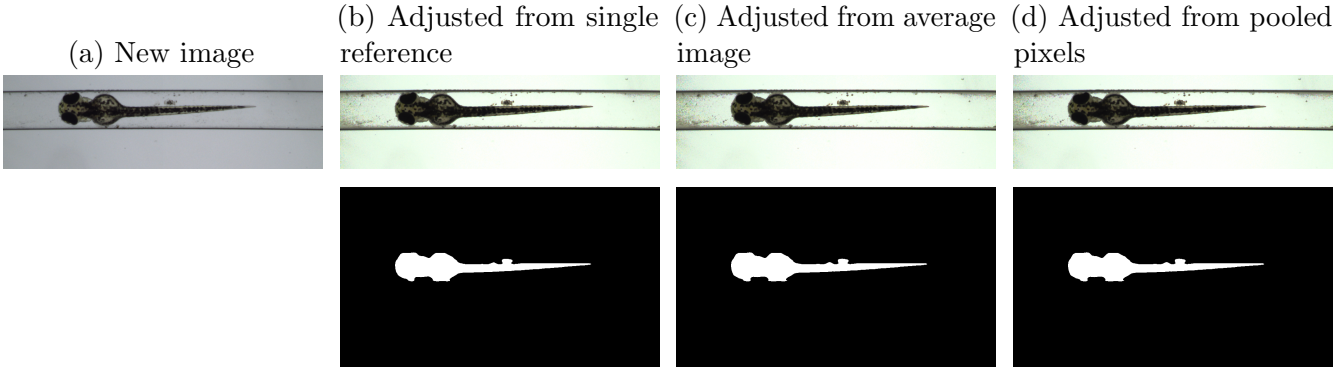


Figure 22: Comparison of the binary masks created from the new adjusted images

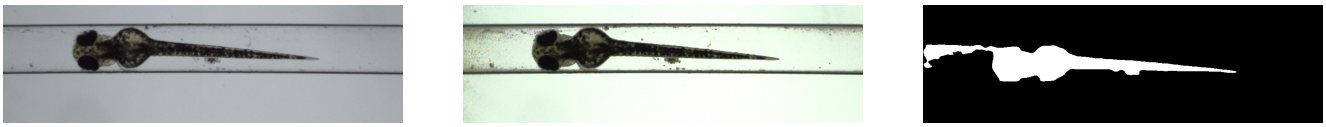


Figure 23: Issue 1: local contrast becomes exaggerated in the adjusted images, leading to a poor segmentation.

(a) View 162 (b) View 171

Figure 24: Issue 2: parts of the tail are visible only in some masks.

4.3.3 Reinhard method

The second color transfer technique tested was the Reinhard method [RAGS01]. Unlike histogram equalization, which matches the full intensity distribution, the Reinhard method transfers only the overall color statistics between two images. To do this, the image is first converted from RGB to the $l\alpha\beta$ (or LAB) space, which can be done using the `rgb2lab` function from the skimage library. The LAB color space describes a color with three values: L for lightness (from black to white) and A and B for the color itself (A from green to red and B from blue to yellow). Its main advantage is that it separates lightness from color, so the brightness and the color of an image can be adjusted independently. In addition, the three channels are close to uncorrelated, which means the statistics of each channel can be transferred separately without distorting the others [RAGS01].

Once in LAB space, the transfer is done per channel. For each of the three channels, the mean and standard deviation of the new image are shifted to match those of a reference image, so that the new image takes on the same average color and the same spread of values as the reference. The image is then converted back to RGB. This makes the colors of the new image resemble those of the reference, while being simpler and less sensitive to individual pixel values than histogram equalization, since only two numbers per channel are matched instead of the whole distribution.

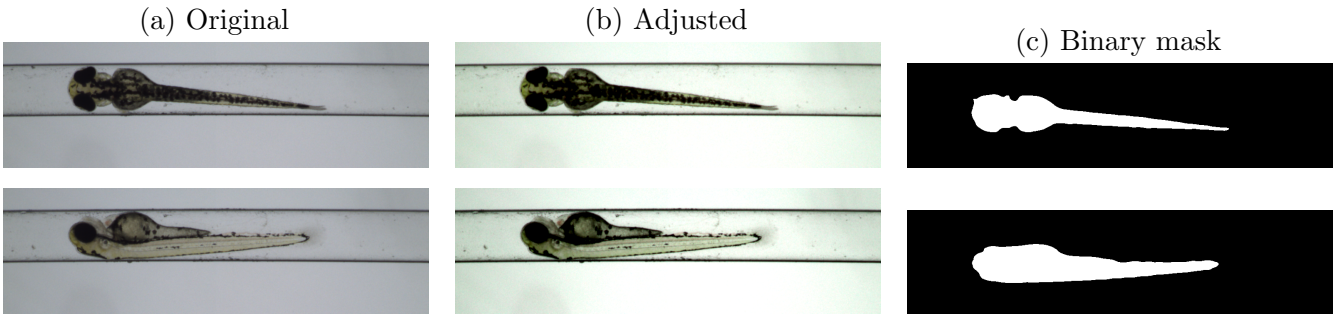


Figure 25: Reinhard color transfer applied to two views of a new dataset. Top row: view 1; bottom row: view 120 (out of 360). From left to right: the original image, the image after Reinhard adjustment, and the resulting binary mask.

Figure 25 shows the adjusted images and the masks obtained from them. Out of the 11 folders of 360 brightfield images, only 3 led to successful reconstructions from adjusted images. Others failed due to reasons explained in the previous section (Figures 23 and 24).

4.3.4 Fluorescent image thresholding

The fluorescent images look completely different from the brightfield images. They have a dark background and a brightly colored silhouette, with the color depending on the stain used. Some examples and their color histograms are shown in Figure 26. Because the segmentation models were never trained on fluorescent images, a different technique had to be used. The high contrast between the fish and the background makes these images well suited to thresholding.

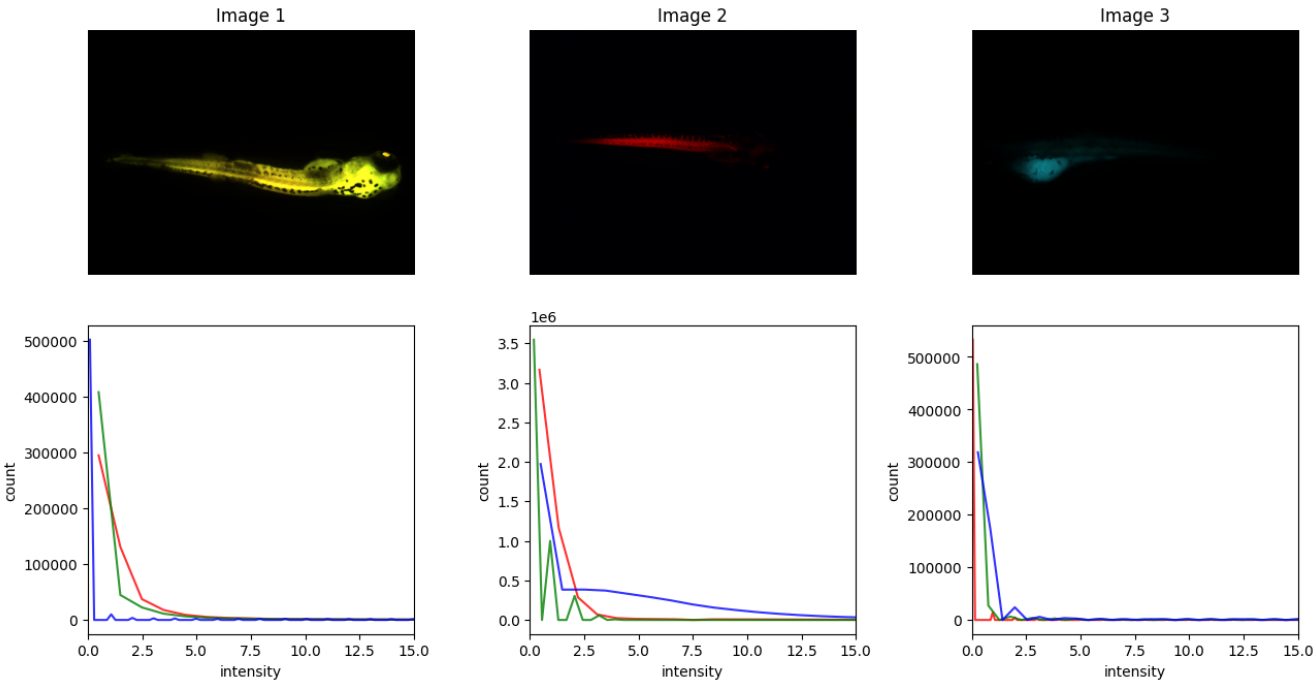


Figure 26: Three types of fluorescent images with color histograms. The pixel intensities are at the low end of the scale and all intensities are within a narrow range.

Thresholding converts an image into a binary mask by comparing every pixel to a threshold value between 0 and 255: pixels below the threshold become black and those above it become white [GW18]. Although thresholding is normally used on grayscale images, it can also be applied when a single channel is present. The fluorescent images have three channels but a very narrow range of colors, so they were collapsed to a single channel by taking, for each pixel, the highest intensity across the three channels.

To find a threshold value, the skimage library offers several functions [vdWSNI+14], which are either local or global. Local thresholding compares each pixel to its neighbors and is used when the background has uneven lighting, while global thresholding computes a single value for the whole image and works best when the background is uniform [SS04]. Since the background of the fluorescent images is uniformly black, global thresholding was used. The skimage function `try_all_threshold` was used to compare all available global methods: isodata, li, mean, minimum, otsu, triangle and yen. Figure 27 shows the output of each method for the three types of fluorescent images. Some thresholding functions, e.g. mean, yen or triangle, detected more of the fish than the other methods. This, however, is undesirable, since the rest of the fish (apart from the fluorescently labeled structures) should remain as background. For all images, the isodata [RC78] and otsu [Ots79] functions gave the most accurate masks, so these two were used.

An initial attempt computed the threshold per image, but this was unreliable, as the masks were not consistent enough for reconstruction. The threshold values of each method were therefore compared across a full dataset (Figure 28). Otsu and isodata were the most stable and gave almost identical thresholds, so a single value of 54 was applied to all images. Even then, the masks were

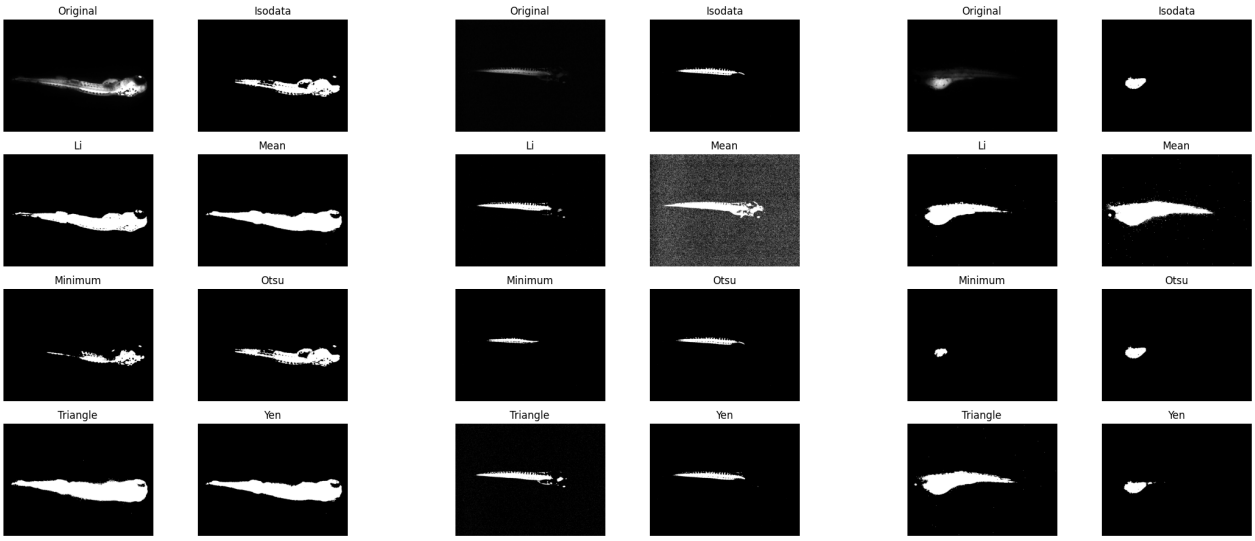


Figure 27: Comparison of thresholding functions for three different fluorescent images.

not consistent enough across views to be combined into a 3D object. Thus, no reconstruction from fluorescent images could be obtained.

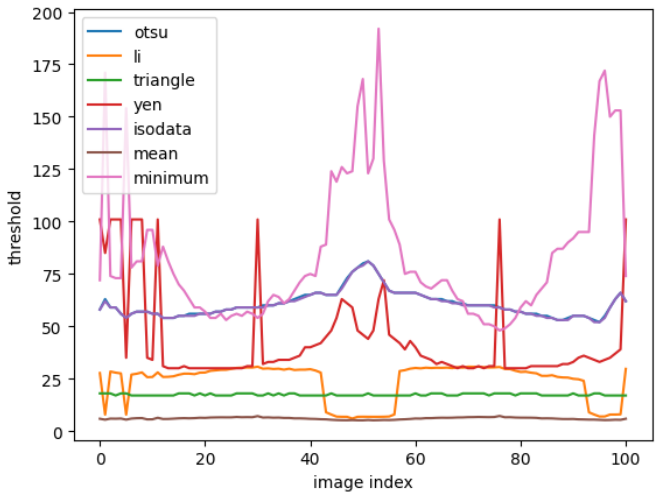


Figure 28: Threshold values computed by each methods for every image in a dataset of 100 images.

4.4 Image-count experiments on brightfield data

Once the segmentation and preprocessing were in place, the reconstructions were run to study how the number of images affects the result. Not all datasets could be reconstructed. All 14 original datasets of 100 images went through the pipeline successfully. Of the 11 datasets of 360 images, only 3 could be reconstructed, and none of the fluorescent datasets could be reconstructed at all.

These failures were caused by the same error in the final reconstruction step. The 3D mesh is

extracted from the carved voxel volume using the marching cubes algorithm [LC87], which builds a surface at a fixed iso-level inside the volume. When the silhouettes are too inconsistent, the intersection carves away so much of the volume that no valid surface remains at that level. In this case the algorithm stops with the error "`Surface level must be within volume data range`" and no mesh is produced. This happened for all fluorescent datasets, where the thresholded masks were too inconsistent (Section 4.3.4), and for most of the 360-image datasets, where some of the views were segmented incorrectly.

4.4.1 Datasets of 100 images

First, 14 datasets of 100 images each were analyzed. From each dataset, reconstructions were obtained using 5, 10, 20, 25, 50, 75 and 100 images. Example reconstructions from a single dataset are shown in Figure 29. When fewer images were used, the 3D mesh resembled a block, most clearly from above, even though the lateral view still looked fish-like. As more views were added, the mesh became progressively thinner. This trend was consistent across all datasets.

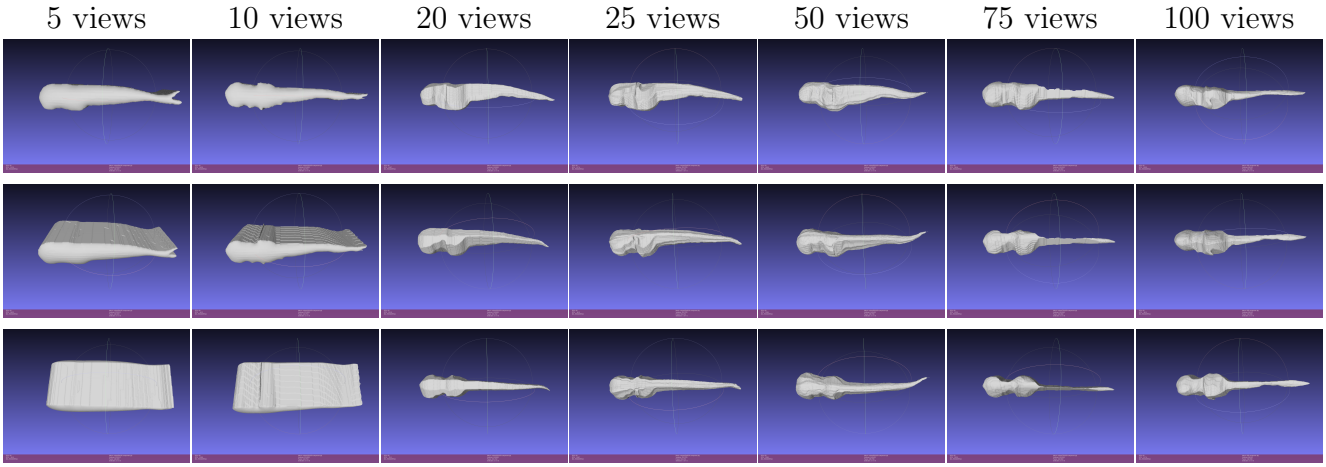
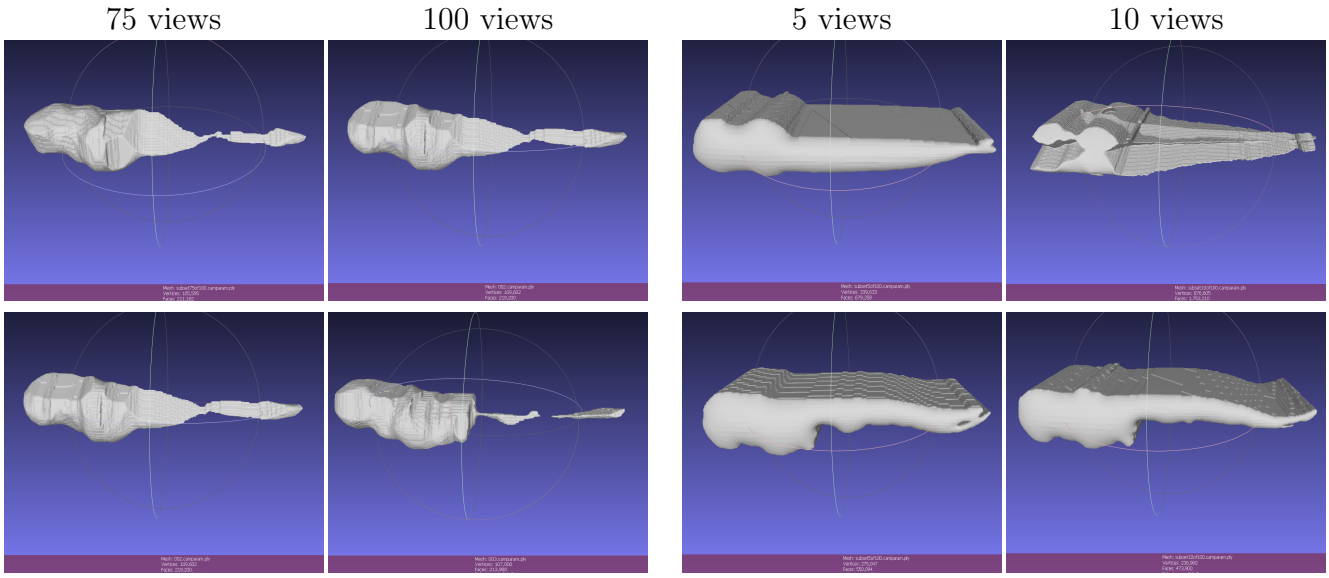


Figure 29: Reconstructions obtained from samples of a single dataset of 100 images, seen from different angles.

Contrary to expectation, adding more views did not always make the reconstructions more accurate. In many cases, the reconstructions from 75 and 100 views were considerably worse than those from 20-50 views, often resulting in loosely connected or even disconnected fragments (Figure 30a). The reconstructions from fewer views, on the other hand, were much thicker and bore little resemblance to a fish (Figure 30b). The most convincing reconstructions were obtained at around 20-50 views.

To measure this more precisely, the volumes and surface areas of all reconstructions were compared against reference distributions. Guo et al. found that for 3dpf zebrafish larvae the volume follows a normal distribution $\mathcal{N}(\mu = 2.53, \sigma = 0.11)$ in $10^8\,\mu m^3$, and the surface area follows $\mathcal{N}(\mu = 3.20, \sigma = 0.15)$ in $10^6\,\mu m^2$ [GVSV17]. Figure 31 shows how the reconstructions from one dataset compare to these distributions; the plots for all datasets can be found in Appendix A. In Figure 31, the reconstructions from 20 and 50 views fit the distribution best, while the one from 25



(a) Overcarving (b) Undercarving

Figure 30: Examples of over- and under-carving resulting from using too many or too few images.

views lies on the edge. The other reconstructions match the earlier observations: those from 5 and 10 views are much larger, while those from 75 and 100 views are much smaller.

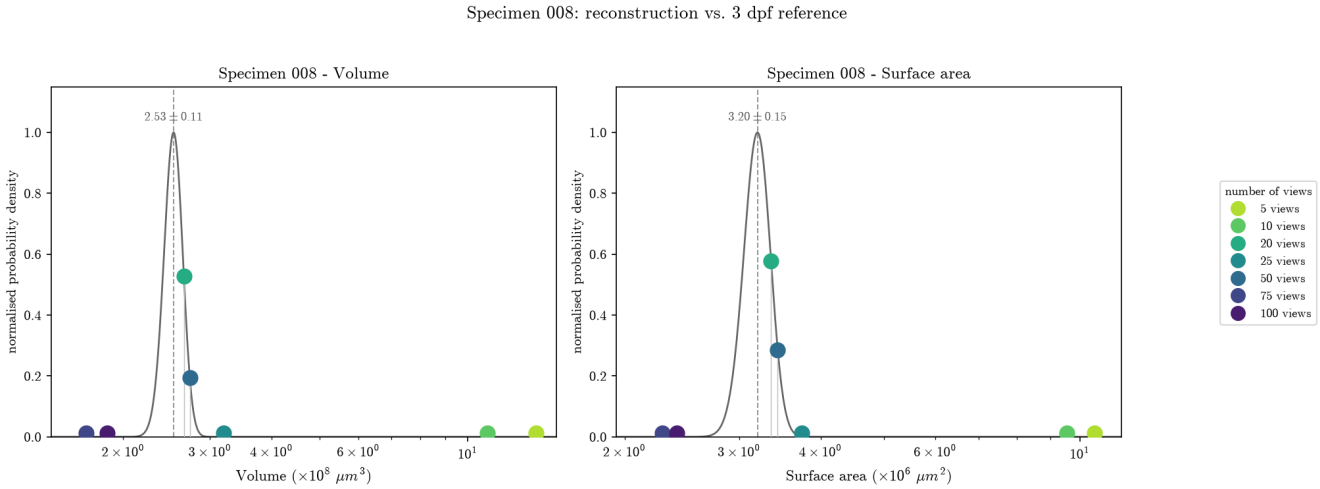


Figure 31: Volume and surface area values obtained from reconstructions from samples of a single dataset of 100 images, plotted against reference distributions for 3dpf zebrafish larvae as found in [GVSV17].

For each number of views, it was then determined whether the measured values fell within the distribution and, if so, how far they lay from the mean, expressed in terms of standard deviations. The results are summarized in Table 1. According to the three-sigma rule, about 99.7% of the values of a normal distribution lie within three standard deviations of the mean [Puk94], so

any value further than that was counted as "out", that is, falling outside of the reference distribution.

Views	$\leq 1\sigma$	$1–2\sigma$	$2–3\sigma$	$> 3\sigma$	In	Out	Total
5	0	0	0	14	0	14	14
10	0	0	0	14	0	14	14
20	5	4	2	3	**11**	3	14
25	0	0	3	11	3	11	14
50	6	4	2	2	**12**	2	14
75	0	0	0	14	0	14	14
100	0	0	1	13	1	13	14

(a) Volume

Views	$\leq 1\sigma$	$1–2\sigma$	$2–3\sigma$	$> 3\sigma$	In	Out	Total
5	0	0	0	14	0	14	14
10	0	0	0	14	0	14	14
20	6	5	2	1	**13**	1	14
25	0	2	3	9	5	9	14
50	9	5	0	0	**14**	0	14
75	0	0	1	13	1	13	14
100	0	2	0	12	2	12	14

(b) Surface area

Table 1: Number of reconstructions (out of 14) falling within each distance band from the reference mean, by number of views. Highest counts falling inside the reference distributions are highlighted in bold.

A number of views was considered good when most of the 14 reconstructions fell inside the reference distribution. Based on this, Table 1 shows that 20 and 50 views gave the best results, as almost all reconstructions fell within three standard deviations of the mean. At 5, 10 and 75 views, on the other hand, none of the reconstructions fell within the distribution.

In line with the qualitative observations, the volumes from too few views were too large, while those from too many were too small. This can be explained by under- and over-carving. In visual hull reconstruction the 3D shape is obtained by intersecting the silhouettes from all views, so a voxel is only kept if it lies inside the silhouette in every view [Lau94]. When only a few views are used, few silhouettes are intersected and little of the volume is carved away, so the reconstruction stays too large (under-carving). As more views are added, more of the volume is removed and the reconstruction becomes thinner. However, because the silhouettes are not perfect, each additional view is also a chance for an inaccurate silhouette to carve away parts that actually belong to the fish. Since voxels can only be removed and never added back, these errors add up, and with many views the reconstruction becomes too small and can break into fragments (over-carving). This is the same effect that caused some reconstructions to fail completely with the marching cubes error described above.

Interestingly, the reconstruction from 25 views was worse than both 20 and 50 views even though it lies between them. If the result depended only on the number of views, 25 views would be expected to fall between its neighbors. This suggests that the result does not depend only on how many views are used, but also on which views are selected. Because the subsets are created by taking every n^{th} image, 25 views corresponds to every fourth image of the full revolution. It is possible that this particular set of angles happens to include several poorly segmented silhouettes, which then carve away too much. As the same angles are selected for every dataset, this would cause a systematic error rather than random noise. This could be tested in future work by repeating the experiment with a different starting image, to see whether 25 views is always worse or only for this particular subset.

4.4.2 Datasets of 360 images

The same analysis was applied to the 3 datasets of 360 images that could be reconstructed. Because only three datasets went through the pipeline, these results are preliminary and should be read with care. Results from one of the datasets can be seen in Figure 32, while the rest can be found in the appendix A. The overall behavior with the number of views was similar to that of the 100-image datasets - more views led to smaller volumes while fewer views corresponded to bigger volumes. However, all the volumes and surface areas were smaller than the reference values, even at the numbers of views that gave accurate results for the 100-image datasets.

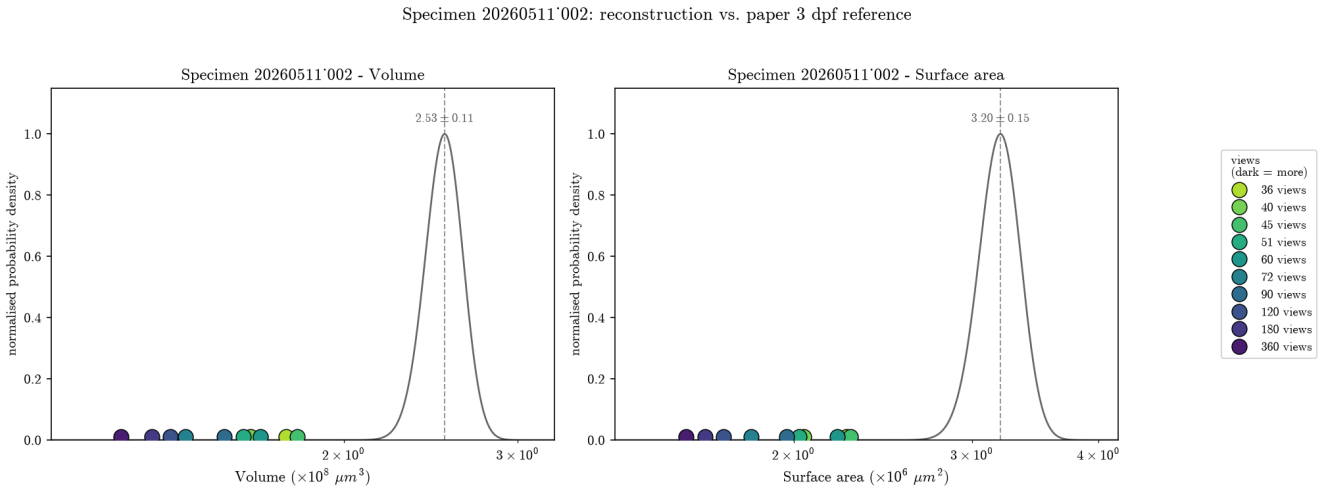
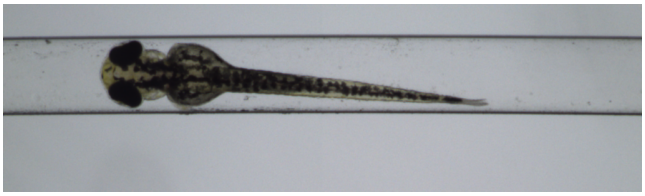
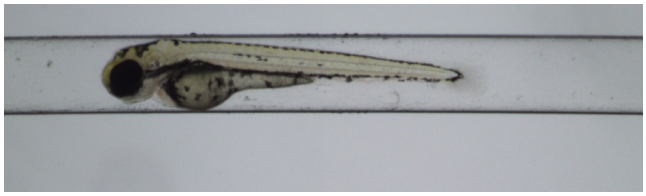


Figure 32: Volume and surface area values obtained from reconstructions from samples of a single dataset of 360 images, plotted against reference distributions for 3dpf zebrafish larvae as found in [GVSV17].

The zebrafish in these datasets were confirmed to be 3dpf, the same age as the reference larvae, so the difference is unlikely to come from the size or shape of the fish. Instead, it points to a problem with the calibration that converts the voxel volume into real-world units. This calibration relies on hard-coded image sizes, which match the original datasets but not the newer 360-image datasets. It has been discovered that the pipeline is optimized only for images of size 1024 by 250 pixels. The images are then padded with 387 pixels at the bottom and top of the image, so that the center of the zebrafish falls at 512 pixels ($(250 \div 2) + 387 = 512$). The newer images were found to be 1024 by 300 pixels, which distorted the calibration, leading to the masks getting rescaled and measurements becoming inaccurate. Moreover, it has been discovered that the 360 images do not span a full revolution (Figure 33). If they did, the starting and ending position should align. However, this was found not to be the case. The final position was still around 90 degrees short of the original position. For this reason, the obtained reconstructions are also not very accurate.



(a) View 1



(b) View 360

Figure 33: First and last view of a 360-image dataset. If the images spanned a full revolution, the last view would align with the first. Here they do not match, which shows that the 360-image datasets do not cover a complete 360-degree rotation of the specimen.

5 Discussion

The main result of this thesis is a desktop application that makes the reconstruction pipeline much easier to use. The user no longer has to edit the code or paste file paths into the terminal, which makes the process faster and less prone to mistakes. The batch mode and the option to continue from already segmented images also make the pipeline more flexible. Still, several problems came up during the work, and most of them point back to the same few weak spots in the pipeline.

The first weak spot is segmentation. The models were trained on images taken in one period with the same microscope settings, so they did not generalize to the newer data. The main difference was the color balance. To deal with this, color transfer techniques such as histogram equalization and the Reinhard method were applied to make the new images look more like the training images. At first the masks looked good, but the equalization also made faint contrasts in the background visible, and the tail of the zebrafish was sometimes lost between two views. Because of this, preprocessing by hand is not a real solution. It only works on some datasets, it has to be redone for every new imaging condition, and it breaks the automation that the whole pipeline is supposed to provide.

The second weak spot is the reconstruction step itself. The visual hull is built by intersecting the silhouettes from all views, so a voxel is only kept if it lies inside the silhouette in every single view. This makes the reconstruction very sensitive to bad masks. If a part of the fish, for example a piece of the tail, is missing in only one image out of a hundred, that part is carved away and cannot come back, because voxels can only be removed and never added. This is exactly what happened in many of the failed reconstructions. This behavior is well known in the literature: a miss in a single view stops the visual cones from intersecting there, which leads to an unavoidable gap in the reconstructed shape [LPC08]. Visual hull reconstruction is also known to be sensitive to errors in the segmentation and in the viewpoint, which can lead to over-carving when the silhouettes are not perfect [LPC08]. The same effect is what produced the marching cubes error ("`Surface level must be within volume data range`") for all fluorescent datasets and for most of the 360-image datasets: once too much of the volume is carved away, no valid surface is left.

This also explains the image-count results. In classic visual hull theory more views can only improve the reconstruction, but that assumes the silhouettes are perfect [Lau94]. In practice the masks are not perfect, so each extra view is also another chance for a bad silhouette to carve away part of the fish. With too few views little is carved away and the reconstruction stays too large (under-carving);

with too many views the small errors add up and the reconstruction becomes too small and can break into pieces (over-carving). This matches the observation that 20 and 50 views gave the best results, while 5 and 10 views were too large and 75 and 100 views were too small. It is worth noting that 25 views was worse than both 20 and 50, even though it lies between them. Because the subsets are always taken as every n^{th} image starting from the same frame, this is most likely a systematic effect: this particular set of angles probably includes a few poorly segmented views that carve away too much. In other words, the result does not depend only on how many views are used, but also on which views are used.

A separate problem showed up in the 360-image datasets, where all reconstructions came out too small, even at view counts that worked well for the 100-image datasets. The fish were confirmed to be 3dpf, the same age as the reference larvae, so the size difference does not come from the fish. Two other causes were found. First, the calibration that converts voxels into real units uses hard-coded image sizes. It assumes images of 1024 by 250 pixels, padded to place the center of the fish at pixel 512. The newer images are 1024 by 300 pixels, so the calibration rescales the masks incorrectly and the measurements come out wrong. Second, the 360 images do not span a full revolution. The first and last views do not line up, and the last view is around 90 degrees short of the first. This fits with the way the VAST works. Guo et al. report that the stepper motor has a minimum step of 0.72 degrees [GVSV17], which means a full revolution takes exactly 500 steps ($500 \times 0.72 = 360$). Taking only 360 images at the minimum step therefore covers just $360 \times 0.72 \approx 259$ degrees, which is about 100 degrees short of a full turn and close to the gap that was observed. The motor also cannot move in exact 1-degree steps, because 1 degree is not a multiple of 0.72, so the requested angles are snapped to the nearest possible step. On top of this, the pipeline assumes that the views span a full 360 degrees when it computes the camera angles, so when the real coverage is smaller, the assumed angles are wrong and the reconstruction is distorted. Together, the wrong calibration and the incomplete rotation explain why the 360-image reconstructions are not accurate.

Finally, all experiments were done on 3dpf larvae. So the results say something about this specific stage, but they cannot be assumed to hold for other ages, where the size and shape of the fish are different. The experiments should also thus be repeated also for larger zebrafish. Lastly, a small technical limitation of the pipeline is that some of the packages used are outdated, which makes the environment harder to set up and to reproduce.

6 Conclusion

This thesis looked at how the existing zebrafish 3D reconstruction pipeline could be made more flexible and easier to use, and at how many input images are actually needed for an accurate reconstruction. The main research question was:

> *How can the existing zebrafish 3D reconstruction pipeline be improved to enhance flexibility through parameterization to accommodate both brightfield and fluorescent imaging?*

To answer this, the question was split into three sub-questions, which are addressed below.

- **How can the reconstruction script be redesigned to allow users without programming experience to operate it easily?**

  This was solved by building a desktop application. Instead of editing the code and pasting file paths into the terminal, the user can now select an input folder, choose a segmentation model, and set the number of images through a simple interface. The app also shows the output of the pipeline while it runs, so the user can follow the process in one place.

- **How can the pipeline be extended to support automated high-throughput processing of images from multiple modes?**

  A batch mode was added that searches a parent folder for all image folders and processes them one after another, with parameters that can be set per dataset. The option to continue from already segmented images, and to convert the masks to PNG, was also added, so that the segmentation can be checked before the rest of the pipeline is run. Support for fluorescent images was explored, but the masks produced by thresholding were not consistent enough to be combined into a 3D object, so a full reconstruction from fluorescent images could not be obtained.

- **What is the minimal number of images required to achieve accurate 3D reconstruction without compromising quality?**

  Because the segmentation models did not work well on the newer data, this was tested on the original brightfield datasets. The results show that more images is not always better: the most accurate reconstructions came from around 20 to 50 views, while both fewer and more views were worse, due to under- and over-carving. Because of this, a single firm minimum cannot be given yet. It also depends on which views are selected, not only on how many. To get a definitive answer, the segmentation models should be improved and the experiments should be repeated on larger datasets and on full 500-image acquisitions.

Overall, the pipeline is now easier to use and gives the user more control, and fewer images can be used for brightfield reconstruction. Extending it fully to fluorescent imaging remains an open problem.

7 Future work

Several things could be improved in future work. The most important step is the segmentation. The models should be retrained, or fine-tuned, on a larger and more varied set of images, including the newer brightfield data. Almost all of the problems in this thesis came from segmentation, so a model that generalizes well would remove the need for manual preprocessing and keep the pipeline automated. A separate model for fluorescent images should also be trained or a new image processing pipeline should be introduced, since the current models cannot handle this kind of data. If some preprocessing is still needed after that, it should be built into the pipeline instead of being done by hand, because doing it separately breaks the automation and makes the results harder to reproduce.

The reconstruction step should also be made more robust. At the moment the visual hull uses strict intersection, so a voxel is only kept if it is inside the silhouette in every view. This means a single bad mask can destroy a real part of the fish. A better option is to relax this rule and keep a voxel if it is inside the silhouette in most views instead of all of them, for example in at least $N - k$ out of N views [LPC08]. Other approaches replace the hard decision with a probabilistic one, where each view votes on how likely a voxel is to be occupied and the votes are combined, such as the occupancy-grid method of Franco and Boyer [FB05] or the Bayesian reconstruction of Grauman et al. [GSD03]. These methods are more tolerant of the occasional bad mask and would likely recover parts like the tail that are currently lost. Trying one of these would be a good next step.

The imaging itself could also be improved. The VAST can take up to 500 images, but with the current camera settings the images taken this way are too poor in quality to use. Since 500 images at the minimum 0.72-degree step is exactly one full revolution, it would be worth adjusting the camera settings so that a full set of 500 good images can be taken. The reconstruction could then be tested at higher view counts and the minimum number of images could be confirmed. At the same time, the calibration should be fixed so that it no longer assumes a single fixed image size, and it should be ensured that the images always cover a full 360 degrees revolution.

It would also be interesting to test different starting positions. Right now the subsets always start from the same image, so the same angles are used every time. This is probably why 25 views was worse than both 20 and 50. Repeating the experiment with a different starting frame would show whether certain view counts are always worse, or only for this particular set of angles, and whether the same pattern appears at other sampling frequencies.

The experiments should also be repeated on more fish. Only 3dpf larvae were used here, so it is not clear yet whether the results hold at other stages, where the fish have a different size and shape. Running the experiments on more specimens and more ages would show how general the findings are. Once the fluorescent images can be segmented reliably, the image-count experiment should be repeated on them as well. This matters most for fluorescent imaging, because each fluorescent image needs a long exposure time, so using fewer images would save a lot of time during acquisition.

Finally, the output structure could be redesigned. At the moment the results are saved inside the input folder, which becomes messy, especially when the pipeline is run on part of the dataset. The

results should be saved in a separate location and the metrics from all reconstructions collected into a single database. This would make it easier to compare datasets and analyze results later, and it could be connected to a website that visualizes the reconstructions. The outdated packages in the original pipeline should also be updated so the environment is easier to set up and reproduce.

References

[ABB+25] Muhammad Ali, Viviana Benfante, Ghazal Basirinia, Pierpaolo Alongi, Alessandro Sperandeo, Alberto Quattrocchi, Antonino Giulio Giannone, Daniela Cabibi, Anthony Yezzi, Domenico Di Raimondo, Antonino Tuttolomondo, and Albert Comelli. Applications of artificial intelligence, deep learning, and machine learning to support the analysis of microscopic images of cells and tissues. *Journal of imaging*, 11(2):59, Autumn 2025.

[CAF+19] Steven Cassar, Isaac Adatto, Jennifer L. Freeman, Joshua T. Gamse, Iñaki Iturria, Christian Lawrence, Arantza Muriana, Randall T. Peterson, Steven Van Cruchten, and Leonard I. Zon. Use of zebrafish in drug discovery toxicology. *Chemical Research in Toxicology*, 33(1):95–118, Oct 2019.

[CKS95] Vicent Caselles, Ron Kimmel, and Guillermo Sapiro. Geodesic active contours. *International Conference on Computer Vision*, Jun 1995.

[CKSP22] Kelda Chia, Anna Klingseisen, Dirk Sieger, and Josef Priller. Zebrafish as a model organism for neurodegenerative disease. *Frontiers in Molecular Neuroscience*, 15:940484, 2022.

[CM02] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002.

[Com26] Corporate Communications. Guide to live-cell imaging, Jan 2026.

[FB05] Jean-Sébastien Franco and Edmond Boyer. Fusion of multi-view silhouette cues using a space occupancy grid. In *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV)*, volume 2, pages 1747–1753, Beijing, China, October 2005.

[GSD03] Kristen Grauman, Gregory Shakhnarovich, and Trevor Darrell. A Bayesian approach to image-based visual hull reconstruction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 187–194, Madison, WI, USA, June 2003.

[GVSV15] Yuanhao Guo, Wouter J. Veneman, Herman P. Spaink, and Fons J. Verbeek. Silhouette-based 3d model for zebrafish high-throughput imaging. In *2015 International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 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*, 2017.

[GW18] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Pearson, 4 edition, 2018.

[GWK+17] Y. Guo, R.C. Van Wijk, E.H.J. Krekels, H.P. Spaink, P.H. Van Der Graaf, and F.J. Verbeek. Multi-modal 3d reconstruction and measurements of zebrafish larvae and its organs using axial-view microscopy. In *2017 IEEE International Conference on Image Processing (ICIP)*, 2017.

[HJV+16] D.L. Hickman, J. Johnson, T.H. Vemulapalli, J.R. Crisler, and R. Shepherd. Commonly used animal models. *Principles of Animal Research for Graduate and Undergraduate Students*, 7(PMC7150119):117–175, Nov 2016.

[KFS+22] E. Kugler, James Frost, Vishmi Silva, K. Plant, K. Chhabria, T. Chico, and P. Armitage. Zebrafish vascular quantification: a tool for quantification of three-dimensional zebrafish cerebrovascular architecture by automated image analysis. *Development (Cambridge, England)*, 149, 2022.

[Lau94] A Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, Jan 1994.

[LC87] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21(4):163–169, 1987.

[LC07] Graham J. Lieschke and Peter D. Currie. Animal models of human disease: zebrafish swim into view. *Nature Reviews Genetics*, 8(5):353–367, 2007.

[LPC08] José Luis Landabaso, Montse Pardàs, and Josep R. Casas. Shape from inconsistent silhouette. *Computer Vision and Image Understanding*, 112(2):210–224, 2008.

[MG17] Richard C. Mohs and Nigel H. Greig. Drug discovery and development: Role of basic biological research. *Alzheimer's Dementia: Translational Research Clinical Interventions*, 3(4):651–657, Nov 2017.

[oM19] National Library of Medicine. Benefits derived from the use of animals, 2019.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[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. *Nature Methods*, 2010.

[Puk94] Friedrich Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994.

[Pul16] Rock Pulak. Tools for automating the imaging of zebrafish larvae. *Methods*, 2016.

[RAGS01] Erik Reinhard, Michael Ashikhmin, Bruce Gooch, and Peter Shirley. Color transfer between images. *IEEE Computer Graphics and Applications*, 21(5):34–41, 2001.

[RC78] T. W. Ridler and S. Calvard. Picture thresholding using an iterative selection method. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(8):630–632, 1978.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science*, 9351:234–241, 2015.

[RKGS08] Emmanuel G. Reynaud, Uroš Kržič, Klaus Greger, and Ernst H. K. Stelzer. Light sheet-based fluorescence microscopy: More dimensions, more photons, and less photodamage. *HFSP Journal*, 2(5):266–275, Oct 2008.

[SCE+00] M N Sawka, V A Convertino, E R Eichner, S M Schnieder, and A J Young. Blood volume: importance and adaptations to exercise training, environmental stresses, and trauma/sickness. *Medicine and science in sports and exercise*, 32(2):332–48, 2000.

[SPM04] S.N. Sinha, M Pollefeys, and L McMillan. Camera network calibration from dynamic silhouettes. *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:195–202, Nov 2004.

[SS04] Mehmet Sezgin and Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13(1):146–165, 2004.

[TKL+13] Wojciech Tarnawski, Vartan Kurtcuoglu, Paweł Lorek, Marcin Bodych, Jan Rotter, Monika Muszkieta, Łukasz Piwowar, Dimos Poulikakos, Michał Majkowski, and Aldo Ferrari. A robust algorithm for segmenting and tracking clustered cells in time-lapse fluorescent microscopy. *IEEE Journal of Biomedical and Health Informatics*, 17(4):862–869, Jul 2013.

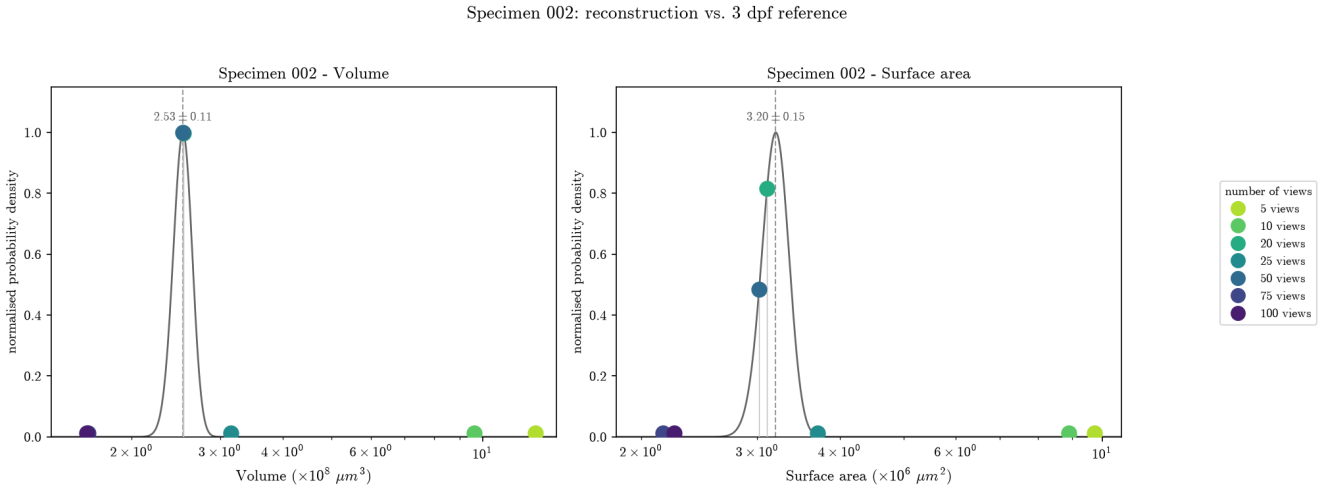
[TZR+19] Tsegay Teame, Zhen Zhang, Chao Ran, Hongling Zhang, Yalin Yang, Qianwen Ding, Minxu Xie, Chenchen Gao, Yongan Ye, Ming Duan, and Zhigang Zhou. The use of zebrafish (danio rerio) as biomedical models. *Animal Frontiers*, 9(3):68–77, Jun 2019.

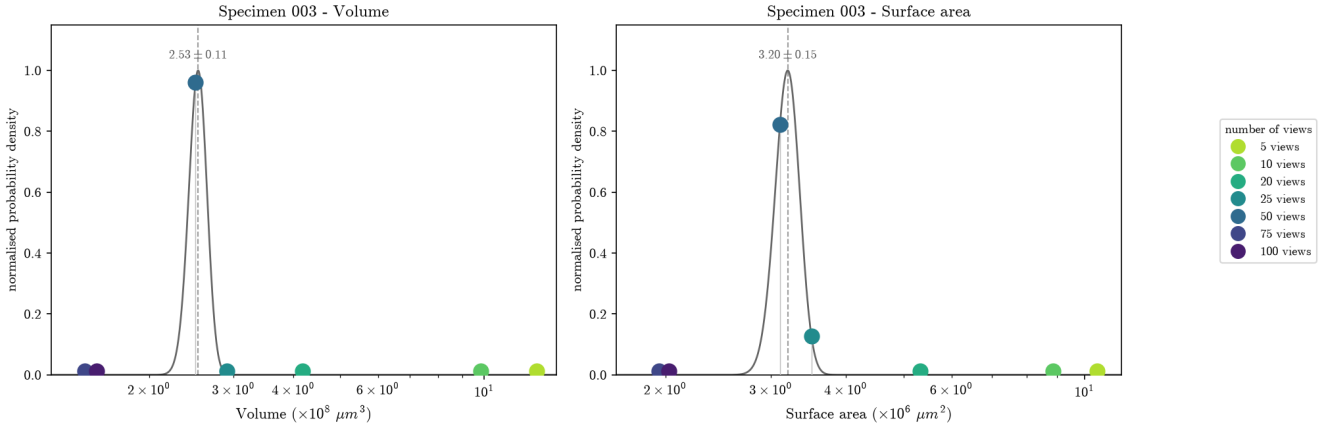
[vdWSNI+14] Stefan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in Python. *PeerJ*, 2:e453, 2014.

[WRZ13] Richard M. White, Kristin Rose, and Leonard I. Zon. Zebrafish cancer: the state of the art and the path forward. *Nature Reviews Cancer*, 13(9):624–636, 2013.

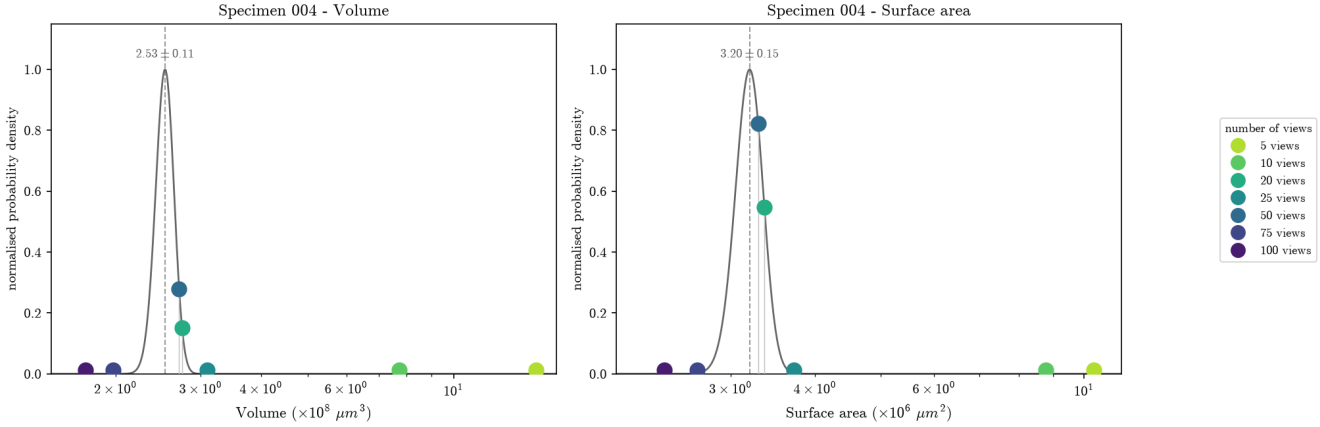
A Appendix

A.1 Plots of all 14 analyzed folders of 100 images

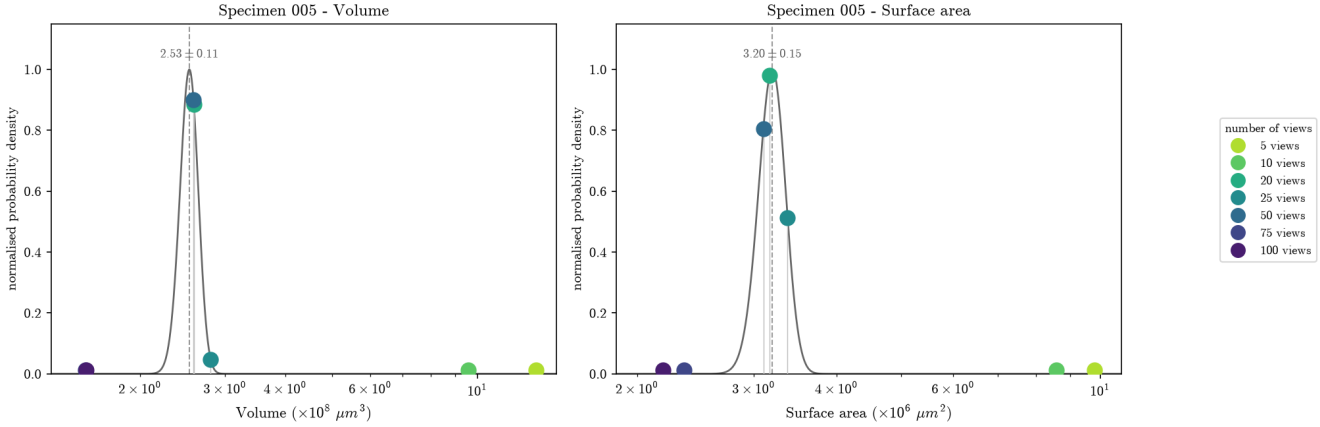




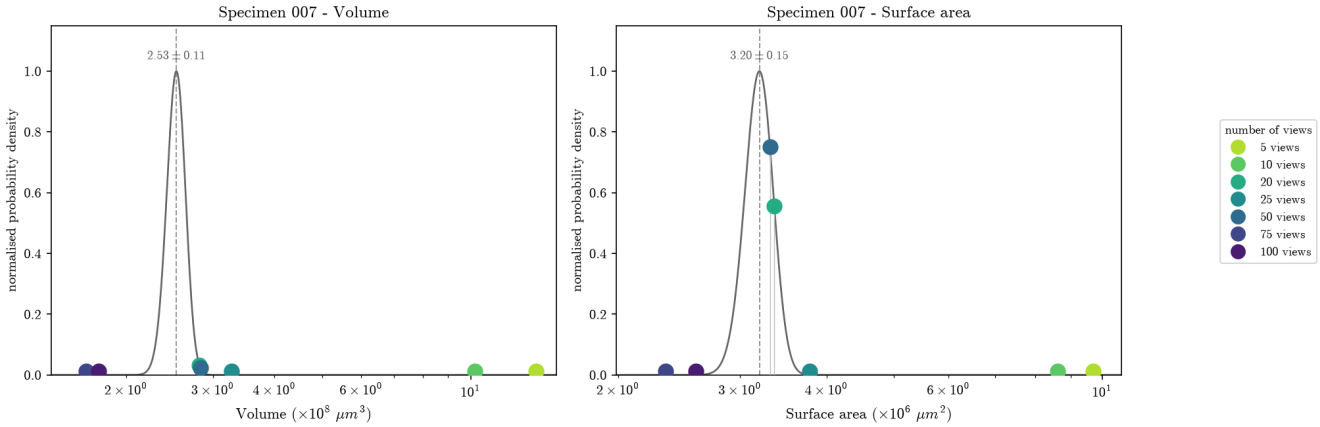
Specimen 004: reconstruction vs. 3 dpf reference



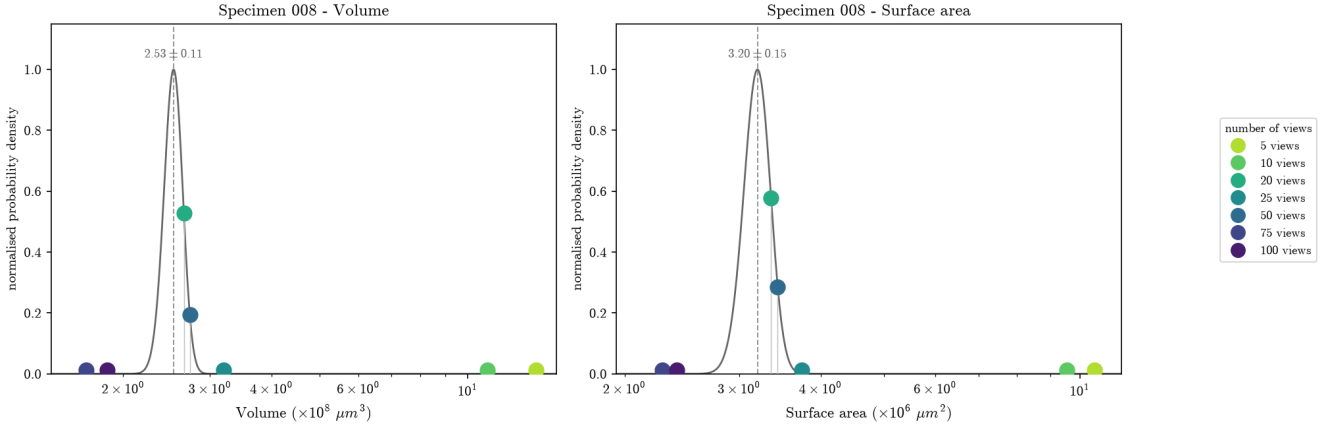
Specimen 005: reconstruction vs. 3 dpf reference



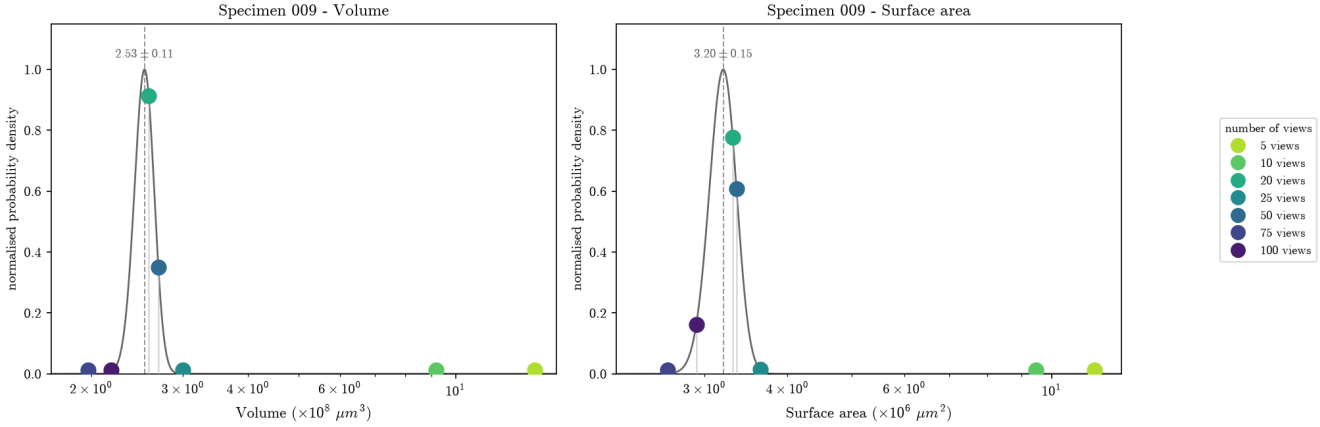
Specimen 007: reconstruction vs. 3 dpf reference



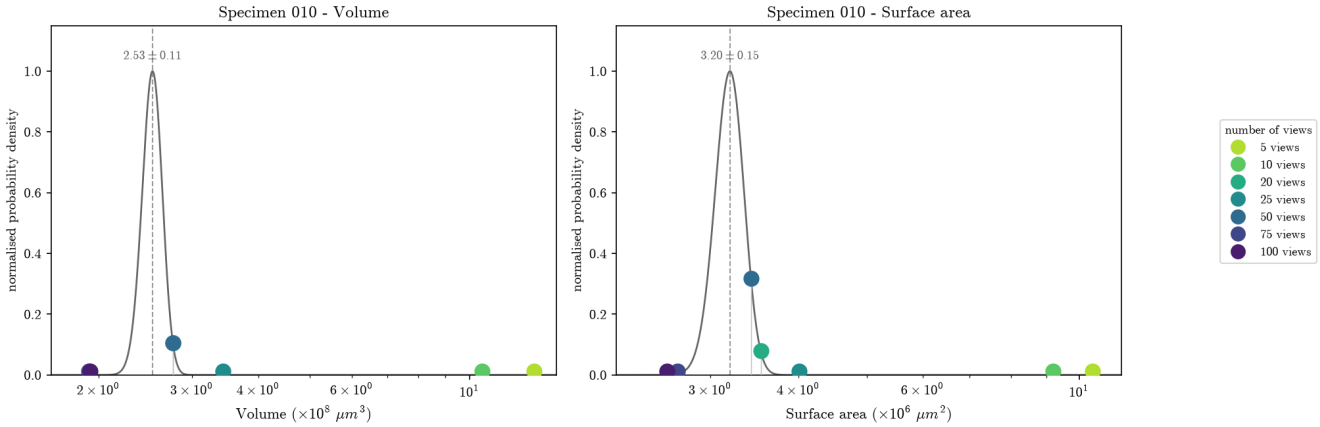
Specimen 008: reconstruction vs. 3 dpf reference



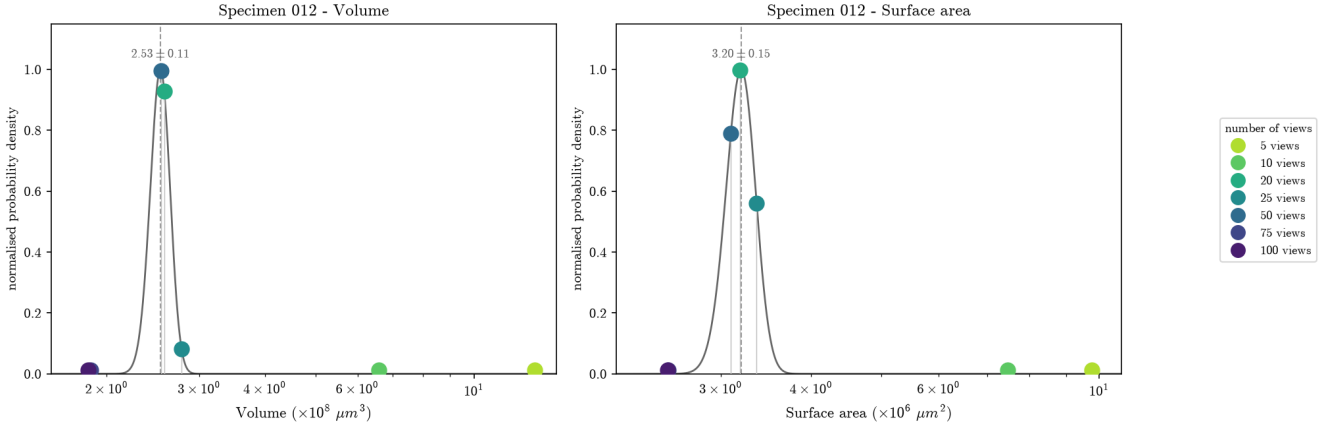
Specimen 009: reconstruction vs. 3 dpf reference



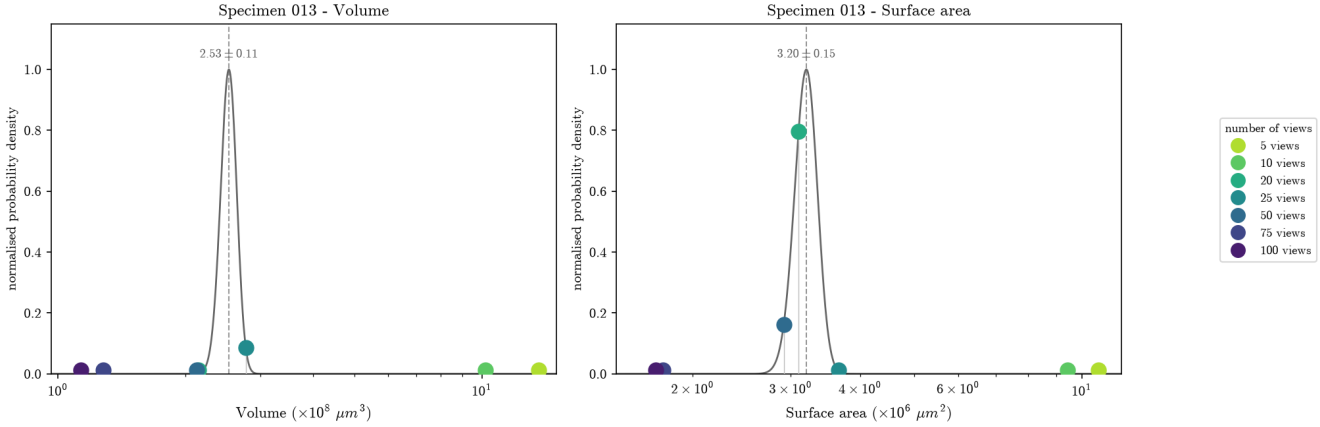
Specimen 010: reconstruction vs. 3 dpf reference



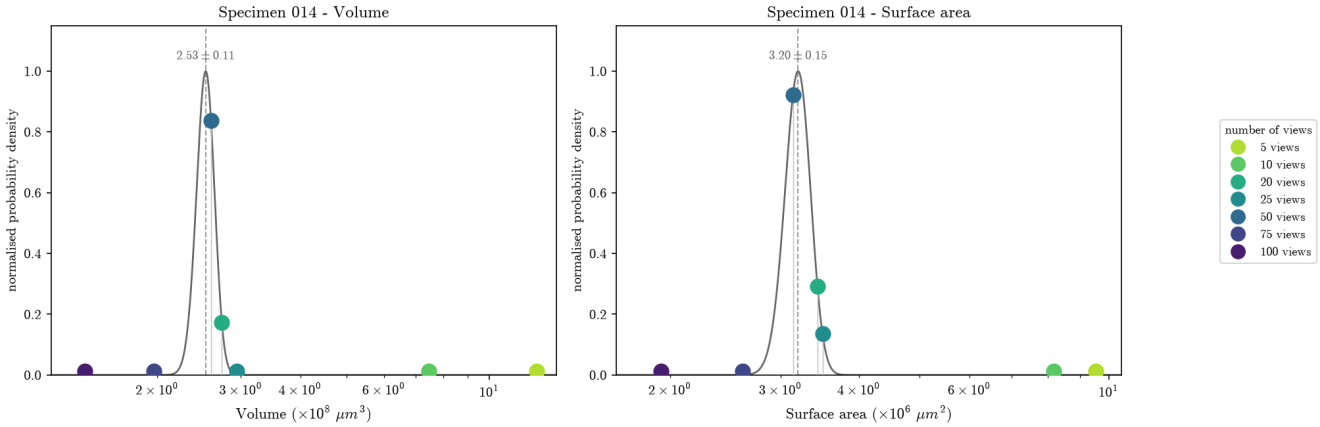
Specimen 012: reconstruction vs. 3 dpf reference



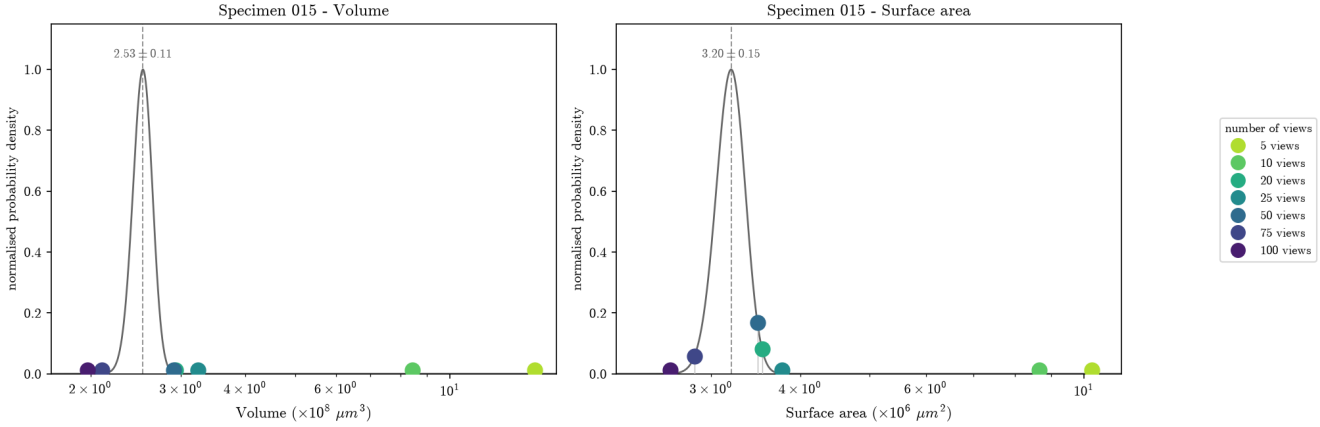
Specimen 013: reconstruction vs. 3 dpf reference



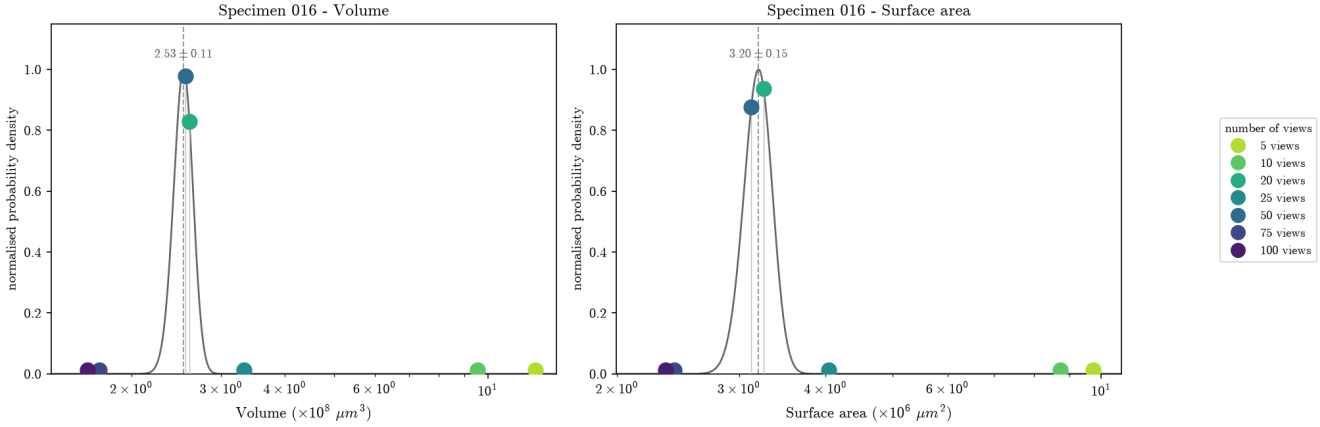
Specimen 014: reconstruction vs. 3 dpf reference



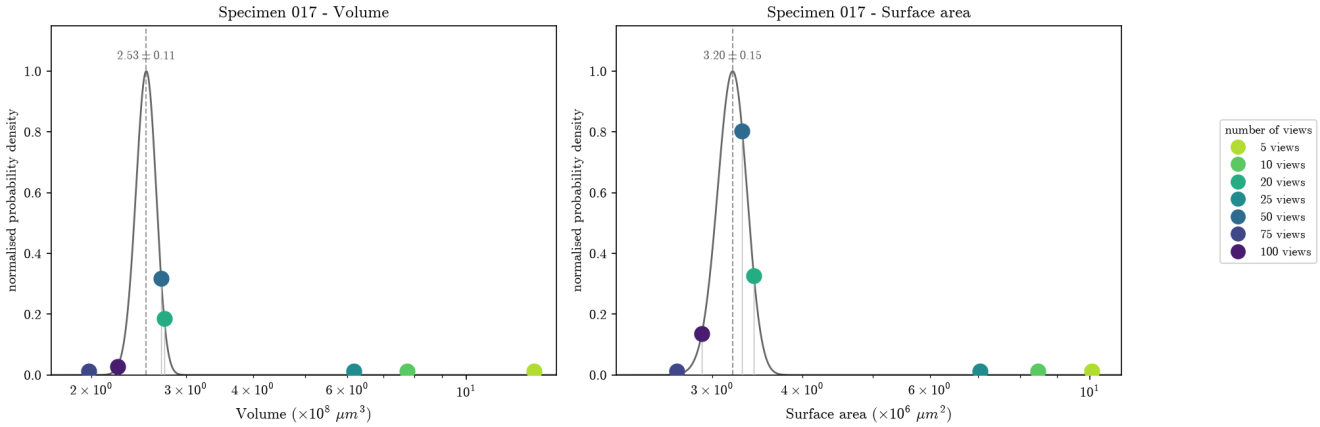
Specimen 015: reconstruction vs. 3 dpf reference



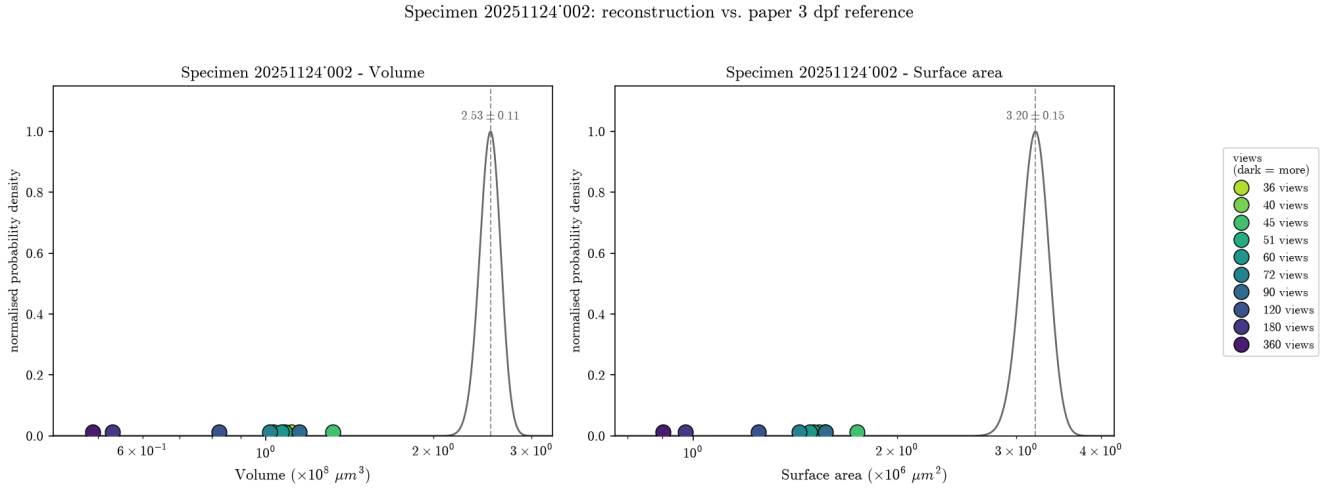
Specimen 016: reconstruction vs. 3 dpf reference

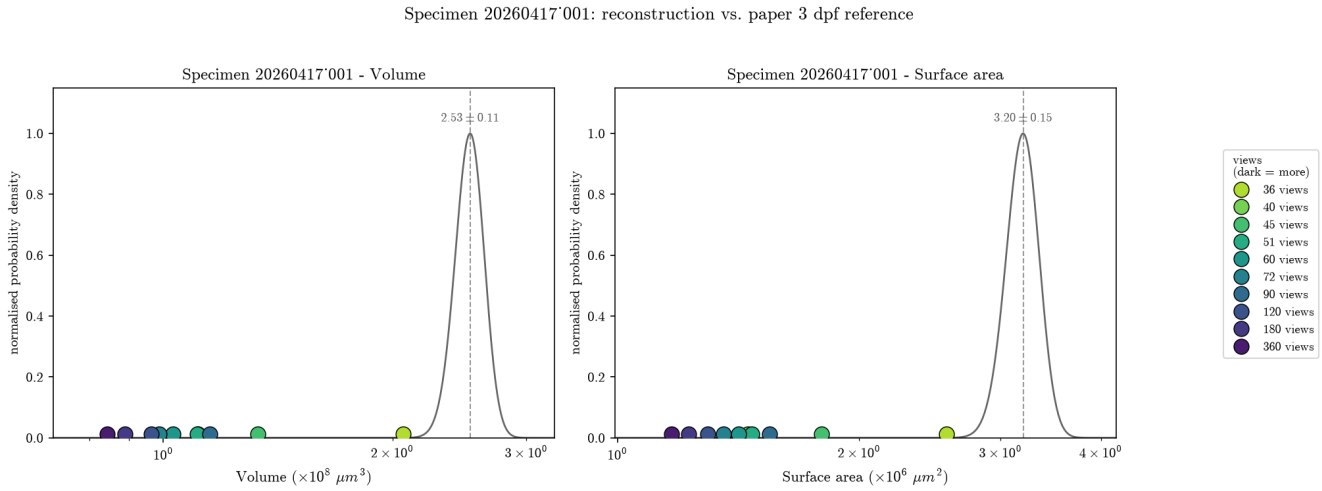


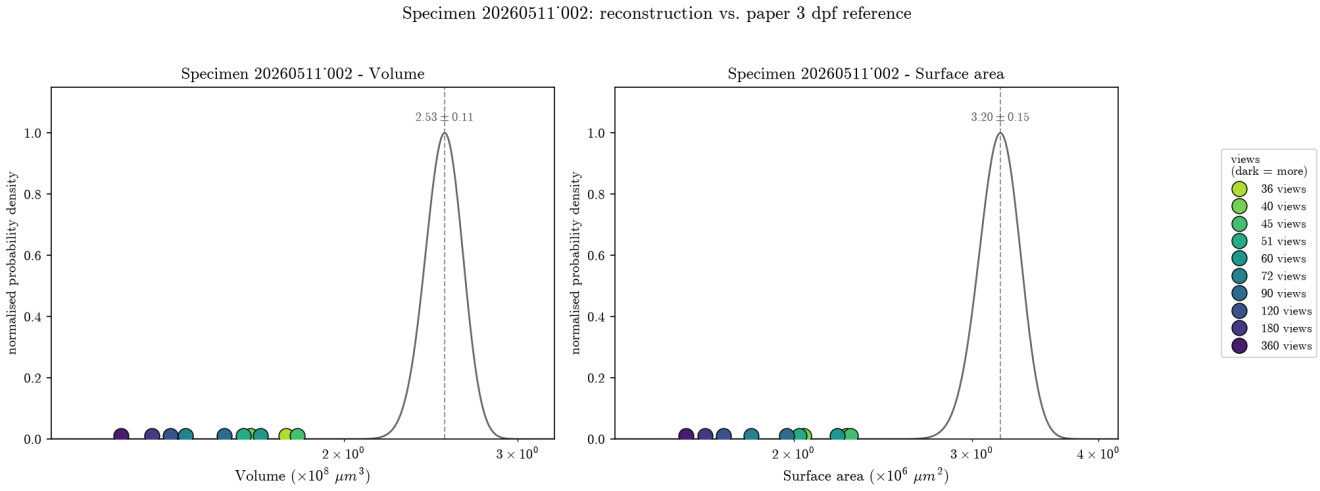
Specimen 017: reconstruction vs. 3 dpf reference



A.2 Plots of the 3 successfully reconstructed dataset of 360 images






Specimen 20260511˙002: reconstruction vs. paper 3 dpf reference
Specimen 20260511˙002 - Volume
2.53 ± 0.11
normalised probability density
Volume (×10^8 μm^3)
Specimen 20260511˙002 - Surface area
3.20 ± 0.15
normalised probability density
Surface area (×10^6 μm^2)
views
(dark = more)
36 views
40 views
45 views
51 views
60 views
72 views
90 views
120 views
180 views
360 views