Robust Real Time Zebrafish High Speed Tail Tracking

Samuel Yang
Stanford University
samuely@stanford.edu

Abstract

The real time detection and quantification of Zebrafish swim behavior from high speed video data is an essential component of a Zebrafish virtual reality system, where body-fixed Zebrafish are suspended in a gel and shown visual stimuli simulating a real world swimming environment. Here, several methods for extracting the Zebrafish tail position are explored and a method for real time tail tracking is developed. Furthermore, with the aim of eventually utilizing supervised or unsupervised machine learning to classify various types of swim behaviors, a method for extracting concise and useful features from the tail position data is developed.

1. Introduction

The goal of this project is to robustly identify and track the position and orientation of a Zebrafish tail in high speed video data sets. Zebrafish are small (1” - 2”) fish that serve as model organisms for several biological research areas. They can be genetically modified to be transparent, with selective regions being fluorescent under certain circumstances[3],[2]. One commonly studied Zebrafish behavior is their swimming behavior[1],[4]. Swimming involves large motions of the tail, and as such, using computer vision techniques to automate the identification and tracking of Zebrafish tail position will be extremely useful in any studies related to their swim behavior.

In particular, this project is a piece of a larger project to build a Zebrafish virtual reality system where a real time method for tail tracking is required. More details are described in Appendix A, but in brief, a Zebrafish is suspended in a gel and imaged from below with a high speed camera as it sees a virtual environment projected in front. Figure 1 shows the setup, and Figure 2 shows a sample image from the high speed camera.

The general approach is to first develop a method for segmenting the tail within each frame, and then from there to extract the tail position across various frames. Once this is accomplished, a method for extracting a useful feature for building a machine-learning classifier is developed. In addition, all methods and code were developed and optimized to run in real time.

2. Background

Zebrafish swim behavior in free swimming Zebrafish has been studied and characterized quite extensively[4]. Zebrafish have a variety of motor swim responses including the optokinetic reflex (OKR), optomotor response (OMR), prey tracking responses and escape swim responses [9]. However, methods for characterizing behavior in freeswimming Zebrafish may not readily apply to the situation of body-fixed Zebrafish where real time methods are required [7].

Developing a robust method for tracking Zebrafish tail movement in real time may be challenging for several reasons. Figure 2 shows a sample image. The zebrafish body
is at a fixed position under a microscope, while its tail is free to move. Many factors can vary across image data sets, including the Zebrafish age (which affects size, shape and transparency), position within the frame and position relative to the background, the Zebrafish lighting angle, the background lighting angle and the camera focus position. Furthermore, the Zebrafish tail may actually move away from the camera, causing defocus aberrations from image to image.

![Sample image from high speed video data set of a Zebrafish captured with a macro lens.](image)

**Figure 2.** Sample image from high speed video data set of a Zebrafish captured with a macro lens. In the background is a microscope objective lens. The goal is to detect tail position within video stream, in real time, and to classify the Zebrafish behavior based on the tail positions.

The Data Set

The data set for this project consists of 11 million frames of high resolution, high speed video data collected using the fish theater setup over the summer (see Appendix), in which various fish were shown various types of visual stimulus in hundreds of experiments. At the outset of this project, this dataset was largely unexplored. In fact, this project actually involved quite a lot of data mining and visualization even before the computer vision and machine learning techniques were developed. After working with this data set over the course of the project here are some important observations about the data.

Within each frame, there is a single Zebrafish whose body is fixed and oriented in the same direction under the microscope (visible as a large circular ring in the background).

What varies from experiment to experiment are factors such as the position within the frame, size and transparency of Zebrafish, lighting (angle, total luminosity), camera frame rate, frame resolution. The background image varies quite a lot as well, depending on the particular preparation of the gel in which the fish is suspended (various parts of the gel are carved out to give the fish varying degrees of freedom).

Additionally the size of the data set, stored on a remote server, made it difficult to visualize and explore. Furthermore, there are segments of recorded video before/after each experiment was actually started which had to be ignored.

Finally, there was quite a large behavioral variation from experiment to experiment (apparent only after the tail segmentation and tracking code was developed, optimized and run on 1 million frames). The most active fish were constantly moving their tail in response to the visual stimulus, attempting various type of swim behaviors. However, some of the least active fish had tail motion as low as 3% of the time.

3. Approach

Given the large and unexplored data set described in the previous section, the general approach toward the project was to continuously iterate back and forth between working with a small hand-picked data set and developing the methods, with each iteration involving more of the data set and the methods requiring less human supervision.

As for the computer vision methods, the approach can be described as a single pipeline logically broken down into these steps.

1. **detect fish tail base**—or the first frame of a given image sequence, identify the pixel coordinate of the point where the fish tail connects to the fixed body. This is accomplished using a combination of edge detection, morphological operations and a Hough transform.

2. **segment fish tail**—given the coordinate of the base of the fish tail from part 1 and a frame buffer containing the last N frames, output an image containing the segmented tail only. A few methods were explored, but a robust background subtraction was found to work best.

3. **fit model for fish tail position**—Given a segmented image tail image, fit a parameterized model of tail position. A modified fourth degree polynomial fitted using a weighted least squares cost function was used.

4. **extract features for fish behavior classification**—Given the fitted parameterized model of tail position in each of the last N frames, compute a suitable set features to be used for swim behavior classification.
5. **classify fish behavior**—Using an (optionally hand-labeled) representative data set of fish swimming behavior, train a classifier on the features from part 4 to distinguish between various swim behaviors. Use this output at for each frame to drive the visual stimulus for the next frame. This section is to be completed in future work.

The following is a flow chart indicating the approach toward the problem at hand, which happens to match the structure of the code as well. In Figure 3, grey cells represent data inputs or outputs, while white cells represent processes (programs).

![Flowchart](image)

Figure 3. Flow chart of the general approach and developed software (with the same structure). The large image data set explored at the same time that methods for tail tracking are developed. The end goal is to build a classifier to classify fish behavior from tail position in real time.

First, the data was simply explored and small subsets were extracted in order to detect the fish tail base position (section 4.1). Once this was working, the actual tail tracking code (center cell) uses this function along with the methods described in sections 4.2 and 4.3. These tail positions are overlaid upon the original video and outputted as the tagged video. Beyond just tagging the positions, we can extract features (section 4.4) to be used to build a classifier (section 4.5) for fish behavior. The dashed cells and paths indicate future work.

The code implementing these methods was first developed in Matlab, but much of it was then optimized to run in real time in python. See Appendix B for more details about the code.

4. **Experiments**

The steps in the approach outlined above will be discussed in detail in each subsection here. Where appropriate, various qualitative and quantitative performance metrics will be described.

4.1. detect fish tail base

The first step of the processing pipeline is to determine the pixel coordinate where the fish tail connects to the body. The particular use for this information will be come evident in the next sections.

Figure 5 shows a small representative subset of fish experiment preparations in which we need to identify the fish tail base coordinate, along with the detected position marked with a white ‘plus’.

The following method, outlined in Figure 4 was developed to generate these results. First, Canny edge detection is used to identify edges [5]. We see a cluster of edges exists surrounding the fish body, but possibly in other areas of the image as well. Next, the borders of the borders of the image are masked off and a morphological closing operator is used to connect the various clusters of edges. The resulting image is then run through an edge detection again, after which a Hough transform searches for the circle representing the microscope objective tip behind the fish body. Finally, the base of the tail coordinate is found within a region just next the objective tip, aligned on the same axis.

![Images](image)

Figure 4. The first stage of the processing pipeline requires detecting the base of the fish tail within a given frame. A combination of edge detectors (a), closing operators (b) and Hough transform (c) are used to locate the coordinate of the base of the fish tail, shown as the white ‘plus’ in (d). (Not shown) image correlation with a template for the two eyes is used for eye detection.

This method was developed on a small subset of images and then run on the entire set of 123 different fish experiments, achieving 70.7% accuracy. Qualitatively, it appears...
that those 20% of failures occurred when the bright disk surrounding the fish body wasn’t there, and the Hough transform fails. After looking into this, I realized that for higher magnification microscope objectives, this objective is actually smaller. After adjusting the search space of radii for the Hough transform to include these smaller radii, the accuracy improved to 92.7%. The remaining failures appear to occur when there is no bright disk at all, causing the Hough transform to fail completely. Within Figure 5, a white box around the plus indicates the Hough transform failed.

**Figure 5.** Fish tail base detection succeeded on 92.7% of the 123 experimental images tested; shown here 20 such images, of which there are 6 of the 9 failures and 14 of the 114 successes. The tail base position is marked by the white ‘plus’.

**extracting fish eyes (bonus)**

This is not related to the goal of trail tracking, but is extremely relevant for the overall goal of correlating fish behavior (including tail and eye motions) with neural activity. A simple extension of the previous method allows for the fish eyes to be located and extracted from the image as well. This information could then be passed to existing code for tracking the orientation of the eyes over time (using SIFT, for example, to generate trackable features for optical flow).

From the previous part, the Hough transform is used to identify the region containing the fish body. This information is then passed to the fish eye extraction code, which uses a template of the fish eyes to compute the correlation. The results are shown in Figure 6.

This code was then run on all of the images for which the Hough transform succeeded in finding a match, which included 112 images. Of those, the eye detection code worked on 93.8% of those images.

**Figure 6.** As an added bonus, the fish eyes can be detected with an additional step after detecting the tail base position. A large ‘plus’ marks the tail base position while cropped images of the eyes are displayed at the lower right. Accuracy was 93.8% on 112 images.

**4.2. segment fish tail**

The task here is to detect the fish tail within the image frame, given only the last N frames. Two classes of approaches both utilizing information from multiple frames were considered here, with both being implemented and tested (by eye, see next section for more quantitative evaluation metric) and the second method was found to work better. Approaches for detecting the tail within a single frame only using methods such as correlation or even some sort of supervised learning were immediately ruled out due to the variations in background image, low signal intensity and noise within the images; it may still be possible to do but perhaps not as robust as utilizing information from multiple frames.
Generating and Tracking Edge Descriptors

This approach toward detecting the tail within an image involves utilizing the standard optical flow procedure. First, a repeatable and distinctive feature descriptor for the fish tail needs to be determined, and from that the tail position could be tracked and segmented in successive frames. Methods such as SIFT could be used to generate features independently for each frame [8].

However, I found a fundamental problem with these approach on this particular data set. Given the variation in backgrounds depending on the preparation (see Figure 5), many of the found features were in the static background. Also, depending on the defocus level of the fish tail, more or fewer features would be found on the tail itself. Hence, if it were just the fish tail on a clean background, then such a feature detector would work fine, but that isn’t the case with this data set.

One possible work around to handling static detected features in the background is to keep track of which features actually move, and which stay stationary from frame to frame, and from that to estimate which features actually correspond to the tail, and which correspond to the tail. This approach may work if the data is being post-processed, such that information from future frames is readily available, or if a long enough history of motion is utilized. Given that some of the fish only moved 3% of the time, this approach would take quite some time to develop and validate.

Background Estimation and Subtraction

The approach I settled on for detecting and segmenting the tail was to first estimate the background image and then to simply perform a background subtraction. The main challenge here is estimating the background. After experimenting with a few methods, a simple one that worked reasonably well was to take the median pixel value across the last \( N \) frames in the frame buffer to form the background image. This is then subtracted from the current frame, producing a noisy segmented tail image. Median filtering is done to remove noise, along with a Gaussian blur. The resulting image is thresholded with a dynamically computed threshold to produce the segmented tail image.

Figure 7 outlines the method for background estimation and subtraction in detail. First, the median pixel value across the last \( N \) frames in the frame buffer is taken to form the background image. This is then subtracted from the current frame, producing a noisy segmented tail image. Median filtering is done to remove noise, along with a Gaussian blur. The resulting image is thresholded with a dynamically computed threshold to produce the segmented tail image.

Figure 8 shows qualitatively the result of this fish tail segmentation in the red channel, along with the original
contrast-enhanced image in the green channel. The bar at the bottom of the image indicates whether motion is detected (red for motion, blue for no motion). As evident from the figure, the fish often stops moving its tail at off-axis positions, but the estimate of the tail position (blue trace) remains accurate.

4.3. fit model for fish tail position

Once we have the segmented image containing only the fish tail, we can treat each pixel as a point in 2D space and to fit a parametric model for the tail position and orientation for this particular frame.

Here, we find the $(x_i, y_i)$ coordinates of the nonzero pixels in the segmented fish tail image, $I_s(x, y)$ from the previous section as well as the pixel values, $\{w_i\}$ where $w_i = I_s(x_i, y_i)$, and attempt to fit an $n^{th}$ degree polynomial function $f(x) = \sum_{j=0}^{n} a_j x^j$ to these points using a weighted least squares fit. To establish a constraint that the polynomial must perfectly fit the base of the tail, $(x_0, y_0)$, obtained from the first stage of the pipeline, we attach a large weight to minimizing $(f(x_0) - y_0)^2$. Furthermore, we know that the tail must be tangent to the fish body’s horizontal orientation at this point as well, so we can add another term to the cost function, $f'(x_0)^2$ which forces the derivative to go to zero at this point as well.

The modified weighted least squares problem for fitting a polynomial to the set of nonzero points in the segmented tail image is:

$$\min_{\{a_j\}} \sum_{i=1}^{N_s} w_i (f(x_i) - y_i)^2 + \lambda \left( (f(x_0) - y_0)^2 + f'(x_0)^2 \right)$$

where $N_s = \dim\{(x_i, y_i)\}$.

After some experimentation, it was determined that a fourth degree polynomial ($n = 4$) yielded exactly enough freedom to accurately represent the space of possible fish tail positions and orientations while avoiding overfitting.

Note that this fitting procedure could be made more robust using robust fitting methods such as RANSAC[6] or perhaps using a Huber penalty function rather than the standard least squares. However, in practice this method seemed to work well enough. Occasionally bright spots away from the fish appear which will skew the fit.

One slight modification had to be made to accommodate a special case where a polynomial fit (and for that matter, attempting to fit any function) breaks down. As shown in Figure 8, occasionally a fish will execute an escape swim maneuver where it’s tail will bend back 180 degrees over or under itself. A work around to avoid this issue is to simply mask off points too far above or below the tail. This did not appear to affect the performance of the fit.

4.4. extract features for fish behavior classification

Once we have the parameterized polynomial model for the current tail position (as well as those from the last N frames), we can start to come up with a good feature descriptor for describing the fish behavior over a local region in time.

One method that comes to mind is to find a simple summary statistic for summarizing the fish tail position at each frame. Naturally, we can look at the curvature of the tail, and infer its behavior from the direction and extent that the tail is bending.

From the previous part, we have the polynomial fit $f(x) = \sum_{j=0}^{n} a_j x^j$. We can take the second derivative, $f''(x) = \sum_{j=2}^{n} a_j (j-1) x^{j-2}$ and evaluate this function, integrating across the entire tail to get the “total curvature”. This single number for each frame, when plotted over time, qualitatively seems to correlate well with fish activity. In simplest terms, a positive curvature corresponds to the tail bending up, and a negative curvature corresponds to a tail bending down. Swimming patterns involve oscillating up/down tail motions which result in sinusoidal curvature across time (see Figure 9). The larger a turn, the larger the curvature as well.

Hence, at least qualitatively, the curvature as defined here appears to be the perfect feature for starting to actually classify fish behavior from a sequence of tail position.

evaluation of results for fish tail position

Because hand labeling the fish tail position for this much data would be too time consuming for the scope of this class project, a simpler method is used instead. The frames with the tail position labeled (see Figure 8) are presented to a human user and a count of the incorrect frames is kept. In total, over approximately 30 experiments and 20,000 frames, the tail tracking was accurate over 99% of the time.

Figure 9. Given the extracted tail positions across the last N frames, a concise feature descriptor of the fish tail behavior can be constructed by (a) simply computing the curvature, a single number representing the sum total second derivative, for each frame. The descriptor (b) of the behavior at a given frame is a vector of dimension N consisting of the curvature of the past N frames (5 here). This feature descriptor could be useful in a correlation analysis with the visual stimuli seen by the fish, or used to build a fish behavior classifier using either clustering or supervised learning.
4.5. classify fish behavior

From the previous section, we have a simple feature for classifying behavior. Before we can start classifying, we need to find a representative training data set containing fish swim behaviors that span all possible behaviors.

(a) forward swim

(b) routine right turn

(c) routine left turn

Figure 10. The curvature as extracted from the last 5 tail positions (traces plotted in blue above each plot) for characteristic swim maneuvers as described in [4]. This feature descriptor could be useful in a correlation analysis with the visual stimuli seen by the fish, or used to build a fish behavior classifier using either clustering or supervised learning.

After covering approximately 25% of the data set of 10 million frames and excluding fish with very little activity, several characteristic swim behaviors were observed, shown in Figure 10. Here, the tail positions for several consecutive time points are shown, along with the computed curvature metric as defined in the previous section. Just visually, it’s clear that the five-dimensional curvature vector quite distinctly correlates with the particular swim behavior.

There were other observed swim behaviors in the data that did not look similar to these, and there are also swim behaviors documented in the literature in [4] that were not observed in this data set. This could be simply that the range of visual stimuli shown to the fish in these particular experiments did not evoke those other swim behaviors, namely the escape response which involves large tail motions.

Hence, to close the loop on the processing pipeline shown in Figure 3 we need to build a fish behavior classifier based on this curvature feature extracted from the fish tail position. However, before this is done, more data needs to be collected to get a more representative set of fish swim behaviors.

There were, however, some interesting observations on just the fish behavior. In the references that studied Zebrafish swim behavior [4, 7], the analysis was done on free swimming fish, and not body-fixed fish. Based on observing the data processed by the tail tracking methods, it seems like there are differences in the fish behavior when they are body-fixed. For example, in the few escape swims attempted, the fish tail never exhibited the counterbend that’s typically seen on the return trajectory. Also, the small J-turns for orienting seem to last longer, possibly due to the lack of visual feedback. With more data, it would be interesting to study these differences in behavior for free swimming and body-fixed Zebrafish.

5. Conclusion

A robust and real time method for extracting Zebrafish tail position from a high speed camera video stream has been developed and tested. In addition, a method for generating features from a sequence of tail positions for real time behavior classification has been implemented.

Through the process of developing these methods, several important lessons have been learned. First, even though this project was intended to be more of a computer vision and machine learning project, it actually required quite a lot of data mining and visualization to simply understand the challenges that would be involved in processing the data. For example, just trying to find portions of the data set where the fish were moving for fish that only moved 3% of the time on average was a challenge that required developing some basic motion detection code just to start visualizing the data.

Also, the process of iterating back and forth between data exploration and method development was essential to covering a sizable portion of the data set. The initial methods for tail position tracking were developed on a mere several hundred frames, but then tested and validated on several thousand frames. Doing further behavior classification would probably require analysis of the total 10 million frames in order to identify segments of characteristic behavior that could easily be labeled. Covering yet another two orders of magnitude of data would probably require further optimization and parallelization of the code to do this efficiently, though once such a behavior classification model has been developed, it should be readily applicable to the real time tracking code without further optimization.

Finally, some standard procedures for algorithm development proved useful again for this project. Most of these methods in this project were first prototyped in Matlab until they were working fairly robustly, and then time was spent optimizing the code in python to run in real time.
ods also started with as much supervision as possible using hand tuned parameters which were later replaced with automatic parameter selection.

Future work

Future work would involve extending upon the behavior classification. To do this, a more representative data set of natural fish behavior would be required. Such a data set would need the behavior to be hand labeled, such that some sort of fish behavior classifier could be trained upon the data. The resulting classification model could then be applied to the tail position data for real time behavior classification. The output of this behavior classifier could then be used to drive the visual stimulus projected by the fish virtual reality theater system.

References


6. Appendix A

This project was done only for this class. It is however being done in the context of a larger research project to develop a Zebrafish virtual reality system. Nothing else (code, methods, etc.) was borrowed from this larger project except for the raw video from the high speed camera, which served as the data set for this computer vision project.

For completeness, however, this larger project will be described in brief here. The goal is to image with a microscope, Zebrafish brain activity during various behaviors such as free swim and prey capture. To achieve this, the Zebrafish must be suspended in place under the microscope, and a small theater system projects a virtual reality for the Zebrafish to see. An additional high speed camera records the Zebrafish from below, from which the position and orientation of its eyes and tail is visible. Data with this setup was collected over the past summer, but none of it had been analyzed. This class project aimed to develop the computer vision tools necessary to efficiently and analyze the data, with the ultimate goal of developing the real-time feedback system required to allow the Zebrafish to control the virtual world, swimming and behaving as it would in natural situations.

7. Appendix B

The software flow chart gives an overview of the code developed for this project. All methods were first prototyped in Matlab. The critical parts of the code were then optimized in python to run in real time (60 frames per second).

The following files were developed for this class project. Bolded names indicate files where significant computer vision methods are implemented.

```python
exploreHSdata.py - extracting subsets of images get_tail_pos.m - detects tail base position test_get_tail_base.m - evaluates tail and eye detection get_tail_base_pos.m - python-to-Matlab helper function get_eye_pos.m - detects eye positions track_tail.py - detects and tracks tail over time process_tail_positions.py - computes tail curvature play_images.py - converts output to video for evaluation
```

The following are library files not written by me:

```python
exploreHSdata.py - loads images wpolyfit.m - weighted least squares polynomial fit houghcircle.m - Hough transform
```