Maneuverability and Heading Control of Compliant Biomimetic Swimming Devices

by

Anirban Mazumdar

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 2007

Signature of Author:_____
Department of Mechanical Engineering
05/11/2007

Certified by:_____
Kamal Youcef-Toumi
Professor of Mechanical Engineering
Thesis Supervisor

Accepted by:_____
John H. Lienhard V
Professor of Mechanical Engineering
Chairman, Undergraduate Thesis Committee

Maneuverability and Heading Control of Compliant Biomimetic Swimming Devices

By

Anirban Mazumdar

Submitted to the Department of Mechanical Engineering on May 11, 2007 in partial fulfillment of the
requirements for the Degree of Bachelor of Science in Mechanical Engineering

ABSTRACT

Biomimetic swimming devices that employ compliant mechanisms have shown promise as an alternative to current biomimetic design approaches that involve the use of complex mechanisms. The additional stealth, ruggedness, and efficiency of this approach means that such devices could perform important tasks such as reconnaissance and underwater mapping. Many of these applications also require high levels of maneuverability and closed-loop control. However, maneuverability and heading control are two areas that are relatively unexplored with regard to such devices. Therefore, in order to study maneuverability and control, this thesis outlines a simple dynamic model to predict the maneuvering behavior of compliant biomimetic swimming devices. A comparison of the model predictions with experimental data is also presented. Lastly, the dynamic model is used to successfully design, simulate and implement a compass-based heading control system.

Thesis Supervisor: Kamal Youcef-Toumi
Title: Professor of Mechanical Engineering

Table of Contents

List of Figures

1. Introduction

1.1 Compliant Biomimetic Aquatic Devices

Autonomous Underwater Vehicles (AUVs) currently perform many essential tasks ranging from ocean floor mapping to littoral reconnaissance. For nearly all of these applications, the ability to follow trajectories and headings is an essential quality. In order to rapidly and accurately follow paths and trajectories without the guidance of a pilot, AUVs must have control systems that enable the vehicle to dynamically follow a heading and correct itself in the event of disturbances and errors.

However, heading control is not simply a control problem. The ability of AUVs to track headings and trajectories is limited not by the control system but also by the maneuverability of the vehicle. While conventional underwater vehicles can achieve turning radii of several body lengths [1], natural organisms such as fish can achieve turning radii on the order of just one body length [2]. As a result, devices that emulate the natural swimming and turning motions of fish have the potential to improve on current designs of underwater vehicles.

The idea of creating biomimetic aquatic devices has been actively pursued in the field of robotics in the last decade. Triantafyllou and Barett [3] designed robotic devices that leading up to the RoboTuna in 1994, and currently there is work underway at the University of Essex by Liu and Hu on a robotic Tuna [4]. For the most part, these designs employ classical mechanisms such as linkages and multiple actuators to recreate the complex motions of fish like the Tuna. However, recently Valdivia produced a design involving the use of body compliance to achieve required body motions [5]. Due to its mechanical simplicity, this new design holds promise as an alternative approach for the design of underwater vehicles.

While the swimming dynamics and forces of such compliant biomimetic swimming devices were explored by Valdivia, heading control has yet to be fully explored. Therefore, studying and implementing heading control for compliant

biomimetic swimming devices would not only further the understanding of the maneuvering behavior of such devices, but would also be an essential step towards the developing a new type of Autonomous Underwater Vehicle.

1.2 Previous Work

As described previously, maneuverability and the design of a control system are related issues. Maneuverability and heading control of underwater vehicles is a problem that has been studied thoroughly for many years. Conventional AUVs such as the Monterey Bay Aquarium Research Institute's AUV (MBARI) [6] have achieved heading control. In addition, J.J. Slotine and researchers at the Woods Hole Oceanographic Institute (WHOI) have performed in depth studies of heading control for AUVs. Similarly, maneuverability and heading control has been demonstrated in biomimetic AUVs such as Anderson's RoboTuna and Lu's robotic tuna.

While studies of maneuverability and heading control exist for biomimetic AUVs that rely on discrete mechanisms, maneuverability and controls have yet to be fully researched with regard to biomimetic AUVs that rely on the use of compliant bodies. Knowledge of turning dynamics is essential to the simulation and design of a control system. Therefore, there exists a significant need for research into both maneuverability and controls for such devices.

1.3 Motivation

Several issues motivate a comprehensive study of maneuverability and controls for compliant biomimetic aquatic devices. It is important to develop and verify a simple model for the maneuvering behavior of such devices. Such a model will be essential to the design of any control system by allowing analysis and simulations to assess instability, command following, and disturbance rejection. In addition, such a model can serve to increase understanding of how the device functions physically and can help provide insights into how maneuvering performance can be improved.

Similarly, the design and implementation of a heading control system can facilitate experimental studies of swimming performance and swimming. Currently, the devices are uncontrolled and sometimes frequently fail to swim along a desired path. In addition, the implementation of such a control system will enable the development of autonomous prototypes. Currently the devices are tethered using cables due primarily to the need for the user to "pilot" the device. Finally, a control system is essential for the pursuit of any applications. For example, it has been suggested that the device be coupled with a camera in order to track targets or paths. Such an application would require the device to track heading commands

1.4 Problem Statement

Currently the area of maneuverability and controls for compliant biomimetic aquatic devices is an area that has not been fully explored. Not only does this hinder analysis of the performance of the devices but it also limits understanding of the impact of a compliant mechanism on maneuvering performance. In addition, the absence of a control system limits the experimental value and applicability of such devices.

This thesis will attempt to address this problem by outlining and experimentally verifying a simple model describing the maneuverability of compliant biomimetic devices. In addition, this thesis will aim to use the aforementioned dynamic model to design, simulate, and implement a heading control system that will enable the device to track heading commands.

1.5 Thesis Outline

This thesis begins with a detailed description of the dynamic model of fish-like maneuverability and its derivation. Chapter 2 describes the design of simulations for assessing the results dynamic model. Chapter 3 describes the design of the heading control system. Specifically, relevant control principles and methods are discussed. In addition, Chapter 3 illustrates the creation of a set of simulations for predicting closed loop performance. Chapter 4 discusses the implementation of the heading control

system, detailing the physical prototype for testing the control system and the associated electronics and control software. Chapter 5 outlines the experimental design and compares the experimental data with the results from the aforementioned dynamic model. Chapter 6 presents the results of the closed-loop control system and compares the results with simulations. Lastly, Chapter 7 contains the conclusions, final recommendations, and ideas for future research.

2. Modeling Maneuvering Dynamics

2.1 Introduction

In order to achieve a proper understanding of how the swimming device will behave dynamically it is important to create a simple model for describing swimming and turning dynamics. This chapter will outline the development and simulation of a simple turning model tailored for the compliant biomimetic swimming devices designed and constructed in the Mechatronics Research Laboratory (MRL). Specifically, this chapter will describe the equations of motion, the estimates for thrust forces and drag coefficients, and the simulation design.

2.2 Equations of Motion

In order to derive the equations of motion, the first step is to establish a coordinate system. While the actual swimming devices have 6 degrees of freedom, this thesis will be restricted to planar motion. Therefore, the device can be treated as having 3 degrees of freedom. Figure 2.1 illustrates the choice of a coordinate system relative to a fixed xy axis; i and j represent the directions parallel and perpendicular to the motion of the swimming device respectively, while θ represents the orientation of the device relative to the fixed xy coordinate system. Note the slightly unconventional coordinate system; the motivation is to help simplify image processing. The three dimensions, i,j, and θ can be used to describe all possible configurations of the swimming device.

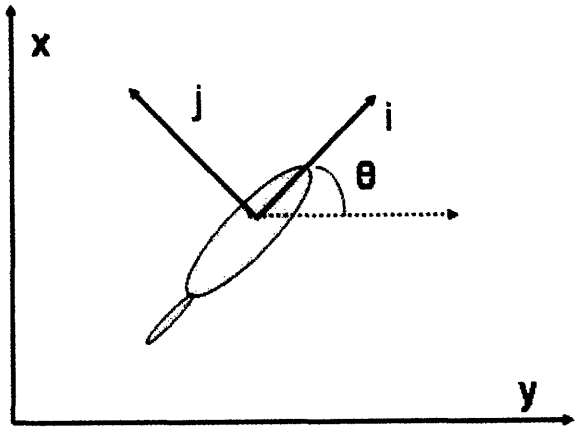

Figure 2.1: Coordinate system used for identify the position and orientation of the swimming device.

By combining this coordinate system with Newtonian dynamics, the three equations governing the motion of the device can be derived. Since the device moves through water, the equations for drag forces on immersed bodies outlined in White [10] are used. First for the direction along the trajectory (*i*):

$$(m + m_{add})\dot{v}_i = T_i - \frac{1}{2}(C_{Di}\rho A_P)_{body} v_i^2 - \frac{1}{2}(C_{Di}\rho A_P)_{fins} v_i^2 \tag{2.1}$$

In these equations T_i represents the thrust force in the *i* direction, C_{Di} represents the fluid coefficient of drag with regard to motion in the *i* direction, m_{add} represents the added fluid mass, and A_p represents the wetted area. Since the densities of the prototype and water are very similar, m_{add} will be assumed to be comparable to the mass *m*.

Similarly, for the direction normal to the swimming direction (*j*), the equation of motion can be written:

$$(m + m_{add})\dot{v}_j = T_j - \frac{1}{2}C_{Dj}\rho A_P v_j^2 \tag{2.2}$$

In these equations T_j represents the thrust force in the *j* direction and C_{Dj} represents the fluid coefficient of drag with regard to motion in the *j* direction. For this direction the fins can be neglected due the fact that the body shape is the dominant shape exposed to the fluid flow.

Lastly, for the orientation of the device, equations can be derived by examining the torques with regard to the center of mass. Note that in this case the fins can be

considered as having negligible effects due to the fact they are situated very near the center of mass and therefore contribute very little to any net moment. Other fin configurations would create two additional moments (caused by the motion of the fish in the *i* direction) about the center of mass. Due to the relative complexity it is instructive to use a free body diagram (Figure 2.2).

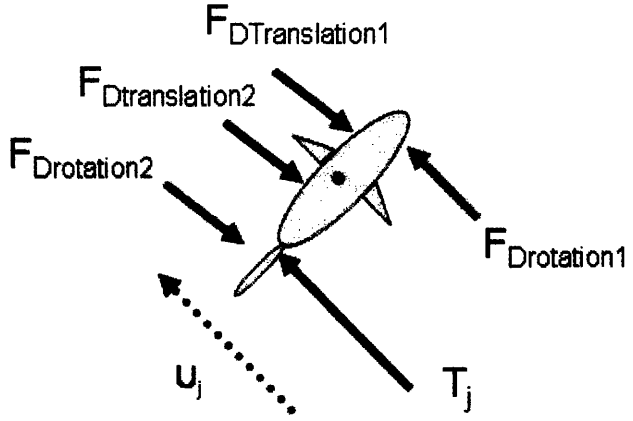

Figure 2.2: Free body diagram of torques on the device.

Breaking the distributed drag force into two separate forces results in the equation:

$$(I_{zz} + I_{add})\ddot{\Theta} = T_j * r_{tail} - F_{Drotation1} * r_1 - F_{Drotation2} * r_2 - F_{Dtranslation2} * r_{t2} + F_{Dtranslation1} * r_{t1} \tag{2.3}$$

Where r represents the respective moment arm, and F_{D1} and F_{D2} represent the fluid drag forces. In addition, I_{add} represents the effect of the added fluid mass in the θ direction. For the purposes of this thesis, I_{add} will be assumed to comparable to the moment of inertia I_{zz}. The drag forces can be calculated using the standard equations for fluid drag:

$$F_{Drotation1} = \frac{1}{2} C_{Dj} \rho A_{p1} (r_1 \dot{\Theta})^2 \tag{2.4}$$

$$F_{Drotation2} = \frac{1}{2} C_{Dj} \rho A_{p2} (r_2 \dot{\Theta})^2 \tag{2.5}$$

$$F_{Dtranslation1} = \frac{1}{2} C_{Dj} \rho A_{p1} (v_j)^2 \tag{2.6}$$

$$F_{Dtranslation2} = \frac{1}{2} C_{Dj} \rho A_{p2} (v_j)^2 \tag{2.7}$$

Lastly, the moment arms (r_1, r_2) can be calculated by borrowing from standard beam bending analysis and integrating over the body:

$$r = \frac{\int_a^b \sigma(u) u du}{\int_a^b \sigma(u) du} \tag{2.8}$$

where σ represents the load per unit length, and u represents a position along the body of the device.

2.3 Estimates for Thrust Force

In order to estimate the thrust force on the device, it is important to understand the swimming motion employed by thunniform swimmers. This swimming motion, characterized by high amplitude tail oscillation [7] what the compliant biomimetic aquatic device attempts to emulate. Estimating the thrust force is therefore a two step process. The first step for estimating the thrust force that results from this motion is to determine the lateral displacement at the tail, and the second step is to use the estimates of lateral displacement to calculate the thrust force.

Using the equations and guidelines set forth by Valdivia [8], the lateral deflection of the tail (H) can be estimated. A rough visualization for deflection of the tail is illustrated in Figure 2.3.

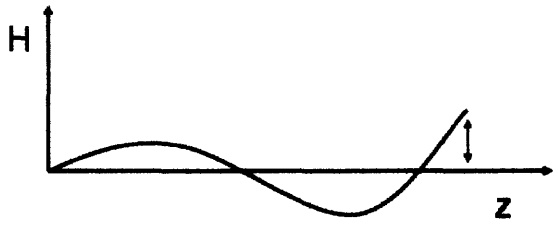

Figure 2.3: Estimated body and tail deflection. A top view of the body's centerline showing swimming motions.

This tail deflection was estimated by Valdivia [8] who treated the tail of the device as a "slender body". The resulting estimates for the tail deflection (H) and the phase (φ) are:

$$H \sim \frac{\frac{M}{(L-a)^2}}{\sqrt{\left(\frac{EI}{(L-a)^4} - (\rho_{device} A_c + m_l)\omega^2\right)^2 + \left(\frac{\mu I \omega}{(L-a)^4}\right)^2}} \tag{2.9}$$

$$\varphi \sim \tan^{-1}\left(\frac{\frac{\mu I \omega}{(L-a)^4}}{\frac{EI}{(L-a)^4} - (\rho_{device} A_c + m_l)\omega^2}\right) \tag{2.10}$$

in these equations, E, ρ_{device}, and μ represent material the material properties of the swimming device (modulus of elasticity, density , viscosity). Similarly, m_l represents the mass per unit length, while A_c and I represent the cross sectional area and cross sectional moment of inertia respectively. Lastly, L represents the length of the device, while a represents the position at which the moment M is applied.

Once the tail deflection and the tail phase are determined, the thrust force can be estimated. Using Sir James Lighthill's Elongated Body Theory, Valdivia [8] obtained an estimate for the average thrust:

$$\langle T \rangle = \frac{m(L)}{4}\left(\omega^2 H(L)^2 \left(1 - \frac{U^2 k^2}{\omega^2}\right) - U^2 H'(L)^2\right) \tag{2.11}$$

where $m(L)$ represents the added mass at the tail, U represents the velocity of the swimming device, $H'(L)$ represents the slope of the tail, and k represents the wave number. The added mass at the tail $m(L)$ can be calculated using the circular approximation described in Videler [9]:

$$m(L) \approx \frac{s(L)^2 \rho_f \pi}{4} \tag{2.12}$$

where the added mass at the tail is estimated by creating a virtual circle with a diameter (s) equal to the body depth.

Finally, the thrust forces in the i and j directions must be determined. Due to symmetry, there will be no net thrust in the j direction when the device swims without a bias (asymmetric swimming motion). However, should a bias exist, there will be a thrust component in the j direction. A simple way to estimate this effect is to examine the thrust vector components.

Figure 2.4: Illustration of tail-thrust vector.

As Figure 2.4 illustrates, the resulting thrust force for small tail angles can be decomposed in the following manner:

$$F_i = F\cos(\psi) \approx F \tag{2.13}$$

$$F_j = F\sin(\psi) \approx F\psi \tag{2.14}$$

$$\tan(\psi) \approx \frac{H}{L-a} \tag{2.15}$$

Since the tail deflections and angles for such prototypes are small, the small angle approximation is applicable. With this approximation, average tail angles can be calculated and used to determine the thrust components.

2.4 Coefficients of Drag

In order to create a complete model that can be used in simulations, estimates for the coefficients of drag for the body (C_{Di}, C_{Dj}) and for the fins (C_{Dfin}) are needed. The first step for estimating the drag coefficients is to estimate the Reynolds number in order to know whether the flow around the swimming device is laminar or turbulent. The Reynolds number (Re) is defined as

$$\mathrm{Re} = \frac{\rho_{water} UD}{\mu_{water}} \tag{2.16}$$

where D represents a characteristic length for the device. The characteristic length (D) can be approximated as the typical length of prototypes (~0.254m). Using previously obtained experimental results [8], the swimming velocity (U) can be estimated as ~0.5 body lengths/s (~0.127 m/s). Using these values and the properties of water, the Reynolds number was calculated to be approximately 3600. This Reynolds number value implies laminar flows. Using the presented area as the characteristic area and published

tables for drag coefficients of smooth bodies [10], the static coefficients of drag can be approximated. For the fins, the shape can be approximated as an airfoil. The tables of typical drag coefficients in Hoerner [11] indicate an estimate of 0.1 for C_{Dfin} is appropriate. The shape of the nose of the device was approximated as an ellipsoid. Using the tables of drag coefficients provided in White [10], a rough estimate rough estimate of 0.3 was used for C_{Di}. For the j direction, the device is exposed to cross flow rather than flow across the nose. Therefore the motion through the fluid can be approximated as a cylinder exposed to cross flow. Again using the tables provided by Hoerner, an estimate of 1.1 was used for C_{Dj}. However, for dynamic conditions and simulations, the dynamic conditions must be taken into account. Experimental studies have revealed that for fish, the dynamic drag can be several times higher than the static drag [12]. Therefore, the coefficients of drag for the body (C_{Di}, C_{Dj}) must be multiplied by a dynamic correction factor (K_D~4). Since the device of concern swims with a fishlike motion, $K_D = 4$ will be used.

2.5 Biased Swimming

In order to simplify simulations and the later controller design, it is useful to develop a terminology to quantify a biased motion. Since the motion of the tail is crated by a moment from the actuator, it is useful to examine the actuator output. If the actuator output (also referred to as a swimming signal) is treated as the oscillation of the moment M applied by the actuator between two points a and b, where a represents the "low" and b represents the "high", the bias B can be quantified in the following manner:

$$B = \frac{M_a}{\max(M)} + \frac{M_b}{\max(M)} \tag{2.17}$$

where M_a and M_b represent the actuator torques at the low and high points respectively, and max(M) represents the maximum achievable deflection. The value for max(M) is assumed to be a positive value, while M_a and M_b can be either positive or negative. This is indicated visually in figure 2.5.

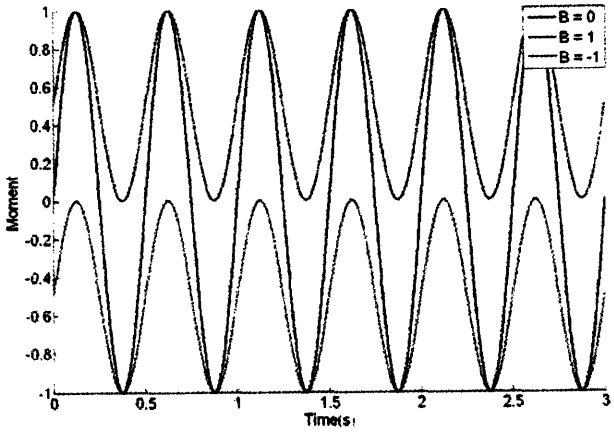

Figure 2.5: A graphical illustration of biased swimming motions

As figure 2.5 illustrates, the result is fairly intuitive; a symmetric swimming motion ($M_a = -M_b$) results in a bias value of 0 while a motion that favors the "low" position will have a negative bias and a motion that favors a "high" position will have a positive bias. The swimming bias (B) will be referred to extensively throughout this thesis, and will be very useful for control system design.

2.6 Simulating Dynamics

Due to the presence of the nonlinear fluidic drag terms, deriving closed form solutions to the equations of motion presents a challenge. While these equations could be linearized, this would limit the applicability of the model to set of operating points. Therefore, simulations provide the best way to understand actual system response.

Matlab's simulink was used to create a software model that could easily be manipulated, simulated and analyzed. Simulink was chosen for two main reasons. First, Simulink is a powerful tool for the simulation and analysis of control systems. Therefore, the simulations that are created for an uncontrolled system can be easily incorporated into designs for closed loop control systems. In addition, the use of Simulink enables the use of block diagram form for entering the equations of motion. The use of block diagram form resulted in an intuitive and simple interface that can easily be viewed, understood,

and debugged. Equations 2.1, 2.2, 2.3, 2.9, 2.10, and 2.11 were used to create the model, and an illustration of the model block diagrams is provided in figure 2.6.

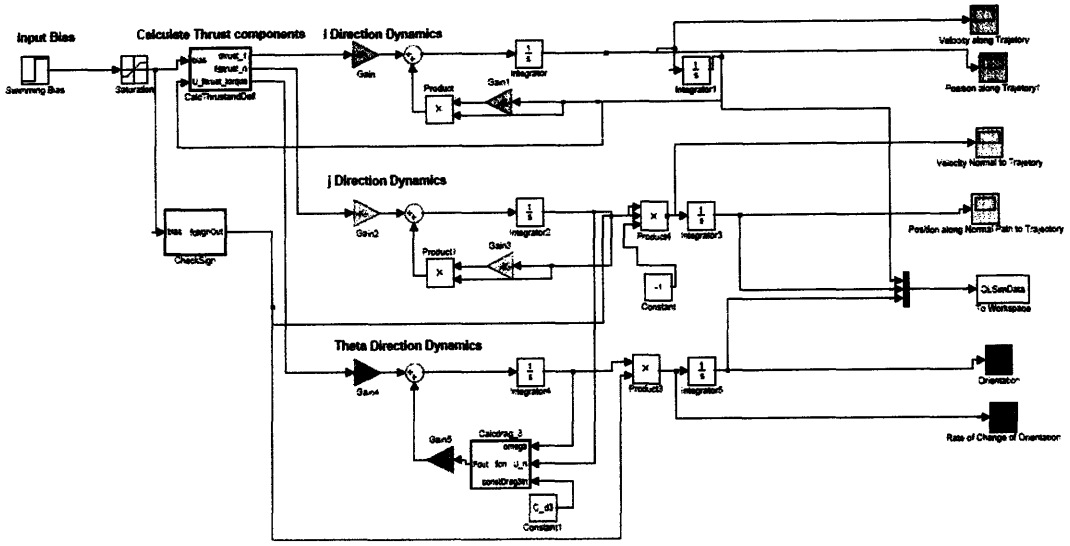

Figure 2.6: Graphical Illustration of the Simulink model for simulating maneuvering dynamics.

The simulation takes a swimming bias signal as the input and uses the aforementioned equations of motion to predict the swimming motions that result from the bias signal.

2.7 Summary

This chapter presented the set of steps necessary for deriving and simulating a model that can approximate the dynamics of a compliant biomimetic swimming device. In this chapter, equations of motion were derived in detail, estimates for thrust were calculated, and drag coefficients were selected. In addition, this chapter described a method with which to quantify biased swimming. Lastly, this chapter culminated with the discussion and description of a Simulink model that can be used to simulate the dynamics from the equations of motion.

3. Heading Control

3.1 Introduction

Heading control is the control of the direction of motion of a vehicle. In this case, heading control will involve attempting to control the swimming direction (i direction). Currently, the biomimetic swimming devices rely on the user to control the heading by visually monitoring the motion of the device and then adjusting actuator signals accordingly. While this strategy may work for surface applications, it will not work for underwater applications as the water will attenuate the radio signals. As a result, there exists a clear need for a closed loop heading control system that does not require user intervention. In addition, heading control is an important area to explore due to the fact that it is an important first step towards more advanced control systems such as trajectory tracking controls or other forms of navigation.

This chapter will outline the development and simulation of a closed loop heading control system tailored specifically for the compliant biomimetic swimming devices designed and constructed in the Mechatronics Research Laboratory (MRL). Specifically, this chapter will describe control system design, the creation of a simulation that makes use of the dynamic model outlined in chapter 2, and the selection of the proper sensor for measuring orientation.

3.2 Controller Design

One way to control heading is to control the absolute orientation of the device. Using this approach, the only sensor measurement required for closed loop control is the absolute orientation. Using these guidelines, a control system can be designed. Figure 3-1 illustrates the controller design in block diagram form.

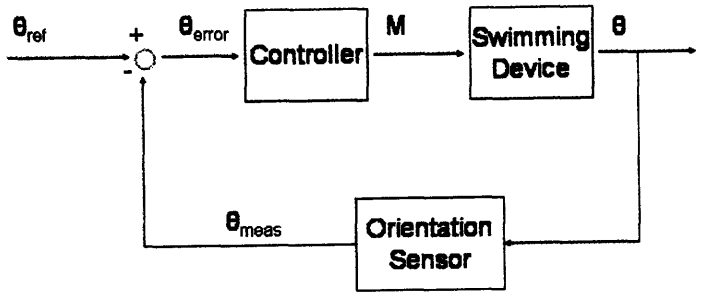

Figure 3.1: Block Diagram for closed loop control of orientation

As figure 3.1 reveals, a reference orientation can be fed into the control system and compared to the measurement from the signal. The resulting error signal is used to calculate the swimming bias which is then used to actuate the device.

3.2.1 Circular Errors

Due to the nature of angular errors, logic must also be included in the control system. For example as figure 3.2 illustrates, when an error of -270°, the controller will attempt to compensate by attempting to turn in the clockwise direction.

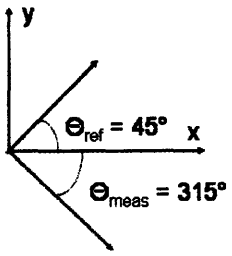

Figure 3.2: Diagram illustrating the nature of angular errors.

However, the diagram clearly illustrates that a counterclockwise turn would be a more logical and efficient approach. These problems occur when the absolute value of the error exceeds 180°. To resolve this error the following operation can be included in the controller logic:

$$\begin{aligned} e &= \theta_{ref} - \theta_{meas} && |\theta_{ref} - \theta_{meas}| \le 180 \\ e &= \theta_{ref} - \theta_{meas} + 360 \quad \text{where} && (\theta_{ref} - \theta_{meas}) < -180 \\ e &= \theta_{ref} - \theta_{meas} - 360 && (\theta_{ref} - \theta_{meas}) > 180 \end{aligned} \tag{3.1}$$

This logic will ensure that the control system attempts to turn in the most logical direction for any angular error.

3.2.2 Calculating the Swimming Bias

The actuator signal (measured with the swimming bias) can be calculated using the error signal. A positive turning bias will cause a counterclockwise turning motion, while a negative turning bias will cause a clockwise turning motion. With this knowledge, the swimming bias can be calculated in the following manner

$$B = -K_c * e$$

where K_c represents the controller gain, and e represents the error signal. Note the negative sign. The negative sign is necessary due to the fact that a positive angular error requires a clockwise turn to correct, while a negative angular error requires a counterclockwise correction.

3.3 Simulating Closed Loop Control

The dynamic models outlined in chapter 2 were combined with the control concepts discussed in section 3.2 in order to create a model that would allow the simulation of closed loop control. Figure 3.3 illustrates the Simulink model used to simulate a closed loop control system. Equations 2.1, 2.2, 2.3 were used as the equations of motion for the device.

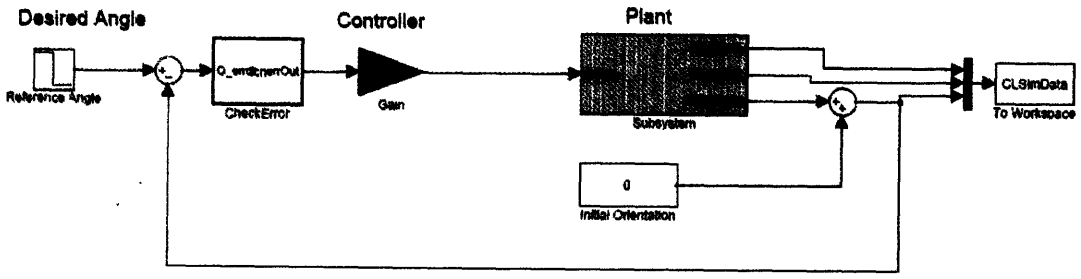

Figure 3.3: Graphical Illustration of the Simulink model for simulating maneuvering closed loop dynamics

The desired orientation (θ_{ref}) serves as the input, and the actual orientation of the device is the measured output. In addition, the simulation outputs the positions and velocities in the i and j directions. The position and orientation data from the simulation can be used to reconstruct the swimming path of the device. Therefore, the swimming path of the device can be viewed and compared with desired paths and trajectories.

3.4 Sensor Selection

For a heading control, there exists a need to measure the orientation of the device relative to an absolute coordinate frame. Three sensors that fit could perform this function are gyroscopes, Global Positioning System (GPS) modules, and compasses. In order to select the proper sensor for the application, it is useful to explore all three options.

3.4.1 Gyroscopes

Gyroscopes are used to measure angular velocities, and our very common in inertial navigation systems. The benefits that exist with regard to gyroscopes are that they are very small, they can be easily interfaced with microcontrollers, and they are stand alone devices that will not be influenced by underwater conditions. The chief drawback with gyroscopes with regard to heading control is that integration is required in order to measure orientation. As a result, only relative orientations can be measured and calibration will be required every time the device is activated. In addition, the need for integration to measure the orientation not only adds computational difficulty, but can also introduce drift into the measurement.

3.4.2 Global Positioning System

GPS modules are used to measure absolute position on the Earth, and are becoming increasingly common in automobiles as well as electronic devices such as cellular phones and hiking equipment. One of the biggest benefits associated with the use of GPS is that GPS systems provide a measurement of the absolute position of the device. In addition, portable GPS systems are now available. There are however, two

big problems associated with the use of GPS. The first issue is resolution. While GPS systems are currently extremely high resolution (10 -20m) [13], resolution on the order of 1m is required for laboratory measurements and precise maneuvers. In addition, water may attenuate the GPS signal, making underwater applications quite difficult.

3.4.3 Compass

The compass has been used as a heading sensor for naval applications for over four centuries. Compasses provide a measurement of orientation relative to the Earth's magnetic field. The biggest benefits of compasses are that they are small, they provide absolute orientation measurements, and digital versions can be easy to interface with microcontrollers. In addition, compasses have already been employed in underwater vehicles such as the RoboTuna [1]. The biggest drawback associated with the use of a compass is that other magnetic fields such as those caused by DC motors or permanent magnets can interfere with the measurement. Since the current prototypes use DC servo motors, this presents a major problem.

3.4.4 Sensor Comparison and Selection

The advantages and disadvantages of each of the three previously mentioned sensors are outlined in table 3.1.

Sensor	Ease of Implementation	Size	Resolution	Vulnerability to Outside Interference	Calibration
Gyroscope	O	O	O	O	O
GPS	-	O	-	-	++
Compass	+	O	+	-	+

Table 3.1: A chart outlining the relative advantages and disadvantages of each sensor type. O signifies a neutral result, while + and – signify advantages and disadvantages respectively.

As the table illustrates, the compass appears to be the best sensor for the heading control. This is due primarily to the fact that the compass will provide absolute

orientation measurements and will be unaffected by underwater situations. While the interference caused by outside magnetic fields is certainly a concern, there exist possible solutions such as shielding or the use of selective sampling.

3.5 Summary

This chapter presented the idea of heading control for a compliant biomimetic swimming device. In this chapter, control concepts were discussed and relevant calculations outlined. In addition, this chapter described the creation of a Simulink model that can be used to simulate the closed loop swimming dynamics of the device. Finally, this chapter included a comparison of various sensing options, and culminated with the choice of a digital compass as the sensor of choice.

4. Implementation

4.1 Introduction

Physical prototypes with functioning control systems are necessary for any experimental studies of maneuverability and controls.

This chapter will discuss the design and fabrication of physical prototypes of compliant biomimetic swimming devices. Specifically, this chapter will describe the selection of the prototype, and the control hardware. In addition, this chapter will cover software for the sensor, construction of the physical prototype, and software for control systems.

4.2 Swimming Device Prototype

Since the decision was made to limit the scope of this thesis to maneuvering dynamics and controls, previously existing swimming device designs were used for all the studies of maneuverability and heading control. Both of the prototypes that were used were Tuna like devices based on a design by Valdivia [8].

This design was chosen for several reasons. First, Tuna due to their high swimming speeds and low drag [7] are creatures that achieve impressive performance characteristics. In addition, the Tuna based prototypes were large enough to contain the necessary circuitry and wiring required for the compass, while also being small enough to perform turning maneuvers within the confines of the MRL tank.

The two Prototypes that were used in this thesis both rely on the novel use of a compliant mechanism to recreate the swimming modes of Thunniform swimmers. In fact, the only major difference between the design for Prototype A and Prototype B is that Prototype B includes two DC servo motor actuated side fins. Other than this difference

the devices are nearly identical; both devices are of comparable geometry, mass, and material properties.

4.3 Control Hardware

A *Plugapod*™ microcontroller from New Micros Inc was used for both the piloting and the closed loop control of the swimming devices. The *Plugapod*™ microcontroller uses a DSP56F803 MPU 16-bit processor and provides 6 PWM (Pulse Width Modulation) outputs as well as 6 Timers, 3 LED control lines, and 3.3 volt and 5.1 volt regulators. For the purposes of this thesis, the *PlugaPod Development Board* from New Micros Inc was used to interface the microcontroller with power supplies and other external circuitry. In addition, the *Plugapod*™ microcontroller was combined with an XBee™ radio chip. This radio chip, used in tandem with a USB radio dongle, enabled wireless communication between a host computer and the microcontroller. Figure 4.1 illustrates the microcontroller, the development board, and the XBee™ radio chip. With regard to software, the *Plugapod*™ microcontroller supports four programming languages: Static C, Small C, Forth, and *Isomax*™.

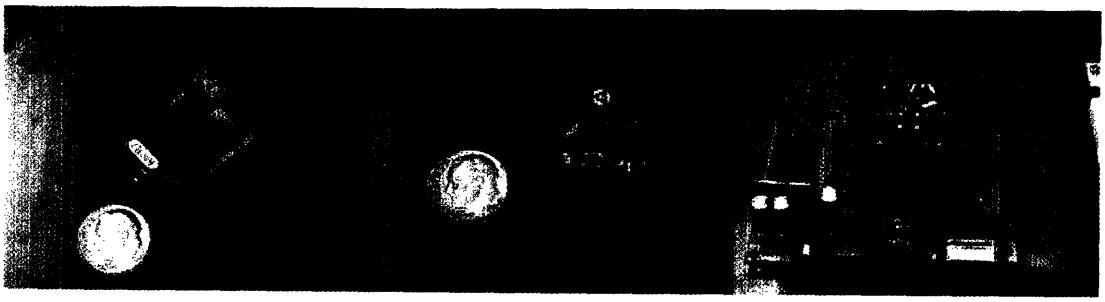

Figure 4.1: From left to right, the *Plugapod*™ chip, the XBee radio chip, and the *Plugapod*™ development board (photos courtesy of Pablo Valdivia [8])

4.4 Sensor

The compass that was selected was a Devantech CMPS03 digital compass. This compass was selected primarily due to its small size (illustrated in figure 4.2) and its use of Pulse Width Modulation (PWM) as its output. The use of a PWM output greatly simplifies the interfacing with microcontrollers. This specific compass adjusts the "high

time" of the square wave in a manner such that the high time is proportional to the angle of the compass. Once the "high time" (t_H) is known, the orientation of the compass (θ_{meas}) can be calculated using the company provided specifications:

$$\theta_{meas} = \frac{(t_H * 1000 - 1)}{0.1} * \frac{\pi}{180} \tag{4.1}$$

the resulting measurement (θ_{meas}) provides an absolute measure (ranging from 0 to 2π) of the orientation of the compass with regard to the local magnetic field.

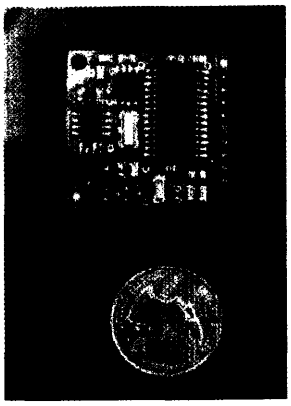

Figure 4.2: Photograph of the Devantech CMPS03 Digital Compass.

4.5 External Circuitry

External circuitry was required to enable the transmission of power and control signals to the prototype, and a simple schematic is provided in figure 4.3. While it would have been possible to construct a completely untethered (no external wires) device, the decision was made to use tethered prototypes due to size and troubleshooting considerations. Two DC power supplies were used to provide power to the system. The first power supply (5.1 V) was used to provide power to the microcontroller. The second power supply (6V) was used to provide power to the DC servo motors. Lastly, The regulated 5 Volt supply from the microcontroller was used to provide power, while a pull up resistor (30 KΩ) was used to provide 5 volts to the unused pins.

PWM signals were used for the control of the DC servo motors. The PWM signals were generated by the microcontroller and routed to the DC motors. Similarly, the PWM output of the digital compass was connected to one of the timers on

microcontroller. Finally, the compass output was connected directly to one of the timing pins on the microcontroller.

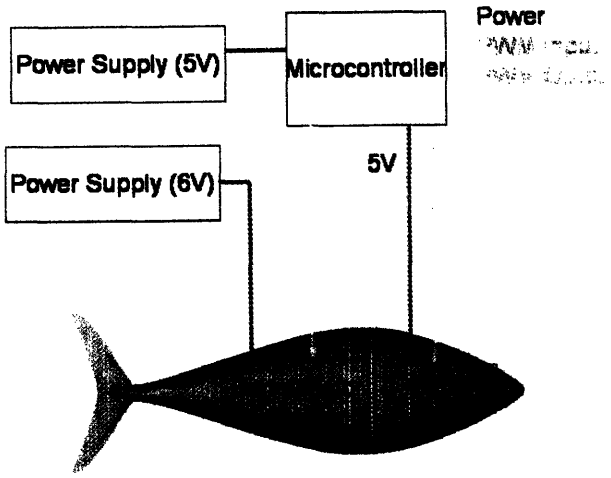

Figure 4.3: A diagram illustrating the electronic connections for the swimming prototype.

Lastly, due to concerns regarding magnetic interference between the DC Servo Motors and the magnetic sensors on the digital compass, the magnetic sensors were removed from the Devantech CMPS03 board and were placed on a separate smaller circuit board. Thin wires were then used to connect the CMPS03 board to the magnetic sensors. Due to the small size of the magnetic sensors, this configuration enabled greater flexibility with regard to sensor placement.

4.6 Fabrication

The device prototypes were fabricated using a casting method developed by Valdivia [8]. In accordance with the hybrid design (body and tail of differing material properties) the head and tail were cast separately. Therefore, casting consisted of a three step process. First, the tail was cast, then the components such as the motors, and sensor were placed within the mold, and finally the body was cast.

The casting of the tail was performed by mixing a 1:1 ratio of parts A and B supplied by EcoFlex Silicone Rubbers, and placing the resulting mixture in the mold. In

order to place the electronics, the mold was reopened and the servo motors, motor housings, transmission mechanisms were placed in the locations indicated within the mold.

Similarly, the compass board and magnetic sensors were placed within the mold. Due to concerns about magnetic interference, the magnetic sensors were carefully removed from the Devantc board and placed in the tip of the nose where they would be as away as possible from the DC servo motors. This configuration is illustrated in figure 4.4. Only prototype B was equipped with a compass. Prototype A was intended for maneuverability studies and was therefore cast without a compass. The last step was to cast the body and head. Quantum Silicone's Q300 product was used for the body and head, and a 1:2 ratio of parts A and B was used. Figure 4.5 provides a view of both prototypes.

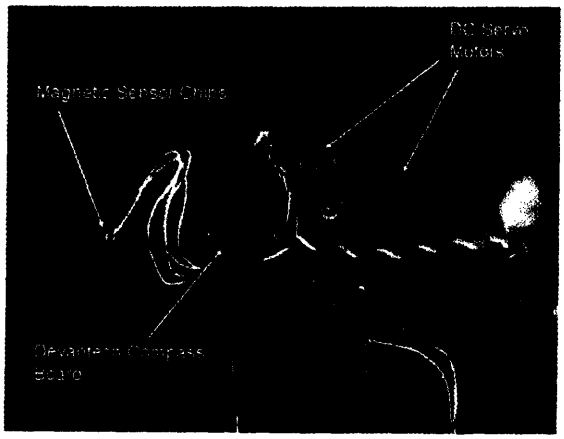

Figure 4.4: Sensor configuration for Prototype B.

Figure 4.5: From left to right, Prototype A and Prototype B.

4.7 Control Software

Two custom written sets of code were used to achieve control of the prototypes. One set of code related to "Piloting" or control of the prototype through commands by the pilot or user. The second set of code attempted to implement a closed loop heading control system. Both sets of code were written in *Isomax*^{TM} and compiled using the NMITerm compiler supplied by New Micros Inc. Isomax was chosen as the programming language due in large part to its intuitive nature and the ease with which it could be compiled and uploaded using the NMITerm program. Once the program is compiled and sent to the microcontroller, the user can communicate with the microcontroller by typing commands into the NMITerm terminal window. These signals can be sent wirelessly to the *Plugapod*^{TM} via the XBee radio.

4.7.1 Piloting

The "piloting software" was written specifically for carrying out experiments where the user can visually monitor the motion of the device. The program enables the dynamic adjustment of swimming parameters such as swimming bias, swimming frequency, swimming amplitude. The user can adjust easily adjust these parameters by using the appropriate keys on the keyboard. When a key is pressed in the terminal window it is sent wirelessly to the microcontroller, and the microcontroller then sends the corresponding signal to the swimming device. The commands were designed in such a way so that their use would be intuitive and similar to a simple computer game. In addition, the microcontroller relays data back to the user. Therefore, the user can verify that the swimming device is actually carrying out the correct commands. This functionality can also be combined with sensors such as the digital compass or thermal sensors. However, it should be emphasized that this program relies heavily on visual feedback in the form of the user or pilot. While this configuration is effective under laboratory conditions, it is highly impractical for applications where the pilot will be unable to visually monitor the device.

4.7.2 Closed Loop Control

The clear limitations of the piloting method outlined above created the need for a closed loop system that could function without constant feedback from a pilot. Unlike the program described above, this program will dynamically adjust the swimming bias based on the signal from the compass. While derivative and integral controllers are feasible, this thesis will be limited to the use of proportional control.

One important consideration with regard to closed loop control related to interference created by the DC Servo Motors within the swimming device. Due to the small size of the prototypes, it was impossible to completely eliminate the magnetic interference. Therefore, the need for a unique sensing scheme emerged. The solution to this problem was to selectively sample the compass signal. Instead of sampling from the compass signal continuously, the compass was read at fixed intervals. During these intervals, the servo motors were returned to their equilibrium positions (reducing the magnetic field) until a compass signal could be read. The delay associated with this sensor reading was ~500ms. Once the compass signal was read, the device resumed its motion.

Finally, the program for closed loop control still allowed the user to maintain supervisory control over the device. The user therefore maintained the ability to turn off closed loop control, stop the fish, or adjust key swimming parameters.

4.8 Summary

This chapter presented an overview of the design and implementation of compliant biomimetic swimming device prototypes. The basic design, the control hardware, and the electronic circuitry were all described in detail. Finally, the chapter closed with a discussion of the two *Isomax*^{TM} programs that were written for the control of the prototypes.

5. Experimental Maneuvering Results

5.1 Introduction

The best way to asses the validity of the dynamic models outlined in this thesis was to carry out controlled experiments with the physical prototypes described in chapter 4. In this chapter, the experimental setup and procedure will be outlined, followed by an overview of the data analysis software. Finally, the experimental results will be compared with the results from the simulations. These results will be used to determine the validity of the model and will also be used to make any necessary changes to the modeling parameters.

5.2 Experimental Setup and Procedure

The experiments were performed in the tank at the Mechatronics Research Laboratory at the Massachusetts Institute of Technology. Figure 5.1 illustrates the 2.5m by 0.6m by 0.6m acrylic tank. A digital camera can be mounted on the frame so that the experiments can be filmed. For these experiments, a Sony DCR-TRV30 NTSC MiniDV digital camera was used. Since the prototypes have swimming frequencies of ~2Hz, the frame rate of 29.97Hz was more than sufficient to capture the swimming dynamics.

Figure 5.1: A photograph of the MRL tank.

The experimental procedure was designed to capture the full dynamic response of the devices. Therefore, experiments began with the prototype at rest within the frame of the camera. The swimming device was then sent the appropriate swimming signal, and its dynamic response to the swimming motion was recorded. Throughout the experiments care was taken to ensure that the power and PWM cables did not exert tension on the prototype and therefore affect the swimming dynamics. These experiments were performed for swimming biases of 0, 0.8, and 1.

5.3 Data Analysis

To fully verify the dynamic models outlined in chapter 2, it was necessary to measure the tail deflection (H), the phase (φ), the position of the center of mass of the swimming device (x_{cm}, y_{cm}), and the orientation of the swimming device (θ). In order to accomplish this task, a Matlab program was written to analyze the video data frame by frame. The program allows the user to manually select the mid body, head, and tail of the device from each frame. Figure 5.2 illustrates a typical frame and the outputs of the Matlab program.

Figure 5.2: From left to right, a typical clip from video data, and the measured head, tail and midbody positions. The left hand image illustrates a typical clip from a set of video data. The right hand image overlays each clip and also marks the head, tail, and mid body position for each frame.

In order calculate the head and tail deflection, the program computes a mean direction associated with the motion of prototype. The deflections can then be calculated by finding the distances of the head and tail from this trajectory. Once the deflection data is obtained, the phase can also be calculated by determining the time lag between the peak deflection at the head and the deflection at the tail.

The final step is to determine the position and orientation of the swimming device so that a trajectory can be created for comparison with the simulation results. The x and y positions of the center of mass can be approximated as the positions of the mid body of the swimming device, and the orientation was measured by calculating the angle created by the head and mid body. Figure 5.3 illustrates the coordinate system with reference to the tank.

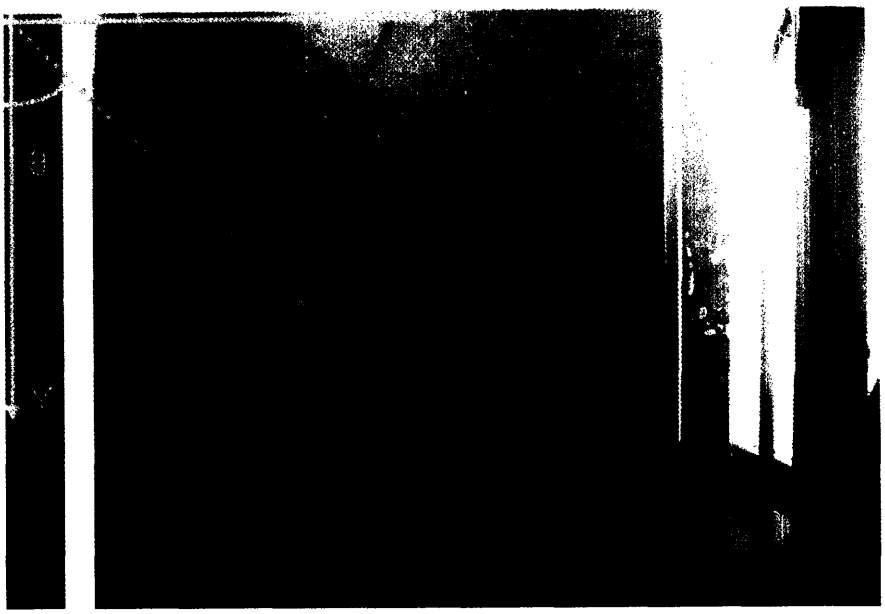

Figure 5.3: A photograph illustrating the coordinate system with regard to the tank. The axis is simply overlaid on an image of the tank. However, the actual coordinate system is created in the matlab software and is used to compute all the position and orientation data for the swimming devices.

5.4 Deflection Analysis

Since the estimates for the tail thrust force components (T_i and T_j) depend heavily on the magnitudes of the tail deflections and phase value, an essential first step was to compare the predictions of the theoretical model with the tail deflections measured using the aforementioned computer program. Figure 5.4 provides a graphical illustration of the predicted tail deflections compared with those predicted by the theoretical model. This figure reveals that the model over predicts the tail deflection by a factor of ~5, causing 80% error. Since the relationship outline by Valdivia [8] was intended only as an order of magnitude estimate, this result is not altogether surprising. A likely source of this discrepancy lies in the age of the prototype. Age causes the material properties (E, μ) to change and these changes cannot be easily predicted or accounted for.

However, since the predictions for the swimming thrust have H^2 dependence, these errors will cause large errors in the prediction for the thrust. In order to evaluate the

rest of the theoretical model it is essential to obtain more accurate tail deflection measurements. Although it is possible to use a correction factor, it is likely that the error is also dependent on the swimming bias. Therefore, for the purposes of this thesis, the measured deflection results from the matlab program will suffice. The use of these estimates will enable a study of the dynamic equations outlined in this thesis. Throughout the rest of this chapter, the measured deflections will be used to directly calculate the swimming thrust components.

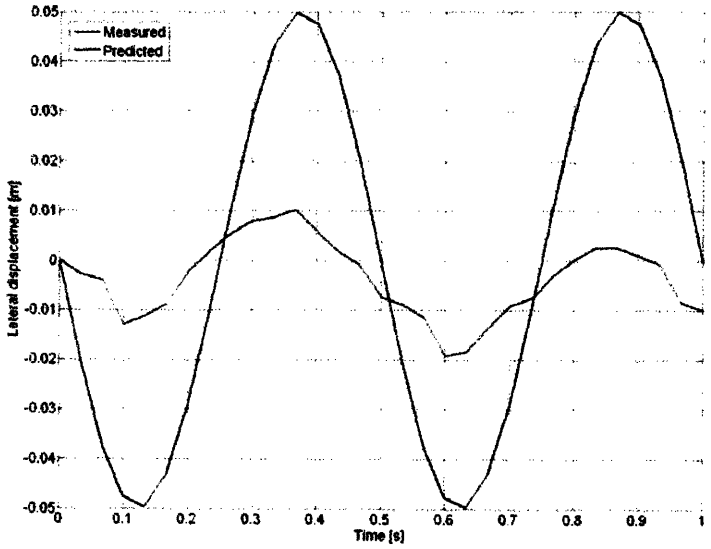

Figure 5.4: A graph showing the predicted and measured tail deflections.

5.5 Assessing the dynamic equations

5.5.1 Translational Dynamics

The first step with regard to assessing the dynamic equations was to study the translational motion from the video data. Since the y direction was the dominant direction of motion throughout all the videos, the y direction was used to assess the steady state translational parameters. Figure 5.5 illustrates the predicted and measured position of the center of mass (y_{cm}), for a trial for a swimming bias $B = 0$. Figure 5.5 reveals that both the transient and steady state responses appear to be incorrect. The slope (at large times) for the predicted data clearly exceeds the corresponding slope for

the experimental data. Similarly, the predicted transient response is far faster than the actual transient response. Based on the dynamic equations, this implies that the thrust calculation, the inertial term ($m + m_{add}$) and / or the drag term (C_{di}) are incorrect. Since the thrust estimates of 0.06 N compare favorably with the measurements taken by Valdivia, it is likely that the inertia and drag terms are the cause of the error. The original dynamic equations only estimated the added fluid mass due to the water. Therefore, it is likely that the translational inertia terms require an increase to correctly account for the added fluid mass. For the purposes of this thesis, the modeling parameters will simply be roughly adjusted so that the predicted transient response matches the measured response. This is accomplished by increasing the translational inertia by a factor of 5 and by increasing the drag term (C_{di}) by a factor of 5.5. Figure 5.5 illustrates that the updated model now matches the experimental data

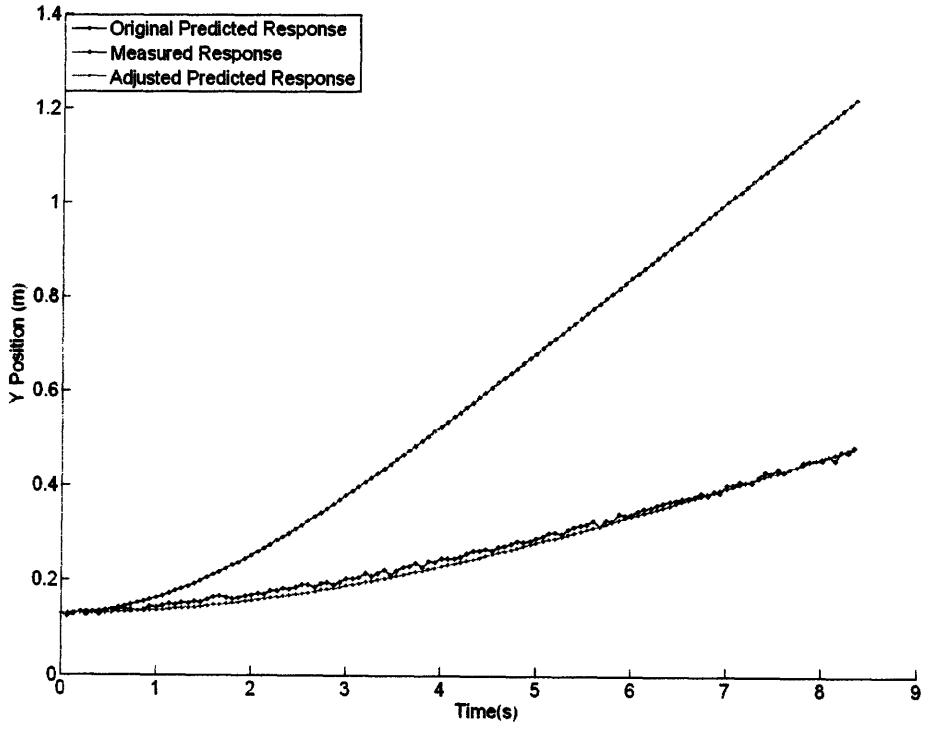

Figure 5.5: A graph illustrating the predicted and measured trajectory of the center of mass in the y direction.

5.5.2 Rotational Dynamics

Once the translational dynamics were assessed and corrected, the next step was to study the rotational dynamics. Figure 5.6 illustrates the predicted and measured orientation (y_{cm}), for a trial for a swimming bias $B = 1.4$. Figure 5.6 reveals that while the predictions for the transient response appear to match the experimental measurements, the steady state responses do not match. From figure 5.6 it can be ascertained that the theoretical model over predicts the steady state angular velocity. This can be caused either by an over estimate of the torque induced by the tail or by an under estimate for the drag (C_{dj}). Since the estimates for the drag coefficients are approximations by their very nature, it is logical to attempt to adjust the model by first adjusting the coefficient of drag. By increasing the coefficient of drag (C_{dj}) by a factor of 3, the discrepancies in the steady state response can be significantly reduced, and figure 5.5 reflects this result. These significant increases in both the translational and rotational drag terms are likely the result of the unsteady fluid dynamics caused by the swimming motions. The drag terms predicted in section 2.4 assumed steady fluid dynamics, and were therefore not completely applicable to fish like swimming motions.

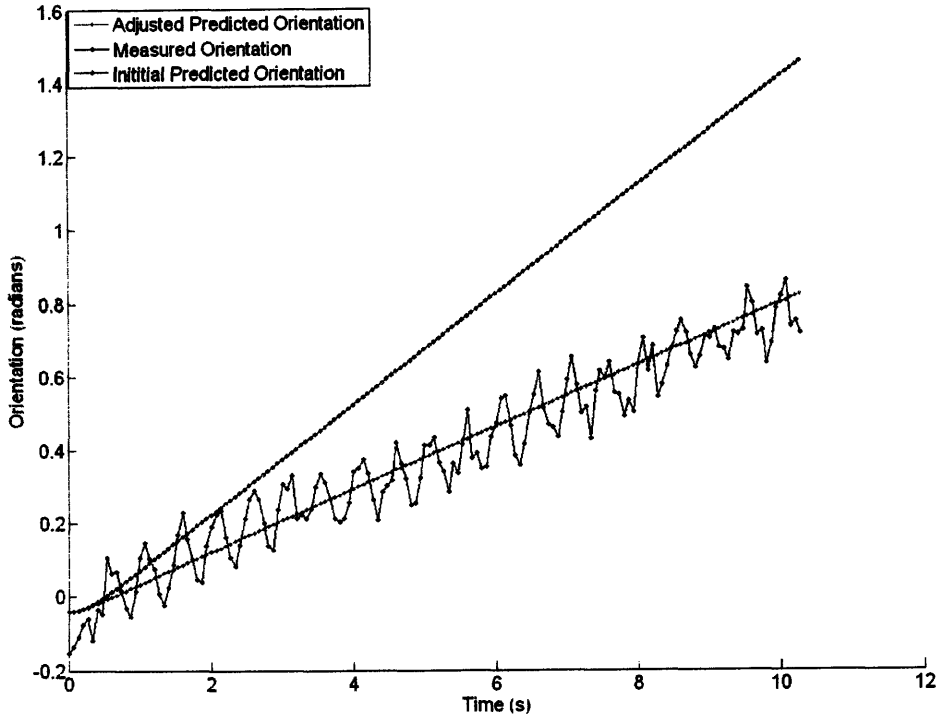

Figure 5.6: A graph illustrating the predicted and measured orientation.

5.6 Updated Model and Experimental Data

The last step was to compare the predictions of the updated model with a range of experimental results. As previously described, experiments were performed with a range of swimming bias signals (*B = 0, 1, 1.2, 1.4).* The data from *B = 1,* and *B= 1.2* are especially relevant since these data sets were not used to recalculate the model parameters. The corresponding tail deflections for each swimming bias are provided in table 5.1. However, it should be emphasized that these measurements are only approximations rather than exact measurements. The updated model parameters and other geometric and material properties with regard to Prototype A are summarized in appendix A.

Bias	H1 (m)	H2 (m)

0	0.01026	0.0130
1	0.00380	0.0143
1.2	0.00338	0.0188
1.4	0.00100	0.0188

Table 5.1: Estimated tail deflections for each experimental trial.

As the figures 5.6 to 5.9 illustrate, the experimental data appears to approximately match the model for all four experiments. In order to quantify the errors, the root mean square of the position error was computed. For these experiments this error ranged from 0.0240m to 0.0887m. When compared to the total distance traveled along the trajectory, this gives errors ranging from 3.5% to 9.1%. Table 5.2 provides a summary of these numerical errors. When assessing these results it is also important to note that the high uncertainties associated with the deflection measurements. These uncertainties would affect the thrust predictions and therefore create the errors that are evident in Figures 5.7 and 5.9. Therefore, it can be concluded that while the theoretical model outlined in this thesis may not be completely accurate, it can serve as a useful too for providing insights and first order estimates with regard to the maneuvering behavior of compliant biomimetic devices.

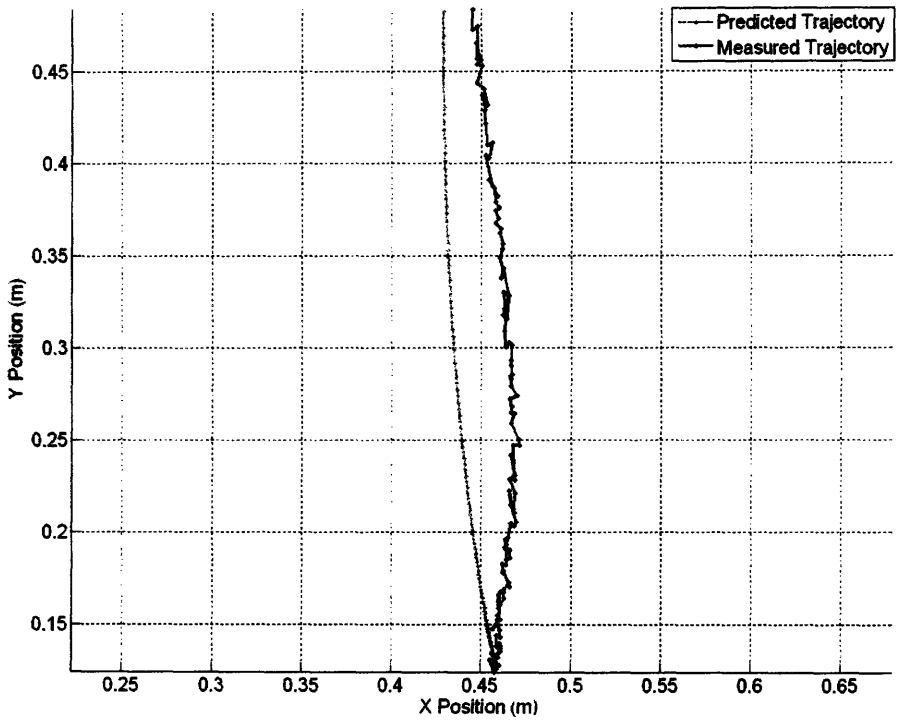

Figure 5.7: A graph of the predicted and measured trajectories for a swimming bias of 0. This result is interesting because the tail deflections imply a turning in one direction while the device actually turns in the other direction. This is likely the result of asymmetries in the swimming device. Even small asymmetries can cause the swimming device to favor a certain direction.

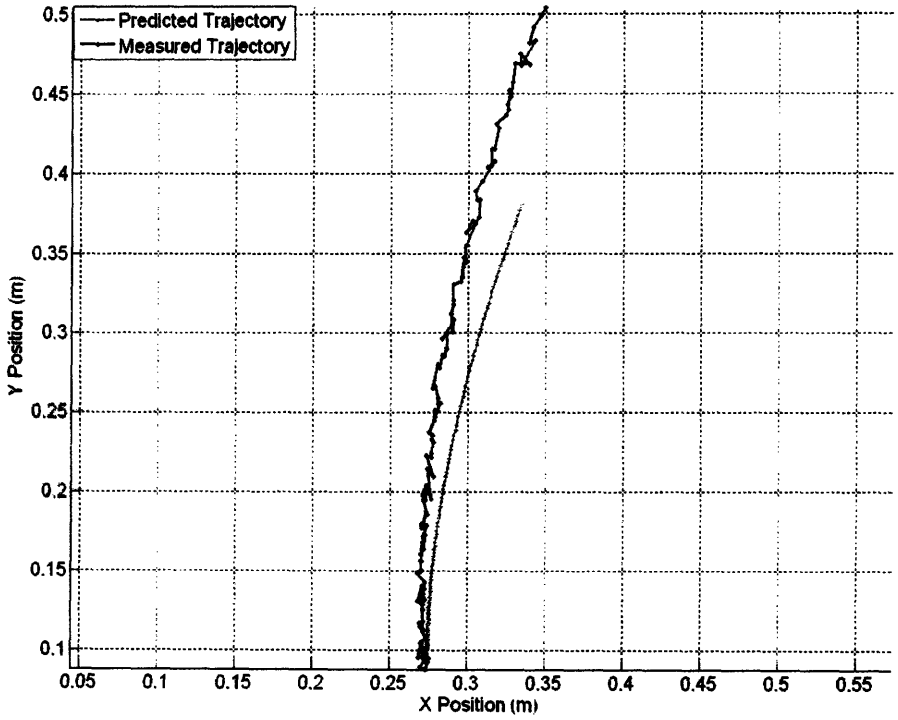

Figure 5.8:A graph of the predicted and measured trajectories for a swimming bias of 1. In this case the model underestimates the forward thrust while overestimating the turning motion. This is most likely the result of errors in the tail deflection measurement.

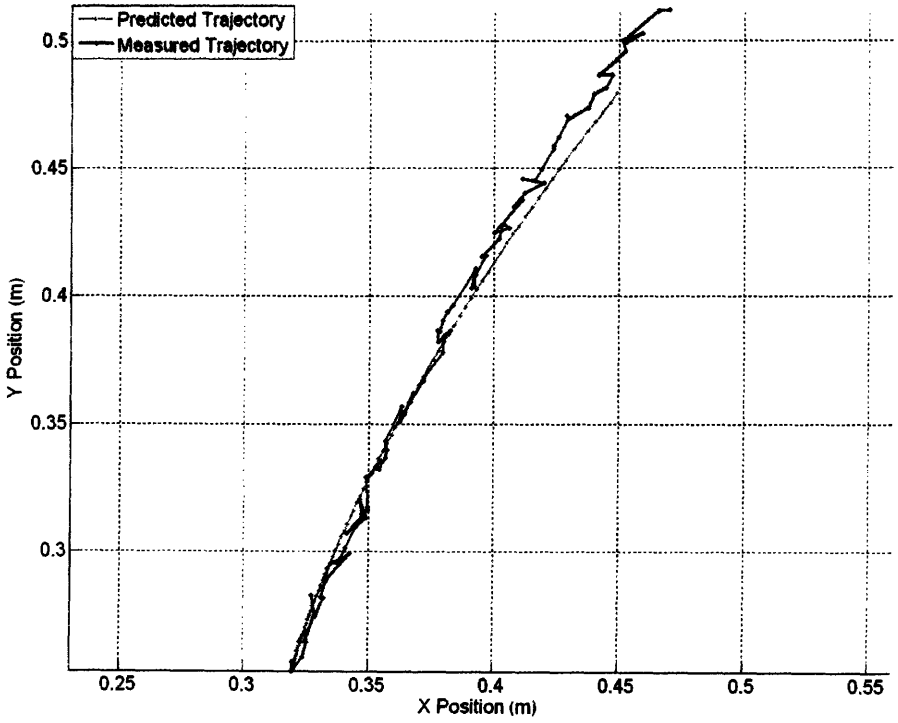

Figure 5.9: A graph of the predicted and measured trajectories for a swimming bias of 1.2. In this case the model appears to predict the experimental behavior quite well.

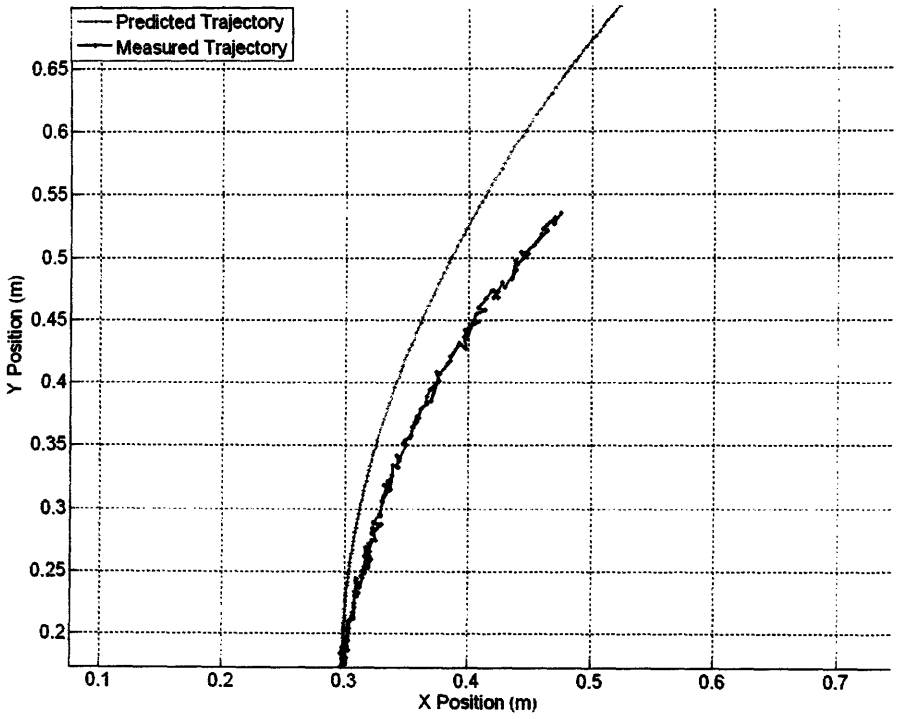

Figure 5.10: A graph of the predicted and measured trajectories for a swimming bias of 1.4. Interestingly this is the exact reverse of the situation illustrated in figure 5.8. In this case the model overestimates the forward thrust and underestimates the turning motion. This again is likely the result of errors in the deflection measurements.

Bias	RMS Error (m)	Percent RMS Error
0	0.0240	3.56
1	0.0634	8.14
1.2	0.0418	7.83
1.4	0.0887	9.12

Table 5.2: Summary of error results for experimental trials.

5.7 Summary

This chapter presented an overview of the experimental setup and procedures for measuring maneuverability. Errors with regard to the tail deflection calculations were discussed and dealt with. In addition, steady state and transient data were used to adjust the appropriate modeling parameters. Finally, the chapter closed with a comparison of the predicted and measured swimming trajectories. It was shown that while the dynamic model outlined in this thesis remains approximate, it can provide first order estimates of maneuvering behavior.

6. Experimental Closed Loop Control Results

6.1 Introduction

With the evaluation of the dynamic model complete, the final step is to use the prototypes described in chapter 4 to carry out experiments relating to closed loop control. In this chapter the experimental setup will be outlined, and the experimental measurements for closed loop control will be compared with the results of the simulation described in chapter 4. Finally the open loop and closed loop experimental results will be compared. The chapter will end with a short discussion of the effectiveness of the control system.

6.2 Experimental Setup

The experimental setup for the closed loop control experiments was very similar to the setup described in chapter 5. The experiments were again performed in the MRL tank, and the experiments were filmed using a Sony DCR-TRV30 NTSC MiniDV digital camera. However, while the maneuverability experiments were carried out using Prototype A, the closed loop measurements were carried out using Prototype B.

The experiments revolved around evaluating the behavior of the open and closed loop system. To study the open loop behavior the device was sent a signal corresponding to a zero swimming bias and allowed to swim freely through the frame of the camera. While the prototype did take readings from the compass at a sampling frequency of 0.25Hz, these readings were not used.

To study the closed loop behavior, the device was sent the appropriate heading, and allowed to swim freely in the tank. Once again the compass was sampled at a frequency of 0.25Hz. However, in this case, the sensor reading was fed back and used to adjust the swimming bias. The controller gains were calculated using equation 6.1 by

determining the minimum angular error that would cause the controller to attempt a fully asymmetric bias ($B = 1$).

$$K_p = \frac{1}{\max(\theta_{err})} \tag{6.1}$$

Since it was evident from the experimental data that turning motions only began at $B \sim 1$, the minimum angular error was chosen to be 10 degrees (0.175 radians). Therefore, for this experiment, a gain of $K_p = 5.729$ was used. Since size and shape of the tank limit the range of headings that can be commanded, experiments were limited to studying a heading parallel to the tank (0°).

6.3 Open Loop Experimental Results

The results of the open loop experiments confirm the need for a closed loop control system. Figure 6.1 illustrates the results of video results of an open loop experiment, while figure 6.2 provides the measurements of the orientation. These figures clearly confirm the need for closed loop control for these devices. Even though the device was sent a swimming signal that was assumed to be symmetric, the device does not swim straight and instead turns approximately 10 degrees of the course of the video.

Figure 6.1: Video data illustrating the open loop trajectory of the device. The device was commanded to swim along the labeled heading. The desired heading is labeled in red while the measured trajectory is labeled in green.

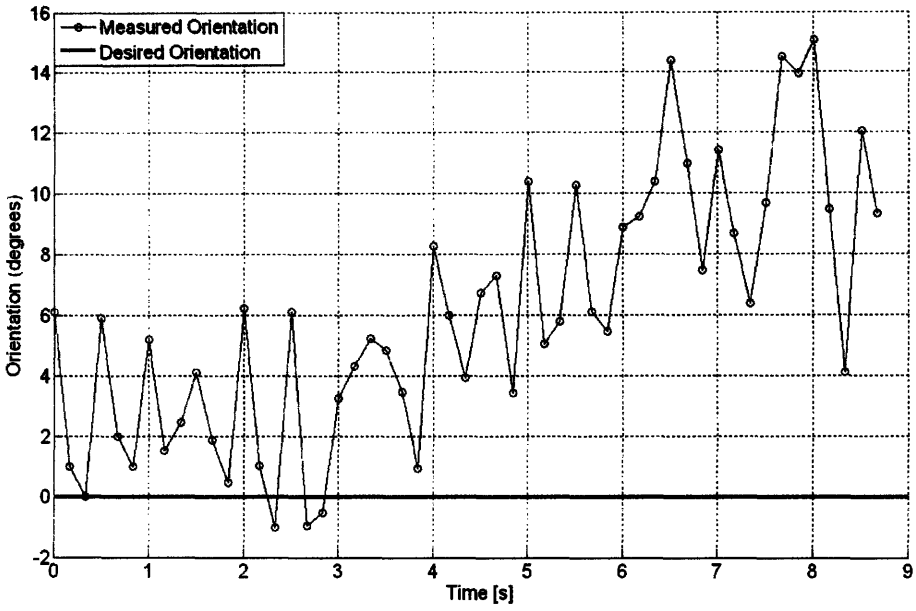

Figure 6.2: Measured open loop orientation of the swimming prototype. This plot illustrates how the orientation of the swimming device does not match the desired orientation and even beings to diverge further from the desired heading.

6.4 Closed Loop Experimental Results

From a visual of the experimental results (figure 6.3), it appears that the use of closed loop control provided much better results. In fact, figure 6.3 illustrates how the device attempts to correct an initial error in its orientation and then attempts to maintain the desired heading. Figure 6.4 provides the measurements of the orientation of the device.

Figure 6.4 reveals how the device attempts to correct the initial error of ~17°, and eventually reduces it to ~5°. While it appears that the error is again increasing towards the end of the graph, this is somewhat misleading due to the fact that the device does not have a chance to correct itself before leaving frame of reference of the video. In fact, the final frames of the video imply that such a correction is taking place.

Figure 6.3: Data illustrating the closed loop trajectory of the device. The device was commanded to swim along the labeled heading. The desired heading is labeled in red while the measured trajectory is measured in green.

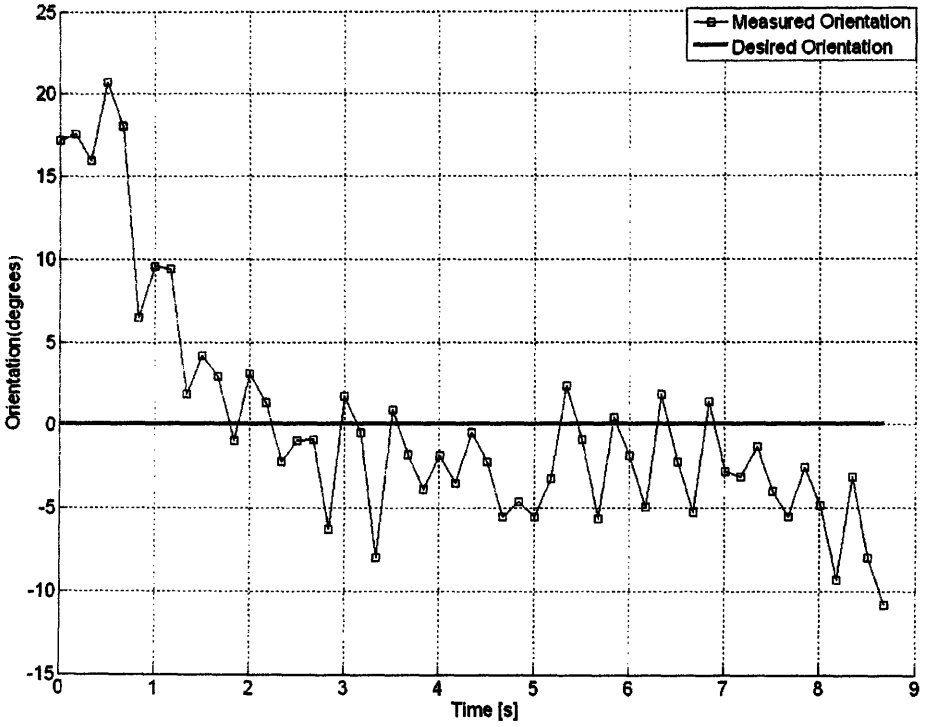

Figure 6.4: Measured closed loop orientation of the swimming prototype. This plot illustrates how the swimming device begins with an error in its orientation and attempts to correct it. From the times 2s to 8s, the device succeeds in correcting its heading and matching the desired orientation. After 8s, the device again beings to diverge (due perhaps to a disturbance) and lacks the space to correct it before leaving the frame of the camera.

Figure 6.5 provides a direct comparison of the open and closed loop orientation results. This figure confirms that the use of closed loop control is far more effective than the open loop approach. While the errors appear to be comparable at t~9s, this should not obscure the fact that the use of closed loop control appears to be working. The significance of this error is difficult to determine due to the fact that the device does not have the chance to recover from this possible disturbance before leaving the frame of the video. More than anything else, this problem illustrates the obvious drawback of sampling the compass at a low frequency. However, it should still be noted that the device maintains a heading that is within 5 degrees of the desired heading for a considerable time (between $t = 2s$ *and* $t = 8s$) before experiencing large errors.

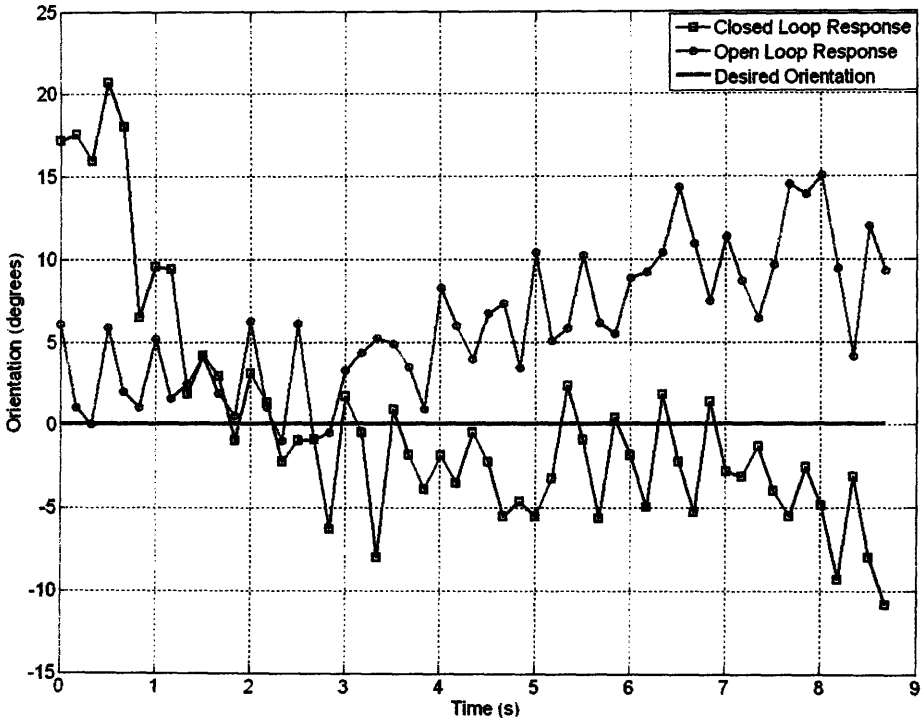

Figure 6.5: A comparison of the open and closed loop orientation measurements. This graph provides a direct comparison of the open loop and closed loop systems. Even though the open loop system begins with a small error, it slowly diverges from the desired orientation and fails to correct itself. In contrast, the closed loop system begins with a much larger initial error, and corrects for it before beginning to diverge towards the end of the camera frame.

6.5 Comparing Experimental Results with Simulation Results

It is also instructive to compare the experimental closed loop results with the results from the closed loop simulation outlined in chapter 4. In order to do this, the simulation was performed with parameters selected to match those of Prototype B. One obvious complication with this approach is that the model for closed loop control requires a relationship between the swimming bias and the tail deflection. While such a relationship exists, it is not always accurate (a fact discussed in detail in chapter 5). In fact chapter 5 revealed that for Prototype A, a correction factor of ~0.2 was needed to make the deflection predictions match the actual results.

In addition, other discrepancies exist with regard to the deflection model. Specifically, the model predicts that a bias signal of 1 would produce completely asymmetric swimming (deflection ranging from 0 to H_{max}). However, experimental data illustrated that this does not occur at a bias of 1 but rather a bias of 1.4. Therefore, while the swimming prototypes were allowed to achieve swimming biases of up to 1.4, the simulation restricted the swimming bias to 1. While these corrections make the Simulink model more accurate, they are also specific changes that can only be applied on a case by case basis.

The results of the adjust simulation are provided in figure 6.6 and figure 6.7, and the figures illustrate that the simulation roughly predicts the actual behavior. While the results for orientation (figure 6.7) are fairly comparable, figure 6.6 reveals that the actual prototype travels significantly less distance over the same amount of time. This discrepancy can be explained by the fact that the simulation did not taken into account the sampling scheme. In other words, the simulation did not predict the degradation in performance that occurs when the servos must be set to their neutral positions in order to read the compass signals. This can also help explain the slower response of the actual control system. The actual system can only sample at 0.25Hz, and therefore travels significant distance before correcting itself.

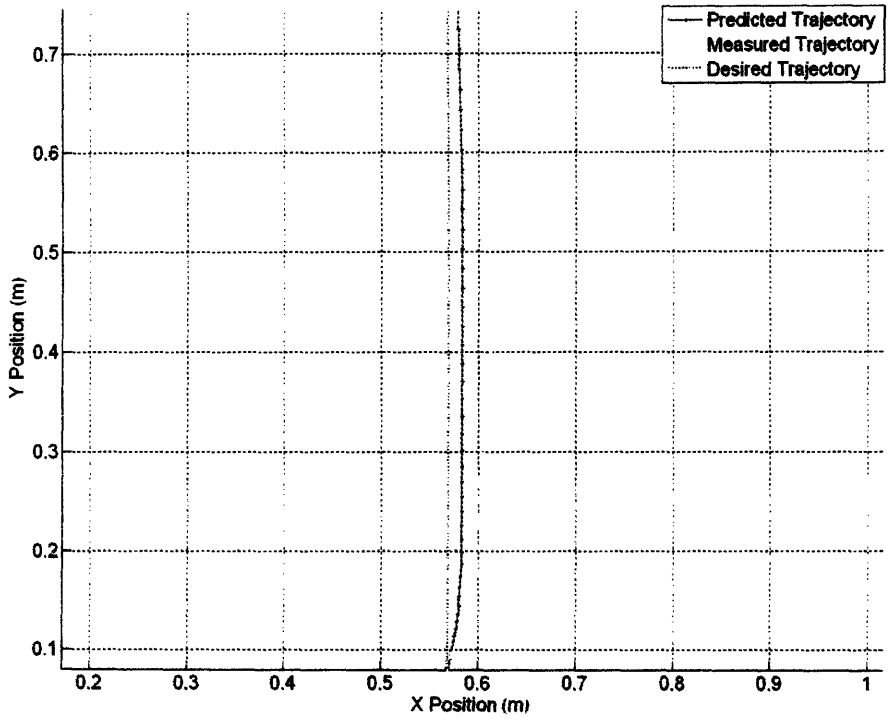

Figure 6.6: A visual comparison of the simulated and experimental trajectories for closed loop heading control. As the figure illustrates, the simulated system corrects itself more quickly and swims at a much greater speed than the actual system. This is due to the fact that the actual system must idle the actuators in order to read the compass signal. This leads to a significant degradation in swimming speed and response time.

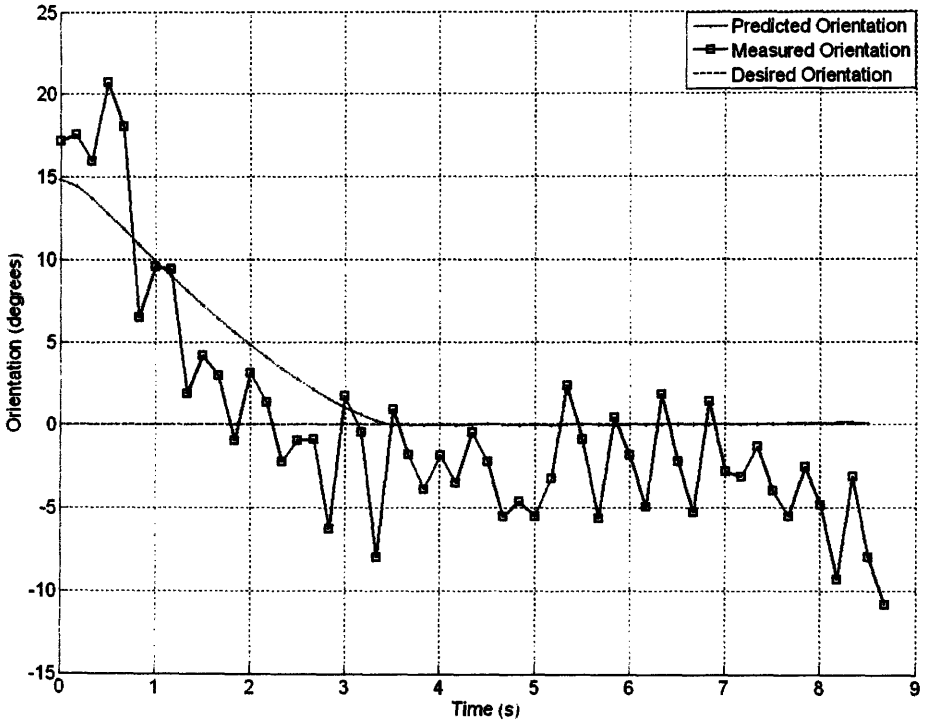

Figure 6.7: A comparison of the simulated and experimental orientation for closed loop heading control.

6.6 Discussion

While the results appear promising, there exist significant caveats. First, the data illustrated in this chapter represents only a small fraction of the experimental trials that were carried out. In many of the trials the swimming device failed to response to control signals and would be unable to properly correct its heading despite swimming at a full bias. This was a direct result of Prototype B's tendency to list and swim along a curved trajectory. This fact was partially illustrated by the open loop result (figure 6.1), but there were cases where the turning was even more pronounced.

It is likely that these problems resulted from two sources. First, due to the presence of the electronics and the compass, the nose of the swimming device was asymmetric in terms of weight distribution. Since even small deviations can affect the swimming behavior, this likely caused the prototype to list and swim in an asymmetric

manner. In addition, the presence of the cables (7 total) likely interfered with the swimming behavior. While great care was taken to reduce this interference, it was impossible to reduce it completely.

Lastly, the compass was never completely removed from magnetic interference. In fact, the compass only provided dynamic reading over certain regions (approximately 90 to -90 degrees in the reference frame of the tank). In addition, the resolution of the compass was reduced. If experiments had taken place in a larger tank, these problems would have been evident. However, the small size of the MRL tank helped mask some of these deficiencies.

However, these problems should not detract from the overall contributions of this chapter. The experimental results outlined in this chapter are certainly promising; closed loop control not only functions properly despite the presence of magnetic interference, but also appears to provide significant improvements over open loop behavior. Finally, the simulations outlined in chapter 4 appear to provide useful predictions of actual system behavior.

6.7 Summary

This chapter presented an overview of the experimental setup and procedures for studying open and closed loop control. The open loop data was compared with the closed loop control data. In addition, closed loop control data was compared with the results of Simulink experiments. Finally, the chapter concluded with a discussion of the results.

7. Conclusion

7.1 Introduction

This chapter summarizes the models and results presented in this thesis. In addition, areas for future research are described in detail. The chapter concludes by discussing the potential applications of this thesis.

7.2 Discussion

This thesis has presented three core contributions. First, a simple model based on Newtonian dynamics for describing the maneuvering behavior of compliant biomimetic swimming devices was outlined. The model made use of the relationships outlined by Dr. Valdivia Y Alvarado [8], and introduced a mathematical method for quantifying the swimming bias.

Second, the model was simulated using Matlab's Simulink, compared to actual experimental data and adjusted accordingly. The updated model was compared with a set of experimental data and shown to provide good first order approximations for the turning behavior of compliant biomimetic prototypes.

Finally, the dynamic model was used to help design and implement a closed-loop heading control system. The control system used a compass to sense absolute orientation and proportional control to dynamically adjust the swimming signal. In addition, this control system incorporated selective sampling so that the magnetic interference caused by the actuators would not affect the sensor readings. Experimental data confirmed that the control system was indeed functioning and controlling the orientation of the swimming device.

7.3 Future Work

While this thesis investigated maneuverability and control of compliant biomimetic swimming devices, significant work remains to be done. First, the experimental results in this thesis exposed a need for a more accurate set of equations for the prediction of the tail deflection. While order of magnitude calculations can be adequate for predicting basic performance characteristics, it is not sufficient for predicting maneuverability. In addition, control systems cannot be properly simulated without a way to predict the tail deflection from the applied moment.

In addition, this thesis has exposed the need for experiments in a larger tank. As described in chapter 6, the full dynamic range of the compass was not even used due to the fact that the tank used for experiments was too small. A larger tank would enable the use of larger prototypes which would immediately solve many of the problems described in this thesis. For example the use of larger prototypes would further reduce magnetic interference between the motors and the compass. Larger prototypes could also contain all the necessary electronics for fully autonomous operation. As a result, tethers would be unnecessary and would no longer interfere with the swimming dynamics. Similarly the use of a larger tank would allow the exploration of the full range of motion of the swimming devices. For example 360° turns could be performed, and the full dynamic range of the control system could be properly assessed.

Lastly, an exciting area of future research is the area of trajectory tracking control systems. Such control systems are essential for navigation and for the performance of complicated missions. With a simple heading control system in place, a part of the trajectory tracking controls problem has been solved. Future research could build on this progress by attempting to control the actual position of the swimming device.

7.4 Applications

The contributions of this thesis are certainly applicable in a variety of ways. First, the dynamic models and simulations that were outlined can serve as useful design and analysis tools. The ability to predict swimming performance and dynamic behavior without always carrying out time consuming experiments is certainly useful. In addition,

these tools will be particularly valuable for the design of larger prototypes which cannot be tested within the laboratory environment.

In addition, the digital control system that was outlined and studied in this thesis serves as a proof of concept for the control of compliant biomimetic swimming devices. This thesis showed that heading control can be achieved with compliant biomimetic swimming devices through the use of a simple proportional control system

Finally, this thesis tackled the difficult control problem of dealing with interference from the actuator. As a result, the control system designed and implemented in this thesis employed an elaborate scheme of sampling the sensor on a periodic basis. The results of chapter 6 illustrate that this approach of periodically idling the actuators and taking sensor readings can be an effective control approach for aquatic devices. Future designs can build on the achievements of this thesis by designing more elaborate compensators and control schemes.

8. Acknowledgements

I would like to thank Professor Youcef-Toumi for his encouragement, advice and guidance over the last year. I would also like to thank Dr. Pablo Valdivia Y Alvarado for introducing me to this exciting project, and taking the time to teach me about mechanical engineering research. Similarly, I would like to acknowledge the members of the Mechatronics Research Laboratory: Dan Burns, Adam Wahab, and Vijay Shilpiekandula for their assistance throughout my tenure at the MRL. In addition, I would like to thank my friends Joshua Jiricek, Christa Margossian, Stephanie Reed, Miguel Saez, and Renee Lizcano for helping me collect all of my experimental data.

Finally, I must express my deep gratitude to my father and mother. Without their constant love and support none of this would have been possible.

Bibliography

[1] Anderson, Jamie M., Chhabra Narender K., "Maneuvering and Stability Performance of a Robotic Tuna", Symposium Stability and Maneuverability presented at the Annual Meeting of the Society for Comparative and Integrative Biology, January 2001, Chicago, Illinois.

[2] Blake, R. W., Chatters L. M., Domenici P., "Turning radius of yellowfin tuna (Thunnus albacares) in unsteady swimming manoeuvres". J. Fish Biol, 1995.

[3] Barrett, David S., The Design of a flexible hull undersea vehicle propelled by an oscillation foil," Master's Thesis, Massachusetts Institute of Technology, 1994.

[4]Liu, Jindong., Hu, Huosheng., "Mimicry of Sharp Turning Behaviours in a Robotic Fish", Proceedings of the International Conference on Robotics and Automation, April 2005, Barcelona, Spain.

[5] Valdivia Y Avarado, Pablo, Youcef-Toumi, Kamal, "Design of Machines with Compliant Bodies for Biomimetic Locomotion in Liquid Environments," Journal of Dynamic Systems, Measurement, and Control, Vol. 128, 2006.

[6] McEwen, Rob., Streitlien, Kurt., "Modeling and Control of a Variable-Length AUV." December 8, 2006.

[7] Colgate, J. Edward., Lynch, Kevin M., "Mechanics and Control of Swimming: A Review," IEEE Journal of Ocean Engineering, Vol. 29, No. 3, July 2004.

[8] Valdivia Y Alvarado, Pablo., "Design of Biomimetic Compliant Devices for

Locomotion in Liquid Environments," PhD Thesis, Massachusetts Institute of Technology, 2007.

[9] Videler, John J., Fish Swimming, New York, Chapman and Hall, 1993.

[10] White, Frank M., Fluid Dynamics, Boston, MA, McGraw Hill, 2003.

[11] Hoerner, Sighard F., Fluid Dynamic Drag: practical information on aerodynamic drag and hydrodynamic resistance, Bakersfield, CA, Hoerner Fluid Dynamics, 1992.

[12] Lighthill, James, Mathematical Biofluiddynamics, Philadelphia, PA, Society for Industrial and Applied Mathematics, 1975.

[13] Dana, Peter H., "Global Positioning System Overview," The Geographer's Craft Project, Department of Geography, University of Colorado at Boulder. Available online at http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

Appendix A

Simulation Parameters

Parameter	Value
Mass (m)	0.34 (kg)
Added mass (m_{add})	1.7 (kg)
Moment of inertia (Izz)	0.00075267 (kg m^2)
Moment of inertial associated with added mass (I_{add})	0.00075267 (kg m^2)
Swimming frequency (f)	2 (Hz)
Coefficient of Drag in i direction (Cd_i)	6.6
Coefficient of Drag in j direction (Cd_j)	13.2
Applied Moment (M)	1.41 (N-m)
Length (L)	0.2667 (m)
Characteristic Length of tail (l_{tail})	0.1052 (m)
Distance from Head to Actuator Plate (a)	0.1486 (m)
Modulus of Elasticity of Tail Material (E)	95650 (Pa)
Viscosity of Tail Material (μ)	90 (Pa-s)
Presented Area in i direction	0.0053 (m^2)
Presented Area in j direction	0.0115 (m^2)
Presented Nose Area in j direction (A_{p1})	0.0053 (m^2)
Presented Tail Area in j direction (A_{p2})	0.0062 (m^2)

Appendix B

List of Vendors

B.1 Digital Compass

Summerour Robotics Corp (www.RoboticsConnection.com)

B.2 Silicon

Quantum Silicones, LLC. (www.quantumsilicones.com): Q300 Silicone gels.

Smooth-On (www.smooth-on.com): EcoFlex Silicone rubbers.

B.3 RC Components

Tower Hobbies (www.towerhobbies.com)

B.4 Microcontroller Components

New Micros Incorporated (www.newmicros.com)

Appendix C

Isomax Closed Loop Control Software

```
(       A simple program to control and acquire data from a robotic fish.  Code
(       written by Ani Mazumdar with Pablo Valdivia


( This program is a simple program that allows a user to test the actual
( swimming performance of a Fish
( This program performs closed loop control using a Devantech CMPS03 compass
( Note, the compass is calibrated for truncated dynamic range and for the specific conditions of
the MRL tank

( Procedure for Use
( 1.    Flash program onto memory
( 2.    Type 'MAIN' into terminal window
( 3.    Control the fish as desired using keyboard
( 4.  To break out of the MAIN function, press the reset button on the board
(       or toggle the power

(       Control commands
(       's' = stop
(       'w' = forward
(       'a' = bias left
(       'd' = bias right
(       '+' = reduce period
(       '-' = increase period
(       'u' = increase low dutycycle value
(       'U' = decrease low dutycycle value
(       'h' = decrease high dutycycle value
(       'H' = increase high dutycycle value
(       'l' = decrease left fin dutyclcle value
(       'L' = increase left fin dutycycle value
(       'r' = decrease right fin dutycycle value
(       'R' = increase right fin dutycycle value
(    'A' = hard turn left
(       'D' = hard turn right
(       'X' = Exit main loop

SCRUB

COLD

DECIMAL

3.30e FCONSTANT VREF EEWORD
1.0e FCONSTANT TREF EEWORD
0.625e FCONSTANT CLOCKCONV EEWORD
```

```
10250.0e FCONSTANT OFFSET EEWORD

10.0e FCONSTANT ERRANGE EEWORD      ( RANGE FOR ERROR
1.0e FCONSTANT THRESHRANGE EEWORD ( RANGE FOR BIAS

( ACTUAL VALUES
( 5500 CONSTANT MINPOS EEWORD  ( Values set for the HiTech Digital Servo
( 9500 CONSTANT MAXPOS EEWORD

( SAFETY VALUES
6500 CONSTANT MINPOS EEWORD
10000 CONSTANT MAXPOS EEWORD

4300 CONSTANT MINPOSL EEWORD ( Left is associated with PWM1
12200 CONSTANT MAXPOSL EEWORD

4300 CONSTANT MINPOSR EEWORD ( Right is associated with PWM2
12200 CONSTANT MAXPOSR EEWORD

1000 CONSTANT RCSCALE# EEWORD


MAXPOS MINPOS + 2/ CONSTANT MIDPOS EEWORD
MAXPOS MINPOS - CONSTANT RCRANGE EEWORD


MAXPOSL MINPOSL + 2/ CONSTANT MIDPOSL EEWORD
MAXPOSL MINPOSL - CONSTANT RCRANGEL EEWORD

MAXPOSR MINPOSR + 2/ CONSTANT MIDPOSR EEWORD
MAXPOSR MINPOSR - CONSTANT RCRANGER EEWORD




VARIABLE CMDCHAR        EEWORD
VARIABLE DUTYH EEWORD
VARIABLE DUTYL EEWORD
VARIABLE DUTYHPRE EEWORD
VARIABLE DUTYLPRE EEWORD

VARIABLE DUTYFINL EEWORD
VARIABLE DUTYFINR EEWORD

VARIABLE TEMP EEWORD
VARIABLE TEMP2 EEWORD
VARIABLE TEMPL EEWORD
VARIABLE TEMPR EEWORD

FVARIABLE T1 EEWORD
```

```
FVARIABLE T2 EEWORD


VARIABLE TRCKTIME EEWORD

VARIABLE TIMEUP EEWORD

VARIABLE STATEL EEWORD
VARIABLE STATEH EEWORD

VARIABLE TIMEI EEWORD
VARIABLE TIMEC EEWORD
VARIABLE TIMEDIFF EEWORD
VARIABLE TIMEDIFF3 EEWORD
0 CONSTANT NULLCHAR EEWORD

DECIMAL 10 CONSTANT 1-SECOND  EEWORD
VARIABLE INTERVAL EEWORD  ( WAIT ASSOCIATED WITH SWIMMING
VARIABLE INTERVAL2 EEWORD ( WAIT ASSOCIATED WITH TURNING
VARIABLE INTERVAL3 EEWORD ( WAIT ASSOCIATED WITH ACQUIRING DATA
VARIABLE ENDPROG EEWORD

VARIABLE BOOLTURN EEWORD

VARIABLE TIMEDIFF2 EEWORD
VARIABLE PWMTIMEX EEWORD
VARIABLE PWMTIMEY EEWORD
VARIABLE BOOLDISP EEWORD
VARIABLE BOOLTEMP EEWORD
VARIABLE BOOLSTOP EEWORD
VARIABLE BOOLCONTRL EEWORD
VARIABLE TIMEPRE EEWORD

VARIABLE BOOLPWM EEWORD
VARIABLE DATAIN EEWORD ( THIS VARIABLE SHOULD BE UNSIGNED
VARIABLE DATAIN2 EEWORD


VARIABLE BOOLREADPWM EEWORD
VARIABLE TIMEPREPWM EEWORD
VARIABLE INTERVALPWM EEWORD
VARIABLE TIMEDIFFPWM EEWORD

VARIABLE CYCLECOUNT EEWORD
VARIABLE INTERVALCHK EEWORD

FVARIABLE OUTPUTL EEWORD
FVARIABLE OUTPUTH EEWORD
FVARIABLE CYCLE EEWORD
FVARIABLE HEADINGDES EEWORD
FVARIABLE DATAIN3 EEWORD
```

```
FVARIABLE COMPVAL1 EEWORD
FVARIABLE COMPVAL2 EEWORD
FVARIABLE HEADERROR EEWORD
FVARIABLE SIGNALL EEWORD
FVARIABLE SIGNALH EEWORD
FVARIABLE KP EEWORD
FVARIABLE ACTSIGNAL EEWORD
FVARIABLE BIAS EEWORD

: SETUP-TD1
TD1 SET-PWM-IN
[ HEX ] D6E @ 0400 OR D6E ! [ DECIMAL ]
; EEWORD



: INIT
        32767 PWMA0 PWM-PERIOD
        MIDPOS PWMA0 PWM-OUT

        32767 PWMA4 PWM-PERIOD
        MIDPOSL PWMA4 PWM-OUT

        32767 PWMA5 PWM-PERIOD
        MIDPOSR PWMA5 PWM-OUT

        0.0e T1 F!

        10.0e HEADINGDES F!


        0.0e COMPVAL1 F!
        0.0e COMPVAL2 F!
        0.0e HEADERROR F!
        0.0e SIGNALL F!
        1000.0e SIGNALH F!
        0.0e ACTSIGNAL F!
        0.0e OUTPUTL F!
        0.0e OUTPUTH F!
        0.0e BIAS F!

        THRESHRANGE ERRANGE F/ KP F!

        500 DUTYFINR C!
        500 DUTYFINL C!
        0 BOOLREADPWM C!
        1 TIMEUP C!
        1 STATEL C!
        0 STATEH C!
        0 TIMEI !
        0 TIMEC !
```

```
        0 TIMEDIFF !
        900 DUTYH !
        100 DUTYL ! ( SHOULD BE 100
        15 TEMP C!
        0 TEMP2 C!
        25 INTERVAL C!
        500 INTERVAL2 !
        3 INTERVAL3 C!
        0 ENDPROG C!
        0 TIMEDIFF2 !
        0 BOOLTURN !
        1 PWMTIMEX !
        1 PWMTIMEY !
        1 BOOLDISP !
        1 BOOLTEMP !
        0 TIMEPRE !
        0 TIMEDIFF3 !
        0 BOOLPWM !
        0.0 CYCLE F!
        0 DATAIN C!
        0 DATAIN2 C!
        0 BOOLSTOP C!
        0 BOOLCONTRL C!

        0 BOOLREADPWM C!
        100 INTERVALPWM !
        0 TIMEPREPWM !
        0 TIMEDIFFPWM !
        100 DUTYLPRE !
        900 DUTYHPRE !
        0 CYCLECOUNT !
        50 INTERVALCHK !

        ISOMAX-START
; EEWORD


: DECIVAL S>F VREF FSWAP 32760.0e F/ F* ;  EEWORD



: GET-AD
  ADC0 ANALOGIN DECIVAL T1 F!
; EEWORD


: WAIT-TIME ( -- )

        TCFTICKS @
        BEGIN
                TCFTICKS @ OVER -
```

```
        INTERVALCHK C@ 1 - > UNTIL
        DROP
; EEWORD

: RCOUT0 ( COUNT FROM 0 TO RCSCALE#
        0 MAX RCSCALE# MIN
        RCRANGE RCSCALE# */
        MINPOS +
        PWMA0 PWM-OUT
;       EEWORD

: RCOUTL ( COUNT FROM 0 TO RCSCALE#
        0 MAX RCSCALE# MIN
        RCRANGEL RCSCALE# */
        MINPOSL +
        PWMA5 PWM-OUT
;       EEWORD

: RCOUTR ( COUNT FROM 0 TO RCSCALE#
        0 MAX RCSCALE# MIN
        RCRANGER RCSCALE# */
        MINPOSR +
        PWMA4 PWM-OUT
;       EEWORD

: CYCLESERVO

        DUTYH @ RCOUT0
        YELLED ON
        WAIT-TIME

        DUTYL @ RCOUT0
        YELLED OFF
        WAIT-TIME
;       EEWORD

: SERVOH
        ( THRESHOLD



        DUTYH @ RCOUT0
        YELLED OFF
        GRNLED OFF
; EEWORD

: SERVOL
        ( THRESHOLD
```

```
        DUTYL @ RCOUT0
        YELLED ON
        GRNLED OFF
; EEWORD

: TURNLEFT
        1 BOOLTURN !
        TCFTICKS @ TIMEI !
        0 TIMEUP !
        1000 RCOUT0
        0 DUTYFINL C!
        500 DUTYFINR C!
        GRNLED ON
; EEWORD

: BIASLEFT
        0 DUTYL !
        500 DUTYH !
        900 DUTYFINL C!
        100 DUTYFINR C!
; EEWORD

: BIASRIGHT
        500 DUTYL !
        1000 DUTYH !
        100 DUTYFINL C!
        900 DUTYFINR C!
; EEWORD

: SWIMSTRAIGHT
        0 DUTYL !
        1000 DUTYH !
        500 DUTYFINR C!
        500 DUTYFINL C!
        0 BOOLTURN C!
; EEWORD

: FISHSTOP
        REDLED ON
        500 DUTYH !
        500 DUTYL !
        500 DUTYFINL C!
        500 DUTYFINR C!
        0 BOOLTURN C!
        0 BOOLCONTRL C!
; EEWORD

: TURNRIGHT
        1 BOOLTURN C!
        TCFTICKS @ TIMEI !
        0 TIMEUP !
```

```
        0 RCOUT0
        500 DUTYFINL C!
        1000 DUTYFINR C!
        GRNLED ON
;       EEWORD

: BRAKE
        500 DUTYH !
        500 DUTYL !
        0 DUTYFINL C!
        1000 DUTYFINR C!
; EEWORD

: DIVE
        500 RCOUT0
        1000 RCOUTL
        0 RCOUTR
; EEWORD

: SETACTUATOR
        COMPVAL1 F@ -0.180e F* COMPVAL2 F!
        HEADINGDES F@ COMPVAL2 F@ F- HEADERROR F!

        180.0e HEADERROR F@ F< IF
                HEADERROR F@ 360.0e F- HEADERROR F!
        THEN

        HEADERROR F@ -180.0e F< IF
                HEADERROR F@ 360.0e F+ HEADERROR F!
        THEN

        ( CALCULATE OUTPUT SIGNAL
        HEADERROR F@ KP F@ F* BIAS F!

        BIAS F@ 500.0e F* ACTSIGNAL F!

        ACTSIGNAL F@ 0.0e F< IF
        ( ACTUATOR SIGNAL IS LESS THAN 0, RIGHT

                1000.0e ACTSIGNAL F@ F+ SIGNALH F!
                0.0e SIGNALL F!
        ELSE
        ( ACTUATOR SIGNAL IS GREATER THAN 0, LEFT
                0.0e ACTSIGNAL F@ F+ SIGNALL F!
                1000.0e SIGNALH F!
        THEN
; EEWORD

: SETVALS
        BOOLCONTRL C@ 0 = IF
                ( We are not using closed loop control
```

```
                DUTYFINL C@ RCOUTL
                DUTYFINR C@ RCOUTR
        ELSE
                500  RCOUTL
                500  RCOUTR

                SIGNALL F@ OUTPUTL F!
                SIGNALH F@ OUTPUTH F!

                SIGNALL F@ 0.0e F< IF
                        000.0e OUTPUTL F!
                THEN

                SIGNALH F@ 0.0e F< IF
                        000.0e OUTPUTH F!
                THEN

                1000.0e SIGNALL F@ F< IF
                        1000.0e OUTPUTL F!
                THEN

                1000.0e SIGNALH F@ F< IF
                        1000.0e OUTPUTH F!
                THEN


                OUTPUTL F@ F>D DROP DUTYL !
                OUTPUTH F@ F>D DROP DUTYH !

                DUTYL @ 700 >  IF
                        700 DUTYL !
                THEN

                DUTYH @ 300 <  IF
                        300 DUTYH !
                THEN

        THEN
; EEWORD


: PRINTDATA
        TCFTICKS @ 10 * . ( Print clock in milliseconds
        INTERVAL C@ .
        DUTYL @ .
        DUTYH @ .
        DUTYFINL C@ .
        DUTYFINR C@ .
        CYCLE F@ F.
        HEADINGDES F@ F.
        ( COMPVAL1 F@ F.
```

```
        COMPVAL2 F@ F.
        HEADERROR F@ F.
        BIAS F@ F.
        ACTSIGNAL F@ F.
        SIGNALL F@ F.
        SIGNALH F@ F.
        BOOLPWM @ .

        CR
; EEWORD

: ENDPROGRAM
        500 DUTYFINL C!
        500 DUTYFINR C!
        500 DUTYL !
        500 DUTYH !
        500 RCOUT0
        500 RCOUTL
        500 RCOUTR
; EEWORD

DECIMAL
: MAIN
        INIT

        BEGIN
                BOOLTURN C@ 0 = IF ( Check if a hard turn has been commanded
                        TIMEUP C@ 1 = IF ( Check timer
                                TCFTICKS @ TIMEI ! ( Reinitialize the Time Variables

                                ( COUNTCYCLE @ 1 + COUNTCYCLE !

                                0 TIMEUP !

                                STATEL C@ 1 = IF ( Check State
                                                SERVOL
                                CR
                                THEN

                                STATEL C@ 0 = IF ( Check State
                                                SERVOH
                                THEN

                        THEN
                THEN

                TCFTICKS @ TIMEC !
                TIMEC @ TIMEI @ - TIMEDIFF !
                TIMEDIFF @ TIMEDIFF2 !

                BOOLTURN C@ 0 = IF ( Check if a hard turn has been commanded
```

```
                TIMEDIFF2 @ INTERVAL @ 1- > IF
                        1 TIMEUP C!
                        STATEL C@ 1 = IF ( Check State
                                                0 STATEL C!
                                        ELSE
                                                1 STATEL C!
                                        THEN
                                THEN
        ELSE
                TIMEDIFF2 @ INTERVAL2 @ 1- > IF
                        0 BOOLTURN C!
                        1 TIMEUP C!
                        500 DUTYFINL C!
                        500 DUTYFINR C!
                THEN
        THEN


TCFTICKS @ TIMEPRE @ - TIMEDIFF3 !
TCFTICKS @ TIMEPREPWM @ - TIMEDIFFPWM !

TIMEDIFFPWM @ 400 < IF ( Sampling Frequency
        ( DO NOT DO ANYTHING
        DUTYL @ DUTYLPRE !
        DUTYH @ DUTYHPRE !
ELSE
        1 BOOLREADPWM !
        500 RCOUT0

THEN


BOOLREADPWM @ 1 = IF


        500 DUTYL ! ( SET TAIL TO EQUILIBRIUM POSITION
        500 DUTYH !
        500 RCOUT0

        REDLED ON

        BOOLPWM @ 0 = IF
                        1 BOOLPWM !
                        WAIT-TIME
                        SETUP-TD1
                        TD1 CHK-PWM-IN DATAIN C!
        ELSE
                TD1 CHK-PWM-IN DATAIN C!

                DATAIN C@ 0 > IF
                        0 BOOLPWM !
```

```
                        0 BOOLREADPWM !
                        DUTYLPRE @ DUTYL !
                                DUTYHPRE @ DUTYH !

                                REDLED OFF
                        TCFTICKS @ TIMEPREPWM ! ( UPDATE TIMING
VARIABLES

                        DATAIN C@ DATAIN2 C!
                        DATAIN2 C@ S>F DATAIN3 F!
                        DATAIN3 F@ CLOCKCONV F/ CYCLE F! ( microsecs
                        CYCLE F@ OFFSET F- COMPVAL1 F! ( Re center
                        SETACTUATOR



                THEN
        THEN
THEN



TIMEDIFF3 @ 5 < IF ( Sampling Frequency
        0 BOOLTEMP !
ELSE
        1 BOOLTEMP !
THEN

        ?KEY IF
                KEY CMDCHAR C!
                        ( GRNLED ON
                CR

                        CMDCHAR C@ 115 = IF ( Character 's', Fish stops
                                FISHSTOP
                        ELSE
                                REDLED OFF

                        THEN


                        CMDCHAR C@ 100 = IF  ( Character 'd', Right Bias
                                BIASRIGHT

                        THEN

                        CMDCHAR C@ 97 = IF ( Character 'a', Left Bias
                                BIASLEFT
                        THEN
```

```
                    CMDCHAR C@ 119 = IF ( Character 'w', Fish moves straight
                        SWIMSTRAIGHT
                    THEN


                    CMDCHAR C@ 43 = IF ( Character '+', Reduce Period
                        INTERVAL C@ TEMP C!
                        TEMP C@ 1 - INTERVAL C!

                    THEN

                    CMDCHAR C@ 45 = IF ( Character '-', Increase Period
                        INTERVAL C@ TEMP C!
                        TEMP C@ 1 + INTERVAL C!
                    THEN

                    CMDCHAR C@ 99 = IF ( Character 'c' BEGIN CONTROL
                        1 BOOLCONTRL C!
                    THEN

(                   CMDCHAR C@ 65 = IF ( Character 'A' HARD TURN LEFT
(                       TURNLEFT
(                   THEN

(                   CMDCHAR C@ 68 = IF ( Character 'D' HARD TURN RIGHT
(                       TURNRIGHT
(                   THEN

(                   CMDCHAR C@ 98 = IF ( Character 'b' brake
(                       BRAKE
(                   THEN

(                   CMDCHAR C@ 66 = IF ( Character 'B' dive
(                       1000 DUTYFINL C!
(                       0 DUTYFINR C!
(                   THEN

(                   CMDCHAR C@ 67 = IF ( Character 'C' climb
(                       300 DUTYFINL C!
(                       700 DUTYFINR C!
(                   THEN

                    CMDCHAR C@ 88 = IF ( Character 'X' END PROGRAM
                        ENDPROGRAM
                        1 ENDPROG C!
                    THEN
        ELSE

        NULLCHAR CMDCHAR C!
        ( GRNLED OFF
THEN
```

```
        SETVALS ( SEND CORRECT SIGNALS TO ACTUATORS

        BOOLTEMP @ 1 = IF ( READ AND DISPLAY
                TCFTICKS @ TIMEPRE !
                PRINTDATA

        THEN
        ENDPROG C@ 1 = UNTIL


; EEWORD
SAVE-RAM
```

Appendix D

Matlab Video Processing Code

```
% Program to determine the trajectory and velocity of a prototype by
% analyzing progressive frames
% Data is also stored in a text file for future analysis
% writen by Ani Mazumdar and Pablo Valdivia Y Alvarado
close all
clear all

file1 = 'BigSalNew_Control_0'; % INPUT VIDEO
file2 = 'ClosedLoop_0_2.txt'; % OUTPUT TEXT FILE

digCheck = 0;
tempDig = 0;
trajMap = zeros(1,1);
boolTraj = 0;
F_width = -1;
scaleFactor = 1;
%F_width_known = 0.0254; For small fish design
F_width_known = 2 * 2.54 / 100; %Width of the fattest part of the fis
(in meters)


boolVel = 1;
%GET INFO

%FISH 1
% For swim and turn I = ? F = ?


% FISH 2

%For Hard turn, I = 122, F = 190
%For Hard Turn 2 , I = 120, F = 245
%For Start Swim I = 70, F = 205
%For Stop I = 165, F = 310
%For 2.3 Coast I = 90, F = 270
%For biasturnright I = 60, F = 210

frameInitial =120;
fileInfo = aviinfo(file1);
numFrames = fileInfo.NumFrames;
frameFinal =415;
interval = 4;



if(frameFinal > numFrames)
    disp('Final Frame is too high, will use all the Frames');
    frameFinal = numFrames;
```

```
end

frameRate = fileInfo.FramesPerSecond;
timeArray = [(frameInitial - 1):interval:(frameFinal-1)] * 1/frameRate;

%if(size(timeArray,2) > numFrames || size(timeArray,2) <numFrames)
%    disp('TIME ARRAY IS INVALID, VELOCITY WILL NOT BE CALCULATED');
%    boolVel = 0;
%end
counter =1;
count = 1;
for(q= frameInitial:interval:frameFinal)



    disp(q);
    mov = aviread(file1,q);
    [X,Map] = frame2im(mov);
    J1 = rgb2gray(X);
    BW1 = J1;
    thresh=0.05;
    BW1=edge(J1,'log');
    if(count == 1)
        trajMap = zeros(size(J1,2),size(J1,1));
        imTemp = zeros(size(J1));
        imPre = BW1;
    end

    figure(1);
    imshow(~BW1);
    axis equal;
    boolSelect = 1;

    while(boolSelect)
        disp('Create a Line Across the fattest point of the Fish,
double click on the second point');
        [BWtemp, x_s, y_s] = roipoly;
        if(size(x_s,1) < 3 || size(x_s,1) > 3)
            disp('You did not select exactly 2 points, please try
again');
            boolSelect = 1;
        else
            boolSelect = 0;
        end
    end
    x_1 = x_s(1);
    x_2 = x_s(2);
    y_1 = y_s(1);
    y_2 = y_s(2);
    boolSelect = 1

    while(boolSelect)
        disp('Double Click on the Head');
        [BWtemp, x_s, y_s] = roipoly;
        if(size(x_s,1) < 2 || size(x_s,1) > 2)
```

```
            disp('You did not select exactly 1 point, please try
again');
            boolSelect = 1;
        else
            boolSelect = 0;
        end
    end
    x_head = x_s(1);
    y_head = y_s(1);
    boolSelect = 1;

    while(boolSelect)
        disp('Double Click on the Tail');
        [BWtemp, x_s, y_s] = roipoly;
        if(size(x_s,1) < 2 || size(x_s,1) > 2)
            disp('You did not select exactly 1 point, please try
again');
            boolSelect = 1;
        else
            boolSelect = 0;
        end
    end
    x_tail = x_s(1);
    y_tail = y_s(1);
    %Calculate Pixel Width across Fish
    if(count == 1)
        F_width = sqrt( (x_2- x_1)^2 + (y_2 - y_1)^2);
        scaleFactor = F_width / F_width_known; % PIXELS PER METER
    end
    %Calculate Midpoint of line segment
    x_m = (abs(x_2+x_1)/2);
    y_m = (abs(y_2+y_1)/2);
    %trajMap(x_m, y_m) = 1;
    positions(1,count) = x_m;
    positions(2,count) = y_m;

    positionsB1(1, counter) = x_tail;
    positionsB1(2, counter) = y_tail;
    positionsB1(1, counter+1) = x_m;
    positionsB1(2, counter+1) = y_m;
    positionsB2(1, counter) = x_m;
    positionsB2(2, counter) = y_m;
    positionsB2(1, counter+1) = x_head;
    positionsB2(2, counter+ 1) = y_head;

    headPos(1, count) = x_head;
    headPos(2, count) = y_head;
    tailPos(1, count) = x_tail;
    tailPos(2, count) = y_tail;

    boolTraj = 1;
    if(count >= 2)
        K=imlincomb(1,double(BW1),1,double(imPre));
        imPre = K;
    end
    counter= counter +2;
```

```
    count = count + 1;

end
if(boolTraj ==1)
    figure(2), imshow(K), hold on


    plot(positions(1,:),positions(2,:),'LineWidth',2,'Color','green');

    % Plot beginnings and ends of lines
    %plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
    %plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

    % Determine the endpoints of the longest line segment

end
totalDist = calcDist(positions(1,:), positions(2,:));


%NOW DO THE KINEMATICS
%First step, fit the trajectory data to a 3rd order polynomial
positionsM = double(positions);
positionsH = double(headPos);
positionsT = double(tailPos);


p_M = polyfit(positionsM(2,:), positionsM(1,:), 1); %FIT x in terms of
Y
p_H = polyfit(positionsH(2,:), positionsH(1,:), 1);
p_T = polyfit(positionsT(2,:), positionsT(1,:), 1);


a_M = p_M(1);
b_M = p_M(2);
a_H = p_H(1);
b_H = p_H(2);
a_T = p_T(1);
b_T = p_T(2);


%y_fit_Int = int16(y_fit);
%y_predict_head = a*headPos(1,:).^2 +b * headPos(1,:) + c;
%figure(5)
%hold on;
%imshow(K);
%plot(x_fit,y_fit_Int,'LineWidth',2,'Color','Red');
plot(positions(1,:), positions(2,:), '.', 'Color', 'g');
hold on;
%plot(x_fit, y_fit,'LineWidth', 4, 'Color', 'b');
plot(headPos(1,:), headPos(2,:), '.', 'Color', 'b');
plot(tailPos(1,:), tailPos(2,:), '.', 'Color', 'r');
%plot(headPos(1,:), y_predict_head, 'o', 'Color', 'b');
hold off;
```

```
%NOW WE HAVE THE TAIL AND HEAD

%CALCULATIONS FOR THE HEAD
mulNeg = 1;
counter = 1;
headDist = zeros(1,size(headPos,2));
for(i = 1: size(headPos,2))

    % Step 1 find x_t, y_t
    x_t = headPos(1,i);
    y_t = headPos(2,i);

    % Step 2 find m_p
    m = 1/a_H;

    m_p = -1/m;

    a = 1/ a_H;

    b = - b_H / a_H;

    %Step 3 solve for x_c
    polytemp = [a - m_p, b - y_t + m_p * x_t];
    x_c = roots(polytemp);

    %Step 3a Check Sign
    if(x_c > x_t)
        mulNeg = 1;
    else
        mulNeg = -1;
    end
    %Step 4 Solve for y_c
    y_c = y_t + m_p*x_c - m_p * x_t;

    headDist(counter) = mulNeg*sqrt( (x_c - x_t)^2 + (y_c - y_t)^2 );
    counter = counter + 1;

end

%CALCULATIONS FOR THE TAIL

mulNeg = 1;
counter = 1;
tailDist = zeros(1,size(headPos,2));
for(i = 1: size(headPos,2))

    % Step 1 find x_t, y_t
    x_t = tailPos(1,i);
    y_t = tailPos(2,i);

    % Step 2 find m_p
    m = 1/a_T;
```

```
    m_p = -1/m;

    a = 1/ a_T;

    b = - b_T / a_T;
    %Step 3 solve for x_c
    polytemp = [a - m_p, b - y_t + m_p * x_t];
    x_c = roots(polytemp);

    %Step 3a Check Sign
    if(x_c > x_t)
        mulNeg = 1;
    else
        mulNeg = -1;
    end
    %Step 4 Solve for y_c
    y_c = y_t + m_p*x_c - m_p * x_t;

    tailDist(counter) = mulNeg*sqrt( (x_c - x_t)^2 + (y_c - y_t)^2 );
    counter = counter + 1;

end


%CALCULATIONS FOR THE Middle
mulNeg = 1;
counter = 1;
medDist = zeros(1,size(positions,2));
medPos = double(positions);



for(i = 1: size(medPos,2))

    % Step 1 find x_t, y_t
    x_t = medPos(1,i);
    y_t = medPos(2,i);

    % Step 2 find m_p
    m = 1/a_M;

    m_p = -1/m;

    a = 1/ a_M;

    b = - b_M / a_M;
    %Step 3 solve for x_c
    polytemp = [a - m_p, b - y_t + m_p * x_t];
    x_c = roots(polytemp);

    %Step 3a Check Sign
    if(x_c > x_t)
```

```
        mulNeg = 1;
    else
        mulNeg = -1;
    end

    %Step 4 Solve for y_c


    y_c = y_t + m_p*x_c - m_p * x_t;

    medDist(counter) = mulNeg*sqrt( (x_c - x_t)^2 + (y_c - y_t)^2 );
    counter = counter + 1;

end

%TRAJECTORY VELOCITY AND ACCELERATION

%CALCULATIONS FOR THE Middle

p_M_x = polyfit(timeArray, positionsM(1,:), 4); %FIT x in terms of Y
p_M_y = polyfit(timeArray, positionsM(2,:), 4);

a_x = p_M_x(1);
b_x = p_M_x(2);
c_x = p_M_x(3);
d_x = p_M_x(4);
e_x = p_M_x(5);

a_y = p_M_y(1);
b_y = p_M_y(2);
c_y = p_M_y(3);
d_y = p_M_y(4);
e_y = p_M_y(5);

a_x_M = p_M_x(1);
b_x_M = p_M_x(2);
c_x_M = p_M_x(3);
d_x_M = p_M_x(4);
e_x_M = p_M_x(5);

a_y_M = p_M_y(1);
b_y_M = p_M_y(2);
c_y_M = p_M_y(3);
d_y_M = p_M_y(4);
e_y_M = p_M_y(5);

y_predict_2 = a_y * timeArray.^4 + b_y * timeArray.^3 + c_y *
timeArray.^2 + d_y * timeArray.^1 + e_y ;
x_predict_2 = a_x * timeArray.^4 + b_x * timeArray.^3 + c_x *
timeArray.^2 + d_x * timeArray.^1 + e_x;


F =  @(x) sqrt( (4*a_x *x.^3 + 3 * b_x * x.^2 + 2 * c_x* x.^1 +  d_x
).^2 + (4*a_y *x.^3 + 3 * b_y * x.^2 + 2 * c_y* x.^1 +  d_y ).^2);
```

```
lengthValsM = zeros(2, size(timeArray,2)-1);
for(i = 1:(size(timeArray,2) -1 ))
    t_val_1 = timeArray(1);
    t_val_2 = timeArray(i + 1);

    lengthValsM(1,i) = quadl(F, t_val_1, t_val_2); %DISTANCE TRAVELED
ALONG TRAJECTORY
    lengthValsM(2,i) = timeArray(i);
end


velValsM = calcDeriv(lengthValsM);
accelValsM = calcDeriv(velValsM);

%CALCULATIONS FOR THE HEAD
p_M_x = polyfit(timeArray, positionsH(1,:), 4); %FIT x in terms of Y
p_M_y = polyfit(timeArray, positionsH(2,:), 4)

a_x = p_M_x(1);
b_x = p_M_x(2);
c_x = p_M_x(3);
d_x = p_M_x(4);
e_x = p_M_x(5);

a_y = p_M_y(1);
b_y = p_M_y(2);
c_y = p_M_y(3);
d_y = p_M_y(4);
e_y = p_M_y(5);

y_predict_2H = a_y * timeArray.^4 + b_y * timeArray.^3 + c_y *
timeArray.^2 + d_y * timeArray.^1 + e_y ;
x_predict_2H = a_x * timeArray.^4 + b_x * timeArray.^3 + c_x *
timeArray.^2 + d_x * timeArray.^1 + e_x;


F =  @(x) sqrt( (4*a_x *x.^3 + 3 * b_x * x.^2 + 2 * c_x* x.^1 +  d_x
).^2 + (4*a_y *x.^3 + 3 * b_y * x.^2 + 2 * c_y* x.^1 +  d_y ).^2);

lengthValsH = zeros(2, size(timeArray,2)-1);
for(i = 1:(size(timeArray,2) -1 ))
    t_val_1 = timeArray(1);
    t_val_2 = timeArray(i + 1);

    lengthValsH(1,i) = quadl(F, t_val_1, t_val_2); %DISTANCE TRAVELED
ALONG TRAJECTORY
    lengthValsH(2,i) = timeArray(i);
end
velValsH = calcDeriv(lengthValsH);
accelValsH = calcDeriv(velValsH);


%CALCULATIONS FOR THE Tail
```

```
p_M_x = polyfit(timeArray, positionsT(1,:), 4); %FIT x in terms of Y
p_M_y = polyfit(timeArray, positionsT(2,:), 4)

a_x = p_M_x(1);
b_x = p_M_x(2);
c_x = p_M_x(3);
d_x = p_M_x(4);
e_x = p_M_x(5);

a_y = p_M_y(1);
b_y = p_M_y(2);
c_y = p_M_y(3);
d_y = p_M_y(4);
e_y = p_M_y(5);

y_predict_2T = a_y * timeArray.^4 + b_y * timeArray.^3 + c_y *
timeArray.^2 + d_y * timeArray.^1 + e_y ;
x_predict_2T = a_x * timeArray.^4 + b_x * timeArray.^3 + c_x *
timeArray.^2 + d_x * timeArray.^1 + e_x;


F =  @(x) sqrt( (4*a_x *x.^3 + 3 * b_x * x.^2 + 2 * c_x* x.^1 +  d_x
).^2 + (4*a_y *x.^3 + 3 * b_y * x.^2 + 2 * c_y* x.^1 +  d_y ).^2);

lengthValsT = zeros(2, size(timeArray,2)-1);
for(i = 1:(size(timeArray,2) -1 ))
    t_val_1 = timeArray(1);
    t_val_2 = timeArray(i + 1);

    lengthValsT(1,i) = quadl(F, t_val_1, t_val_2); %DISTANCE TRAVELED
ALONG TRAJECTORY
    lengthValsT(2,i) = timeArray(i);
end


velValsT = calcDeriv(lengthValsT);
accelValsT = calcDeriv(velValsT);


% %CALCULATE TURNING RADIUS

t = timeArray;
x_dot = 4 * a_x_M* t.^3 + 3 * b_x_M * t.^2 + 2 * c_x_M * t.^1 + d_x_M;
y_dot=  4 * a_y_M* t.^3 + 3 * b_y_M * t.^2 + 2 * c_y_M * t.^1 + d_y_M;
x_ddot = 12 * a_x_M * t.^2 + 6 * b_x_M * t +2* c_x_M;
y_ddot = 12 * a_y_M * t.^2 + 6 * b_y_M * t +2* c_y_M;
curvature = (x_dot .* y_ddot - y_dot .* x_ddot) ./ ((x_dot.^2 +
y_dot.^2).^1.5);

p_turn = polyfit(positionsM(1,:), positionsM(2,:), 2);
a_turn = p_turn(1);
b_turn = p_turn(2);
c_turn = p_turn(3);
```

```
x_predict_t = positionsM(1,:);
y_predict_t = a_turn * x_predict_t.^2 + b_turn* x_predict_t.^1 +
c_turn;

y_t_dot=  2 * a_turn* x_predict_t.^1 +  b_turn  ;
y_t_ddot = 2 * a_turn;


curvature_2 = y_t_ddot ./ ((1 + y_t_dot.^2).^1.5);
turnRadArray_2 = 1 ./ curvature_2;

turnRadArray = 1./curvature;
turnRad = 1 / (2 * a_turn) * 1 / scaleFactor / (10 * 2.54/100);

%CALCULATE TURNING ANGLE
del_X = positionsH(1,:)- positionsM(1,:);
del_Y = positionsH(2,:) - positionsM(2,:);
phi_vals(1,:) = atan2(del_X , del_Y);
phi_vals(2,:) = timeArray;
p_phi = polyfit(timeArray, phi_vals(1,:), 2);

a_phi = p_phi(1);
b_phi = p_phi(2);
c_phi = p_phi(3);

phi_vals(2,:) = timeArray;
phi_vals2(2,:) = timeArray;

phi_vals2(1,:) = a_phi * t.^2 + b_phi * t + c_phi;


omega_vals = calcDeriv(phi_vals);
omega_vals2 = calcDeriv(phi_vals2);


% count0 = 1;
% n = size(omega_vals, 2);
% sum0 = 0;
% interval0 = 2;
% for (i = 1:1:n)
%     if (mod(i,interval0) == 0)
%         sum0 = sum0 + omega_vals(i);
%         mean0(1,count0) = sum0/interval0;
%         mean0(2, count0) = omega_vals(2,i);
%         count0 = count0 + 1;
%         sum0 = 0;
%     else
%         sum0 = sum0 + omega_vals(1,i);
%     end
% end
```

```
% PLOT DATA
figure(2);
hold on;
plot(x_predict_2, y_predict_2, '*','color', 'g');
plot(x_predict_2H, y_predict_2H, '*','color', 'b');
plot(x_predict_2T, y_predict_2T, '*','color', 'r');
totalDist = totalDist/scaleFactor;
headDist = headDist/scaleFactor;
tailDist = tailDist / scaleFactor;
medDist = medDist / scaleFactor;



figure(5)
hold on;
% plot(timeArray, headDist, 'b');
% plot(timeArray, tailDist, 'r');
% plot(timeArray, medDist, 'g');
plot(timeArray,headDist,'b',timeArray,
tailDist,'r',timeArray,medDist,'g','LineWidth',2);
grid;
title('Clear Bass F 5.0 Hz');
xlabel('Time [s]');
ylabel('Lateral displacement [m]');
legend('Head','Tail','Mid Body');
positions = double(positions);

figure(6)
hold on
plot(lengthValsM(2,:), lengthValsM(1,:)/scaleFactor, 'g');
plot(lengthValsH(2,:), lengthValsH(1,:)/scaleFactor, 'b');
plot(lengthValsT(2,:), lengthValsT(1,:)/scaleFactor, 'r');
grid on;
xlabel('Time [s]');
ylabel(' Distance Traveled Along Trajector(m)');
legend('Mid Body', 'Head', 'Tail');

figure(7)
hold on
plot(velValsM(2,:), velValsM(1,:)/ scaleFactor, 'g');
plot(velValsH(2,:), velValsH(1,:)/ scaleFactor, 'b');
plot(velValsT(2,:), velValsT(1,:)/ scaleFactor, 'r');
grid on;
xlabel('Time [s]');
ylabel('Velocity Traveled Along Trajector(m)');
legend('Mid Body', 'Head', 'Tail');

figure(8)
hold on
plot(accelValsM(2,:), accelValsM(1,:)/ scaleFactor, 'g');
plot(accelValsH(2,:), accelValsH(1,:)/ scaleFactor, 'b');
plot(accelValsT(2,:), accelValsT(1,:)/ scaleFactor, 'r');
grid on;
xlabel('Time [s]');
```

```
ylabel('Acceleration Traveled Along Trajector(m)');
legend('Mid Body', 'Head', 'Tail');

figure(9)
hold on
plot(timeArray, phi_vals(1,:) * 180 / pi);
plot(timeArray, phi_vals2(1,: )* 180/pi, 'r');
grid on;
xlabel('Time [s]');
ylabel('Phi (degrees)');

figure(10)
hold on
plot(omega_vals(2,:), omega_vals(1,:)* 180 / pi);
plot(omega_vals(2,:), omega_vals2(1,:)* 180 / pi, 'r');
grid on;
xlabel('Time [s]');
ylabel('Omega (deg/s)');

figure(11)
hold on
plot(timeArray, turnRadArray / scaleFactor / (10*2.54/100))
hold on
plot(timeArray, abs( turnRadArray_2) / scaleFactor / (10 * 2.54 /100) ,
'r');
grid on;
xlabel('Time [s]');
ylabel(' Turning Radius (bodyLengths)');

figure(12)
axis equal
hold on
plot(positionsM(1,:), positionsM(2,:), '*g');
plot(x_predict_2, y_predict_2, '.','color', 'g');
grid on;
title('Compare Fit');
xlabel('X Position');
ylabel('Y Position)');
plot(positionsH(1,:), positionsH(2,:), '*b');
plot(x_predict_2H, y_predict_2H, '.','color', 'b');
plot(positionsT(1,:), positionsT(2,:), '*r');
plot(x_predict_2T, y_predict_2T, '.','color', 'r');
legend( 'Actual Data Middle' , 'Fitted Data Middle', 'Actual Data Head'
, 'Fitted Data Head', 'Actual Data Tail' , 'Fitted Data Tail');

figure(13)
axis equal
hold on
plot(positionsM(1,:)/ scaleFactor, positionsM(2,:)/ scaleFactor, '*g');
plot(x_predict_t/ scaleFactor, y_predict_t/ scaleFactor, 'r');

disp('Turning Radius in Body Lengths');
disp(turnRad);
```

```
%WRITING DATA TO FILE
fid = fopen(file2, 'w+');
dataM(:,1) = timeArray;
dataM(:,2) = positionsM(1,:);
dataM(:,3) = positionsM(2,:);
dataM(:,4) = positionsH(1,:);
dataM(:,5) = positionsH(2,:);
dataM(:,6) = positionsT(1,:);
dataM(:,7) = positionsT(2,:);
dataM(:,8) = ones(1,size(timeArray,2)) * scaleFactor;
for(i = 1: size(dataM,1))
    fprintf(fid, '%12.6f %12.6f %12.6f %12.6f %12.6f %12.6f %12.6f
%12.6f\n ', dataM(i,1), dataM(i,2), dataM(i,3), dataM(i,4), dataM(i,5),
dataM(i,6), dataM(i,7), dataM(i,8));
end
fclose(fid);
```