EPFL

CREATE Lab Semester Project

2DOF Fish Closed-Loop Control

by Ricardo Francisco
Robotics Master's

Professor
Prof. Josie Hughes

Supervisors
Nana Obayashi

January 2, 2026

Contents

1 Introduction

Bio-inspired robotic fish are a growing topic in robotics, as they offer several advantages over conventional propeller-based prototypes. In particular, they generate less noise pollution in the environments they study and can achieve high maneuverability. However, these advantages come with significant challenges, since the dynamics of such systems are highly nonlinear and often underactuated, making control design non-trivial.

This project builds upon a previous master's thesis conducted in the CREATE Lab on a two-degree-of-freedom (2DOF) robotic fish prototype [1]. That work investigated the advantages of introducing a second control point in the caudal fin in addition to tail actuation. The presented project, however, focuses on the study of the Strouhal number and the cost of transport, considering only open-loop control.

To progress toward closed-loop control, it is first necessary to formalize and consolidate the existing work. This includes identifying appropriate sensors to close the control loop—specifically, a vision-based system using a camera—and establishing a reliable interface between the motor command implementation and the visual feedback. Although closed-loop control is not implemented in this project, two promising approaches are identified: bio-inspired control strategies, such as those presented in [2], and optimal nonlinear control methods, such as model predictive control, as explored in [3].

The objective of this project is therefore to build upon the previous thesis work and establish a structured framework for testing closed-loop control strategies in the future. This framework includes methods for commanding the motors of the robotic fish, acquiring camera footage, detecting and tracking the fish, and performing image processing to extract relevant state information for control. This work provides the necessary experimental and software infrastructure to enable systematic evaluation of closed-loop control strategies for bio-inspired robotic fish.

2 Methods

Before moving on to closed-loop control, it is first necessary to understand how the robotic fish works.

The robotic fish consists of a rigid forward hull and a flexible articulated spine that enables compliant bending in the horizontal plane. Its motion is decomposed into two degrees of freedom:

- **Tail flexion:** Lateral bending of the tail is achieved by actuating the spine to generate a bending moment, producing an undulatory motion analogous to fixed-end beam bending.
- **Caudal-fin rotation:** The caudal fin is independently actuated, allowing precise bidirectional control of its angular position during swimming maneuvers.

This motion can be visualized in Fig. 1.

2.1 Controlling the fish

Each degree of freedom is actuated individually: the tail is controlled by a Dynamixel motor, and the caudal fin is controlled by a servo motor. By commanding each motor, it is possible to control the overall movement of the fish. Detailed information on the mechanical assembly can be found in Appendix 5.

For the implementation, a simple approach was used: a Python program running on a laptop, connected to the fish via an umbilical cable. Motor commands were sent using a lab-developed

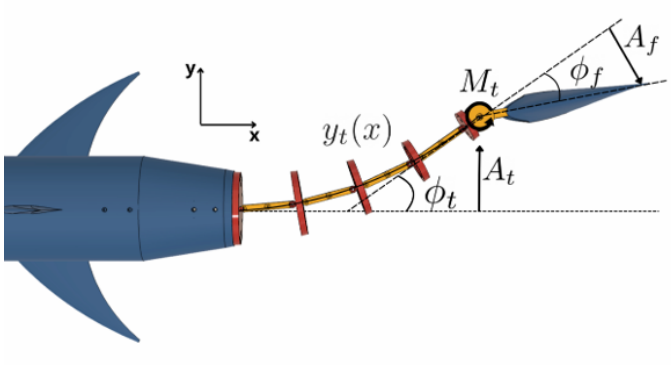


Figure 1: Kinematics of the robotic fish

Python package that interfaces with the Dynamixel U2D2 motor controller to command the tail. The servo motor controlling the fin was commanded through an Arduino running the corresponding code for position control.

2.2 Identification of the Fish

To define a suitable method for the identification of fish, it is necessary first to determine which states of the fish must be estimated. Based on the previous work presented in [1], closed-loop control mainly targets the surge velocity and the yaw angle of the robotic fish. In this work, these states are estimated by tracking the position of the fish over time.

By computing the difference in position between consecutive frames and dividing by the corresponding time interval, the surge velocity can be estimated. Consequently, the vision processing pipeline follows the sequence outlined below:

fish detection → estimation of real-world position → velocity computation → state estimation

Obtaining the Real-World Position

To estimate the real-world position of the robotic fish, three AprilTags with known locations are used. These markers are reliably detected using OpenCV functions. Knowing their positions in the world frame allows the use of a Perspective-n-Point (PnP) method to estimate the camera pose.

Specifically, the P3P algorithm is employed to compute the rotation and translation matrices that relate the world coordinate frame to the camera frame, using the known 3D marker positions, their corresponding 2D image projections, and the camera intrinsic parameters. This procedure follows the approach described in the OpenCV documentation on pose estimation opencv_pnp.

The resulting projection relationship can be expressed as:

$$\mathbf{P}_{\text{image}} = \mathbf{T}\,\mathbf{\Pi}\,\mathbf{R}\,\mathbf{P}_{\text{world}}, \tag{1}$$

where $\mathbf{\Pi}$ is a 3×4 identity matrix augmented with a column of zeros, and $\mathbf{R}$ and $\mathbf{T}$ represent the rotation and translation of the camera with respect to the world frame.

By inverting this process, the real-world position of the fish is obtained by back-projecting the detected pixel position into the world frame. A ray is projected from the camera center through the image point and intersected with the plane $z = 0$, which corresponds to the plane where the AprilTags are located and where the fish operates at a known height.

Identifying the Fish and Its Orientation

Identifying the orientation of the fish is significantly more challenging than detecting the AprilTags. For this, classical computer vision techniques without the use of markers were selected, as attaching a marker to the fish is impractical due to its shape.

The orientation estimation pipeline is composed of the following steps as shown in Fig. 2:

- **Image Preparation** Transformed image to grayscale and applied gaussian filter to reduce any unwanted noise .
- **Adaptive thresholding:** Used instead of global thresholding to improve robustness to lighting variations and ensure consistent segmentation.
- **Morphological dilation:** Applied to fill gaps in the thresholded image and obtain a solid foreground region.
- **Contour extraction:** Contours are extracted using OpenCV's `findContours` function and filtered based on their pixel area to remove small black zones and noise.
- **PCA analysis:** Principal Component Analysis is performed on the contour image points to compute the center of mass and the principal axis, which corresponds to the fish orientation.
- **Final orientation disambiguation:** Since PCA yields an orientation with a 180° ambiguity, the width of the fish is sampled along its principal axis. The direction with the smaller average width is identified as the head of the fish, as the nose is narrower than the body.

Note: This processing pipeline is applied only to the pool area. Since the pool boundaries are known, the input image is cropped to this region to avoid interference from external objects near the pool.

This pipeline requires several tuning parameters, which may need to be adjusted depending on the lighting conditions:

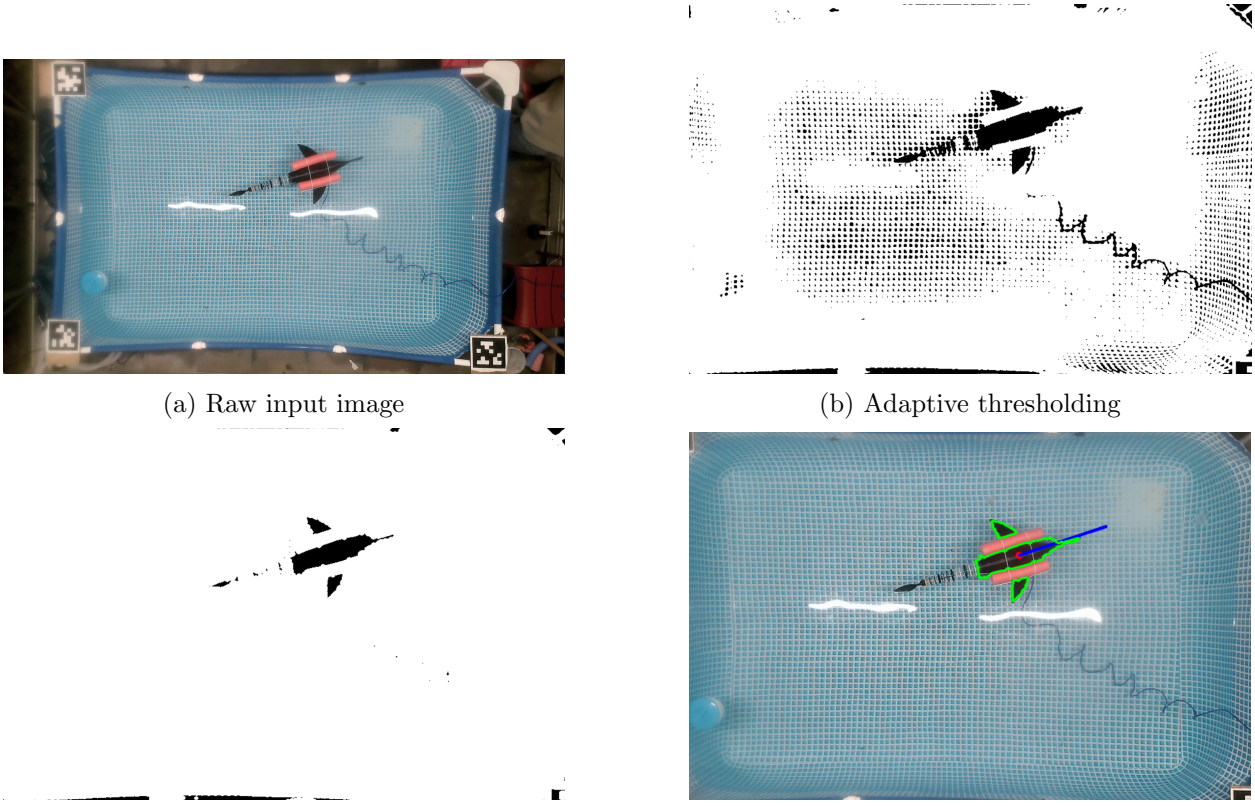
- Gaussian filter kernel size
- Adaptive thresholding window size and constant
- Morphological dilation kernel size and number of iterations
- Minimum and maximum pixel area thresholds for contour filtering

The process assumes that the principal axis obtained from the contour corresponds to the longitudinal axis of the fish.

3 Experimental Setup

With the identification and control frameworks in place, it is possible to set up the experimental environment to evaluate closed-loop control strategies. Unfortunately, due to a failure of the Dynamixel motor, full closed-loop experiments could not be conducted. Nevertheless, the complete connection and control scheme intended for the experiments is illustrated in Fig. 3.

As shown in the figure, the main Python control code runs on a laptop, which acquires the camera feed through a USB-A connection. The laptop is also connected via USB to a U2D2 adapter, which interfaces with a powered hub supplying both power and communication to the Dynamixel motor. In parallel, the laptop communicates with an Arduino board through a serial



(a) Raw input image

(b) Adaptive thresholding

(c) Morphological dilation

(d) Contour extraction and PCA orientation

Figure 2: Computer vision processing pipeline used for fish detection and orientation estimation.

USB connection. The Arduino runs the corresponding firmware required to receive commands from the Python program.

The Arduino generates a PWM signal to control the servo motor, providing both the control signal and a common ground reference. Through the umbilical cable, the U2D2 adapter and servo control signals are routed to the robotic fish, including the required power supply (+12 V) and ground connections.

With the connections in place, the robotic fish can be deployed in a pool measuring 248 cm by 146 cm with a depth of 65 cm. Since the fish is not able to float autonomously, it is equipped with two buoyancy floats to maintain it at the surface.

Three AprilTags are placed in three corners of the pool, with their positions measured and defined in the software. These known spatial references are used to perform the calibration required for the vision-based localization algorithm.

4 Results

4.1 Metric Position Estimation

To validate the accuracy of the metric position estimation, an initial experiment was conducted using five markers. Three markers were used for spatial calibration, while two additional markers with known ground-truth positions were placed in the pool to evaluate the estimation accuracy. This setup allowed a direct comparison between the estimated positions and the measured ground-

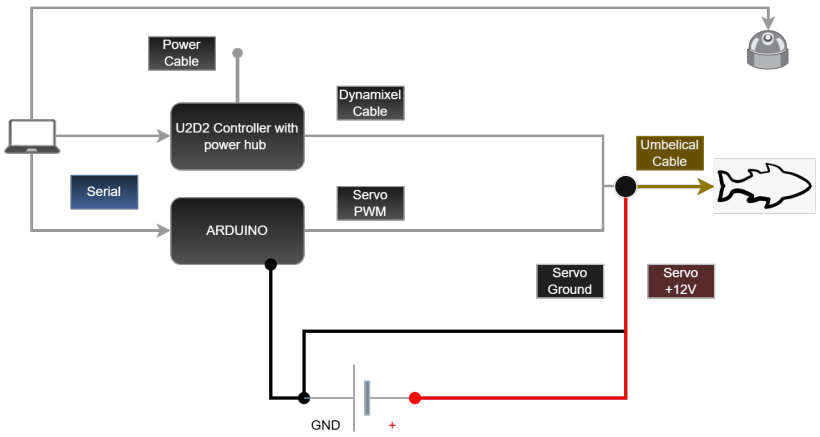


Figure 3: Setup for the robotic fish experimental platform.

truth values.

The resulting position estimates are shown in Fig. 4, where the horizontal axis corresponds to the x coordinate and the vertical axis to the y coordinate. For the first test marker, the estimated position was (1.3914, 0.6654) m, compared to the ground-truth position of (1.35, 0.70) m, resulting in a Euclidean error of 0.0539 m. For the second marker, the estimated position was (0.4053, 0.8133) m, with a ground-truth position of (0.37, 0.80) m, yielding an error of 0.0377 m. On the left lower corner the vision metric axis can be observerd, which match the pool shape.

These results indicate that the vision-based localization framework is capable of estimating metric positions with an accuracy on the order of a few centimeters, which is sufficient for the intended closed-loop control experiments.

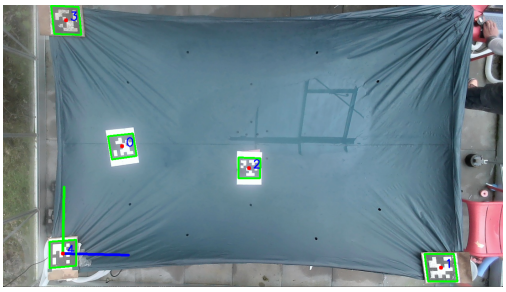
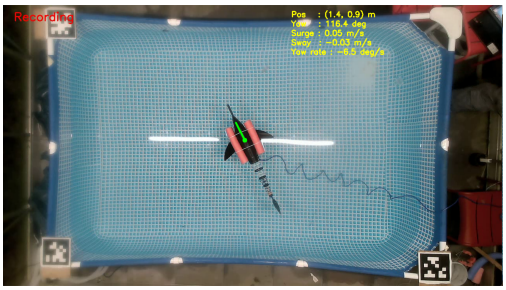


Figure 4: Metric position estimation validation using markers.

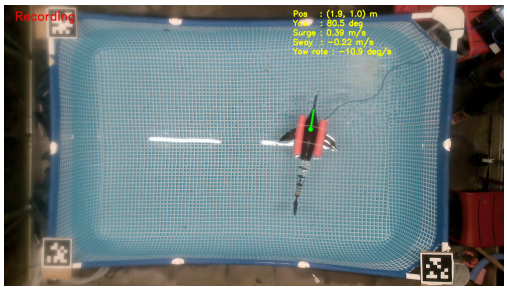
4.2 Fish Tracking

Due to the Dynamixel motor not working, the fish was not able to move on its own. Instead, the fish was manually displaced using the umbilical cable in order to validate the full state identification pipeline.

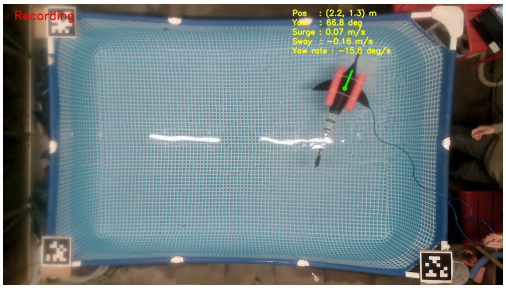
Figure 5 shows four representative frames extracted from different instances of the recorded video sequence. Each frame illustrates the detection of the fish, including position and orientation estimation. Despite the overall good results, a few mismatches can be observed, particularly in subfig. 5d.



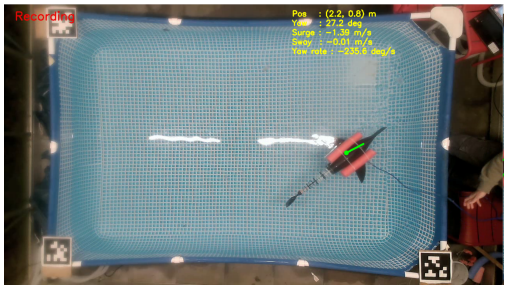
(a) Frame 1



(b) Frame 2



(c) Frame 3



(d) Frame 4

Figure 5: Fish tracking and state identification at four different time instances.

5 Discussion & Conclusion

For the results presented in Section 4, it should be noted that the experimental validation is limited in scope due to time constraints and hardware issues. Nevertheless, the experiments demonstrate that the metric position estimation achieves centimeter-level accuracy, which is sufficient for the intended application and validates the feasibility of the proposed approach.

The localization performance could be improved through a more accurate camera calibration by reducing the reprojection error via careful image selection and proper checkerboard coverage. Additional validation could include controlled experiments in which the fish moves along straight trajectories at constant speed, allowing ground-truth surge and yaw measurements to be compared with the estimated values and enabling a more quantitative performance assessment.

As is common in classical computer vision approaches, the proposed pipeline is sensitive to lighting conditions. Since the experiments are conducted outdoors, variations in sunlight exposure and glare can affect the detection performance. Although the method is generally robust, sudden lighting changes may introduce errors. In particular, the yaw estimation exhibits noticeable noise, on the order of 2 to 5 degrees, which can lead to spikes in the estimated yaw rate depending on the sampling frequency, especially during turning maneuvers.

Overall, the proposed vision-based identification framework performs reliably and is straightforward to use, as it is implemented entirely in Python. It can be readily extended with closed-loop controllers for surge and yaw, as suggested in the introduction.

A relatively simple improvement would be to replace the classical computer vision pipeline with a learning-based approach, such as a YOLO model, to detect the fish and estimate its orientation. By training a custom dataset under varying lighting conditions and fish configurations, such a model could provide a more robust and straightforward detection method.

The developed vision package enables future work on closed-loop control, since the software infrastructure is already in place and can be directly augmented with control algorithms.

References

[1] G. Veigas Marques, "Two degree-of-freedom tail-fin mechanism for efficient and agile robotic fish undulatory swimming," CREATE Lab – Computational Robot Design & Fabrication Lab, EPFL STI IGM CREATE Lab, Master's Thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, Jul. 2025.

[2] G. Manduca et al., "A bioinspired control strategy ensures maneuverability and adaptability for dynamic environments in an underactuated robotic fish," *Journal of Intelligent & Robotic Systems*, vol. 110, no. 69, 2024. DOI: 10.1007/s10846-024-02080-9.

[3] M. L. Castaño and X. Tan, "Model predictive control-based path-following for tail-actuated robotic fish," *Journal of Dynamic Systems, Measurement, and Control*, 2019. DOI: 10.1115/1.4043152.

Appendix

Fish Assembly Documentation

- Master's thesis of fish development: appendix/GabrielThesis.pdf
- User manual for the robotic fish prototype: appendix/UserManualFish.pdf

Git Code

The code developed is available in: code