GLIDING ROBOTIC FISH: CONTROL AND EXPLORATION UNDER LOCALIZATION UNCERTAINTIES

By

Demetris Coleman

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering — Doctor of Philosophy

2024

ABSTRACT

Autonomous underwater vehicles have a variety of applications such as environmental monitoring, search and rescue, ocean exploration, and fish tracking. One such class of these vehicles is gliding robotic fish, which realizes energy-efficient locomotion and high maneuverability by combining buoyancy-driven gliding and fin-actuated swimming. The goal of this dissertation is to endow gliding robotic fish with advanced control capability and autonomy, to facilitate their ultimate applications in aquatic environments.

First, an overview of the gliding robotic fish platform GRACE is presented and design improvements for the third generation of GRACE are discussed. These include adding Iridium satellite-based communication for remote operation, making the robot more robust for ocean operation, and developing a miniaturized version (Miniglider) to enable rapid testing of functionality and control algorithms.

Second, a backstepping-based trajectory tracking controller for the energy-efficient gliding-like motion of gliding robotic fish is proposed. The controller is designed to track the desired pitch angle and reference position in 3D space. In particular, under-actuation is addressed by exploiting the coupled dynamics and introducing a modified error term that combines pitch and horizontal position tracking errors. Two-time-scale analysis of singularly perturbed systems is used to establish the convergence of all tracking errors to a neighborhood around zero. The effectiveness of the proposed control scheme is demonstrated via simulation and experimental results.

Next, incorporating observability into control schemes is discussed. Incorporating observability can enhance an observer's ability to recover accurate estimates of unmeasured states, minimize estimation error, and ultimately, allow the original control objective to be achieved. The use of control barrier functions (CBFs) is proposed to enforce observability and thereby encourage convergence of state estimates to the true state in output feedback control schemes. The pro-

posed approach is compared to a model predictive control (MPC)-based alternative that optimizes a weighted combination of an observability surrogate function and the control objective. Motivated by the applications of fish tracking and navigating in GPS-denied environments, the problem of target tracking, when only the distance to the target is measured, is addressed. It is found that both approaches are comparable in terms of observability and estimation error, but the CBF-based approach has an edge in terms of computational efficiency. Experimental validation of the CBF-based scheme is conducted with a Miniglider.

To complete this body of work, a strategy for the exploration of unknown scalar fields under localization uncertainty is proposed. The strategy hinges on the concept of the multi-fidelity Gaussian processes (GPs) and sampling-based motion planning for information gathering. It uses multi-fidelity GPs to approximate the environmental field by assigning location-measurement pairs to a particular fidelity based on the level of uncertainty in the location estimate. An informative trajectory planner is then designed that plans not only where the robot should go, but also what types of motion (e.g. swimming, gliding, etc.) the robot should use to best gather information for the reconstruction of the field. Experiments are carried out on a Miniglider for the task of mapping the light field in an indoor tank. The results show that using a multi-fidelity GP model provides a better reconstruction of the field in terms of the weighted mean squared error when compared to using standard GP regression, where the localization error is ignored.

Dedicated to my mother,
Carlene Bradley,
and grandparents,
Sarah and Ira Bradley,
for everything you've done.

ACKNOWLEDGMENTS

I would like to express my greatest gratitude to my advisor and director of the Smart Microsystems Lab at Michigan State University, Prof. Xiaobo Tan. I was fortunate to work with and learn from him and some of my future senior graduate students during an undergraduate research program at Michigan State. After allowing me to join the Smart Microsystems Lab, he provided continuous support, advice, mentoring, and motivation throughout my Ph.D. journey in the field of underwater robotics. He has had a profound effect on my growth and development, both academically and as a person.

I want to thank my academic committee members Dr. Bopardikar, Dr. Srivastava, and Dr. Li for their insightful comments in the areas of control and robotics, general guidance, advice, and willingness to always make time to help me through problems. I also want to thank Dr. Percy Pierre and Dr. Nelson Sepulveda for their support and their work with the Sloan Engineering Program.

I also owe gratitude to many of my fellow members of the Smart Microsystems Lab. I would like to give special thanks to John Thon for his continuous support, his advice, and his crucial contributions to the mechanical aspects of the robots we built. I also would like to specifically thank Osama Enassr, Maria Castaño, Pratap Solanki, Mohaammed Al-Rubaiai, and Thassyo Pinto for taking the time to teach me and discuss problems with me throughout my Ph.D. journey. I would also like to thank them, all the other students in the Smart Microsystems and D-CYPHER Labs, and the close friends I've made for years of companionship, support, and wonderful memories.

I also want to acknowledge the support from the funding agencies that made this work possible: the National Science Foundation (IIS 1848945, ECCS 2030556, ECCS 1446793, IIS 1715714), the Department of Education (Grant #P200A180025), the United States Geological Survey, and the Rose Graduate Fellowship Fund in Water Research.

Finally, I want to thank friends and family for their unconditional love. I especially want to thank my mother, Carlene Bradley, my grandparents, Sarah and Ira Bradley, my cousin, Trinity, and my siblings, both by blood and chosen. I could not have finished my Ph.D. program without their understanding, support, and encouragement.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Overview and Applications of Underwater Gliders

The concept of underwater gliders was introduced by Henrey Stommel just over a decade before the turn of the century [1]. Underwater gliders are known for their high energy efficiency and exceptional capability for long-duration operations. They use variable buoyancy, hydrofoils, and a shifting center of gravity to realize horizontal travel. They are typically operated to achieve steady-state motion patterns, including sawtooth-like rectilinear gliding, and spirals induced by controlling the vehicle's roll angle or by deflecting control surfaces. The success of the early gliders such as SLOCUM [1], Spray [2], and Seaglider [3] has inspired the development of other underwater vehicles that exploit gliding [4], [5]. One of these is the gliding robotic fish [5, 6], which achieves both high energy efficiency and high maneuverability by combining the gliding mechanism with the tail-actuated maneuvers of robotic fish [7]. This makes the design concept better suited for smaller and more complex environments than typical underwater gliders while maintaining the capability to operate for long periods of time in large open environments such as the ocean. The gliding robotic fish has demonstrated promise in environmental sensing and fish tracking applications [8, 9].

Over the past two decades, climate change, the occurrence of multiple water crises, a growing interest in sustainability, and a general interest in understanding aquatic environments have made

underwater gliders an indispensable tool for oceanographers and marine scientists [9–11]. They have been used for many applications in ocean physics, chemistry, and biology. In addition, the data they provide improve models of ocean circulation, hurricane intensity forecast models, and weather forecasting. However, traditional underwater gliders or ocean gliders tend to be large (50 kg or more) and expensive (upwards of $50,000). While the size is not an issue in large open waters such as the ocean and the price is cost-effective when compared to manning and deploying large research vessels, operation of multiple ocean gliders may be cost-prohibitive in smaller aquatic environments such as ponds and inland lakes and the gliders lack the maneuverability to be effective in such environments. The gliding robotic fish can bring the benefits of ocean gliders to these areas while maintaining the capability to operate in large open-water environments. This research aims to advance the capabilities of the gliding robotic fish by enhancing motion control and state estimation and enabling autonomous exploration under the constraints of aquatic environments.

1.2 Control of Underwater Glider

A fundamental step in advancing the capabilities of the gliding robotic fish is the development of control algorithms. The energy-efficient gliding motion of the robot is closely related to the control of underwater gliders. Early work in control of gliders saw the use of PID controllers for their simplicity [2], [3]. More advanced and model-based control methodologies have been proposed in the past two decades. For example, Leonard and Graver used a linear quadratic regulator on linearized dynamics to control the magnitude of velocity on a steady-state glide path [12], [13]. Isa and Arshad analyzed the use of a neural network as a model predictive controller and a gain-tuner algorithm to control the pitch angle and linear velocities based on a linearized glider model [14]. Wang et al. used model predictive control for depth regulation along with a PID controller for

maintaining heading [15]. Neural network-based control was used to implement a self-tuning PID controller to track the velocity along a single axis in the inertial frame [16]. Nag et al. [17] compared fuzzy logic control against PID for pitch and depth tracking. Mahmoudian and Woosely developed an efficient path planning strategy that concatenates equilibrium turning and gliding motions and then implemented the strategy using PID controllers to reach a specified center of gravity and center of buoyancy [18]. Zhang et al. [19] used nonlinear passivity-based control to stabilize the glide path of a glider in the sagittal plane with a whale-like tail. Sliding mode control has also been explored because of its robustness to disturbances. Castaño and Tan proposed a sliding mode controller for the simultaneous stabilization of pitch and yaw [20]. Yang and Ma used sliding mode control to track trajectories of the pitch angle and ballast mass [21]. Mat-Noh et al. used a linearized glider model to compare an Integral Super Twisting Sliding Mode controller with several other sliding mode variants for stabilizing a gliding path between 30 and 45 degrees [22]. In [23], several different control strategies for underwater gliders are compared.

This rich history of control for underwater gliders provides a starting point for developing control algorithms for the gliding robotic fish. Furthermore, building on this history and advancing it will also benefit the control of underwater gliders.

1.3 Output Feedback Control and Range-based Localization

When tracking fish or navigating in underwater environments, access to the states of the system may not always be available. In this case, output feedback control becomes a necessary problem to tackle. Output feedback control is an important problem for nonlinear systems, where a popular class of approaches involves the pairing of a state feedback controller and an observer. While there are a variety of observer choices for nonlinear systems, such as Luenberger-like observers

[24], high gain observers [25], sliding mode observers [26], moving horizon estimators [27], and variations of the Kalman filter [28] and the particle filter [29], the separation principle does not hold in general for nonlinear systems. This makes output feedback difficult because of the potential conflict between the state estimation and control objectives.

Recent works have increasingly explored measures of observability and the posterior estimation error covariance matrix to improve estimation performance [30–36]. While some propose metrics of the observability Gramian or the observability matrix as tools for improving state estimation, several works that focus on path planning and trajectory optimization in environments with multiple landmarks advocate for the posterior estimation error covariance to aid in improving estimation for nonlinear system [31, 32]. In [32], the authors argued that optimizing measures of the observability Gramian as a surrogate for the estimation performance may provide irrelevant or misleading trajectories for planning under observation uncertainty. They instead suggested using measures of the posterior Fisher information matrix. As an example, they used the trace of the covariance matrix produced by a Kalman filter as a metric to improve observability while planning the path.

Motivated by robots operating in GPS-denied environments, we are particularly interested in control of such robots where the position is estimated based on measurements of their range to some beacon. Measurements of the distance to a beacon or another robot can be obtained using the hardware embedded in many modern communication systems used on robots. Autonomous underwater vehicles (AUVs) are one such class of robots that regularly operate in GPS-denied environments and have become valuable for a multitude of applications [37, 38]. While AUVs are unable to utilize radio frequency-based communication solutions to provide range measurements in underwater environments, advances in technology have led to acoustic communication tools, such as the Woods Hole Oceanagraphic Institute's micromodem, that can also aid in localization

and navigation [39, 40].

The capabilities of acoustic technology led many researchers to explore the use of static beacons or surface vehicles as communication and navigation aids (CNAs) to underwater vehicles [41–46]. A particularly interesting instance of this class of problems is the single beacon navigation (SBN) problem and its variants. In the SBN problem, an AUV estimates its position using inertial sensors, the knowledge of its dynamic model, and the measurement of its range to a single beacon while locomoting. The SBN problem is often important for AUVs because techniques such as simultaneous localization and mapping are not always applicable due to environments with sparse landmarks and low visibility.

Several groups have studied the observability of the SBN problem. Hinson et al. used the condition number of the empirical observability Gramian in path planning to improve observability in a uniform flow field [47]. Arrichiello and coauthors investigated the observability of relative localization of two AUVs equipped with velocity, depth, and range measurement sensors [34]. They proposed an observability metric, which is utilized in this work, and showed how the range between the vehicles and the angle between the relative velocity and relative position vectors affect the localization performance. Antonelli et al. explored estimation based on range measurements between two robots moving in a 3D environment, where the kinematic model is considered [33]. The authors of [48] studied the observability of SBN with the kinematic model of an AUV moving in the horizontal plane and provided an explanation of when the position of the vehicle can be found using only measurements of the distance from a static beacon.

While many works study the observability of the SBN problem, there is less work using observability metrics for control in the context of the SBN problem. The authors of [49] developed a controller for homing in on a static beacon using range measurements. The controller was inspired by previous results on observable paths but used a heuristic approach based on a covariance

threshold to achieve observable maneuvers. In [50], the authors considered improving the localization estimation of multiple AUVs following fixed paths using a surface vessel moving with constant speed as a CNA. For steering control of the CNA, they proposed the path inertia method, which chooses a heading rate by solving a minimization problem based on the system observability Gramian and an empirical Gramian-based method; in particular, the system is simulated forward in time for a finite set of possible control inputs over a fixed time period, and the heading rate that minimizes the empirical local observability Gramian is chosen. The path inertia and empirical Gramian methods are computed at every time step, assuming the AUVs are static beacons and that the heading control input will be applied indefinitely.

Output feedback for nonlinear systems is still an open problem that necessitates more study and proposed solutions. While unique to nonlinear systems, incorporating observability into the control strategy can help improve an observer's ability to produce good state estimates for scenarios such as range-based target tracking. For this reason, more work in the spirit of [49] and [50] is needed to make advances in output feedback control for nonlinear systems.

1.4 Exploration Under Localization Uncertainty

Many robotic applications that use a mobile robot to traverse an environment and collect data require precise localization of the data collected by the robot. In general, it may not be known a priori where data should be collected. Examples of these applications include search and rescue [51], multi-target search [52], and environmental sampling [53]. Gaussian process (GP) regression is commonly employed to efficiently construct a surrogate model of the field as a function of the position without measuring every point when exploring a spatial field. GP regression enables the development of sampling strategies that help decide informative paths to sample along based on

previously observed data. These strategies can be beneficial in many applications where robots are deployed to collect spatio temporal data such as temperature, fish population density, or air quality.

Underwater gliders [1–3] or gliding robotic fish [5, 6], have energy-efficient motion that allows them to operate for long periods of time. This makes them attractive for performing long-term autonomous operations but requires that they spend significant time underwater, which often causes intermittent access to localization aids such as GPS and large localization uncertainty due to the attenuation of radio waves in water. This localization uncertainty makes it challenging to accurately localize collected data or effectively perform autonomous exploration. In some cases, techniques such as simultaneous localization and mapping can be used [54]. However, they require the environment to be sufficiently rich in features, are computationally intensive, and there are many scenarios (e.g., computationally limited robots) where this is not a viable option and the position of the vehicle has to be estimated with dead reckoning and or model-based estimation. This makes standard Gaussian process regression, which assumes precise inputs (position and time for a spatio temporal process), no longer directly applicable.

Prediction with localization uncertainty can be obtained as a posterior predictive distribution using Bayes' rule, but it generally has no analytical closed-form solution and must be approximated [55, 56]. Several researchers have proposed different methods for handling uncertainty in the inputs of a GP [55–62]. Some methods to approach the problem include approximating the posterior predictive statistics of GP regression using Monte Carlo sampling or using Laplace's method [55] and combining Jacobi over-relaxation and discrete-time average consensus [56]. Other methods consist of numerically calculating the expectation of the kernel function [57, 58] and developing kernels that account for localization error [59]. Some works, e.g. [60, 61], develop GP training methods for input that is corrupted by independent, identically distributed (i.i.d.) Gaussian noise. In [62], the authors consider both measurement and test locations to be uncertain and treat in-

puts as probability distributions to account for the uncertainty. Many of the mentioned approaches leverage sensor networks, assume the input is measured with a constant distribution for localization uncertainty, or consider a particular covariance kernel and do not address input with variable uncertainty.

1.5 Overview of Contributions

The research has contributions in the areas of development and improvements of the gliding robotic fish, model-based control for gliding-type underwater robots, observability-based control with applications to range-based target tracking, and autonomous exploration in aquatic environments. These are summarized in the following subsections.

1.5.1 Development and Improvements to Gliding Robotic Fish

The gliding robotic fish, GRACE, has been through several iterations. The first-generation design of these robots was successfully used in sampling harmful oil and algae bloom [63], but did not allow for ease of inspection, hardware upgrades, or adaptability to different sensor payloads. The second-generation gliding robotic fish improved the design by incorporating a resealable hull that facilitated the rapid testing of different electrical designs and allowed regular inspection and maintenance to be performed. In addition, a larger, replaceable sensor payload that can be customized to fit different applications was included and computational power, data storage capacity, and battery capacity were dramatically improved to facilitate carrying out longer missions. This allowed it to be used as a mobile acoustic telemetry platform [8]. The third generation, discussed in this research, focuses on allowing for further modularity of the system, reduction of the manufacturing and fabrication costs, increasing remote operation capabilities, and addressing mechanical issues of

the second-generation design. Along with the third-generation design, a miniature gliding robotic fish, Miniglider, was developed that allows for rapid testing of algorithms in a large indoor tank.

1.5.2 Backstepping-Based Trajectory Tracking

While extensive work has been done in glider control, the focus of these approaches has been mainly on stabilization based on linearized models, or single-input-single-output control of heading, pitch, velocity, or depth. Trajectory tracking is a fundamental and valuable ability for robots exploring complex environments. In particular, it enables improved performance for applications in oceanography, marine science, water quality monitoring, and surveillance, and has direct relevance to various sampling and target-tracking applications in the underwater environment. Extensive work has been done on trajectory tracking and path following for propeller-driven underwater vehicles [64–71]. For underwater gliders, however, position control or trajectory tracking in the 3D space is scarce. One of the very few examples considering the full dynamic model of a gliding system is [72], where the authors proposed an adaptive backstepping controller for tracking the velocity magnitude, yaw angle and pitch angle of an underwater glider. However, position tracking is often times more valuable when, for example, operating in cluttered underwater environments. In this work, a backstepping-based trajectory-tracking controller that tracks 3D position is proposed for gliding-based underwater robots focusing on the model of a gliding robotic fish. The difficulty in controlling the 3D position of gliding-type robots lies in the fact that only the pitch moment and vertical motion are actively controlled during gliding motion. Like multi-rotor drones, the robot must orient itself to achieve planar motion. Unlike multi-rotor drones, this does not produce an orientation-dependent thrust vector. Instead, planar motion is achieved through the lift forces applied on the wings and control surfaces from the surrounding water. The magnitude of the lift force is heavily dependent on the pitch angle and the vertical velocity. In addition, a gliding

robot usually cannot produce a yaw moment without a non-zero velocity and non-zero pitch angle. Three control inputs are used to simultaneously track the pitch angle, due to its strong influence over the planar movement, and the 3D position. To facilitate the control design, the tracking errors are expressed in a cylindrical coordinate system with its origin coinciding with that of the robot's body-fixed frame. Furthermore, an error function modifying the pitch tracking error with a term dependent on the horizontal position error is introduced to address the problem of under-actuation. An intuition is then given to explain why the proposed controller is able to achieve tracking of all four components of the reference trajectories (pitch and 3D position). The convergence of the position and pitch angle tracking errors is rigorously demonstrated using two-time-scale analysis of singularly perturbed systems. Simulation studies and experimental results are presented to show the efficacy of the proposed control design. The proposed approach is compared to a PID controller and a baseline backstepping controller not using the modified error. Then, a procedure for estimating crucial parameters present in the model is presented and employed for a miniature gliding robotic fish. Lastly, a model-based observer is implemented to estimate the body-fixed velocities, which are otherwise not directly accessible from onboard sensors, before implementing the proposed approach on the miniature gliding robotic fish and showing its advantage over a well-tuned PID controller and a baseline backstepping controller.

1.5.3 Observability-Aware Target Tracking

Motivated by the applications of fish tracking and navigating without position measurements, a control framework to incorporate observability into control schemes is proposed. The aim is to design control strategies that minimize state estimation error by enhancing an observer's ability to accurately reconstruct the state estimates while also achieving satisfactory performance on a nominal control objective. An approach to incorporating observability in output feedback control

by using control barrier functions (CBFs) is proposed. The approach enforces local weak observability through CBFs based on properties of the observability matrix.

In recent years, CBFs have been heavily studied in the context of safety-critical controllers [73–76]. Some works also leverage CBFs to facilitate multi-objective control in multi-robot systems [77–80]. CBFs enforce forward invariance of a set with respect to the state of a dynamical system, and thereby, ensure safety properties [73]. This particular ability is shown to work with any locally Lipschitz controller for control affine systems through a computationally efficient optimization scheme. The ease of computation and theoretical guarantees for forward invariance makes this approach an attractive solution for improving observability. One can leverage barrier function or the related control barrier certificates for improving observability by designing a barrier function that captures the set of states that ensure a system is locally observable.

These observability-based control techniques are applied to the target tracking problem when the range to a target is measured in place of the relative position. The proposed approach is compared with an alternative, model predictive control (MPC)-based method that adds observability metrics to the cost function. The approaches are shown to be similar in terms of control performance and estimation performance, but the proposed approach is significantly less computationally expensive.

1.5.4 Enabling Autonomous Exploration Under Localization Uncertainty

One primary goal of gliding robotic fish and other gliding-like robots operating in aquatic environments is data collection. Currently, it is common to do this by having humans operate the robots through direct control or methods such as waypoint or behavior selection. Designing algorithms to imbue the robot with autonomy when operating can increase efficiency. Specifically for the gliding robotic fish, previous sampling algorithms focus primarily on column sampling [9, 53] or

gliding with fixed depth [81]. In [9], column samples are taken in a uniform grid chosen by a human operator based on prior knowledge of the process being sampled. Ergodic control and mutual information for a Gaussian process regression model are used to plan a 2D trajectory for a unicycle model in [53] and column samples are taken after the robot moves a fixed distance from the previous sample location. These algorithms do not utilize the full capabilities of the gliding robotic fish, consider only the 2D position for planning purposes, and do not consider localization errors.

This work builds upon [81] by leveraging multi-fidelity GP regression to model the environment and to incorporate localization uncertainty through localization uncertainty-dependent fidelity levels. The idea of multi-fidelity GPs stems from the works of [82] and [83] and have gained attention in recent years as a modeling tool [52, 84–87]. The original motivation for multi-fidelity GPs was to approximate high-fidelity data in a computationally efficient manner by using surrogate models whose accuracy drops as they become computationally cheaper. Recent works have shown that the multi-fidelity GP model extends nicely to sensing with downward-facing cameras, where fidelity is dependent on the vertical distance from a 2D field being surveyed by a vehicle [52, 87].

A notable difference between the present work and the multi-fidelity models presented in [82, 83] is the source of the difference in fidelity levels. In the aforementioned works [52, 82, 83, 87], the fidelity level is caused by the output data from less accurate models or lower-quality sensing given the true input data, whereas the fidelity in the present work is due to inaccurate input data (location). The present work seeks to leverage this lower fidelity input data to aid in efficiently reconstructing a spatial measurement field.

In addition to the proposed multi-fidelity model, this work develops an adaptive sampling algorithm that considers the multiple modes of operation for the gliding robotic fish while planning trajectories for data collection. A miniature gliding robotic fish is used to experimentally validate

the proposed approach. The experiments show that, compared to using a standard GP regression model that ignores localization uncertainty, the proposed approach lowers the reconstruction error when measured by the weighted mean-squared error (WMSE).

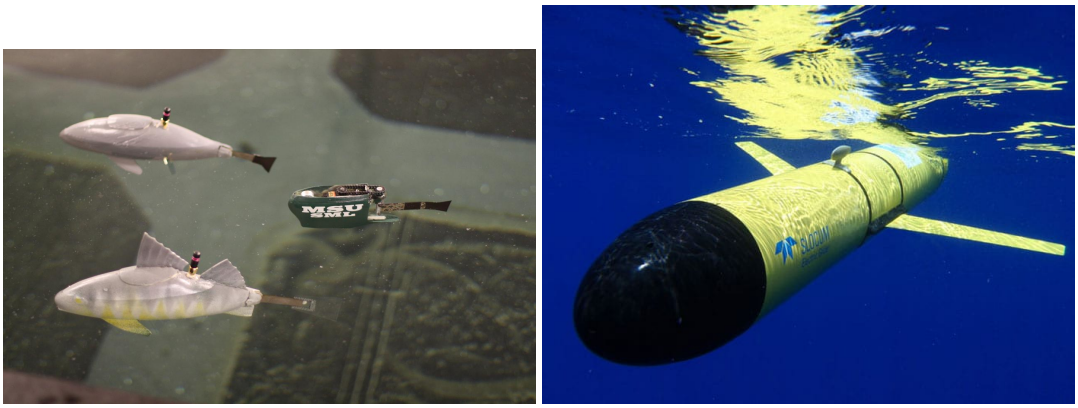
Chapter 2

Design of Gliding Robotic Fish

2.1 Gliding Robotic Fish

The gliding robotic fish, GRACE, was developed in the Smart Microsystems Laboratory at Michigan State University. The design concept of GRACE was inspired by the energy-efficient motion of underwater gliders and the high maneuverability of robotic fish (Figs 2.1). Underwater gliders, renowned for their energy efficiency, achieve motion by changing buoyancy to move vertically through the water and changing the center of mass to orient which generates hydrodynamic forces on the wings of the robot. Some underwater gliders have rudders to steer, while others change the roll angle to steer, but in both cases, change in heading direction as well horizontal movement are passive actuation mechanisms that take advantage of the natural physics of the robot. This allows them to achieve steady state motion without the need to constantly move actuators which contributes greatly to their energy efficiency, but also makes them slow, limits maneuverability, and unsuitable for environments such as ponds and inland lakes.

In opposition, robotic fish are bio-inspired robots that typically have actuators controlling fin-like appendages that can be used for swimming and rapid maneuvers such as quick turns or breaking [88, 89]. Outside of achieving quick maneuvers, these types of robots have to constantly move the actuators in rhythmic patterns such as oscillating the tail to achieve meaningful motion. This can quickly drain energy and limit the operation time of the robots. By combining the concepts,



(a) (b)

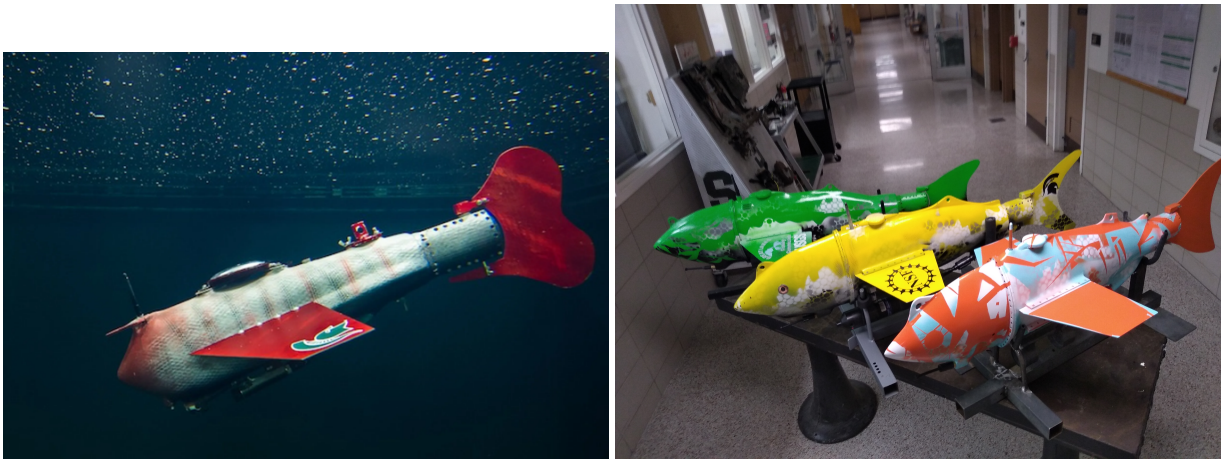
Figure 2.1: Inspirations for gliding robtic fish: (a) school of robotic fish developed in the Smart Microsystems Laboratory and (b) a Slocum glider.

the gliding robotic fish retains the capabilities of both robotic fish and underwater gliders but can adapt for a specified task and use the strengths of one motion mechanism to compensate for the weaknesses of the other.

2.1.1 Past Iterations

Table 2.1 lists some aspects of the previous iterations of GRACE (see Fig. 2.2) reported in [5,6,90]. The first-generation prototype of GRACE, "GRACE 1", was successfully deployed for oil spill detection in Kalamazoo River, Michigan, and for monitoring algae bloom in Wintergreen Lake, MI [9, 63]. While the robot was useful, it had a few problems as a research platform. The hull design of GRACE 1 [5] was permanently sealed with a pressure sensor and GPS, both permanently grafted into the robot shell. This meant accessing the internal electronics or mechanics for upgrades or repairs would effectively require breaking the robot. The first-generation design is also limited in the sensor payload capacity and the battery capacity. It carried one interchangeable environmental sensor.

The second-generation gliding robotic fish, "GRACE 2", improved upon many aspects of



(a) (b)

Figure 2.2: Previous iterations of GRACE. (a): Generation 1 robot gliding. (b): fleet of generation 2 robots.

Table 2.1: Mechanical specifications for Previous Iterations of GRACE.

Component	GRACE 1 Description	GRACE 2 Description
Hull dimensions (L $\times$ W $\times$ H)	$65 \times 15 \times 18$ cm	$103 \times 20 \times 30$ cm
Tail to nose length	90 cm	140 cm
Wingspan	75 cm	60 cm
Weight	9 kg cm	20 kg
Robot hull material	Carbon fiber	Carbon fiber with 3D printed interface
Buoyancy Module Capacity	100 mL	190 mL
Payload	Turner Design Cyclops 7F sensor (crude oil or blue green algae sensor) Temperature sensor	Dissolved Oxygen and Temperature sensor, Photosynthetically Active Radiation sensor, 2 Turner Design Cyclops 7F sensors (Freshwater Blue Green Algae sensor and Chlorophyll), Acoustic receiver

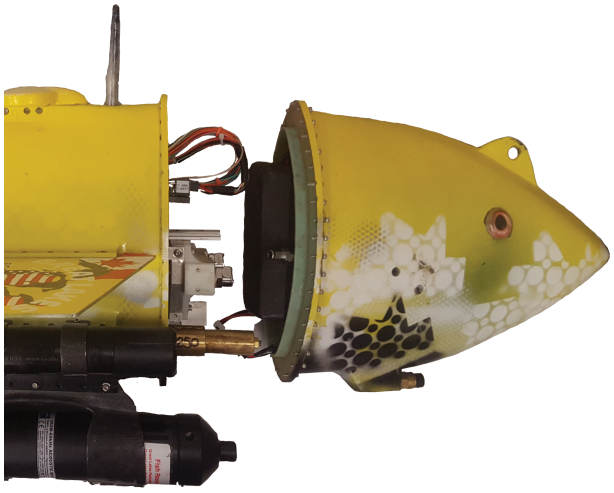


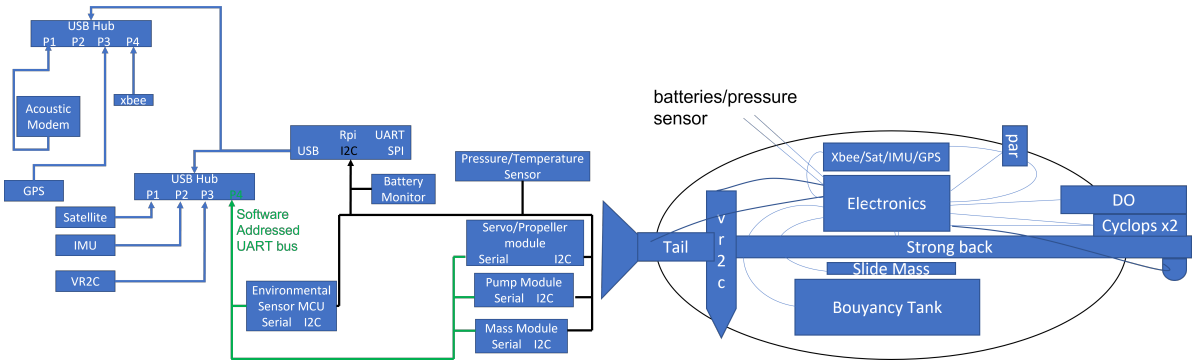
Figure 2.3: Sideview of GRACE 2 with open interface exposing batteries, electronics, mass rack, and glide mechanism.

GRACE 1 and experienced several electrical and mechanical enhancements over the course of its operation. It was sub-versioned as 2.0, 2.1, and 2.5 to distinguish between the incremental upgrades during that period. On the mechanical side, the new design for GRACE 2 addressed the issues of serviceability by introducing a 3D printed interface (see Fig. 2.3) that forms a watertight seal by compressing a custom O-ring between the nose of the robot and the rest of its body. A "gliding mechanism" was designed that combined a movable mass and the buoyancy tank. It also housed the main electronic circuit and an internal rack to carry extra mass (adjustable to a stationary position) and could slide in and out of the robot's main body for maintenance and inspection. Increasing the size of the buoyancy-control tank (volume of approximately 190 mL) and using the mass rack allowed the robot to adapt to a range of different payloads. This enabled the implementation of a changeable sensor harness that could be replaced to accommodate different sensor sets. One such sensor was the VR2Tx acoustic receiver. It is designed as a stand-alone sensor for detecting acoustic tags. Mounting it to GRACE 2 allowed preliminary testing of the robot as a mobile acoustic telemetry platform [8].

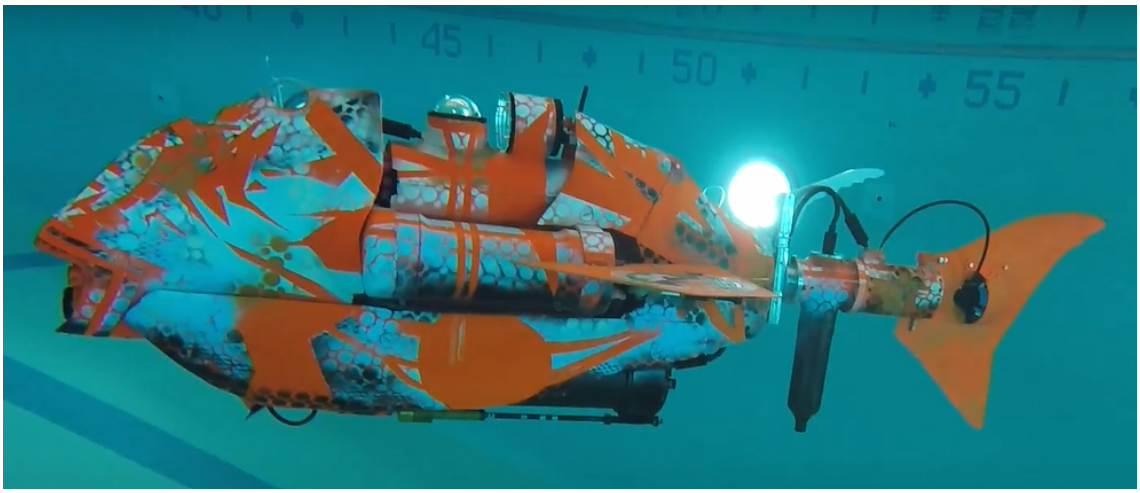
The battery capacity for GRACE 2 was increased significantly and a waterproof connector

was introduced to act as a switch and charging port. The initial electrical design for GRACE 2.0 utilized an architecture similar to the first-generation GRACE, with the exception of introducing a second microcontroller (MCU) dedicated to communicating with the sensor bundle. The power system for the GRACE 2.0 design used linear regulators to regulate battery voltages to 3.3V and 5V, which provided power to the electronics and the sensors. GRACE 2.1 switched to switching regulators for the power design to mitigate excessive heat that caused issues in GRACE 2.0. It also streamlined electrical connections. Both GRACE 2.0 and 2.1 lacked onboard storage of data. They relied on the limited memory on the microcontroller while underwater and sending data to a laptop via XBee RF modules.

GRACE 2.5 was an intermediate step towards the current iteration "GRACE 3.0". It served as a testbed for initial ideas for the new electrical design and software. A Raspberry Pi (RPi) Zero W computer was included to handle high-level tasks (e.g. communicating with the base station, mission control, data storage). The introduction of the RPi also added reliable data storage through the 8 GB micro-SD card that ran its operating system and allowed the MCUs to perform low-level tasks only, such as reading sensor values and controlling actuators. It also enabled rapid testing of functionality without needing to open the robot and adjust the low-level microcontroller code. GRACE 2.5 also added an additional water-proof connector that was used to switch between adding a propeller or the VR2C acoustic receiver which, could be integrated into the robot's electronics to collect the receiver data in real-time, unlike the VR2Tx. Issues with GRACE 2.5 included long times and high resources needed for fabrication, manufacturing, and repairs. Most mechanical components of the robot needed to be custom-built (and, at times, hand-crafted) which lowers productivity for a platform still in the prototyping and testing stage. Malfunctions of the pressure sensor or GPS caused significant downtime, as they were grafted into the body of the robot. The GRACE 2 design was also limited in depth due to compression of the shell leading to



(a) (b)



(c)

Figure 2.4: Depiction of communication structure (a) for electronic circuits and sensors. Schematic (b) and initial prototype (c) of GRACE 3.

a volume decrease large enough to negate the ability to surface using the buoyancy control alone. GRACE 2.5 provided invaluable insight for the next iteration. Many of the electrical designs and much of the software in the previous iterations formed the basis for GRACE 3 which is discussed next.

2.1.2 GRACE Generation 3

Based on issues with previous iterations of Grace, the GRACE 3 design takes a modular approach focused on reducing the cost and time needed for fabrication. It utilizes a series of individu-

Table 2.2: Mechanical specifications for GRACE 3 and Miniglider.

Component	GRACE 3 Description	Miniglider Description
Hull dimensions (L $\times$ W $\times$ H)	$100 \times 30 \times 53$ cm	$40 \times 12 \times 12$ cm
Tail to nose length	170 cm	57 cm
Wingspan	85 cm	37 cm
Weight	33 kg	5 kg
Robot hull material	Carbon fiber, Aluminum, PVC, Acrylic, Foam	Anodized aluminum, Acrlyic, PLA
Buoyancy Module Capacity	420 mL	50 mL
Payload	Dissolved Oxygen and Temperature sensor, Photosynthetically Active Radiation sensor, 2 Turner Design Cyclops 7F sensors (Freshwater Blue Green Algae sensor and Chlorophyll), Vemco VR2C, WHOI micromodem	

ally sealed components that can be quickly removed and replaced to change payload, minimizing downtime if a repair is needed. The components are also based on readily available material to minimize custom-built parts. Additional capabilities beyond that added in GRACE 2.5 include increased battery capacity, quickly swappable sensors, an Iridium satellite communication device for remote communication, an external mass rack for faster balancing with static mass, time synchronization using the GPS pulse per second signal (unused in previous iterations), and a WHOI micromodem that can be used for communication and ranging. It also enables speed control of the mass and buoyancy pump actuators whereas they only accepted setpoint commands in previous versions. The circuitry was also upgraded to fit the modular nature of the new robot design.

Pictures of the initial prototype and schematic of the robot are shown in Fig. 2.4. The robot is built around 6 base components: the tail module, the communication and navigation module, the slide mass module, the buoyancy control module, the battery modules, and the master electronics module. Additional components such as sensors can be added as needed. The modules are encased

in an aluminum, foam, and carbon fiber structure designed to hold the modules in place, protect them from damage, and provide buoyancy where needed. The modules are electrically connected using waterproof connectors. Fig. 2.4 depicts the communication structure between the MCUs, Raspberry Pi, and sensors in the first prototype. GRACE 3 made use of the USB communication on the Raspberry Pi to communicate directly with certain sensors. This gave more control over sensors that could be programmed and more flexibility for possible sensor payloads. It also introduced a software addressed UART communication line to accommodate the MCUs for actuator modules, environmental sensors, and possible additional custom MCUs.

In the initial prototype, the tail module holds the servo and propeller, the buoyancy tank holds a linear actuator and syringe, and the slide mass module holds a linear actuator and movable mass. All of the actuator modules also hold the electronics needed to control their respective actuators. Power and redundant communication lines (UART and I2C) are supplied through the waterproof connections which allow the actuators to be quickly swapped in case of a failure. The two battery modules, buoyancy pump, and slide mass are made from PVC pipe. Acrylic watertight enclosures from BlueRobotics are used for the master electronics and communication and navigation modules. The communication and navigation module contains an IMU, GPS, Iridium satellite module, and an RF Xbee module, while the master electronics module houses the WHOI micromodem circuit, a pressure and water temperature sensor, the power regulation circuit, the Raspberry Pi Zero, a microcontroller to collect analog sensor data, and a USB-TTL conversion circuit to communicate with several other sensors. The watertight enclosures enable easily swappable sensors and payloads. Some components of GRACE 3 and Miniglider (discussed in the next section) are shown in Table 2.3.

Custom software was designed for the Raspberry Pi to run the robot. The software structure was designed to mirror the modularity of the robot. It consisted of several base programs to in-

Table 2.3: Selected Components Used in Current Generation GRACE Robots.

Component Name	MiniGlider Component Model	GRACE 3 Component Model
Master Computer	RPi 4	RPi Zero W
Microcontroller	Microchip dsPIC6014A	Microchip dsPIC6014A, dsPIC2011 (x3)
Battery	Tenergy NiMH 9.6 V	Batteryspace High Power Polymer Li-Ion (x2), 18.5v @ 388 Wh
Data Storage	32 GB SD Card	
Mass Actuator	Actuonix L16-P Linear Actuator with Feedback	
Tail Servo	Hitec HS-646WP	Hitec HS-7980TH
Bouyancy-control Actuator	Actuonix P16-P Linear Actuator with Feedback	ServoCity 6” Stroke 180 lb Thrust Heavy Duty Linear Actuator
RF Communication Module	XBee-Pro 900HP RP-SMA	
IMU	LSM9DS1	VectorNav VN100S
Pressure/Temperature Sensor	BlueRobotics Bar30	
GPS		GPS 18x LVC
Camera	RPi Cam V1	
Acoustic Receiver		Vemco VR2C
Acoustic Modem		WHOI Micromodem
Turner Designs Cyclops-7 Submersible Sensor		sensor-C (Chlorophyll), sensor-P (Algae)
Dissolved Oxygen and Temperature sensor		In-Situ RDO PRO-X

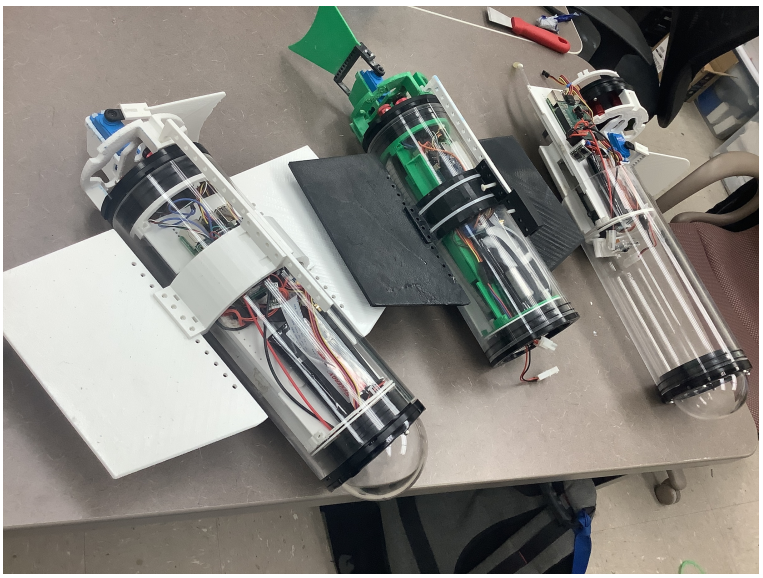
terface with the sensors and actuators which we call ”nodes”. Each node consists of an interface functionality and a socket communication link that can be accessed by multiple other programs to send and receive node-specific commands and data. Programs for higher-level tasks can then be written that communicate with the nodes to retrieve relevant data or send actuator commands. This allows updates to the software to be flexible. For instance, if the IMU needs to be replaced with a different model or the Iridium satellite module needs to be replaced with a cellular communication device, it can be done with minimal or no changes to high-level tasks that depend on the IMU and satellite communication.

2.2 Miniature Gliding Robotic Fish

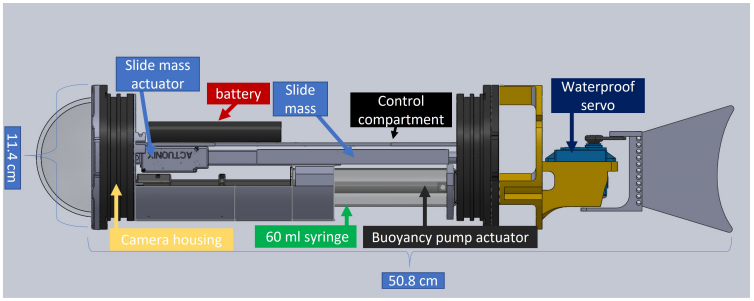
2.2.1 Miniglider

In addition to GRACE 3.0, a miniature gliding robotic fish was developed for rapid testing of algorithms in a controlled environment. A breakdown of the internal structure and actuation system is shown in Fig. 2.5. The robot actuation system consists of two linear actuators (Actuonix P16-P and L16-P, both with a stroke of 10 cm) with position feedback, a 60 ml syringe, a sliding mass, and a waterproof servo (Hitec HS-646WP). The servo controls the tail angle δ, one linear actuator controls the position of a slide mass, and the other linear actuator along with the syringe controls the net buoyancy. The body of the Miniglider is constructed from a BlueRobotics 4-inch series enclosure. It also features a 3D-printed wing mount, a set of wings, a servo mount, and a tail, all with a water-resistant coating.

The power electronics consists of a Tenergy 9.6V NiMH battery directly driving the linear actuators through an STMicroelectronics L298 motor driver and a custom PCB with 5V and 7V switching regulator circuits. The control electronics consist of a Raspberry Pi 4 as the main com-



(a)



(b)

Figure 2.5: Pictures of fleet of Minigliders (a) and SolidWorks design (b) revealing internal mechanical structure.

puter, an RF Xbee module for communication, a custom Raspberry Pi shield containing the motor driver, a Microchip EMC1701 battery monitor, a DsPic30F6014A microcontroller used to control the actuators and read analog sensors, and interfaces for various sensors. Onboard sensors include an LSM9DS1 IMU, a BlueRobotics Bar30 pressure sensor, and a Raspberry Pi Camera V1. Additional sensors such as GPS can be added through the Raspberry Pi USB ports and analog sensor interfaces on the custom shield. The Miniglider software architecture mirrors that of GRACE 3 to encourage testing GRACE 3 algorithms on Miniglider and enable easy transition of algorithms tested on the Miniglider to GRACE 3.

2.2.2 Experimental Setup

The experimental setup for Miniglider includes a large indoor tank measuring 4.6 m long, 3.1 m wide, and 1.2 m deep indoor tank equipped with several 15 cm by 15 cm AprilTags [91, 92] and three overhead webcams. This setup is pictured in Fig. 2.6. The AprilTags were made from Trotech $\frac{1}{16}$ inch Laserable Plastics. The pose T_{tag}^{W} (location and orientation) of each AprilTag was measured with respect to a fixed point, so their relative positions could be used to localize the Miniglider robot using its onboard camera. The known AprilTag poses are used to calculate the pose T_{MG}^{W} of the Miniglider with the formula

$$T_{MG}^{W} = T_{tag}^{W}(T_{tag}^{cam})^{-1}T_{MG}^{cam}$$

in real-time based on the measured relative pose T_{tag}^{cam} of the individual AprilTags with respect to the camera. The poses $T \in SE(3)$ are homogeneous transformation matrices. A similar process was used to estimate the position of the overhead cameras, relative to one another. This enables localization of the Miniglider, consistently across the three cameras, by attaching AprilTags to it.

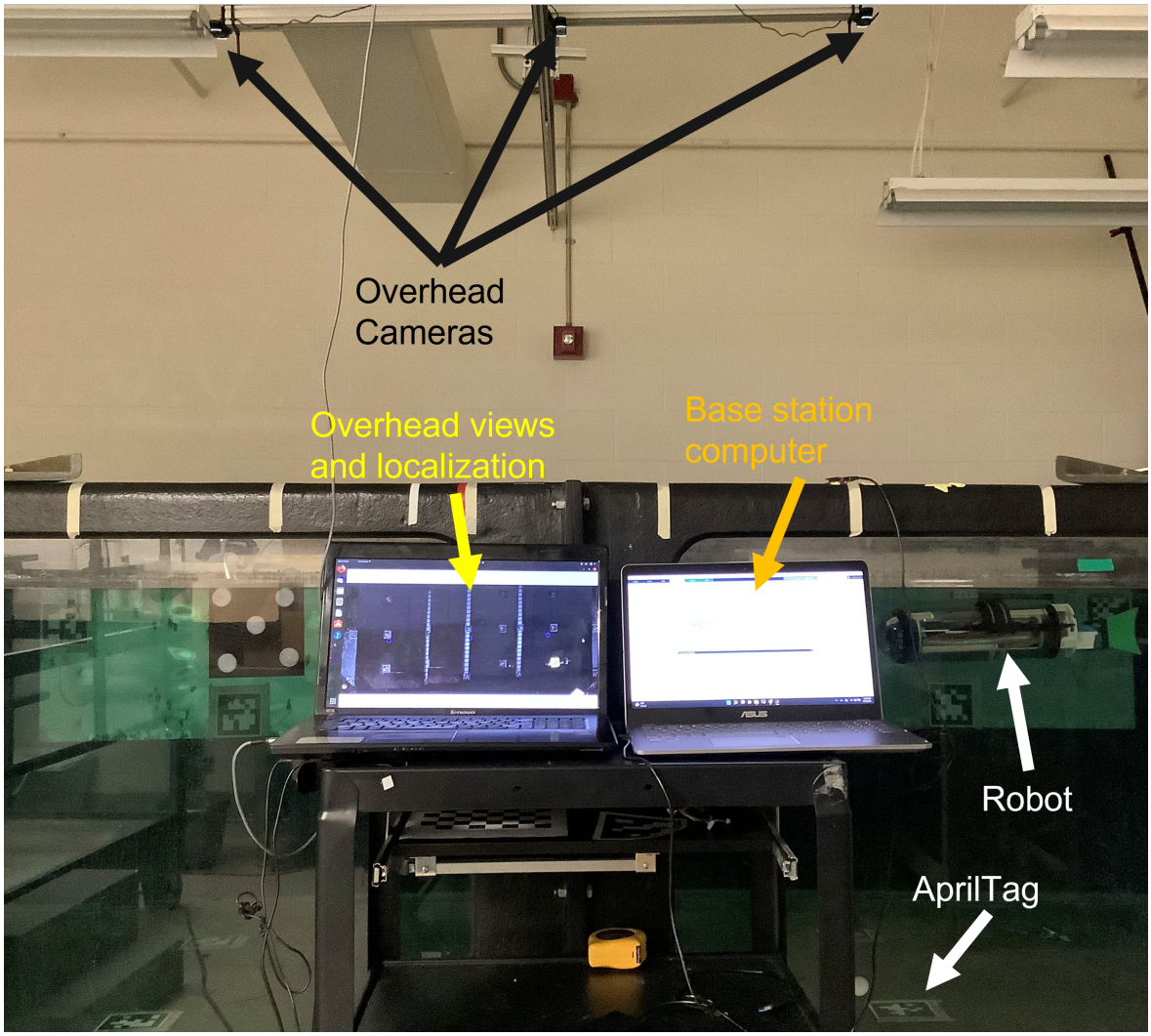


Figure 2.6: Miniglider robot operating in a large indoor tank during experiment. Tank-mounted AprilTags and onboard camera or robot-mounted AprilTags and overhead cameras can be used to localize the robot.

2.3 Summary and Future Improvements

Previous iterations of the gilding robotic fish platform and the upgrades that have been made as of this work have been discussed. Additional capabilities such as Iridium satellite communication and acoustic communication have been added, mechanical issues were addressed, and a major change in the design approach was taken that caters to rapid prototyping and testing. In addition, a miniature gliding robotic fish was developed along with an experimental testbed to allow testing in a controlled environment. While the current design does allow faster prototyping, testing, and more modularity with respect to the sensor payload, there are still areas of improvement that can be addressed. In fact, a sub-version GRACE 3.1 is already under development to address a few of the issues. The first is that the PVC design, while cheap and disposable, was permanent once assembled and could still take considerable time to be constructed. While the design philosophy encourages backup modules to be available, it is still of interest to have the actuator modules not be permanent or custom-built. 3.1 is exploring using BlueRobotics enclosures to house the actuators and batteries. This will also lead to a slimmer, more streamlined body profile as opposed to 3.0. Second, the master computer will be upgraded to the Raspberry Pi 4 with a custom shield to interface with some of the sensors, as it has proven to be space efficient and give a significant increase in computational power for Miniglider.

Other improvements, left for the future, should be easier to incorporate with the current design than previous ones. These include the implementation of solar charging for these robots, incorporating leak detection sensors, adding a quick release for the static weight to aid in emergency surfacing, adding a water conductivity sensor, adding a scanning sonar or beam sonar, and adding one or more cameras. It may also be of interest to purchase pre-made power electronics to speed development and reduce cost. All of the suggested improvements will increase the ability of the

robot to act autonomously and the complexity of tasks it can carry out. Particularly, the introduction of a camera and sonars will be useful for furthering applications in fish tracking and giving the robot the ability to perform obstacle avoidance and bottom detection when near the floor of a body of water.

Chapter 3

Backstepping Control of Gliding Robotic Fish for Pitch and 3D Trajectory Tracking

In this chapter, a backstepping-based trajectory-tracking controller is proposed for gliding-like trajectories using the gliding robotic fish. Section 3.1 describes the system model and the problem formulation. In Section 3.2, the controller is designed using three control inputs to simultaneously track the pitch angle. Then an intuition is then given to explain why the proposed controller is able to achieve tracking of all four components of the reference trajectories (pitch and 3D position) before rigorously demonstrating how the proposed scheme achieves convergence of the position and pitch angle tracking errors using two-time-scale analysis of singularly perturbed systems. In, Sections 3.3 and 3.4, simulation studies and experimental results are presented to show the efficacy of the proposed control design, and a procedure for estimating crucial model parameters is presented and employed for the miniature gliding robotic fish. A model-based observer is also implemented to estimate the body-fixed velocities, which are otherwise not directly accessible from onboard sensors.

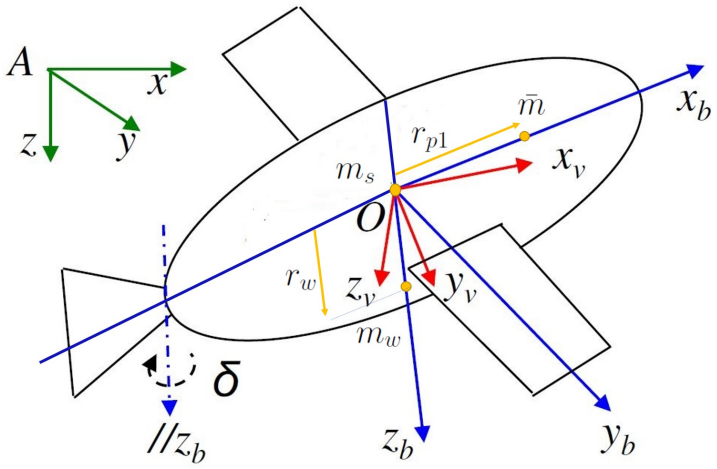


Figure 3.1: Illustration of robot reference frames and mass distribution.

3.1 System Modeling and Problem Formulation

3.1.1 Gliding Robotic Fish Model

The robot has two relevant reference frames shown in Fig. 3.1. The first is the inertial frame, represented by A_{xyz}. The origin A is a fixed point in space with an axis A_z along the direction of gravity and axes A_x and A_y defined in the horizontal plane. The body-fixed frame is denoted by $O_{x_b y_b z_b}$, with the origin O at the geometric center of the glider body, axis O_{x_b} along the body longitudinal axis pointing toward the robot's front, axis O_{z_b} perpendicular to the O_{x_b} axis in the sagittal plane of the robot pointing towards bottom of the robot, and axis O_{y_b} formed according to the right-hand orthonormal principle with respect to O_{x_b} and O_{z_b}. The glider is modeled as a 6-degree-of-freedom (DOF) rigid body with an internal moving mass, a water tank, and a servo-actuated tail that has its own axis of rotation parallel to the robot's O_{z_b} axis, at an offset along the O_{x_b} axis. While the tail can be used for both propulsion and steering, this work only focuses on its steering capability. The internal movable mass is restricted to the longitudinal axis by a linear actuator and has significant influence over the robot's pitch angle. Lastly, a linear actuator-

driven syringe pump controls the negative net buoyancy, which is given as the sum of the mass m_s (including mass of water in tank) not contributing to the moment and located at O, the internal movable mass $\bar{m}$, and the non-uniformly distributed mass m_w minus the mass m of the water displaced by the robot. This can be expressed as $m_0 = m_s + \bar{m} + m_w - m$, where $m_0 < 0$ causes the robot to float and $m_0 > 0$ causes the robot to sink. The robot essentially controls m_0 by changing the amount of water in the tank. In summary, the control inputs include the negative net buoyancy m_0, the distance r_{p1} of the movable mass from the body-frame origin, and the tail angle δ.

The state vector consists of the position $b_i = [x, y, z]^T$ of the robot, the orientation with respect to the inertial frame, and the body-fixed linear velocities $v_b = [v_1, v_2, v_3]^T$ and body-fixed angular velocities $\omega_b = [\omega_1, \omega_2, \omega_3]^T$. With the orientation represented by Euler angles (roll, pitch, and yaw), $\Psi = [\phi, \theta, \psi]^T$, the state vector can be written as

$$X = [x, y, z, \phi, \theta, \psi, v_1, v_2, v_3, \omega_1, \omega_2, \omega_3]^T. \tag{3.1}$$

The dynamic equations are

$$\begin{cases} \dot{b}_i = R v_b \\ \dot{\Psi} = R_\omega \omega_b \\ \dot{v}_b = M^{-1}((M v_b) \times \omega_b + m_0 g R^T k + F_{ext}) \\ \dot{\omega}_b = J^{-1}(-\dot{J}\omega_b + (J\omega_b) \times \omega_b + (M v_b) \times v_b + T_{ext} \\ \quad + m_w g r_w \times (R^T k) + \bar{m} g r_p \times (R^T k)) \end{cases} \tag{3.2}$$

where R, written as

$$R = \begin{pmatrix} c_\theta c_\psi & c_\psi s_\theta s_\phi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta s_\psi & c_\phi c_\psi + s_\theta s_\phi s_\psi & c_\phi s_\theta s_\psi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix},$$

is a 3×3 rotation matrix parameterized by the Euler angles $\Psi = [\phi, \theta, \psi]^T$ following the ZYX convention. c_q and s_q with $q = \phi, \theta, \psi$ represent sine and cosine of the variable in the subscript. R_ω, written as

$$R_\omega = \begin{pmatrix} 1 & \tan(\theta)\sin(\phi) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{pmatrix}$$

is a 3×3 matrix that relates the body-fixed angular velocities to Euler angle rates.

$M = \text{diag}\{m_1, m_2, m_3\}$ is the added mass matrix incorporating the effect of the surrounding fluid, g is Earth's gravitational constant, $k = [0,0,1]^T$, and $F_{ext} = R_{bv}[-D, F_s, -L]^T$ is the hydrodynamic force vector. $J = \text{diag}\{J_1, J_2, J_3\}$ is the added inertia matrix, $T_{ext} = R_{bv}[M_1, M_2, M_3]^T$ is the hydrodynamic moment vector, $r_p = [0,0,r_{p1}]^T$, and $r_w = [0,0,r_{w3}]^T$ is the position of the center of gravity of the non-uniformly distributed mass m_w. R_{bv} is a 3×3 rotation matrix parameterized by the angle of attack $\alpha = \arctan\frac{v_3}{v_1}$ and the side-slip angle $\beta = \arcsin\frac{v_2}{\sqrt{v_1^2+v_2^2+v_3^2}}$ that maps the hydrodynamic forces and moments from the velocity reference frame to the body-fixed frame. It is given by [5]

$$R_{bv} = \begin{pmatrix} \cos(\alpha)\cos(\beta) & -\cos(\alpha)\sin(\beta) & -\sin(\alpha) \\ \sin(\beta) & \cos(\beta) & 0 \\ \sin(\alpha)\cos(\beta) & -\sin(\alpha)\sin(\beta) & \cos(\alpha) \end{pmatrix} \tag{3.3}$$

The hydrodynamic forces and torques, including lift L, drag D, side force F_s, roll moment M_1,

pitch moment M_2, and yaw moment M_3, are given as [5]

$$\begin{cases} D = \frac{1}{2}\rho V^2 S(C_{D0} + C_D^{\alpha}\alpha^2 + C_D^{\delta}\delta^2) \\ F_s = \frac{1}{2}\rho V^2 S(C_{F_S}^{\beta}\beta + C_{F_S}^{\delta}\delta) \\ L = \frac{1}{2}\rho V^2 S(C_{L0} + C_L^{\alpha}\alpha) \\ M_1 = \frac{1}{2}\rho V^2 S(C_{M_R}^{\beta}\beta + K_{q1}\omega_1) \\ M_2 = \frac{1}{2}\rho V^2 S(C_{M_0} + C_{M_P}^{\alpha}\alpha + K_{q2}\omega_2) \\ M_3 = \frac{1}{2}\rho V^2 S(C_{M_Y}^{\beta}\beta + K_{q3}\omega_3 + C_{M_Y}^{\delta}\delta) \end{cases} \tag{3.4}$$

where the parameters associated with K_q and C notations are hydrodynamic constants, ρ is the fluid density, S is the characteristic surface area of the robot, and V is the magnitude of v_b.

For convenience, the linear and angular velocity dynamics are abstracted as

$$\begin{bmatrix} \dot{v_1} \\ \dot{v_2} \\ \dot{v_3} \\ \dot{\omega_1} \\ \dot{\omega_2} \\ \dot{\omega_3} \end{bmatrix} = \begin{bmatrix} f_{v11} + a_{v1} r_{31} u_1 + f_{v12} u_3 + f_{v13} u_3^2 \\ f_{v21} + a_{v2} r_{32} u_1 + f_{v22} u_3 + f_{v23} u_3^2 \\ f_{v31} + a_{v3} r_{33} u_1 + f_{v32} u_3 + f_{v33} u_3^2 \\ f_{\omega 11} + f_{\omega 12} u_3 \\ f_{\omega 21} + a_{\omega_2} r_{33} u_2 \\ f_{\omega 31} + a_{\omega_2} r_{32} u_2 + f_{\omega 32} u_3 \end{bmatrix} \tag{3.5}$$

where $u_1 = m_0$, $u_2 = r_{p1}$, and $u_3 = \delta$ are the controls, a_{vi} $(i = 1,2,3)$ and a_{ω_2} are constants, r_{3j} $(j = 1,2,3)$ are corresponding elements of R, and f_{vij} $(i,j = 1,2,3)$ and $f_{\omega ij}$ $(i,j = 1,2,3$, when present) are the corresponding nonlinear functions of the state vector.

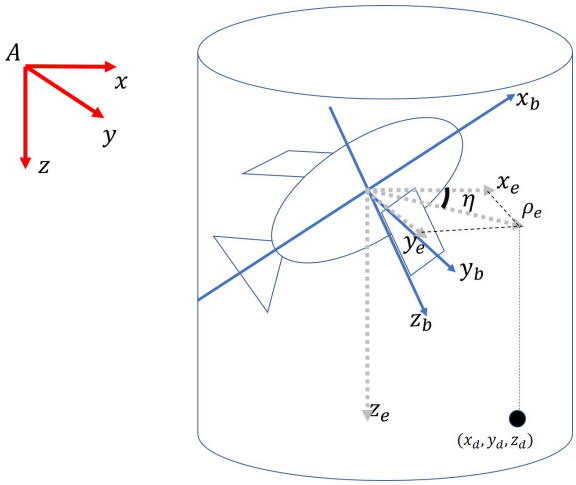


Figure 3.2: Illustration of the robot error frame. A is the inertial frame and the point (x_d, y_d, z_d) is the desired position for the robot. The position error vector (x_e, y_e, z_e) is the difference between the desired position and the center of the robot. The axes x_b, y_b, and z_b represent the body-fixed coordinate frame.

3.1.2 Problem Formulation

The problem of trajectory tracking involves controlling a robot to follow a time-dependent path.In this work, the aim is to have the robot pose $P = [x, y, z, \theta]^T$, consisting of the 3D position and the pitch angle θ, follow a trajectory in the inertial coordinate system. The desired path is given by $P_d(t) = [x_d(t), y_d(t), z_d(t), \theta_d(t)]^T$. $\dot{P}_d(t)$ and $\ddot{P}_d(t)$ are assumed to be bounded and sufficiently smooth with $|\theta_d| < \frac{\pi}{2}$. It is also assumed that $P_d(t)$ is dynamically feasible, which means that it is achievable given the constraints of the robot dynamics and control inputs. To solve this tracking problem, the inertial frame error $P_e(t) = [x_e, y_e, z_e, \theta_e]^T$ is defined as

$$P_e(t) = \begin{bmatrix} x_d - x \\ y_d - y \\ z_d - z \\ \theta_d - \theta \end{bmatrix} \tag{3.6}$$

and each error is regulated to zero. The error vector has four variables to be regulated, while the system has only three control inputs. This is handled by writing the error vector in a form that reduces the number of errors that need to be regulated to 0. According to Fig. 3.2, the Cartesian errors can be rewritten in the cylindrical coordinate system. This can be done by representing the position error vector (x_e, y_e) in the plane by its magnitude $\rho_e = \sqrt{x_e^2 + y_e^2}$ and angle $\eta = \arctan(y_e, x_e)$ suitably defined to give the correct quadrant. The vector is expressed in the inertial frame A, but attached to the origin of the robot's body-fixed frame $O_{x_b y_b z_b}$. The cylindrical representation of the error vector becomes $P_e^c(t) = [\rho_e, \psi_e, z_e, \theta_e]^T$ where $\psi_e = \eta - \psi$, denotes the difference between the direction of the planar tracking error vector, η , and the yaw angle ψ. When $\psi_e = 0$, the robot will point in the direction of fastest reduction of the planar tracking error. Regulating ρ_e, z_e, and θ_e to zero is equivalent to regulating P_e to zero.

3.2 Backsteping-based Control Design

3.2.1 Overview of Control Design

To handle the under-actuated nature of the robot, inspiration is taken from Do and Pan's work [66]. These authors used insight from how a ship helmsman steers a boat to minimize lateral position error, to design a controller for underactuated ships; in particular, minimizing the heading error could be temporarily sacrificed to minimize the position error. Following a similar logic, this work takes advantage of the natural motion of the gliding robotic fish to minimize the planar position error while temporarily sacrificing pitch tracking. To do this, an error function

$$\xi = \theta_e - c f_{\xi 1}(\theta_g) f_{\xi 2}(\rho_e \cos(\psi_e)) \tag{3.7}$$

is introduced, where $\theta_g = \theta - \alpha$ is the glide angle, α is the angle of attack defined in Section 3.1 $f_{\xi 1}$ and $f_{\xi 2}$ are bounded odd, increasing functions satisfying $f_{\xi 1}(0) = 0$, $f_{\xi 2}(0) = 0$, and $c > 0$ is a constant satisfying $c\bar{b}_1\bar{b}_2 < \pi/2$, where $\bar{b}_1$ and $\bar{b}_2$ be the upper bounds of $|f_{\xi_1}|$ and $|f_{\xi_2}|$, respectively. These conditions are satisfied by the choice

$$\xi = \theta_e - c\tanh(\theta_g)\tanh(\rho_e\cos(\psi_e)) \tag{3.8}$$

with $c < \pi/2$. Eq. (3.8) is adopted in the simulation and experiments in this work. With the error function ξ, the modified tracking error vector $P_{e_a} = [z_e, \psi_e, \xi]^T$, is defined and will be used in the backstepping control design. The derivative of the error vector P_{e_a} can be expressed in terms of the state variables. In particular, with $x_e = \rho_e\cos\eta$ and $y_e = \rho_e\sin\eta$, one can derive

$$\begin{cases} \dot{\rho}_e = \cos(\eta)\dot{x}_e + \sin(\eta)\dot{y}_e \\ \dot{\eta} = \dfrac{1}{\rho_e}(\cos(\eta)\dot{y}_e - \sin(\eta)\dot{x}_e) \end{cases} \tag{3.9}$$

which will be useful in computing $\dot{\xi}$ and $\dot{\psi}_e$ later.

With the above formulation, trajectory tracking becomes a stabilization problem with respect to the error vector. The control objective is now to drive the modified error vector P_{e_a} to the origin. Later it is discussed how the convergence of P_{e_a} to zero implies the convergence of all elements of the original tracking error vector P_e to a neighborhood of zero. To drive P_e to zero, the variables $\zeta_1 = \dot{z}_e$, $\zeta_2 = \dot{\psi}_e$, $\zeta_1 = \dot{\xi}$ are defined. The physical control inputs appear in the derivatives of ζ_i. They need to be chosen to render $(z_e, \psi_e, \xi, \zeta_1, \zeta_2, \zeta_3)$ convergent to zero. This can be done by making $\dot{\zeta}_1 = -k_1\zeta_1 - k_z z_e$, $\dot{\zeta}_2 = -k_2\zeta_2 - k_\psi\psi_e$, and $\dot{\zeta}_3 = -k_3\zeta_3 - k_\xi\xi$, where k_z, k_ψ, k_ξ, k_1,k_2,

and k_3 are positive constants to be chosen. The system can be rewritten in a block diagonal form:

$$\begin{bmatrix} \dot{z}_e \\ \dot{\zeta}_1 \\ \dot{\psi}_e \\ \dot{\zeta}_2 \\ \dot{\xi} \\ \dot{\zeta}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -k_z & -k_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -k_\psi & -k_2 & 0 & \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -k_\xi & k_3 \end{bmatrix} \begin{bmatrix} z_e \\ \zeta_1 \\ \psi_e \\ \zeta_2 \\ \xi \\ \zeta_3 \end{bmatrix} \tag{3.10}$$

It can be easily shown that the eigenvalues of the linear system (3.10) are pairs of the form $-\frac{k_i}{2} \pm \frac{\sqrt{{k_i}^2 - 4k_a}}{2}$ for $i = 1,2,3$ and $a = z, \psi, \xi$. These all have negative real parts as long as the gains $k_1, k_2, k_3, k_z, k_\psi, k_\xi$ are positive, implying that the state $(z_e, \psi_e, \xi, \zeta_1, \zeta_2, \zeta_3)$ is asymptotically stable.

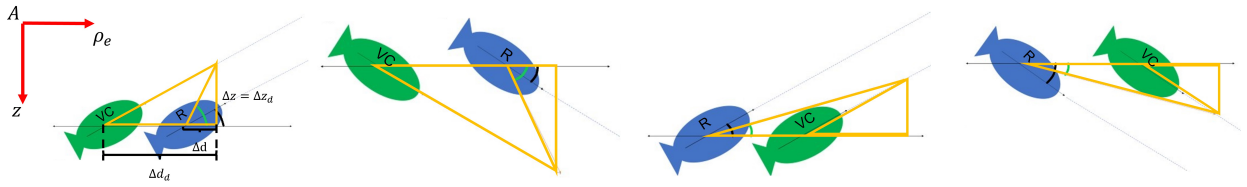
The above analysis enables us to choose inputs u_1, u_2, and u_3 to ensure the convergence of $(z_e, \psi_e, \xi, \zeta_1, \zeta_2, \zeta_3)$ to zero. The equations $\dot{\zeta}_i$ can be rewritten as

$$\dot{\zeta}_1 = f_{11}u_1 + f_{12}u_2 + f_{13}u_3 + f_{14}u_3^2 + f_{15} = -k_1\zeta_1 - k_z z_e$$

$$\dot{\zeta}_2 = f_{21}u_1 + f_{22}u_2 + f_{23}u_3 + f_{24}u_3^2 + f_{25} = -k_2\zeta_2 - k_\psi \psi_e$$

$$\dot{\zeta}_3 = f_{31}u_1 + f_{32}u_2 + f_{33}u_3 + f_{34}u_3^2 + f_{35} = -k_3\zeta_3 - k_\xi \xi$$

where $f_{ij} = \frac{\partial \dot{\zeta}_i}{\partial u_j}$ for $i = 1,2,3,\ j = 1,\cdots,4$ and $f_{i5} = \dot{\zeta}_i - \sum_{j=1}^{4}(u_j f_{ij})$ with $u_4 = u_3^2$. These equations give us the means to solve for the inputs such that the desired values of $\dot{\zeta}_i$ are achieved. The equations can be written in a matrix form to solve for the control inputs as follows (where f_{12},



(a) (b) (c) (d)

Figure 3.3: Illustration of the desired behavior for the robot (R), when tracking trajectory is given by a virtual copy (VC) gliding in a plane for four different cases. Black angle marker represents $\theta = \theta_d$ and green angle marker represents $\theta = \theta_d + cf_{\xi 1}(\theta_g) f_{\xi 2}(\rho_e \cos(\psi_e))$. Δz and Δd represent the vertical travel and horizontal travel, respectively, of the robot when $\theta = \theta_d + cf_{\xi 1}(\theta_g) f_{\xi 2}(\rho_e \cos(\psi_e))$, while Δz_d and Δd_d represent the vertical travel and horizontal travel, respectively, of the virtual copy.

f_{31}, and f_{34} are zero):

$$\begin{bmatrix} f_{11} & 0 & f_{13} & f_{14} \\ f_{21} & f_{22} & f_{23} & f_{24} \\ 0 & f_{32} & f_{33} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_3^2 \end{bmatrix} = \begin{bmatrix} \Gamma_1 \\ \Gamma_2 \\ \Gamma_3 \end{bmatrix} \tag{3.11}$$

where $\Gamma_1 = -f_{15} - k_1\zeta_1 - k_z z_e$, $\Gamma_2 = -f_{25} - k_2\zeta_2 - k_\psi \psi_e$, and $\Gamma_3 = -f_{35} - k_3\zeta_3 - k_\xi \xi$. An approach to solving (3.11) for the control is further discussed in Section 3.3.

3.2.2 Analysis of the Closed-Loop System

Under the control law (3.11), one can guarantee that ξ, ψ_e, and z_e all approach zero. Note that the original tracking goal is for all errors of P_e in Eq. (4.11) to approach zero. The error system after

implementing the control law is given by

$$\begin{bmatrix} \dot{z}_e \\ \dot{\psi}_e \\ \dot{\xi} \\ \dot{\rho}_e \\ \dot{\zeta}_1 \\ \dot{\zeta}_2 \\ \dot{\zeta}_3 \end{bmatrix} = \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ \cos(\eta)\dot{x}_e + \sin(\eta)\dot{y}_e \\ -k_1\zeta_1 - k_z z_e \\ -k_2\zeta_2 - k_\psi \psi_e \\ -k_3\zeta_3 - k_\xi \xi \end{bmatrix} \tag{3.12}$$

with $\theta_e = \xi + cf_{\xi 1}(\theta_g) f_{\xi 2}(\rho_e \cos(\psi_e))$.

An intuition for why $P_{e_a} = 0$ implies that P_e approaches 0 can be gleaned from a geometric perspective. Similar to flight kinematics, glider kinematics can be expressed in terms of the velocity magnitude and a glide angle θ_g [5, 12, 13]. The glide angle is defined as $\theta_g = \theta - \alpha$ and can be approximated by $\frac{\Delta z}{\Delta d}$ for a constant glide angle, where Δd and Δz are the horizontal and vertical distances traveled in a given amount of time, respectively. Using this approximation as a basis, Fig. 3.3 gives an intuitive understanding of what the controller is designed to accomplish. It illustrates idealized scenarios where a robot modifies its glide angle to minimize the horizontal tracking error. For instance, Fig. 3.3(a) shows the gliding robotic fish ahead of the desired position and at the desired pitch angle. If it maintains this pitch and depth rate, the distance to the desired position will remain constant. If it increases the pitch while maintaining the depth rate, the distance to the desired position decreases. In practice, α is often small compared to θ, making $\theta_g \approx \theta$. This means that, in essence, perturbing the pitch angle effectively changes the glide path, which slows or speeds up horizontal travel (for a given vertical travel speed), thus enabling the robot to catch

up to the desired trajectory in the planar position.

Now, multi-time-scale analysis of singularly perturbed systems is used to prove this rigorously and show how the other error states converge to a neighborhood of the origin. For the ease of presentation, it is assumed $\phi = 0$ (in practice the roll ϕ is close to zero). Multi-time-scale analysis of singularly perturbed systems [25, 93, 94] is used as a tool for model reduction through a small parameter ε. It separates the model of a system into multiple time scales, allowing for analysis of the complete system to be broken down into analysis of reduced models and their interconnections. A brief overview of two-time-scale analysis is given in Appendix A. Following the time-scale analysis framework, the system in Eq. (3.12) can be rewritten as a two-time-scale system as elaborated next.

A natural separation of the time-scales is between $X_1 = \rho_e$ (and θ_e) and the states controlled with the backstepping design in Section 3.2.1, $X_2 = [z_e, \psi_e, \xi, \zeta_1, \zeta_2, \zeta_3]^T$. For the analysis below, θ_e is not included separately due to its algebraic relationship with ξ, ψ_e, and ρ_e.

One can show that as the gains $k_1, k_2, k_3, k_z, k_\psi, k_\xi$ in (3.12) get larger, the dynamics for X_2 gets faster. To see this, consider gains of the following relationships (inspired by the form of the eigenvalues from the subsystems in (3.10)): $k_z = c_1 k_1^2$, $k_\psi = c_2 k_2^2$, and $k_\xi = c_3 k_3^2$, for some $c_1 > 0$, $c_2 > 0$, $c_3 > 0$. To scale the gains, let $k_i = \frac{k_i^0}{\varepsilon}$, for a given nominal value $k_i^0 > 0$, $i = 1, 2, 3$. The

dynamics (3.12) can then be represented in the following two-time-scale system:

$$\dot{X}_1 = f(t,X_1,X_2) \stackrel{\triangle}{=} \cos(\eta)\dot{x}_e + \sin(\eta)\dot{y}_e \tag{3.13}$$

$$\varepsilon\dot{X}_2 = g(X_2,\varepsilon) \stackrel{\triangle}{=} \begin{bmatrix} \varepsilon\zeta_1 \\ \varepsilon\zeta_2 \\ \varepsilon\zeta_3 \\ -k_1^0\zeta_1 - \frac{c_1(k_1^0)^2}{\varepsilon} z_e \\ -k_2^0\zeta_2 - \frac{c_2(k_2^0)^2}{\varepsilon}\psi_e \\ -k_3^0\zeta_3 - \frac{c_3(k_3^0)^2}{\varepsilon}\xi \end{bmatrix} \tag{3.14}$$

It can be shown that the eigenvalues of the dynamics for X_2 are $\frac{(-1\pm\sqrt{1-4c_i})k_i^0}{2\varepsilon}$ for $i = 1,2,3$. Therefore, the rate of dynamics of X_2 will scale with $1/\varepsilon$.

Theorem 1. *Consider the system (3.13) and (3.14). With the control law (3.11), it can be shown that there exists an ε^* such that the system (3.12) is uniformly ultimately bounded within a neighborhood around the origin for all $\varepsilon < \varepsilon^*$, and $P_{e_a}(t) \to 0$ implies $P_e(t)$ will converge to a bounded region around the origin.*

Sketch of Proof. The proof follows Theorem 2 in Appendix A to prove the claims. In particular, Assumptions 5—9 are satisfied for the two-time-scale-system (3.13) and (3.14). Assumption 5 requires that the origin $(X_1 = 0, X_2 = 0)$ is an isolated equilibrium point, and there exists a function $X_2 = h(t,X_1)$ such that $g(h(t,X_1),0) = 0$, and a class κ function κ_ρ such that $||h(t,X_1)|| \leq \kappa_\rho(||X_1||)$. Assumption 6 entails finding a Lyapunov function for the reduced system $\dot{X}_1 = f(t,X_1,h(t,X_1))$. To verify A.1 and A.2, note that for any X_1, $X_2 = h(t,X_1) = [0,0,0,0,0,0]^T$ is a unique root of $g(X_2,\varepsilon) = 0$. The dynamic equation of X_1 is then analyzed under the constraints $\zeta_1 = \zeta_2 = \zeta_3 = 0$, $z_e = 0$, $\psi_e = 0$, and $\xi = 0$, as shown next.

First, the identity $v_b = R_{bv}[V,0,0]^T$ is used to express the body-fixed linear velocities in terms of the velocity magnitude V, angle of attack α, and side-slip angle β. Using this identity, the derivatives of the position can be expressed as $\dot{b}_i = RR_{bv}[V,0,0]^T$, resulting in

$$\dot{b}_i = \begin{pmatrix} V\cos(\beta)\cos(\psi)\cos(\theta-\alpha) - V\sin(\beta)\sin(\psi) \\ V\cos(\beta)\sin(\psi)\cos(\theta-\alpha) + V\sin(\beta)\cos(\psi) \\ -V\sin(\theta-\alpha)\cos(\beta) \end{pmatrix} \tag{3.15}$$

Per the assumption on the dynamic feasibility of the reference trajectory $P_d(t)$, one can derive the desired pitch angle θ_d, yaw angle ψ_d, angle of attack α_d, side slip angle β_d, and velocity magnitude V_d for a reference robot from $P_d(t)$, with $|\theta_d - \alpha_d| < \pi/2$. In particular, one could choose ψ_d to be consistent with the projection of the desired velocity on the horizontal plane, making $\beta_d = 0$. Expressing $\dot{x}_e$ and $\dot{y}_e$ using Eq. (3.15) and plugging them into $\dot{\rho}_e$, results in

$$\dot{\rho}_e = \cos(\eta-\psi_d)V_d\cos(\theta_d-\alpha_d) - \cos(\psi_e)V\cos(\beta)\cos(\theta-\alpha) - \sin(\psi_e)V\sin(\beta) \tag{3.16}$$

From Eq. (3.15), $\dot{z}_e = -V_d\sin(\theta_d-\alpha_d) - (-V\sin(\theta-\alpha)\cos(\beta))$, implying $V = \frac{\zeta_1 + V_d\sin(\theta_d-\alpha_d)}{\sin(\theta-\alpha)\cos(\beta)}$. Substituting this equation for V into $\dot{\rho}_e$ yields

$$\begin{aligned}\dot{\rho}_e = {} & \cos(\eta-\psi_d)V_d\cos(\theta_d-\alpha_d) - \cos(\psi_e)\frac{\zeta_1 + V_d\sin(\theta_d-\alpha_d)}{\sin(\theta-\alpha)}\cos(\theta-\alpha) \\ & - \sin(\psi_e)\frac{\zeta_1 + V_d\sin(\theta_d-\alpha_d)}{\sin(\theta-\alpha)}\tan(\beta)\end{aligned}$$

The dynamics of ρ_e for the reduced system, with $\psi_e = 0$ and $\zeta_1 = 0$, can now be written as

$$\dot{\rho}_e = \cos(\psi-\psi_d)V_d\cos(\theta_d-\alpha_d) - \frac{V_d\sin(\theta_d-\alpha_d)}{\sin(\theta-\alpha)}\cos(\theta-\alpha)$$

which can be expressed as

$$\dot{\rho}_e = -V_d(\frac{\sin(\theta_e - \alpha_e)}{\sin(\theta - \alpha)}) + V_d(\cos(\psi - \psi_d) - 1)\cos(\theta_d - \alpha_d) \tag{3.17}$$

with $\alpha_e = \alpha_d - \alpha$. The second term in Eq. (3.17) is bounded between $-2V_d$ and 0. One can add and subtract $V_d \sin\theta_e$ to the numerator of the first term in Eq. (3.17). Then, using the constraint $\theta_e = cf_{\xi 1}(\theta_g)f_{\xi 2}(\rho_e)$ when $\xi = 0$, it can be shown that

$$\dot{\rho}_e = -V_d\left(\frac{\sin(cf_{\xi 1}(\theta_g)f_{\xi 2}(\rho_e))}{\sin(\theta_g)}\right) + \eta_0(t) + V_d(\cos(\psi - \psi_d) - 1)\cos(\theta_d - \alpha_d) \tag{3.18}$$

where $\eta_0(t) = -V_d\frac{2\cos(\frac{2\theta_e - \alpha_e}{2})\sin(\frac{\alpha_e}{2})}{\sin(\theta_g)}$ is considered a perturbation. The case where the perturbation $\eta_0 = 0$ (the case $\eta_0 \neq 0$ will be dealt with afterward) can now be analyzed. It can be seen that the sign of the first term in Eq. (3.18) is negative for a nonzero pitch angle unless $\rho_e = 0$, in which case the term becomes 0 since $f_{\xi 1}$ and f_{ξ_2} are bounded, odd functions, and $|cf_{\xi 1}(\cdot)f_{\xi 2}(\cdot)| < \frac{\pi}{2}$. In the case $\theta_g = 0$, L'Hopital's rule can be used to show that $\dot{\rho}_e \leq -V_d c f'_{\xi 1}(\theta_g)f_{\xi 2}(\rho_e) \leq 0$, with the last equality holding true only when $\rho_e = 0$, since the derivative of $f_{\xi 1}$ is positive. So the nominal reduced system for X_1 can be shown to satisfy Assumption 6 with the Lyapunov candidate $V_{ss} = \frac{1}{2}\rho_e^2$ and $\frac{\partial V_{ss}}{\partial X_1}f(t, X_1, h(t, X_1)) \leq -\gamma_0 a_1^2(\|X_1\|)$ for a positive constant γ_0, where $a_1(\cdot)$ is a class κ function. The existence of the Lyapunov function also implies that $\rho_e = 0$ is a stable equilibrium point of f satisfying the assumption 5.

Assumption 7 requires the existence of a Lyapunov function for the perturbation $X_2 - h(t, X_1)$. Because $h(t, X_1)$ is a zero vector, this is satisfied by the Lyapunov function $V_A = \frac{1}{2}X_2^T X_2$ and analysis from the backstepping design in Section 3.2.1.

Assumption 8 requires the growth of the difference between the system model $f(t, X_1, X_2)$ and

the reduced system model $f(t,X_1,h(t,X_1))$ to be bounded. Using Eq. (3.16), this can be satisfied with $\frac{\partial V_{ss}}{\partial X_1}[f(t,X_1,X_2)-f(t,X_1,h(X_1))] = -\rho_e(\cos(\psi_e)-1)V\cos(\theta-\alpha)\cos(\beta)-\rho_e\sin(\psi_e)V\sin(\beta) \leq \gamma_1 a_1(\|X_1\|)\|X_2\|$, for a sufficiently large constant γ_1.

Assumption 9 has to do with bounding the growth of the perturbation $g(X_2,\varepsilon)-g(h(t,X_1),0)$ and $\frac{\partial V_A}{\partial t}+\frac{\partial V_A}{\partial X_1}f(t,X_1,X_2)$. Since $\frac{\partial V_A}{\partial X_1}f(t,X_1,X_2)=0$ and V_A is independent of t, the second part of assumption 9 is satisfied. The first part can be satisfied using $\frac{\partial V_A}{\partial X_2}[g(X_2,\varepsilon)-g(h(t,X_1),0)] = X_2^T\varepsilon\dot{X}_2 \leq \varepsilon\gamma_3||X_2||^2$ for any non-negative constant γ_3 since $X_2^T\dot{X}_2$ is rendered negative definite by the controller (3.11).

Since conditions 5-9 are satisfied, a Lyapunov function for the system (3.13, and 3.14) can be constructed as

$$v(X_1,X_2) = (1-d)V_{ss}(X_1)+dV_A(X_2)$$

where $0<d<1$ and there exists an ε^* such that for all $\varepsilon\in(0,\varepsilon^*]$, the equilibrium $X_1=0, X_2=0$ is uniformly asymptotically stable.

To fulfill Assumptions 5 and 6 the previous analysis relied on η_0 being 0. To handle the case $\eta_0\neq 0$, one can use the theory for nonvanishing perturbations presented in [25] (Lemma 9.3). It can then be concluded that, when $\eta_0\neq 0$, the system (3.13) is ultimately bounded in a neighborhood around the origin. In particular, consider again the candidate Lyapunov function $V_{ss}(X_1)=\frac{1}{2}\rho_e^2$, but for the perturbed system. The derivative $\dot{V}_{ss}(X_1)$ along the the trajectory of $\dot{X}_1 = f(t,X_1,h(t,X_1))$ now satisfies

$$\begin{aligned}\dot{V}_{ss}(X_1) &\leq -a_1^2(||X_1||)+\|\frac{\partial V_{ss}}{\partial X_1}\|||\eta_0|| \\ &= -a_1^2(||X_1||)+\|X_1\|||\eta_0||\end{aligned}$$

Suppose η_0 satisfies the bound $||\eta_0||\leq\Delta$ for all $t\geq t_0$, $X_1\in D=\{X_1\in\mathbb{R}^1\,|\,||X_1||<r\}$ for some

$r > 0$. It can then be shown that $\dot{V}_{ss} < 0$ whenever $\|X_1\| > a_1^{-1}(\sqrt{r\Delta})$. In other words, $\|X_1\|$ is ultimately bounded by $a_1^{-1}(\sqrt{r\Delta})$. ■

3.3 Simulations

The backstepping controller proposed in this paper is compared against two baseline controllers operating on the errors z_e, ψ_e, and θ_e to show its effectiveness. The first is a PID controller and the second is another backstepping controller (which uses θ_e instead of the modified error ξ). Note that the baseline backstepping design is equivalent to the proposed design with $c = 0$, so the analysis from Section 3.2.1 applies and guarantees the convergence of the three aforementioned errors. The simulation is carried out using MathWorks Simulink and the model parameters used for simulation are based on those estimated for the physical system to be described in Section 3.4. Actuation is limited to the range of $[-27.9, 22.5]$ g for m_0, $[-1.92, 1.92]$ radians for δ, and $[-55.5, 44.5]$ mm for r_{p1}. Limits are also placed on the actuation rates for m_0, r_{p1}, and δ with $|\dot{m}_0| < 1.72\frac{\text{g}}{\text{s}}$, $|\dot{r}_{p1}| < 12.5\frac{\text{mm}}{\text{s}}$, and $|\dot{\delta}| < 1.05\frac{\text{rad}}{\text{s}}$.

To solve for the control (u_1, u_2, u_3) for the proposed method, Eq. (3.11) can be algebraically manipulated to produce a quadratic equation for u_3, with u_1 and u_2 expressed quadratic and linear, respectively, in u_3. In simulation and experiments in this work, control computation is further simplified by making mild assumptions of $\phi = 0$ and $\ddot{\eta} = 0$, both of which are reasonable given that ϕ and $\ddot{\eta}$ are close to zero under typical operating conditions of a gliding robotic fish. In addition, the angle of attack α may not be available for measurement in practice. But given $|\theta| >> \alpha$ during typical gliding operation, the glide angle $\theta_g = \theta - \alpha$ can be approximated by the pitch angle θ, as adopted in simulation and experiments in this work. With these assumptions, f_{21}, f_{22}, f_{24}, and

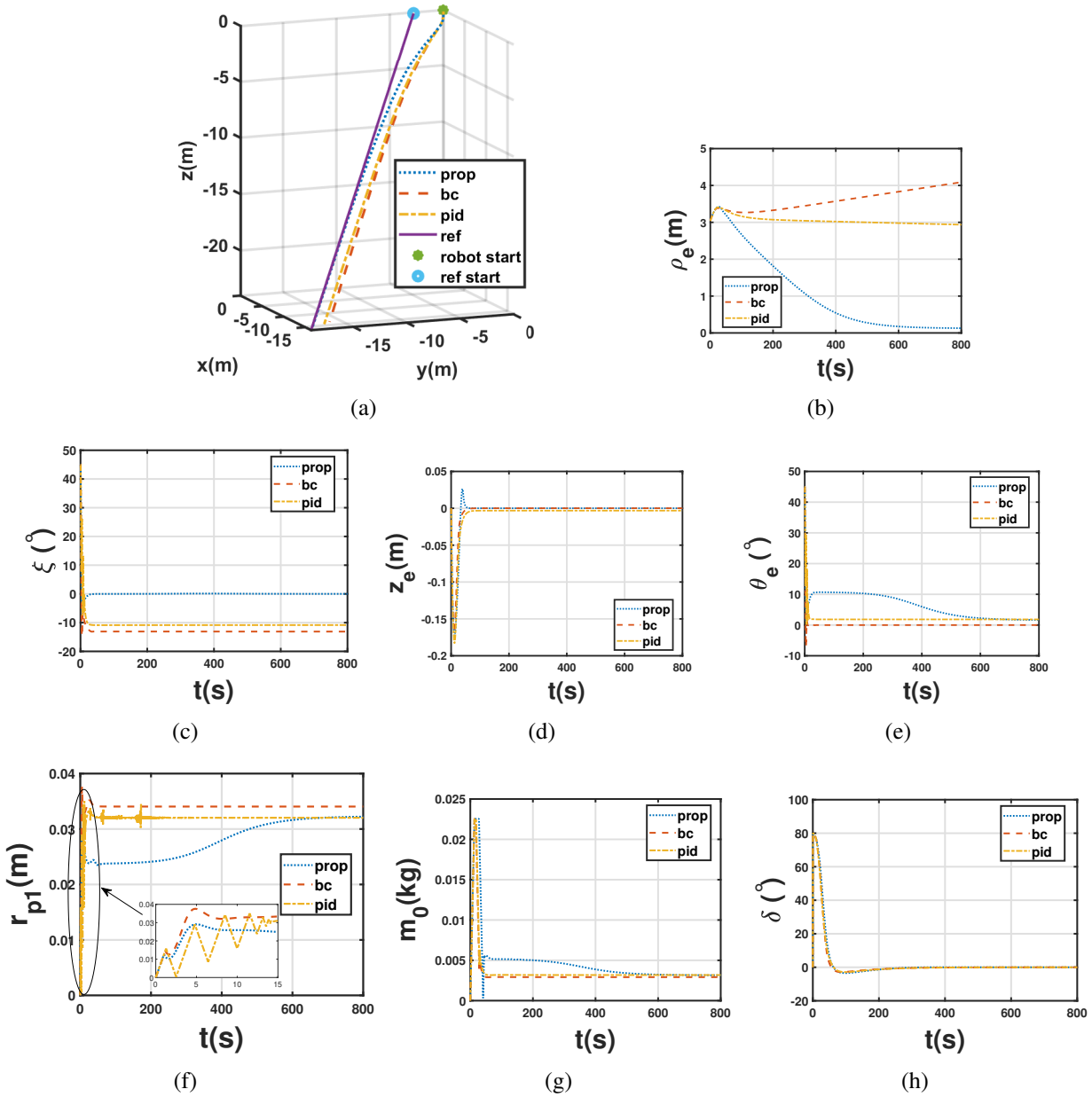


Figure 3.4: Simulation results with a linear gliding reference trajectory. The legends "prop", "bc" and "pid" indicate results from the proposed backstepping controller, the baseline backstepping controller, and the PID controller, respectively. (a): Reference and controlled trajectories in the 3D space; (b)-(e): the trajectories of tracking errors $(\rho_e, z_e, \theta_e, \xi)$; (f)-(h): the trajectories of the control inputs (r_{p1}, m_0, δ).

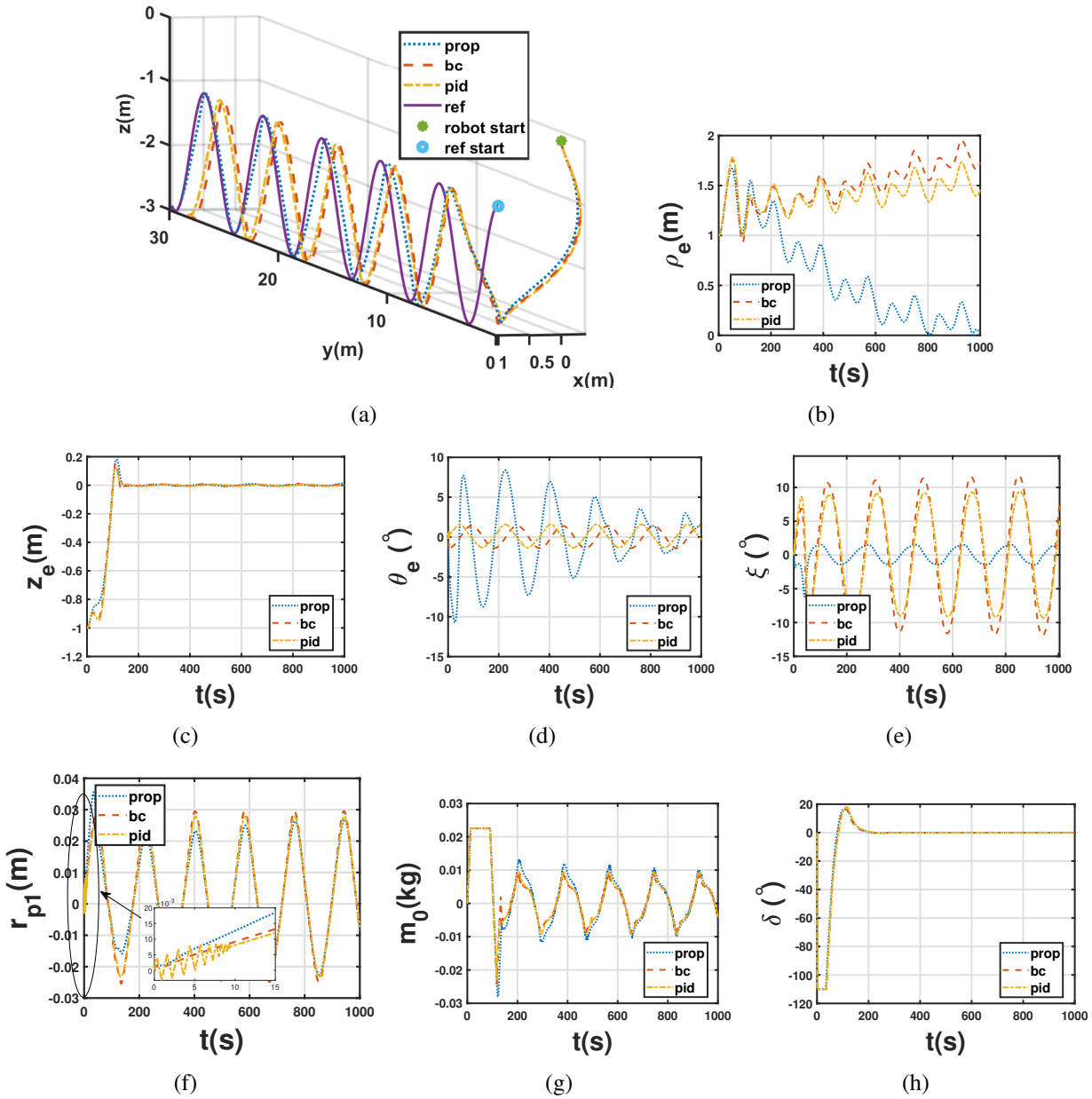


Figure 3.5: Simulation results with a sawtooth-like reference trajectory constrained to a vertical plane. The legends "prop", "bc" and "pid" indicate results from the proposed backstepping controller, the baseline backstepping controller, and the PID controller respectively. (a): Reference and controlled trajectories in the 3D space; (b)-(e): the trajectories of tracking errors $(\rho_e, z_e, \theta_e, \xi)$; (f)-(h): the trajectories of the control inputs (r_{p1}, m_0, δ).

f_{33} in (3.11) vanish, resulting in the following expressions for the control:

$$\begin{cases} u_3 = \dfrac{\Gamma_2}{f_{23}} \\ u_1 = \dfrac{1}{f_{11}}(\Gamma_1 - f_{13}u_3 - f_{14}(u_3)^2) \\ u_2 = \dfrac{\Gamma_3}{f_{32}} \end{cases} \tag{3.19}$$

In simulation, the tracking error ψ_e is redefined as $\psi - \frac{1}{2}(\eta^+ + \eta^-)$ if $\rho_e < \varepsilon_0$ for some small $\varepsilon_0 > 0$ and $\psi - \eta$ otherwise, where $\eta^+ = \arctan(y_e^+, x_e^+)$ and $\eta^- = \arctan(y_e^-, x_e^-)$ with $x_e^\pm = x_e + l\cos(\psi \pm \frac{\pi}{2})$ and $y_e^\pm = y_e + l\sin(\psi \pm \frac{\pi}{2})$, for some small $l > 0$. This allows the tracking error to be defined at the point of singularity when $\rho_e = 0$. ε and l are taken to be 0.05 and 0.15, respectively, in simulation.

The PID control consists of a set of three controllers. The error ψ_e is used to calculate δ with gains $k_p = 1$, $k_i = 0.001$ and $k_d = 1$. The error θ_e is used to calculate r_{p1} with gains $k_p = 1$, $k_i = 0$ and $k_d = 2$. The error z_e is used to calculate m_0 with gains $k_p = 1$, $k_i = 0$ and $k_d = 10$. The gains for the PID controller were chosen using the Matlab PID gain tuner and then manually tuned to refine performance on one of the reference trajectories. The gains for both backstepping controllers are $k_z = 1$, $k_\xi = 1$, $k_\psi = 1$, $k_1 = 10$, $k_2 = 1$, and $k_3 = 2$. These satisfy the conditions from the design in Section 3.2.1. For the proposed controller, c is chosen as $\frac{\pi}{9}$ for ξ. The parameters for all three controllers are kept the same over all trajectories.

Two reference trajectories are used, including a linear gliding pattern with a constant pitch angle and a constant depth rate and a sawtooth-like gliding pattern. The desired trajectories are parameterized as time-parameterized vector paths $[x_d(t), y_d(t), z_d(t), \theta_d(t)]$. It is worth noting that the results only show the actual values of the control inputs (after rate and magnitude saturation) as opposed to the values computed by the controllers. However, they coincide except for during

small segments of the trajectories.

3.3.1 Simulation Results

Table 3.1 shows a summary of root-mean-squared tracking errors for both reference trajectories. Fig. 3.4 shows the simulation results for tracking the first reference parameterized as

$$P_d(t) = \begin{bmatrix} 3+0.02t \\ 1.5+0.02t \\ 0.03t \\ -\frac{\pi}{4} \end{bmatrix}$$

In both this section and next section the units for the first three components of P_d are m and that for the last component of P_d is radian. This trajectory particularly highlights the effect of the proposed controller. Here, all three controllers have similar responses in δ which orients the robot so that it faces the desired horizontal plane position. For r_{p1}, the PID controller activates the rate constraint and is much more aggressive in the initial transient than the backstepping controllers, while the proposed controller is slightly more aggressive in the control of m_0. The PID controller and the baseline backstepping controller quickly track the pitch angle and the depth, but the resulting trajectories never converge to the desired horizontal position. The PID controller has a slight error in the pitch tracking allowing it to slowly decrease ρ_e and the baseline backstepping controller actually results in e_ρ increasing over time. On the other hand, the proposed backstepping controller, quickly tracks the depth with a bit of overshoot. Instead of tracking θ_e, it tracks ξ, which temporarily sacrifices perfect pitch tracking in order to achieve the desired planar position. The maximum deviation of ξ is dependent on c and the current value of θ. The difference in orientation

Table 3.1: Summary of RMS tracking errors (in all cases angles in radians).

Linear Glide Reference Trajectory			
	Proposed	Baseline BS	PID
z_e	0.0510	0.0450	0.0630
ρ_e	2.0920	3.4910	3.1110
θ_e	0.2500	0.2130	0.2210
ξ	0.2130	0.3010	0.2720
Sawtooth-like Reference Trajectory			
	Proposed	Baseline BS	PID
z_e	0.5280	0.4350	0.3690
ρ_e	0.9690	1.3870	1.3640
θ_e	0.0920	0.0190	0.0180
ξ	0.0260	0.1230	0.1210

increases the value of m_0 required to achieve the depth rate and results in ρ_e decreasing much more rapidly than for the PID controller and the baseline backstepping controller. As ρ_e converges to 0, θ_e converges to 0 and m_0 and r_{p1} converge to values similar to their counterparts from the PID control and baseline backstepping control.

Fig. 3.5 shows the simulation results for the case of the sawtooth-like reference trajectory parameterized as

$$P_d(t) = \begin{bmatrix} 0.03t \\ 1 \\ 2-\cos(\frac{\pi}{90}t) \\ -\frac{2\pi}{9}\sin(\frac{\pi}{90}t) \end{bmatrix}$$

It can be seen that, under the proposed controller, the 3D position tracking error (as reflected by ρ_e and z_e) converges to a small neighborhood of zero, and the (oscillating) pitch tracking error shows a consistently decreasing amplitude. On the other hand, while the oscillating θ_e values under the PID controller and the baseline backstepping controller gain smaller amplitudes quicker than the proposed controller, ρ_e values under both show an increasing trend and never converge towards zero.

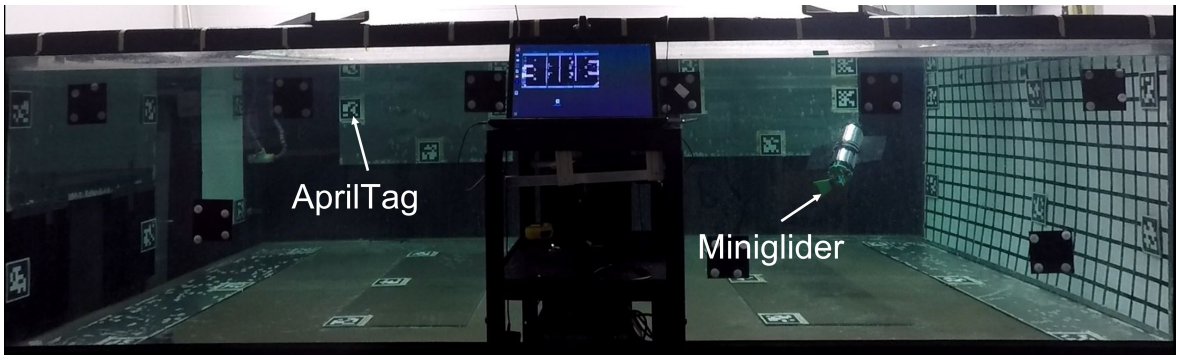


Figure 3.6: Miniglider robot operating in a large indoor tank during an experiment. Onboard sensors are used to estimate body-fixed velocities, while AprilTags and the onboard camera are used to localize the robot.

Figure 3.7: Picture of the Miniglider robot.

3.4 Experiments

3.4.1 Experimental Setup

Experiments are carried out with the miniaturized gliding robotic fish, Miniglider, discussed in Section 2.2.1 and the tank setup discussed in Section 2.2.2 (pictured in Fig. 3.6). A breakdown of the internal structure and actuation system of the robot is shown in Fig. 2.5. The robot actuation system consists of two linear actuators (Actuonix P16-P and L16-P, both with a stroke of 10 cm) with position feedback, a 60 ml syringe, a sliding mass, and a waterproof servo (Hitec HS-646WP). The servo controls the tail angle δ, one linear actuator controls the position $r_{p1} = (\mu - r_{p1_c})r_{p1_s}$ of

the slide mass, and the other linear actuator along with the syringe controls the net buoyancy $m_0 = (\mu - m_{0_c})m_{0_s}$ where $\mu \in [0,1]$ is the normalized linear actuator position for the corresponding case. The subscripts c and s represent the actuator positions corresponding to the base setting $(r_{p1} = 0, m_0 = 0)$, and scaling factors, respectively.

Orientation, angular rates, and depth are measured from onboard sensors including an LSM9DS1 IMU, a BlueRobotics Bar30 pressure sensor, and a Raspberry Pi Camera V1. In addition to the orientation and depth measurements from the onboard sensors, the position and the orientation of the Miniglider are also estimated based on the AprilTags.

3.4.2 Parameter Estimation

Table 3.2: Miniglider Model Parameters.

Parameter	Value	Parameter	Value
m_{0_c}	0.446	r_{p1_c}	0.445
m_{0_s}	-0.051 kg	r_{p1_s}	-0.1 m
m_1	15.011 kg	S	0.013 m^2
m_2	6.077 kg	C_D^α	39.50 rad^{-2}
m_3	8.291 kg	$C_{M_P}^\alpha$	0.279 m/rad
J_1	0.801 kg-m^2	C_L^α	24.66 rad^{-1}
J_2	0.076 kg-m^2	$C_{F_s}^\beta$	-4.650 rad^{-1}
J_3	1.60 kg-m^2	$C_{F_s}^\delta$	-3.529 rad^{-1}
$\bar{m}$	0.287 kg	C_{L0}	0.588
g	9.82 m/s^2	C_{D0}	1.985
C_{MR}^β	0.631 m/rad	K_{q1}	-11.97 m-s/rad
m_{w3}	0.819 kg	K_{q2}	-14.96 m-s/rad
r_{w3}	0.011 m	K_{q3}	-12.1 m-s/rad
C_{MY}^β	14.0 m/rad	C_{MY}^δ	-0.210 m/rad
ρ	992.2 kg/m^3	C_D^δ	4.694 rad^{-2}
C_{M_0}	0.321 m		

The model parameters of the Miniglider are estimated through a combination of computational

fluid dynamics (CFD) simulations and particle swarm optimization (PSO) [95]. SolidWorks 2020 is used to create a model of the Miniglider and the ANSYS 2020 Fluid Flow (Fluent) work flow is used to create a mesh file and run simulations of a static Miniglider in a water tunnel. From this simulation, the hydrodynamic parameters of C_D, C_{F_s}, C_L, C_{M_1}, C_{M_2}, and C_{M_3} can be identified based on (see (5.18))

$$\begin{cases} C_D = C_{D0} + C_D^{\alpha}\alpha^2 + C_D^{\delta}\delta^2 \\ C_{F_s} = C_{F_S}^{\beta}\beta + C_{F_S}^{\delta}\delta \\ C_L = C_{L0} + C_L^{\alpha}\alpha \\ C_{M_1} = C_{M_R}^{\beta}\beta + K_{q1}\omega_1 \\ C_{M_2} = C_{M_0} + C_{M_P}^{\alpha}\alpha + K_{q2}\omega_2 \\ C_{M_3} = C_{M_Y}^{\beta}\beta + K_{q3}\omega_3 + C_{M_Y}^{\delta}\delta. \end{cases} \tag{3.20}$$

By setting two of the three variables (α, β, and δ) to zero, and varying the third, polynomial curve fitting is used to estimate the model parameters in Eq. (3.20) except K_{q1}, K_{q2}, and K_{q3}.

PSO is then used with the entire parameter vector taken as the state space of the particles in the PSO algorithm (details in Appendix B). 100 particles are used with parameter vectors randomly generated from a uniform distribution across a bounded search space. One of these particles has a subset of the parameters replaced with measurable parameters (such as $\bar{m}$) and CFD hydrodynamic parameter estimates. The particles are optimized by minimizing the sum of weighted errors between the partial state data measured from open loop trajectories and their simulated values based on the parameter vectors from the particle swarm. Table 3.2 contains the best performing estimate of the parameter vector. In order to run the PSO-based parameter estimation, data was collected from open loop control of the Miniglider over several different control trajectories. .

3.4.3 Controller Implementation

The proposed controller, PID controller, and backstepping controller are all implemented on the Miniglider's Raspberry Pi 4 for control experiments. The controllers and a model-based observer are implemented in Python 3 along with AprilTag-based localization. The AprilTag based-localization uses known AprilTag poses (position and orientation) and the camera-relative AprilTag measurements to produce a pose estimate of the robot in the world frame for each detected AprilTag. A Kalman filter is then used to fuse AprilTag positions, AprilTag yaw angles, and depth from the pressure sensor. It outputs the 3D position, yaw angle, and the derivatives of the aforementioned state variables. When measurments are available, the Kalman filter reports an estimation variance of 5 to 6 cm for the x and y coordinates and a variance of 1 to 3 cm for the z coordinate. When AprilTag measurements are unavailable for an extended period of time, the model-based observer (discussed below) is used to propagate the planar position and the IMU is used to propagate the yaw angle.

The model-based observer is used to produce estimates of the linear body-fixed velocities $\hat{v}_b$ and is driven by the depth estimation error $z-\hat{z}$. It is based on Eq. (3.2) and formulated as

$$\dot{\hat{b}}_i = R\hat{v}_b + K_2 \begin{bmatrix} 0 \\ 0 \\ z-\hat{z} \end{bmatrix}$$

$$\dot{\hat{v}}_b = M^{-1}(M\hat{v}_b \times \omega_b + m_0 g R^T k + F_{ext}(\hat{v}_b)) + K_1 R^T \begin{bmatrix} 0 \\ 0 \\ z-\hat{z} \end{bmatrix}$$

$\dot{\hat{b}}_i$ is used to generate $\hat{z}$ for the driving error $z-\hat{z}$ and maintain a planar position estimate when

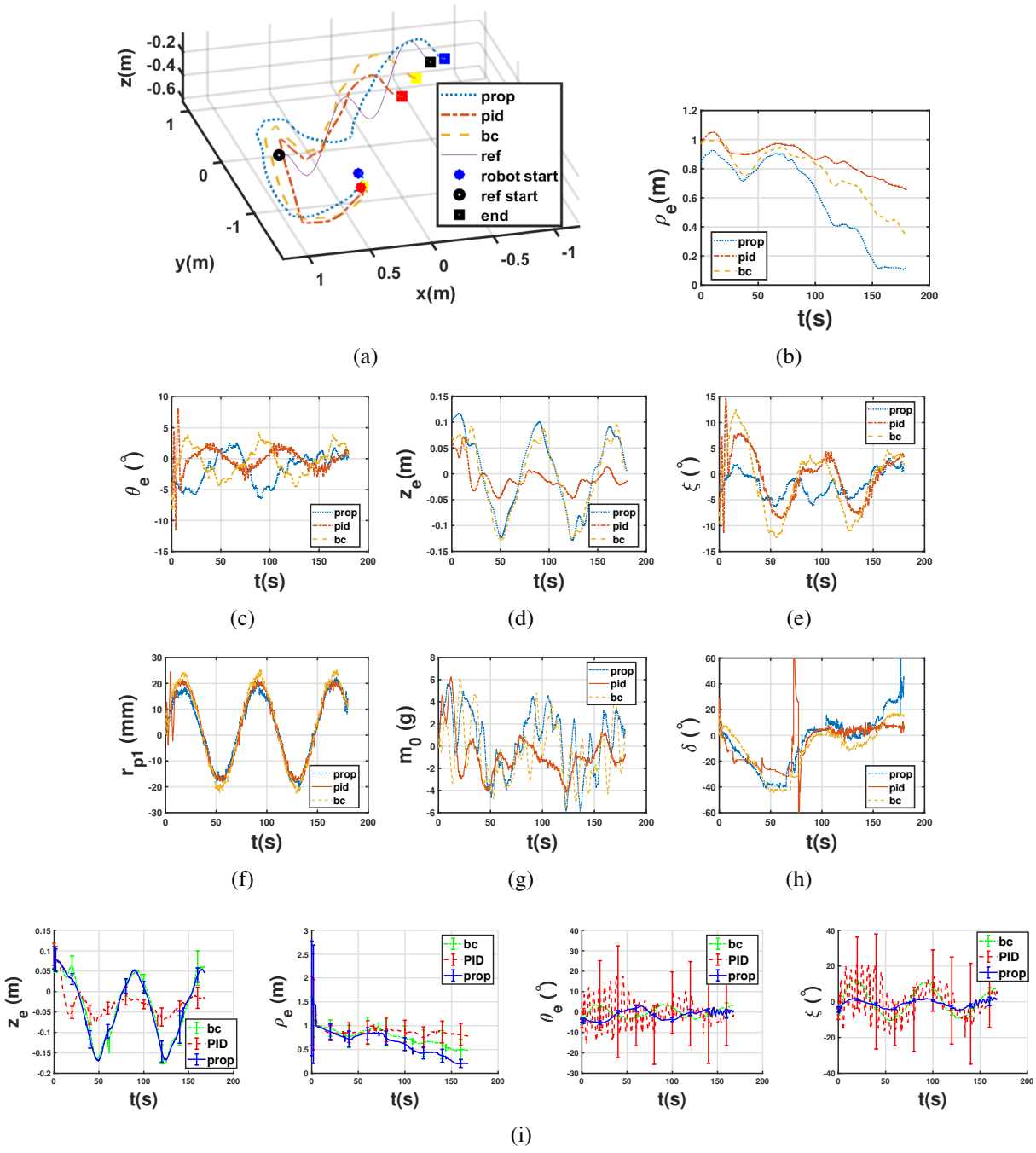


Figure 3.8: Experimental results with a diagonal glide plane reference trajectory. The legends "bc" and "pid" indicate results from the proposed backstepping controller and the PID controller, respectively. (a): Reference and controlled trajectories in the 3D space; (b)-(e): the trajectories of tracking errors $(\rho_e, z_e, \theta_e, \xi)$; (f)-(h): the control command (cmd) calculated by the two controllers as well as the trajectories of the achieved control inputs (r_{p1}, m_0, δ). (i): statistics of subset of the errors.

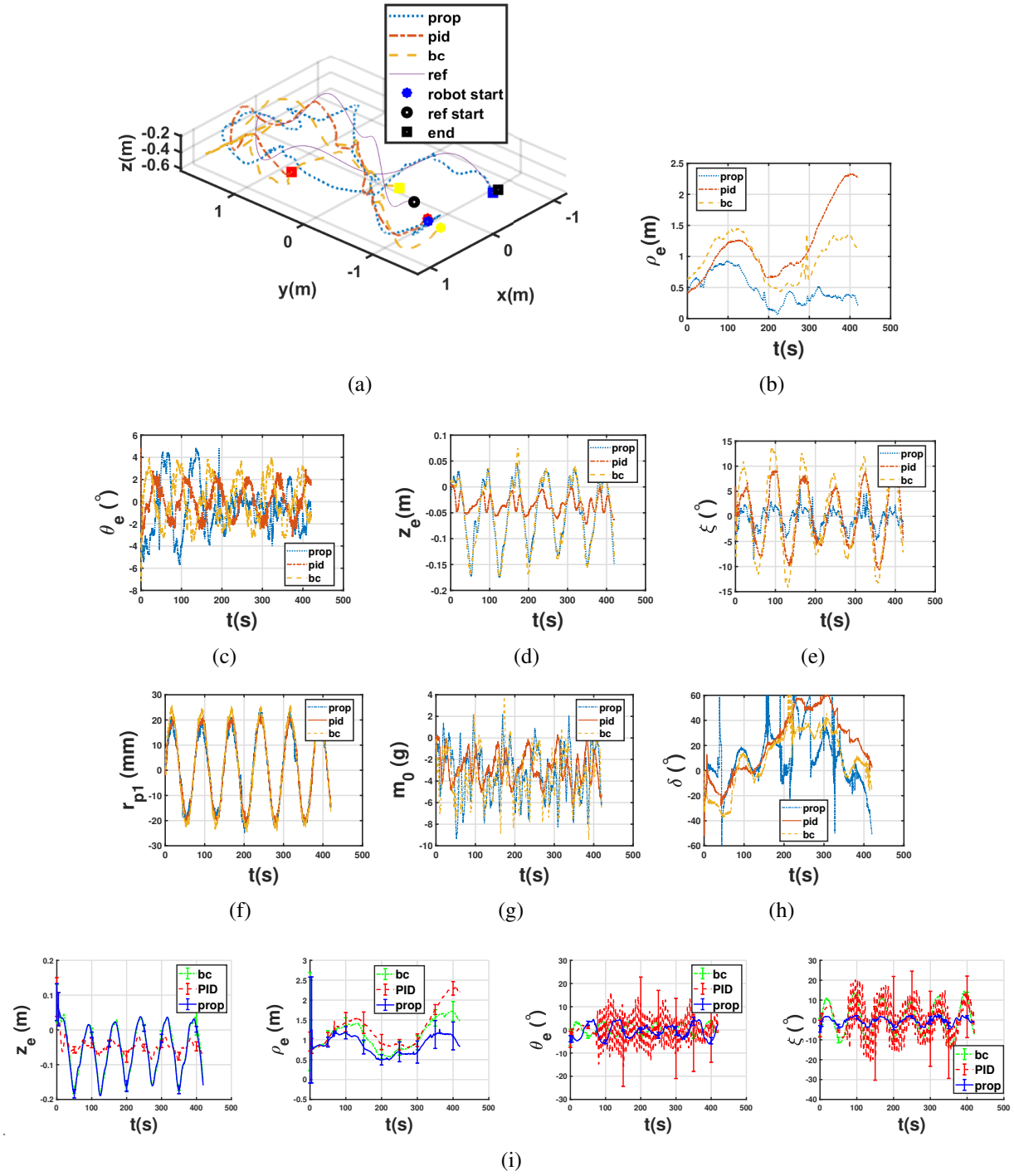


Figure 3.9: Experimental results with a diagonal glide plane reference trajectory. The legends "bc" and "pid" indicate results from the proposed backstepping controller and the PID controller, respectively. (a): Reference and controlled trajectories in the 3D space; (b)-(e): the trajectories of tracking errors $(\rho_e, z_e, \theta_e, \xi)$; (f)-(h): the control command (cmd) calculated by the two controllers as well as the trajectories of the achieved control inputs (r_{p1}, m_0, δ). (i): statistics of subset of the errors.

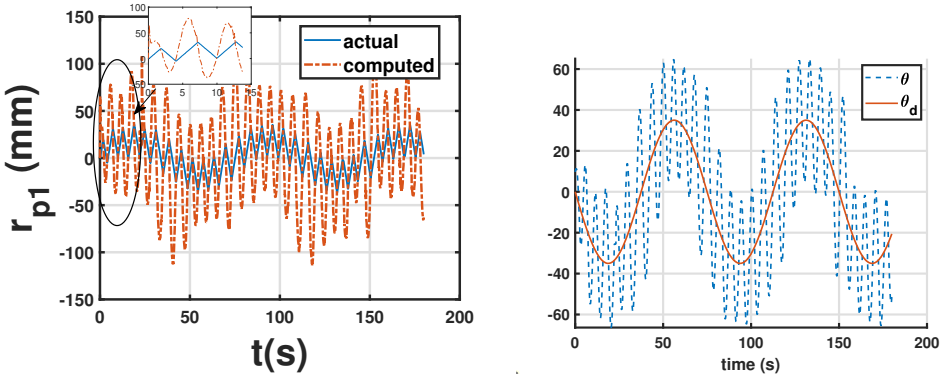


Figure 3.10: Example of r_{p1} and θ when PID controller induces oscillations.

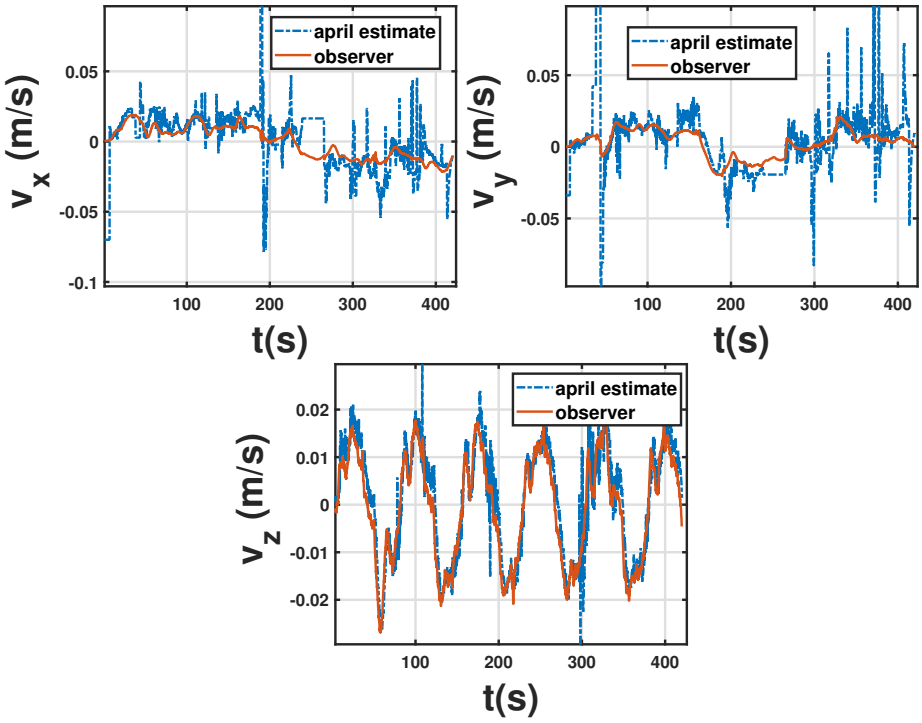


Figure 3.11: The body-fixed velocity estimates from the observer converted into the inertial frame and compared with the inertial velocities estimated from the AprilTag measurements.

no AprilTags are available for measurement. K_1 and K_2 are diagonal gain matrices. As input, the velocity observer takes the current estimate of the body-fixed linear velocities v_b, the rotation matrix $R(\phi,\theta,\psi)$ (ϕ and θ obtained from the IMU, ψ calculated from the IMU and AprilTag Kalman filter), the body-fixed angular velocity ω_b from the IMU, the control inputs m_0 and δ, the current estimate $\hat{z}$ of the depth, and the depth z from the pressure sensor. The AprilTag position estimate is not used in the model-based observer due to the occasional large localization error and relatively long periods with no available measurements.

The controller uses the same sources as the observer for depth, body-fixed angular velocity, and orientation. It also takes the estimated body-fixed linear velocities $\hat{v}_b$ from the model-based observer. The position feedback is obtained from a combination of the AprilTag Kalman filter and model-based observer as previously explained.

All controllers were operated at roughly 10 Hz. The PID gains were tuned experimentally, and were chosen as $k_p = 0.08$, $k_i = 0$, $k_d = 0.1$ for depth control, $k_p = 0.08$, $k_i = 0.05$, $k_d = 0.0375$ for pitch control, and $k_p = 1$, $k_i = 0.001$, $k_d = 1$ for yaw control. For both the baseline backstepping controller and the proposed controller, the gains are chosen as $k_z = 0.08$, $k_\psi = 1$, $k_\xi = 4$, $k_1 = 0.9$, $k_2 = 0.1$, and $k_3 = 4$ with $c = \frac{\pi}{9}$ for ξ.

3.4.4 Experimental Results

The controllers are tested on two reference trajectories; the first parameterized as

$$P_d(t) = \begin{bmatrix} -1 + 0.012t \\ -1 + 0.01t \\ 0.35 - 0.2\cos(\frac{2\pi}{75}t) \\ -\frac{7\pi}{36}\sin(\frac{2\pi}{75}t) \end{bmatrix} \tag{3.21}$$

Table 3.3: Summary of RMS tracking errors (in all cases angles in radians) for experiments.

Reference Trajectory in Eq. (3.21)			
	Proposed	Baseline BS	PID
z_e	0.070	0.070	0.039
ρ_e	0.643	0.768	0.873
θ_e	0.0412	0.03787	0.2631
ξ	0.0349	0.1119	0.3359
Reference Trajectory in Eq. (3.22)			
	Proposed	Baseline BS	PID
z_e	0.071	0.068	0.050
ρ_e	0.884	1.115	1.297
θ_e	0.0496	0.0332	0.1181
ξ	0.0295	0.1237	0.1933

and the second parameterized as

$$P_d(t) = \begin{bmatrix} -1.5\cos(\frac{\pi}{270}t + \frac{\pi}{4}) \\ -1.2\cos(\frac{\pi}{270}t + \frac{\pi}{4})\sin(\frac{\pi}{270}t + \frac{\pi}{4}) \\ 0.35 - 015\cos(\frac{2\pi}{75}t) \\ -\frac{7\pi}{36}\sin(\frac{2\pi}{75}t) \end{bmatrix} \tag{3.22}$$

For each trajectory, ten trials are run for each controller. Fig. 3.8 and Fig. 3.9 show the results of a single trial and statistics across all trials for the first and second trajectory, respectively. Both figures show the path and the reference in 3D, the post-processed errors (z_e, ρ_e, θ_e, and ξ), and the actual achieved values of the control inputs for a single trajectory. The control commands coincide with the actual control values during most of the respective experiments. In addition, the means of the errors across the 10 trials are shown with error bars depicting standard deviation. A summary of root-mean-squared tracking errors is shown in Table 3.3. Of the 10 trials for the PID control scheme, the pitch control induced large oscillations 6 times for the first trajectory and 3 times (trials that immediately induced oscillations were discarded) for the second trajectory causing degraded

performance in the overall tracking.

Statistically, the PID controller is more precise in tracking the depth, while the baseline backstepping controller and the proposed controller offer very similar depth tracking performance to one another. This is also reflected in the single trials as well. The control m_0 is slightly biased indicating some model inaccuracy which may affect the model-based controllers more than the PID controller, allowing it to perform better at tracking depth. The single trials indicate that all three controllers are able to provide good tracking of their respective reference pitch angles; however, the statistical results show that the PID can induce large oscillations quite often. For many of the trials, the control command for the PID controller was initially more aggressive for r_{p1}. The actuator for r_{p1} was unable to keep up with the commanded value, which was likely the cause of oscillations in pitch tracking. An example of this can be seen in Fig. 3.10. Both the single trials and the statistical results indicate that the propose controller is superior in tracking the horizontal position. The baseline backstepping controller provides a particularly good comparison for the proposed controller because the depth profiles are almost identical suggesting the difference in planar tracking error is largely due to the pitch tracking control scheme.

The results of the velocity observer are shown for a single trial in Fig. 3.11. The body-fixed velocity estimates are rotated by the orientation to compute the inertial frame velocities. These match well for the depth velocity and the general trend matches for the x and y velocities.

3.5 Summary and Future Work

In this chapter, a novel backstepping-based controller is presented for a gliding robotic fish. It is able to track a reference trajectory for 3D position and pitch angle using only three actuation inputs. The introduction of a hybrid error function, combining the pitch tracking error with the

planar position tracking error, was key to enabling successful tracking. This novel error, the depth tracking error, and the difference between the yaw and the target-point direction (in the horizontal plane) were are all shown to be regulated to zero under the proposed backstepping control law. Through time-scale analysis, it was further shown that, with a vanishing hybrid error function, both the pitch error and the magnitude of the planar tracking error converge to a region around zero at a slower time-scale. The proposed controller was then evaluated with both simulation and experiments using a mini-glider robot. These results supported the efficacy of the proposed approach in tracking the 3D position and the pitch angle. Its advantages were further demonstrated via the comparison with two alternative schemes, a PID controller and a baseline backstepping controller not using the hybrid error function.

Future directions for this work include considering the dynamics of the actuators in the controller synthesis, which is expected to more naturally accommodate the actuator constraints. The effect of the tail under rapid movement should also be considered in the control design. This may enable design of a controller capable of taking advantage of the energy-efficient gliding-like motion from the buoyancy control as well as rapid maneuvers enable by the tail. Also, since the system parameters can change over time due to changing of parts (such as wings of different designs), it will be of interest to examine adaptive backstepping control schemes. It is also of interest to study output feedback or partial state feedback control theory for the tracking problem. In particular, while a preliminary observer design showed promise in the experimental implementation, establishing a systematic observer framework for underwater gliders remains an open problem. Lastly, the model used in this work assumes an ideal environment, but in field conditions, the robot may be exposed to dynamic disturbances from waves and water currents. In that case, online model estimation and controller robustification methods are of interest in ensuring robust tracking performance.

Chapter 4

Incorporation of Observability in Control

This chapter explores the incorporation of observability into control schemes. It proposes the use of control barrier functions (CBF) to achieve this. The method is applied to the application of ranged-based target tracking and is compared to an approach using model predictive control (MPC) to achieve the same task. While this chapter focuses on ranged-based target tracking due to its applicability to tracking fish in underwater environments, the ideas presented offer a promising approach to addressing output feedback for nonlinear systems. Section 4.1 reviews some concepts of observability and control barrier functions. In Section 4.2, the problem formulation and the proposed approach are provided. An example problem with a unicycle model for the robot is discussed in Section 4.3. Simulation results are presented in Section 4.4 followed by a discussion of the experimental setup and results in Section 4.5.

4.1 Background Material

4.1.1 Nonlinear Observability Rank Condition

A general nonlinear system modeled by

$$\begin{cases} \dot{x} = f(x,u), \\ y = h(x), \end{cases} \tag{4.1}$$

with state $x \in \mathbb{R}^n$, input $u \in \mathbb{R}^m$, and output $y \in \mathbb{R}^o$, is said to be *observable* if there is a one-to-one correspondence on some time interval $[t_0, t_1]$ between the set of initial states x_0 and the trajectories of the input-output pair $(u(t), y(t))$ for $t \in [t_0, t_1]$ [96]. Observability for nonlinear systems can be studied using the concept of local weak observability introduced in [97]. With the Lie derivatives of the output vector $h(x)$ defined as

$$\begin{aligned} \mathscr{L}_f^0 h &= h(x), \\ \mathscr{L}_f^l h &= [\nabla_x \mathscr{L}_f^{l-1} h(x)] f(x,u), \text{for } l \geq 1, \end{aligned}$$

the nonlinear observability matrix for the system in Eq. (4.1), evaluated at some $x = x_1$, is given by

$$\mathscr{O}(x_1, u) = \begin{bmatrix} \nabla_x \mathscr{L}_f^0 h(x_1) \\ \nabla_x \mathscr{L}_f^1 h(x_1) \\ \vdots \\ \nabla_x \mathscr{L}_f^l h(x_1) \end{bmatrix}, \tag{4.2}$$

where $l \geq \frac{n}{m}$ is a positive integer index.[1] The observability rank condition for nonlinear systems states that the system (4.1) is *locally weakly observable* at x_1 if there exists an input u, such that the resulting matrix $\mathscr{O}(x_1, u)$ is full rank. In some cases, it may take several Lie derivatives of the measurement function to confirm that a system is observable. However, it is generally not known a priori how many Lie derivatives of the measurement function are needed to achieve full rank or if the system is simply unobservable.

[1] Alternatively, the matrix can be constructed by stacking the matrices that result from taking the Lie derivatives of each element in the vector $y = h(x)$ separately.

4.1.2 Control Barrier Functions

Consider a general nonlinear system

$$\dot{x} = f(x) \tag{4.3}$$

with $x \in D \subset \mathbb{R}^n$ and $f(\cdot)$ being locally Lipschitz. Define a set $\mathscr{C}$, its boundary $\partial\mathscr{C}$, and its interior $\text{Int}(\mathscr{C})$ by

$$\mathscr{C} = \{x \in D \subset \mathbb{R}^n : B(x) \geq 0\}, \tag{4.4}$$

$$\partial\mathscr{C} = \{x \in D \subset \mathbb{R}^n : B(x) = 0\}, \tag{4.5}$$

$$\text{Int}(\mathscr{C}) = \{x \in D \subset \mathbb{R}^n : B(x) > 0\}, \tag{4.6}$$

for a smooth function $B(x) : \mathbb{R}^n \to \mathbb{R}$. The set $\mathscr{C}$ is forward invariant if the initial condition $x_0 = x(0) \in \mathscr{C}$ implies that $x(t) \in \mathscr{C}, \forall t > 0$. If there exists an extended class $\mathscr{K}_\infty$ function $\alpha : (-a, b) \to (-\infty, \infty)$ with $a, b > 0$ such that the Lie derivative $\mathscr{L}_f B(x)$ of $B(x)$ satisfies

$$\mathscr{L}_f B(x) \geq -\alpha(B(x)), \tag{4.7}$$

for all $x \in D$, then $B(x)$ is a zeroing barrier function[2] and $\mathscr{C}$ is forward-invariant with respect to the system in Eq. (4.3) [75].

Control barrier functions are a synthesis tool derived from barrier functions and are used to enforce forward-invariance of a set $\mathscr{C}$ for a control system. In the case of an affine control system

$$\dot{x} = f(x) + g(x)u \tag{4.8}$$

[2]Both zeroing barrier functions and the closely related reciprocal barrier functions are discussed in [75]. Here, we focus only on the zeroing barrier functions.

with f and g being locally Lipschitz, the state $x \in D \subset \mathbb{R}^n$, and input $u \in U \subset \mathbb{R}^m$, if there exists an extended class $\mathscr{K}_\infty$ function α such that

$$\sup_{u \in U} [\mathscr{L}_f B(x) + \mathscr{L}_g B(x) u] \geq -\alpha(B(x)), \forall x \in D, \tag{4.9}$$

then $B(x)$ is a control barrier function and the set $\mathscr{C}$ is forward-invariant with respect to the system (4.8) for

$$K_{cbf}(x) = \{u \in U : \mathscr{L}_f B(x) + \mathscr{L}_g B(x) u + \alpha(B(x)) \geq 0\}$$

where $K_{cbf}(x)$ is the set of all Lipschitz continuous controls that render the set $\mathscr{C}$ forward-invariant [75, 76].

4.2 Problem Formulation and Proposed Approach

4.2.1 Problem Formulation

This work aims to solve the target tracking problem given only measurements of distance to a target. The objective is to find an output feedback controller that improves the observability of the tracker's location relative to the target, while simultaneously achieving the goal of following the target's position trajectory. The nonlinear system

$$\begin{cases} \dot{X} = f(X, U), \\ Y = h(X), \\ \dot{\hat{X}} = g(Y, U, \hat{X}), \end{cases} \tag{4.10}$$

is considered with state $X \in \mathbb{R}^n$, input $U \in \mathbb{R}^m$, and output $Y \in \mathbb{R}^o$. $\hat{X}$ is the state estimate generated by an observer.

For ease of presentation (and without the loss of generality), a two-dimensional (2D) setting is considered. Define

$$\tau_e(t) = \begin{bmatrix} x(t) - x_{ta}(t) \\ y(t) - y_{ta}(t) \end{bmatrix} \tag{4.11}$$

as the tracking error between the tracker and the target positions, (x, y) and (x_{ta}, y_{ta}), respectively, and

$$\kappa_e(t) = \tau_e(t) - \hat{\tau}_e(t) \tag{4.12}$$

as the error between the true and estimated relative position vectors, where $\hat{\tau}_e$ is an estimate of (4.11). The tracker's state vector is assumed to be available with the exception of its position. Then the aim is to design a controller U that depends on Y and $\hat{X}$ to minimize the 2-norm of both the tracking error τ_e and the estimation error κ_e.

4.2.2 Proposed Approach

Modifying a nominal state feedback controller with a control barrier function based on the observability matrix for the system is proposed as a solution. The approach exploits the observation that the observability for nonlinear control systems often depends on both the state and the control input. In general, because the standard CBF formulation does not cover the scenario of control-dependent barrier functions, the integral control barrier function (iCBF) formulation presented in [98] is needed to treat observability matrices of the form $\mathscr{O}(X,U)$. In this work, the special case of observability matrices $\mathscr{O}(X)$ that depend only on the state is treated. The intuition behind the approach is as follows.

Given the system (4.10), the observability matrix $\mathscr{O}(X)$ is constructed as shown in (4.2). Consider the inverse condition number $C^{-1}(\mathscr{O}(X))$ and the determinant of observability Gramian $\det(\mathscr{O}(X)^T\mathscr{O}(X))$.[3] The condition number for a general matrix A is defined as $C(A) = \|A\|_p\|A^+\|_p$ (where A^+ denotes a pseudo inverse) and depends on the chosen norm p. The 2-norm is commonly chosen leading to the condition number being the ratio of the maximum and minimum singular values and its inverse has been commonly used as a measure of observability [32, 34, 36]. If $\bar{\lambda}(X) = [\lambda_1, ..., \lambda_n]^T$ is a vector containing the eigenvalues of $\mathscr{O}(X)^T\mathscr{O}(X)$ (squared singular values of $\mathscr{O}(X)$), then we have

$$C^{-1}(\mathscr{O}(X)) = \frac{\sqrt{\min(\bar{\lambda})}}{\sqrt{\max(\bar{\lambda})}}$$

and

$$\det(\mathscr{O}(X)^T\mathscr{O}(X)) = \prod_1^n \lambda_i.$$

Since the determinant can be expressed as the product of the eigenvalues, both functions are only zero when at least one eigenvalue of the observability Gramian is zero and the observability matrix becomes rank-deficient. Therefore, the conditions $C^{-1}(\mathscr{O}(X)) > 0$ and $\det(\mathscr{O}(X)^T\mathscr{O}(X)) > 0$ are both necessary and sufficient to satisfy the nonlinear observability rank condition. They both are non-negative functions that define a set of state and control pairs that render the system locally weakly observable while the boundaries $C^{-1}(\mathscr{O}(X)) = 0$ or $\det(\mathscr{O}(X)^T\mathscr{O}(X)) = 0$ represent conditions under which the system is not.

Based on this insight, the use of $C^{-1}(\mathscr{O}(X))$ and $\det(\mathscr{O}(X)^T\mathscr{O}(X))$ to define zeroing barrier functions to enforce local observability is proposed.[4] Unlike reciprocal barrier functions, which exclude the zero level-set when enforcing forward invariance, the zeroing barrier functions include

[3] Note that this is equivalent to the squared determinant $\det(\mathscr{O}(X))^2$ when $\mathscr{O}(X)$ is square.

[4] Only considering the numerator of the inverse condition number is also an appropriate choice.

the zero level-set in the forward-invariant set. So the proposed functions have to be modified such that $B(X) = \bar{B}(X) - \varepsilon_B$ for a sufficiently small constant $\varepsilon_B > 0$ to exclude the zero level-set, where $\bar{B}(X)$ represents one of the proposed functions. This is similar to what is done for collision avoidance and energy management in [76].

Given a nominal controller $U_{nom} = k(X)$ designed to achieve a desired task (an example design for the problem in this work is given in Section 4.4), either $C^{-1}(\mathscr{O}(X))$ or $\det(\mathscr{O}(X)^T \mathscr{O}(X))$ can be used to define a CBF and be paired with the nominal controller to perform the control task as well as possible while producing observable trajectories with respect to a particular measurement function $h(X)$. The CBF formulation can be applied directly using the optimization problem

$$\begin{cases} U^* = \text{argmin}_U \quad J_B(U) = \|U - U_{nom}\|^2 \\ \qquad \text{subject to} \\ \qquad\qquad \dfrac{\partial B_o}{\partial X} f(\hat{X}, U) + \alpha(B_o(\hat{X})) > 0 \\ \qquad\qquad U \in \mathscr{U} \end{cases} \tag{4.13}$$

where $\mathscr{U}$ is the set of all admissible controls, α is a class $\mathscr{K}_\infty$ function, and $B_o(\hat{X})$ is the observability-based CBF evaluated at the estimated state $\hat{X}$. If the dynamics $f(X,U)$ are affine in the input U, the problem (4.13) can be solved efficiently using quadratic programming as mentioned in previous works [73–76].

A potential issue with choosing the proposed functions to define CBFs is that they are not guaranteed to be smooth or continuously differentiable, which is a fundamental requirement for CBFs. If both the dynamic equations $f(X,U)$ and the measurement function $h(X)$ have at least $l+1$ continuous derivatives, where l is the number of Lie derivatives used to compute the observability matrix, then each element of the observability matrix will be continuously differentiable. Since

the determinant of a matrix is a polynomial function of its elements, $\det(\mathscr{O}(X)^T\mathscr{O}(X))$ is also continuously differentiable and thus the standard CBF machinery will apply. On the other hand, the inverse condition number $C^{-1}(\mathscr{O}(X))$ is a non-smooth function of the matrix $\mathscr{O}$, and thus when it is used as a CBF, some modifications are needed. The work in [80] extends the theory of CBFs to non-smooth barrier functions (NBFs) and uses the min and max functions to create Boolean logic [5] for multiple CBFs. Theorem 2 of [80] relaxes the requirement of continuous differentiability for CBFs, allowing NBFs to be continuous, locally Lipschitz, regular, and have a set-valued Lie derivative that satisfies a constraint similar to Eq. (4.7). Then Proposition 3 of [80] shows how to implement quadratic programs for control-affine systems and NBFs resulting from using min and max functions on a set of NBFs. The key difference from the standard CBF implementation is that a set of constraints is required to be satisfied rather than a single constraint. Proposition 3 can be applied to $C^{-1}(\mathscr{O}(X))$ by considering each of the singular values of the observability matrix as individual NBFs.

The proposed approach can be summarized in three major steps:

1. Choose an observability barrier function $B_o(X)$.
2. Choose the controller U_{nom}.
3. Solve the optimization problem (4.13) for U^*.

Remark 1. *One can guarantee that the system can be made locally weakly observable with respect to the measurement function $h(X)$ provided that there exist at least n rows of the observability matrix that can be made independent through particular choices of state and control pairs. This follows from the nonlinear observability rank condition and the forward-invariance of the barrier function condition ensuring that all eigenvalues of $\mathscr{O}^T\mathscr{O}$ are nonzero. However, in practice, the approach would be considered for output feedback. This could lead to the true CBF constraint not*

[5] Boolean logic was also implemented in [79] using sum and products. This can be used to interpret $\det(\mathscr{O}(X)^T\mathscr{O}(X))$ in terms of the singular values of $\mathscr{O}(X)$.

being enforced due to the observability matrix being constructed from the state estimate. Numerically calculating the observability Gramian through simulation of the system with perturbed initial conditions as done in recent works with empirical observability Gramians [99, 100] may provide a pathway to circumvent this issue. A second approach would be to modify the barrier constraint in Eq. (4.13) *to use conservative estimates of the barrier function and its derivative or Lipschitz constants for the barrier function and its derivative, similar to what is done for the effect of model uncertainty on CBF constraints in [101] and [102]. A detailed investigation of these is left for future work.*

4.3 Range-based Target Tracking with Unicycle Model

In this section, we compute the observability matrix for a specific model. This will allow us to construct the observability-based barrier function needed for the proposed approach and gain intuition about what conditions are required to achieve observability.

We consider the tracker dynamics as a unicycle model that can be used for approximating a wide class of systems. The state is taken as $X = [x, y, \psi, v]^T$ with dynamics (notice that this system is affine in the control)

$$\dot{X} = \begin{bmatrix} v\cos(\psi) \\ v\sin(\psi) \\ u_1 \\ u_2 \end{bmatrix}, \tag{4.14}$$

and the measurement function,

$$h(X, x_{ta}(t), y_{ta}(t)) = \begin{bmatrix} \|\tau_e\| \\ \psi \\ v \end{bmatrix}, \tag{4.15}$$

where $\psi \in [0, 2\pi)$ is the heading angle and v is the linear velocity. The input $U = [u_1, u_2]^T$ directly controls the angular velocity and the linear acceleration of the tracker.

To construct the observability matrix for the range-based target tracking problem, it is useful to remove the time dependence by considering the observability of the relative kinematics for the target-tracker system as done in [34]. The state of the relative kinematics can be expressed as

$$\begin{bmatrix} X_r \\ V_r \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} x_r \\ y_r \end{bmatrix} \\ \begin{bmatrix} v_{xr} \\ v_{yr} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} x - x_{ta} \\ y - y_{ta} \\ \dot{x} - v_{x_{ta}} \\ \dot{y} - v_{y_{ta}} \end{bmatrix}, \tag{4.16}$$

where V_r is the relative velocity of the system and can be expressed as a function of the tracker state X and the target velocity, $[v_{x_{ta}}, v_{y_{ta}}]^T$. We further note that studying the observability based on the range squared, $\frac{1}{2}\|\tau_e(t)\|^2$, simplifies computations while preserving the observability properties. Note that $\frac{1}{2}\|\tau_e(t)\|^2$ is only used for analysis and constructing the observability matrix. It is not used when implementing an observer as any measurement noise would be amplified and the transformed noise distribution would become non-Gaussian even if the measurement noise is Gaussian.

The relative system can be written as $X_p = [x_r, y_r, \psi, v]^T$ with dynamics

$$\dot{X}_p = \begin{bmatrix} v_{xr} \\ v_{yr} \\ u_1 \\ u_2 \end{bmatrix}. \tag{4.17}$$

Note that the measurement function in Eq. (4.15) can now be written as a function of X_p. Since $\frac{1}{2}\|\tau_e(t)\|^2 = \frac{1}{2}(x_r^2 + y_r^2)$, the observability matrix, constructed from reordering the rows associated with the first two Lie derivatives, is given by

$$\mathscr{O} = \begin{bmatrix} x_r & y_r & 0 & 0 \\ v_{xr} & v_{yr} & * & * \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{4.18}$$

where $*$ represents non-zero entries that in general will not affect observability. Therefore, to enforce local weak observability of the relative position, only the sub-matrix

$$\mathscr{O}_p = \begin{bmatrix} x_r & y_r \\ v_{xr} & v_{yr} \end{bmatrix}, \tag{4.19}$$

is needed which gives $\det(\mathscr{O}_p) = v_{yr}x_r - v_{xr}y_r \neq 0$ as a necessary and sufficient condition for $\mathscr{O}$ to be full rank. Coincidentally, $\det(\mathscr{O}_p)^2 = \det(\mathscr{O}^T\mathscr{O})$.

The following transformations connect the inverse condition number and the determinant of $\mathscr{O}_p$. An expression for the condition number was derived in [34] by rewriting $\mathscr{O}_p$ in polar coordinates as

$$\mathscr{O}_p = \begin{bmatrix} \|X_r\|\sin(\lambda) & \|X_r\|\cos(\lambda) \\ \|V_r\|\sin(\beta) & \|V_r\|\cos(\beta) \end{bmatrix}, \tag{4.20}$$

with $\beta = \pi - \arctan(\frac{v_{yr}}{v_{xr}})$ and $\lambda = \pi - \arctan(\frac{y_r}{x_r})$. Defining the variables $\gamma = \frac{\|X_r\|}{\|V_r\|}$ and $\theta = \beta - \lambda$, the inverse condition number can be calculated $C^{-1}(\mathscr{O}_p) = \frac{2\gamma|\sin(\theta)|}{\gamma^2+1+\sqrt{\gamma^4+2\gamma^2\cos(2\theta)+1}}$. Note that unlike $\det(\mathscr{O}_p)$, $C^{-1}(\mathscr{O}_p)$ is not continuously differentiable, so it is not used as a barrier function in this work. However, as mentioned Section 4.2.2, theory for non-smooth CBFs could be applied to use this function. We rewrite the expression as

$$C^{-1}(\mathscr{O}_p) = \frac{2\|V_r\|\|X_r\||\sin(\theta)|}{\|X_r\|^2 + \|V_r\|^2 + \Gamma} \tag{4.21}$$

with $\Gamma = \sqrt{\|X_r\|^4 + 2(\|V_r\|\|X_r\|)^2\cos(2\theta) + \|V_r\|^4}$, to show that the numerator of the inverse condition number is a scalar multiple of the absolute value of the determinant, which can be expressed as

$$\begin{aligned} \det(\mathscr{O}_p) &= v_{yr}x_r - v_{xr}y_r \\ &= \|V_r\|\|X_r\|\sin(\theta). \end{aligned} \tag{4.22}$$

There are three scenarios for which $\det(\mathscr{O}_p) = 0$; moving directly toward or away from the target ($v_{yr}x_r = v_{xr}y_r$), being directly on top of the target ($x_r = y_r = 0$), and matching the target velocity vector exactly ($v_{yr} = v_{xr} = 0$). Each of these cause issues with local weak observability because neighboring states will be indistinguishable from the output alone. For instance, if we consider discrete time-steps, the measurements alone will not allow determination of the relative position if $x_r = y_r = 0$ since every point on a circle corresponding to the range measurement

would produce the same output. The same issue would occur if the relative velocity is always such that $v_{yr} = v_{xr} = 0$. If $v_{yr}x_r = v_{xr}y_r$, the measurements will not reveal whether the unicycle is moving behind the target and moving faster than it or in front of the target and moving slower than it. It is important to recall that local weak observability is being enforced and that if local weak observability is not continuously enforced, it is possible (though more difficult to verify analytically) that observability can still be achieved [97].

Notice that the observability matrix is written only in terms of the relative position and relative velocity, making the results agnostic to the particular form of the kinematics if the relative velocity is available. However, the target position, and therefore, the target velocity may not be available in many cases. When the target position is not known, the relative velocity has to be obtained using the range measurement. One way to do this is by taking the unicycle velocity and heading as known and estimating the target velocities using the unicycle model and the range measurement.

While conditions that will make the system observable can be identified, in general designing a controller that satisfies the conditions and is able to track the target may be difficult. However, using an observability-based barrier function, one can modify a predefined tracking controller to satisfy the nonlinear observability rank condition while still tracking the target. This is demonstrated in simulations and in experimentation in the subsequent sections.

4.4 Simulations

In simulation, the target follows a predefined time-dependent trajectory $(x_{ta}(t), y_{ta}(t))$. The range between the target and the unicycle, the unicycle velocity, and the unicycle heading are measured. The measurements are corrupted with additive, zero mean, Gaussian noises and an extended Kalman filter (EKF) is used to estimate the relative position between the target and the tracker. A

secondary Kalman filter is used to estimate the target velocities using the target position as measurements. The unicycle has a maximum magnitude of $\frac{\pi}{4}$rad/s for the turn rate u_1, and a maximum magnitude of 0.01 m/s^2 for the acceleration u_2.

A pre-defined controller U_{nom} that performs well under state feedback is used to achieve tracking. The controller is designed to follow the target with an offset ε to avoid the bearing becoming undefined if $x_r = y_r = 0$ and is given as

$$U_{nom} = \begin{bmatrix} -k_\psi(\psi - \arctan(y_l, x_l)) \\ k_\rho(\sqrt{x_l^2 + y_l^2})\cos(\psi - \arctan(y_l, x_l)) \end{bmatrix}$$

where $x_l = -x_r - \varepsilon\cos(\psi)$, $y_l = -y_r - \varepsilon\sin(\psi)$. The constant parameters k_ρ, k_ψ, and ε are taken as 0.1, 1, and 0.25, respectively. The first component of U_{nom} is chosen so that the heading of the unicycle will point toward the target, while the second component is chosen to accelerate toward or away from the target based on the heading and range from the desired tracking offset. Note that the equation for U_{nom} only describes the form of the controller. The controller would be implemented with the state variables replaced with their estimates based on the measurements.

Before comparing the proposed approach with the MPC-based approach, simulations comparing four different settings (state feedback vs output feedback, and whether to include CBF or not) are conducted to gain insight into the behavior of the closed-loop system including the role of CBF under the proposed method. Fig. 4.1 shows the diagrams for the state feedback and output feedback control loops augmented with the CBF. While clearly one can only use output feedback in a practical scenario, the state feedback cases are included to provide a performance benchmark for comparison (since it would be the best possible case), as well as help one see the role of observability requirement in the system behavior by separating the effect of imperfect state estimation from the controller implementation. For this reason, the EKF is run using the range, the linear

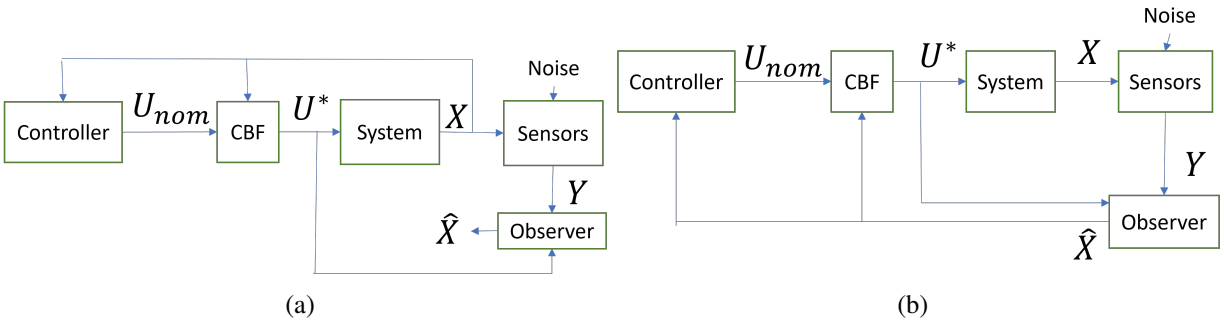


Figure 4.1: Control loop diagrams for (a) state feedback and (b) output feedback schemes with observability-based CBF.

velocity, and the heading measurements even when using state feedback. For a fair comparison between the different cases, the noise distribution, the initial state, and the observer parameters are held constant for each simulation scenario.

The performance of the controllers is quantified in terms of time-averaged estimation error ($\frac{1}{T}\int_0^T \|\kappa_e\| dt$), time-averaged tracking error ($\frac{1}{T}\int_0^T \|\tau_e\| dt$), and cumulative inverse condition number over the simulated trajectory ($\int C^{-1} dt$). Cumulative and time-averaged performance metrics are used as opposed to instantaneous ones so that insight about the entire trajectory is incorporated. The cumulative inverse condition number can be thought of as capturing a cumulative measure of observability that increases if information about the state is gained and remains flat otherwise.

4.4.1 Simulation Results on Different Control and CBF Settings

The noise variance for the range, the unicycle velocity, and the unicycle heading measurements are set to 0.1 m^2, 0.1 $(\text{m/s})^2$, and 0.0175 rad^2, respectively. Fig. 4.2 (a)-(d) show the resulting trajectories of the tracker for four cases: with and without the control barrier function, and when using state and output feedback, respectively. Fig. 4.2 (e) shows a case similar to that of Fig. 4.2 (d). The difference between the two is that v_{xr} and v_{yr} are evaluated by using known target velocities in Fig. 4.2 (d) and by estimating the target velocities in addition to the relative position

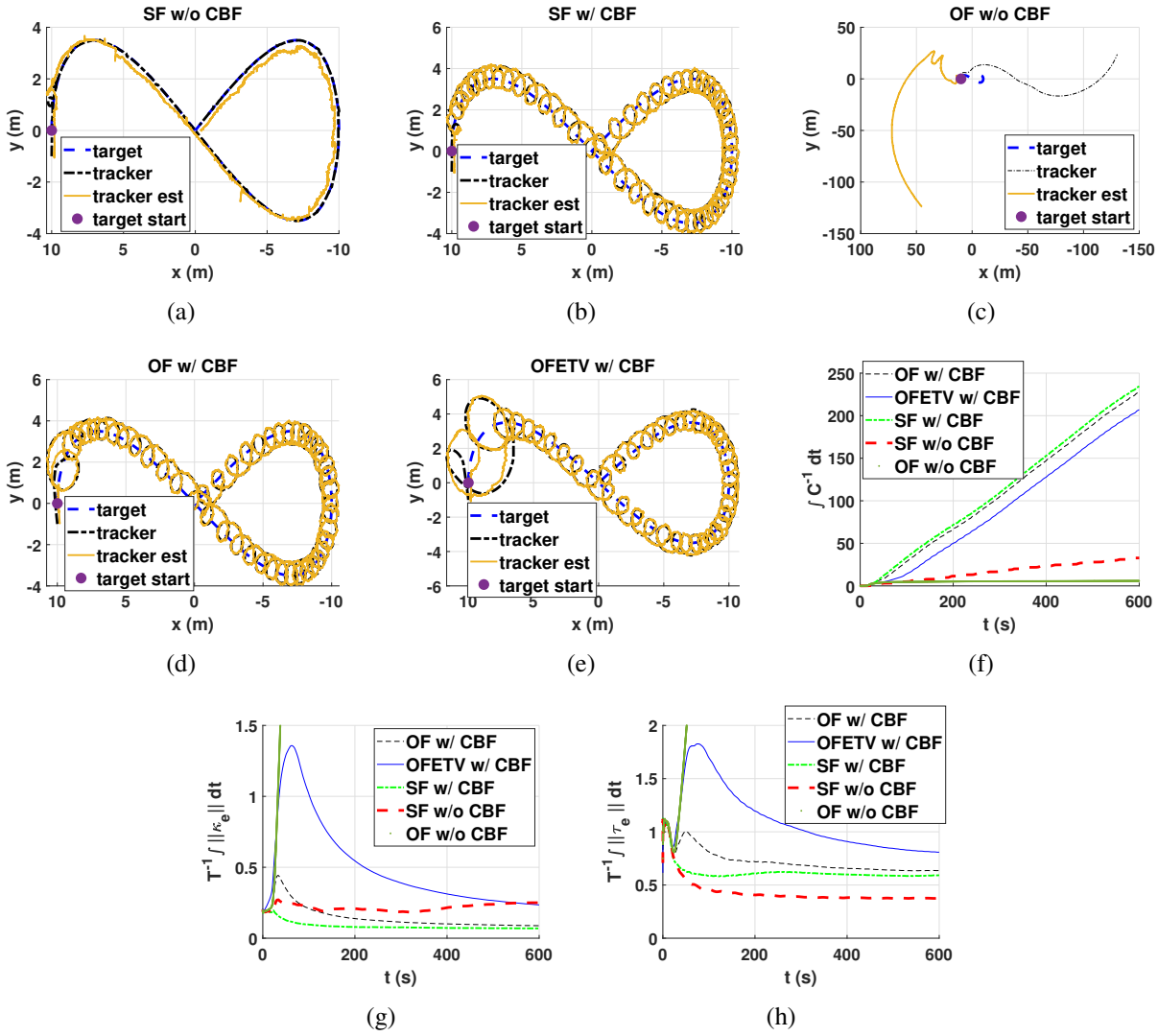


Figure 4.2: Simulation comparison of five controllers that focuses on the role of CBF: state feedback without CBF (SF w/o CBF)), state feedback with CBF (SF w/CBF), output feedback without CBF (OF w/o CBF), output feedback with CBF and known target velocities (OF w/CBF), and output feedback with CBF and estimated target velocities (OFETV w/CBF). (a)-(e) showing the target trajectory (target), the tracker trajectory, and the estimated tracker trajectory (tracker est) under different controllers. (f) showing the cumulative inverse condition number over time, (g) showing the time-averaged estimation error over time, and (h) showing the time-averaged trajectory over time.

in Fig. 4.2 (e). Fig. 4.2 (a)-(e) also depict the corresponding estimated trajectory and the true target trajectory, respectively. From the simulation results, the tracker is able to track the target in four of the five cases (state feedback without CBF, state feedback with CBF, and output feedback with CBF with the target velocities known or estimated) but failing when using output feedback without the CBF. While the nominal controller under state feedback tracks the target by moving straight toward it (Fig. 4.2 (a)), augmenting it with the CBF causes an orbiting behavior that circles the target as it moves (Fig. 4.2 (b)). The same behavior is observed when the output feedback is used while incorporating the observability-based CBF (Fig. 1 (d) and (e)). The orbiting behavior has been reported in previous work localizing with respect to a single beacon [34, 45] and corresponds with $\theta = \pm\frac{\pi}{2}$ in Eq. (4.21) which was shown to be a maximizer for a given range and velocity magnitude in [34]. In cases where this orbiting behavior is not acceptable, one could tune the extended class K_∞ function α to modify the orbiting behavior. An example of this is shown for a static target in Appendix C. The unicycle model may also be replaced with a more accurate model of the system being used which may reveal additional conditions that can be used to ensure that the system's observability matrix is full rank. The nominal controller fails to track the target under output feedback without the CBF, due to divergence of the EKF state estimates (Fig. 4.2 (c)), which shows the significance of incorporating the observability in output feedback control.

Fig. 4.2 (f)-(h) show how the cumulative inverse condition number, the time-averaged estimation error, and the time-averaged tracking error evolve over time for a single run. The inverse condition number is indicative of how well posed the estimation problem is, with a higher number being better. The cumulative inverse condition number remaining flat for the case of output feedback without the CBF indicates the condition number is zero or near zero for most of the trajectory. The higher rates for the cases when the controller is augmented with the CBF indicate that, for most of the trajectory, the inverse condition number is higher than that of the case of state feedback with-

out the CBF. The time-averaged estimation error has an inverse relationship with the cumulative inverse condition number for most of the trajectory, except that during the initial transients the estimation error for the cases of the output feedback augmented with the CBF is relatively large before dropping. From Fig. 4.2 (h), state feedback alone achieves the best tracking performance (unsurprisingly). When the state feedback controller incorporates observability requirement via the CBF, the tracking performance degrades as the system deliberately deviates from the desired trajectory to ensure the system observability. For both output feedback cases incorporating the CBF, the time-averaged state estimation error and tracking error are bounded, where the case with the target velocity estimated shows poorer performance as expected. When the CBF is not incorporated, both the estimation error and the tracking error quickly grow unbounded.

Table 4.1 shows the statistics for 10 trials for the metrics at the last time step of the simulation. The initial position estimate is perturbed from the true state by a zero-mean Gaussian distribution with a variance of 0.065 m^2 for each component in the trials. The results mirror those of the single trials shown in Fig. 4.2. When using the CBF under output feedback, the time-averaged estimation error performance is slightly worse than the case with the CBF under state feedback. The performance decreases further under output feedback with estimated target velocities (output-ETV). The time-averaged estimation error is roughly 3 of that under output feedback with CBF (and known target velocities) when using state feedback without the CBF. This implies that the estimation performance is improved when the controller is paired with the observability-based control barrier function. On average, the cumulative inverse condition number is highest when using the CBF with state feedback, closely followed by using the CBF with output feedback. It is lowest under output feedback with no CBF. The time-averaged tracking error is lowest when using only the tracking controller with state feedback, but under output feedback without the CBF, the tracking performance is worst. With the addition of the CBF, the time-averaged tracking error, under both

state and output feedback (even when the target velocities are estimated), is slightly larger than but comparable to that under just state feedback without CBF. The time-averaged tracking error is higher with the addition of the CBF due to the orbiting maneuvers induced by the CBF. This prevents the unicycle from perfectly tracking the target.

Table 4.1: Statistics of performance metrics for 10 simulation runs for each case, showing both the mean and the standard deviation for each metric.

CBF	no	no	yes	yes	yes
Feedback type	state	output	state	output	output-ETV
mean $\frac{1}{T}\int_0^T \|\kappa_e\| dt$	0.26	100.1	0.07	0.08	0.14
std $\frac{1}{T}\int_0^T \|\kappa_e\| dt$	0.027	10.3	0.001	0.009	0.049
mean $\frac{1}{T}\int_0^T \|\tau_e\| dt$	0.37	54.6	0.59	0.62	0.70
std $\frac{1}{T}\int_0^T \|\tau_e\| dt$	0.001	4.9	0.002	0.016	0.065
mean $\int C^{-1} dt$	32.8	8.3	234.4	230.1	219.0
std $\int C^{-1} dt$	0	2.2	0	2.9	8.6

4.4.2 Comparison with the MPC-Based Approach

4.4.2.1 Overview of the MPC-based Approach

In [103], an alternative method that jointly optimizes a cost function for the control objective and a surrogate function for the state estimation performance was proposed. The proposed CBF-based

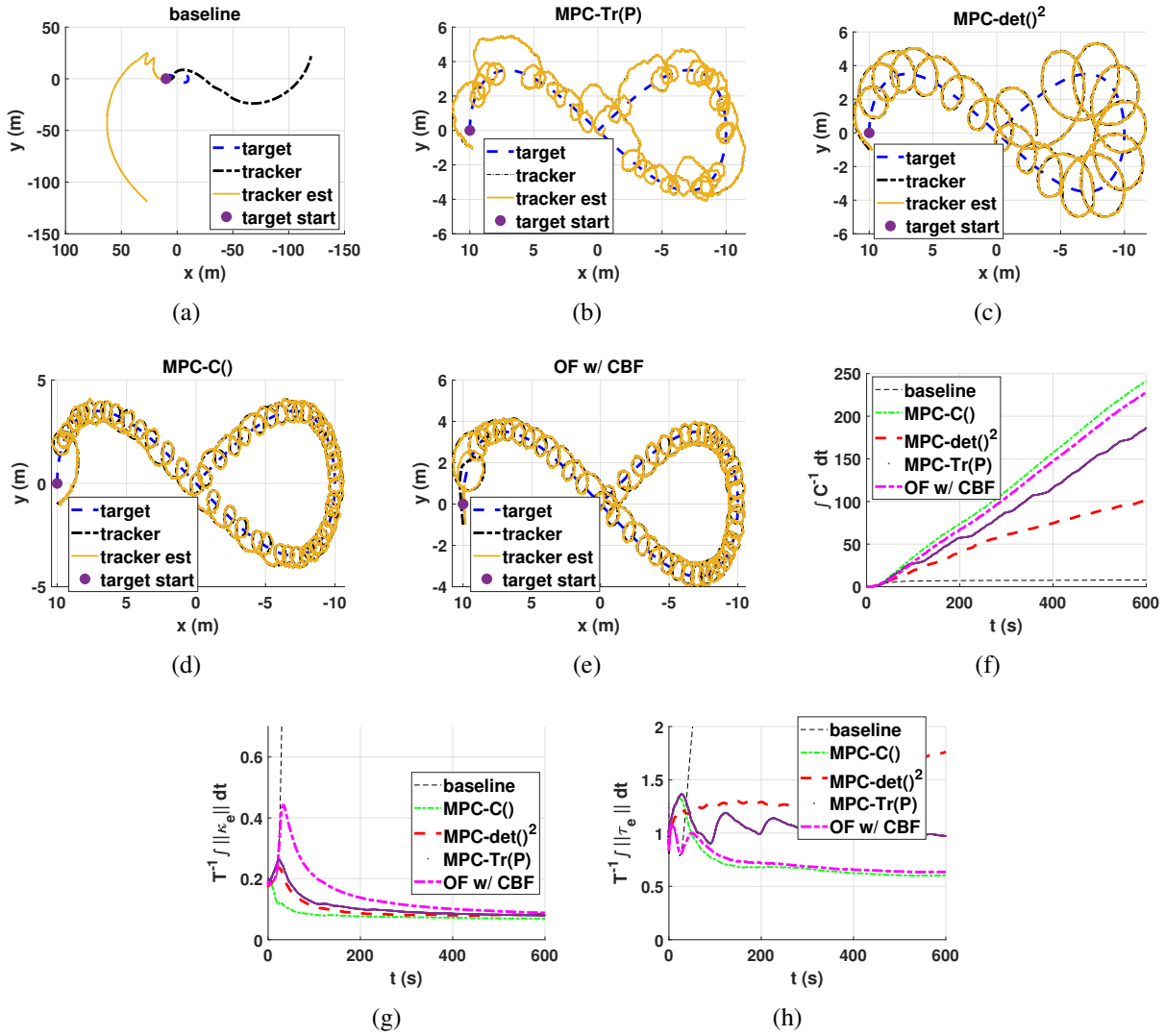


Figure 4.3: Simulation results for five controllers that focus on the comparison of the proposed method with the MPC-based approach: (a)-(e) showing the target trajectory (target), the tracker trajectory, and the estimated tracker trajectory (tracker est) under different controllers: (a) baseline MPC ($O_0 = 0$), (b) MPC with the trace of the covariance matrix $O_1 = Tr(P)$, (c) MPC with the negative determinant squared $O_2 = -\det()^2$, (d) MPC with condition number $O_3 = C()$, and (e) output feedback with CBF. (f) showing the cumulative inverse condition number over time, (g) showing the time-averaged estimation error over time, and (h) showing the time-averaged trajectory over time.

approach is compared to this alternative method. The alternate method can be stated as

$$\begin{cases} \min\limits_{U(\cdot)} \quad J_M(X_0,U(\cdot)) = \int_{t_0}^{t_1} aL(X(t),U(t)) \\ \qquad\qquad + (1-a)O(X(t),U(t))\mathrm{d}t \\ \text{subject to} \\ \qquad\qquad X(t_0) = X_0 \\ \qquad\qquad \dot{X} = f(X,U) \\ \qquad\qquad q(X,U) = 0 \end{cases} \tag{4.23}$$

where $a \in [0,1]$ is a weighting parameter, $L(X,U)$ is a running cost for the control objective, and $O(X,U)$ is a surrogate function for observability for minimizing the estimation error.Three different choices for $O(X,U)$ are considered: the trace of the estimation error covariance matrix $O_1(X,U) = Tr(P)$ of the EKF[6], the negative squared determinant of the observability matrix $O_2(X,U) = -\det(\mathscr{O}_p)^2$, and the condition number of the observability matrix $O_3(X,U) = C(\mathscr{O}_p)$. The function q captures additional constraints. Taking $L(X,U)$ as $\|\tau_e\|^2$, MPC can be used to numerically solve the optimization problem in (4.23).

The MPC-based approach was tested using state feedback to tune the parameter a for each of the surrogate functions O_i until satisfactory performance was achieved. The MPC-based approach is then used under output feedback with each surrogate function and compared to the CBF-based approach under output feedback in Section 4.4.1. We use the Nonlinear MPC tool box in Matlab [104] to solve problem (4.23) with the cost function $J(X_0,U(\cdot)) = \int_{t_0}^{t_1} a\|\tau_e\|^2 + (1-a)O_i\mathrm{d}t$ for the MPC-based approach. It is worth noting that when $O_i = Tr(P)$, the cost is modified as $J_M(X_0,U(\cdot),P)$ to take the current estimation error covariance as an argument to act as the initial

[6]Note that P is not directly a function of X and U, but rather a function of the initial state estimate and the measurements Y.

covariance in the optimization problem. In addition, the EKF equations become a set of extra equality constraints used to predict the output of the EKF in the optimization problem. For all MPC-based methods, the prediction horizon is 1 second with 5 time steps of 0.2 seconds and the reference position of the target is considered static over the optimization period. The optimization is limited to 10 iterations, which provides a good trade-off between the execution speed and the accuracy of the solution to the optimization problem.

4.4.2.2 Comparison with the MPC-Based Approach and Discussion

Fig 4.3 (a)-(e) shows the tracker paths generated for the 4 MPC-based controllers and the CBF-based controller, the estimated paths, and the associated performance metrics of a single trial (for each controller). Table 4.2 shows the statistics of the performance metrics over 10 trials. With the exception of the baseline MPC controller, which does not consider the observability requirement, tracking is consistently achieved among the MPC controllers. Minimizing the condition number produces the most well-rounded results for the MPC approach.

From both Fig. 4.3 and Table 4.2, the proposed CBF-based offers estimation and tracking performance comparable to the successful MPC schemes, but at a fraction of computational cost, as can be seen in Table 4.3. The CBF-based approach is at least an order of magnitude faster since the pre-defined controller in Section 4.4 has a closed form and the quadratic program can be solved efficiently. The baseline MPC could also be used as the nominal controller for the CBF-based approach, which would make its computation time close to the baseline MPC, but still faster than jointly optimize the tracking cost and surrogate functions of the estimation error.

In addition to the computational advantage, the CBF also removes the need to tune the parameter a in the MPC-based approach, is relatively simpler to implement, and has the benefit that the theoretical guarantees of CBFs can be used to ensure observability while the MPC-based approach

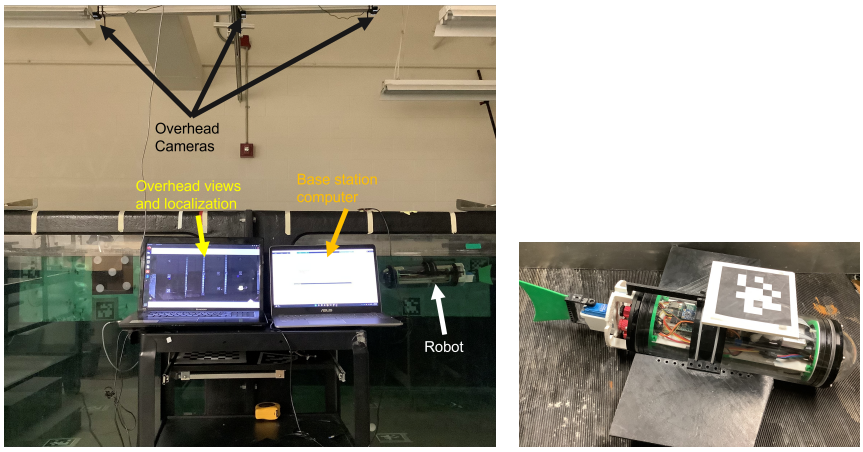
does not readily provide guarantees. However, the MPC-based approaches can provide better initial performance if tuned well. This can be seen in Figs. 4.3 (b), (c), and (d) when the MPC-based approaches begin their trajectories moving in arc-like paths while the CBF-based approach has an initial path that more closely resembles moving directly toward the target as the baseline controller does.

Table 4.2: Statistics of performance metrics for simulation comparison between the MPC approach and the CBF approach over 10 trials for each controller. For the MPC approach, four cases are considered: $O_0 = 0$, $O_1 = Tr(P)$, $O_2 = -\det(\mathscr{O})$, $O_3 = C(\mathscr{O})$. Both the mean and the standard deviation are shown for each controller.

	O_0	O_1	O_2	O_3	CBF
mean $\frac{1}{T}\int_0^T \|\kappa_e\| dt$	96.6	0.09	0.08	0.07	0.08
std $\frac{1}{T}\int_0^T \|\kappa_e\| dt$	10.7	0.005	0.011	0.002	0.009
mean $\int \|\tau_e\| dt$	53.1	1.16	1.36	0.61	0.62
std $\frac{1}{T}\int_0^T \|\tau_e\| dt$	4.9	0.165	0.634	0.007	0.016
mean $\int C^{-1} dt$	9.2	167.6	145.6	239.2	230.1
std $\int C^{-1} dt$	3.9	11.4	54.5	1.9	2.9

Table 4.3: Average computation time per step for four different MPC controllers and the proposed CBF-based approach in simulation. Here $O_0 = 0$, $O_1 = Tr(P)$, $O_2 = -\det(\mathscr{O})$, $O_3 = C(\mathscr{O})$. The evaluation is based upon an 8th generation Intel Core i5 processor.

	Mean computation time for controllers
$MPC(O_0)$	0.036 s
$MPC(O_1)$	0.182 s
$MPC(O_2)$	0.051 s
$MPC(O_3)$	0.051 s
CBF	0.003 s



(a) (b)

Figure 4.4: Photos of (a) the experimental setup and (b) the robot.

4.5 Experiments

The CBF-based approach is further validated with experiments using the miniature gliding robotic fish and large indoor tank discussed in Sections 2.2.1 and2.2.2, respectively. The three overhead cameras and an AprilTags attached to the robot are used to localize the robot within the tank, to collect the ground truth information on the robot's position, and to produce the (emulated) range measurements. The range measurements are calculated from the x, y positions output by the AprilTag algorithm and the position of a virtual target. Pictures of the experimental setup and the robot are shown in Fig. 4.4. The robot implements a time-dependent sinusoidal motion for the tail and accepts bias, frequency, and amplitude inputs for the sinusoidal function from a base station computer. In the experiments, the tail-beat frequency is held constant, while the base station computer sends the bias and amplitude commands to the robot. The base station computer uses Matlab to retrieve the localization data and to calculate the desired acceleration and turn rate, treating the robot as a unicycle. PID controllers generate bias and amplitude outputs based on the difference between the acceleration and turn rate inputs from the controller and the estimated

values from the camera-based localization.

Experiments are carried out in a manner similar to the simulations in Section 4.4.2.1 and the MPC-based approaches use the same settings as the MPC controllers in the simulations. Five trials are run for each controller under output feedback. The initial position estimate is randomly chosen from a Gaussian distribution around the true position with the variance taken as $0.125\ \mathrm{m}^2$ for both of the components of the position. The controllers are set identical to the ones used in simulation, but with control constraints set as $|u_1| \leq \frac{\pi}{4}$ and $0 \leq u_2 \leq 0.01$.

Fig 4.5 shows the robot paths generated under the MPC-based controllers using the different surrogate functions as well as the proposed CBF-based controller, the estimated robot paths produced in each case, and the target path for a single trial (under each controller). In addition, it depicts the associated performance metrics while the statistics of the performance metrics over all 5 trials for each controller is shown in Table 4.4.

Although the overall performance in experiments is not as good as in simulation, the robot is still consistently able to track the target to some extent while maintaining a satisfactory estimate of its position when using the observability-based control strategies. The trajectories produced when using the CBF-based approach and the MPC-based approach with $O = C(\mathscr{O}(X,U))$ share the orbiting behavior of their simulated counterparts, but with much larger, more irregular orbits. The mismatch is due to the robot's limited maneuverability and inability to fully emulate a unicycle. Particularly, the model does not perfectly capture the robot dynamics and the oscillatory movement of the robot's servo-actuated tail is not always able to perfectly execute the strategies calculated by the different controllers. The CBF-based approach and the MPC-based approach with $O = C(\mathscr{O}(X,U))$ produced the highest cumulative inverse condition numbers. The other observability-based controllers have less of a resemblance to their simulated counterparts, but still produce satisfactory tracking and estimation performance. Table 4.4 shows that the average track-

ing performance is similar for all of the observability-aware controllers. When using $O = Tr(P)$, there is a slightly higher estimation error than the other observability-aware controllers. This may be due to the approach taking significant time to calculate the control input as shown in Table 4.5. The calculation time for the control input is roughly 3 times of those of the other MPC-based metrics and about 10 times of that of the CBF-based method. Without incorporating observability, the tracking error and the estimation error tends to grow until the robot is no longer in the field of view of the cameras.

Table 4.4: Statistics of performance metrics for experimental comparison between the MPC approach and the CBF approach over 5 trials for each controller. For the MPC approach, four cases are considered: $O_0 = 0$, $O_1 = Tr(P)$, $O_2 = -\det(\mathscr{O})^2$, $O_3 = C(\mathscr{O})$.

	O_0	O_1	O_2	O_3	CBF
mean $\frac{1}{T}\int_0^T \|\kappa_e\| dt$	0.65	0.12	0.07	0.07	0.07
std $\frac{1}{T}\int_0^T \|\kappa_e\| dt$	0.486	0.077	0.014	0.022	0.016
mean $\frac{1}{T}\int_0^T \|\tau_e\| dt$	8.5	0.8	0.8	0.7	0.7
std $\frac{1}{T}\int_0^T \|\tau_e\| dt$	9.6	0.038	0.04	0.038	0.051
mean $\int C^{-1} dt$	17.6	26.7	29.1	38.4	43.8
std $\int C^{-1} dt$	13.27	0.96	3.98	4.0	7.72

Table 4.5: Mean time for control calculation during experiments with $O_0 = 0$, $O_1 = Tr(P)$, $O_2 = -\det(\mathscr{O})^2$, $O_3 = C(\mathscr{O})$.

	Mean computation time for controllers
$MPC(O_0)$	0.021 s
$MPC(O_1)$	0.127 s
$MPC(O_2)$	0.036 s
$MPC(O_3)$	0.042 s
CBF	0.011 s

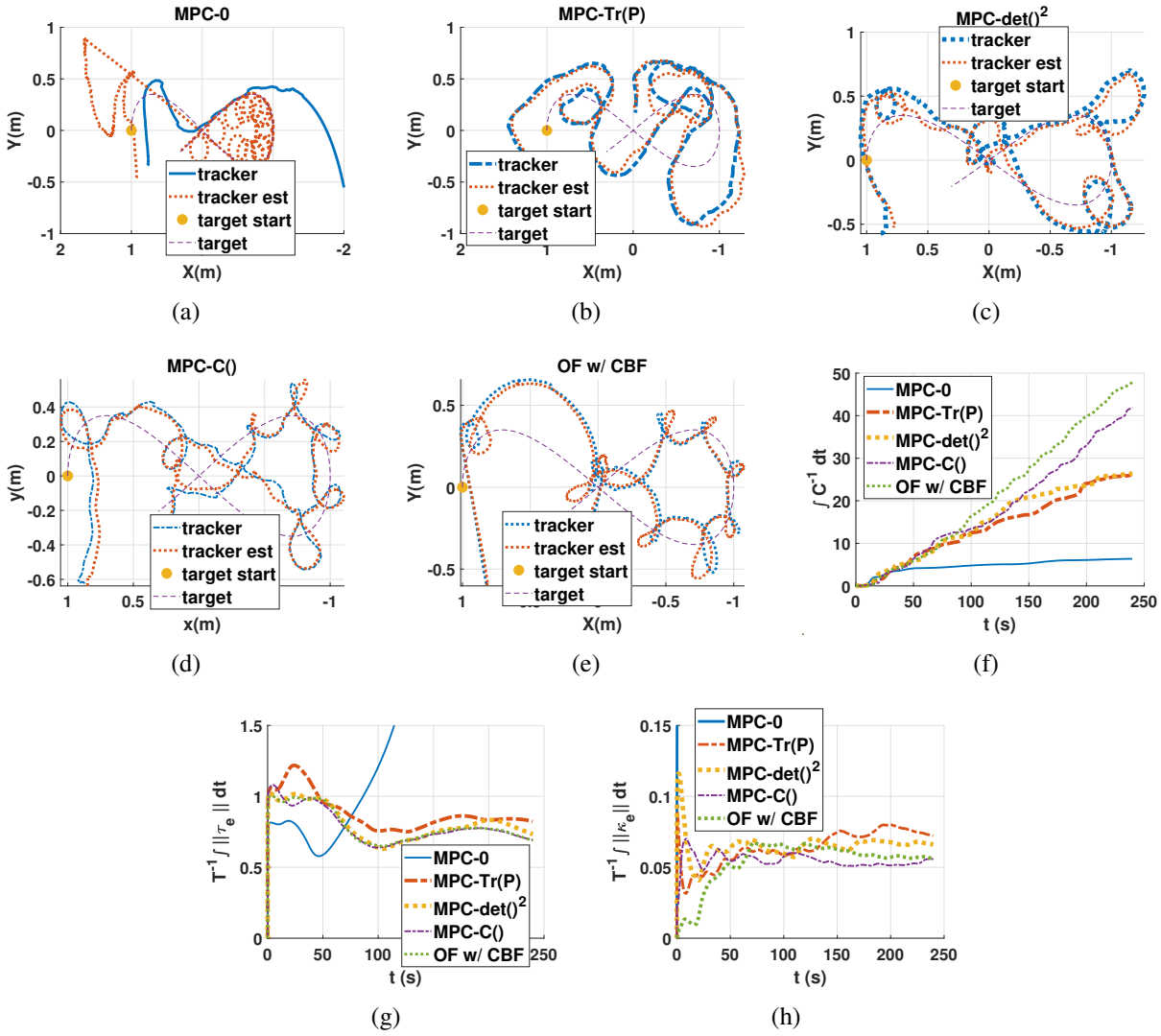


Figure 4.5: Experimental comparison of five controllers. (a)-(e) showing the target trajectory (target), the tracker trajectory, and the estimated tracker trajectory (tracker est) under different controllers:(a) baseline MPC ($O_0 = 0$), (b) MPC with the trace of the covariance matrix $O_1 = Tr(P)$, (c) MPC with the negative determinant squared $O_2 = -\det()^2$, (d) MPC with condition number $O_3 = C()$, and (e) CBF approach. (f) showing the cumulative inverse condition number over time, (g) showing the time-averaged estimation error over time, and (h) showing the time-averaged trajectory over time.

4.6 Conclusion and Future Work

In this work, the problem of output feedback tracking for nonlinear systems while accommodating observability requirements was considered. The motivating application was the problem of target tracking under range measurement when the tracker's position is unknown. The observability of a unicycle model was analyzed and it was shown how the proposed approach can be applied. Simulations and experiments were carried out to quantify the efficacy of the controller. The proposed approach using control barrier functions to enforce observability was compared to an approach that jointly optimizes the control objective and an observability metric using MPC. The MPC-based approach may provide more flexibility in how observability is optimized but is more computationally demanding for systems with large state spaces or that need long optimization horizons. The MPC-based approach also requires a careful selection of a weight that balances observability and the control objective in order to obtain acceptable performance. Deviating from the target trajectory tends to result in lower estimation error, but higher tracking error. Finding a weighting parameter that balances this appropriately can take time, and such a parameter may vary with different operating conditions and with the choice of the observability surrogate function. The CBF-based approach circumvents the computational complexity and manual tuning by enforcing the forward-invariance of an observable set through the CBF constraint at every control instant. However, the MPC-based approach only requires the observability matrix or the covariance matrix, and the associated functions can be numerically computed during the optimization, while the CBF-based approach requires the closed-form expression for the barrier function. While this is not a problem when one uses the determinant squared, a closed form for the condition number may be difficult to find and it is not immediately clear how one would use the trace of the estimation error covariance matrix as a barrier function. Combining the benefits of the proposed approach

with those of MPC in a principled manner would be an interesting direction for future work.

Beyond the application of range-based target tracking, CBF-based observability enforcement can be generally useful for any control system with a well-designed observer and a state-dependent and/or control-dependent observability matrix. Given that several observer designs such as the Luenberger-like observers in [24] and extended Kalman filters rely on the assumption that the observability matrix is full rank, the most useful application may be the estimation and control of unmeasured state variables for systems with nonlinear coupling between the output and state of interest. Examples include velocity tracking for autonomous underwater vehicles or drones in the absence of velocity measurements and operating in GPS-denied environments, estimation of disturbances such as wind for fixed-wing aircraft, or control of bio-inspired robots with coupled actuation and sensing. It would also be interesting to extend this approach to parameter estimation given the ability to model parameters as states with no dynamics. One additional application is to study system trajectories needed to obtain observability for a particular measurement function given state feedback and a nominal control objective.

In this work, the extended Kalman filter is selected to obtain state estimates. For future work, it would be of interest to test performance under different types of observers such as particle filters or high-gain observers. In addition, obtaining theoretical insight into the effect of estimation error using these approaches is of interest. Finally, designing a control barrier function that enforces observability and incorporates measurement noise and estimation error statistics would make the approach robust in stochastic, high-noise settings.

Chapter 5

Exploration of Unkown Scalar Fields using Mult-ifidelity Gaussian Process Regression Under Localization Uncertainty

This chapter considers a gliding robotic fish moving in a 3D environment that is tasked with reconstructing a spatial process using Gaussian process (GP) regression. Section 5.1 gives a brief review of GP regression. In Section 5.2, the problem is described and the use of multi-fidelity Gaussian process regression is proposed to incorporate data associated with uncertain locations. An expanded model of the gliding robotic fish is given in Section 5.3. Section 5.4 provides a description of an informative trajectory planner. Using the proposed approach and the trajectory planner, an adaptive sampling algorithm is developed for the exploration and mapping of unknown scalar fields. In Section 5.5, the adaptive sampling algorithm is tested experimentally on the Miniglider robot and the field reconstruction performance is quantified via the weighted mean squared error between the model prediction and the true field. The results show that using a multi-fidelity GP provides improvements over using a single-fidelity GP to model the field.

5.1 Review of Gaussian Process Regression

GP regression is a tool for function approximation that is commonly used to reconstruct spatial fields. It is popular due to its basis in Bayesian statistics and ability to not only predict the value of a function at an unmeasured sample point, but also provide a confidence level associated with the prediction.

A GP is fully specified by a mean function $m(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}$, and a covariance function $K(\cdot,\cdot;\theta) : \mathbb{R}^{d\times a} \times \mathbb{R}^{d\times b} \times \mathbb{R}^p \to \mathbb{R}^{a\times b}$ with elements $K_{ij}(\mathbf{x},\mathbf{x}';\theta) : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}_{\geq 0}$ (the dependence on θ will be suppressed for brevity) and hyperparameters $\theta \in \mathbb{R}^p$, for any input vectors $\mathbf{x}$, $\mathbf{x}' \in \mathbb{R}^d$. Given a set of input vectors $\mathbf{X} = [\mathbf{x}^{[1]},\ldots,\mathbf{x}^{[n]}]$ and an associated set of scalar measurements $\mathbf{y} = [\tilde{v}^{[1]},\ldots,\tilde{v}^{[n]}]^T$ assumed to have an additive, zero-mean Gaussian noise, the posterior mean and variance of a Gaussian process can be predicted at any set of test input $\mathbf{X}^* = [\mathbf{x}^{*[1]},\ldots,\mathbf{x}^{*[q]}]$. The posterior mean $\mu(\mathbf{X}^*)$ and variance $\Sigma(\mathbf{X}^*)$ at inputs $\mathbf{X}^*$ can be predicted via the equations (see [105] for more details)

$$\begin{aligned}\mu(\mathbf{X}^*) &= m(\mathbf{X}^*) \\ &\quad + K(\mathbf{X}^*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2 I_n]^{-1}(\mathbf{y} - m(\mathbf{X}))\end{aligned} \tag{5.1}$$

$$\begin{aligned}\Sigma(\mathbf{X}^*) &= K(\mathbf{X}^*,\mathbf{X}^*) \\ &\quad - K(\mathbf{X}^*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2 I_n]^{-1}K(\mathbf{X},\mathbf{X}^*),\end{aligned} \tag{5.2}$$

where σ_n^2 is the measurement noise variance.

In order for the GP to be a generative model of the data, the hyperparameters θ must be chosen appropriately. If these are not known in advance, they can be learned by maximizing the log

marginal likelihood of the observations. The optimal hyperparameters can be found as

$$\theta^* \in \text{argmax}_{\theta \in \mathbb{R}^p} \{-\frac{1}{2}\mathbf{y}^T K^{-1}\mathbf{y} - \frac{1}{2}\log|K|\}, \tag{5.3}$$

where $|K|$ represents the determinant of K. The objective function in Eq. (5.3) can be optimized via gradient methods [105]. Note that since this is a non-convex problem, the resulting θ^* is usually computed using multi-start gradient ascent.

5.2 Problem Statement

Consider a robot that moves in the 3D space, i.e., $\mathbf{x} \in \mathbb{R}^3$. The vehicle is assumed to have a state space $X \in \mathbb{R}^p$ with $\mathbf{x}$ being a subset of the states in X, control inputs $U \in \mathbb{R}^l$, and continuous nonlinear dynamics with state-dependent measurements available at discrete time instances t_i, i.e

$$\begin{cases} \dot{X} = f(X,U) \\ Y_i = h(X(t_i)) \end{cases} \tag{5.4}$$

for a non-negative integer i. The vehicle is tasked with sampling and reconstructing a static scalar measurement field using a point sensor and is assumed to have resource constraints such as budget on the total time or total energy. We assume that there exists a measurement model

$$\bar{v} = \zeta(\mathbf{x}), \tag{5.5}$$

where $\zeta(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$ is an unknown function that perfectly describes the measurement field with input $\mathbf{x}$. It is also assumed that its measurement $\tilde{v}$ at any $\mathbf{x}$ is corrupted by a zero mean Gaussian

noise ε_v with variance σ_n^2, i.e.,

$$\tilde{v} = \zeta(\mathbf{x}) + \varepsilon_v. \tag{5.6}$$

In addition, the robot is assumed to have access to its location in the horizontal plane (x, y) only at the surface of the body of water, but experiences localization errors when sensing underwater due to attenuation of the GPS signal. At each time step, the robot's position estimate $\hat{\mathbf{x}} = \mathbf{x} + \varepsilon_x$ (generated from a state estimator such as a Kalman filter) of its true location $\mathbf{x}$ is assumed to have a Gaussian estimation error ε_x with zero mean and a diagonal covariance matrix and the elements of the covariance matrix are assumed to increase with time until the vehicle receives a position measurement. As a result, a dataset $\mathbf{z} = (\mathbf{y}, \hat{\mathbf{X}})$ is collected, where the vector of field measurements $\mathbf{y}$ is defined as in Section 5.1 and $\hat{\mathbf{X}} = [\hat{\mathbf{x}}^{[1]}, ..., \hat{\mathbf{x}}^{[n]}]$. Using the dataset $\mathbf{z}$, an estimate $\hat{\zeta}(\mathbf{x}; \mathbf{z})$ of the measurement field $\zeta(\mathbf{x})$ can be constructed. The end goal of this work is to enable the robot to autonomously collect a dataset $\mathbf{z} = (\mathbf{y}, \hat{\mathbf{X}})$ such that $\hat{\zeta}(\mathbf{x}^*; \mathbf{z})$ sufficiently approximates $\zeta(\mathbf{x}^*) \forall \mathbf{x}^* \in \mathbf{X}^*$, where $\mathbf{X}^*$ is a set of locations that sufficiently cover the measurement field.

5.2.1 Field Surrogate Model

In this work, GP regression is used to estimate the field $\zeta(\mathbf{x})$. One drawback of GP regression is that it assumes the input $\mathbf{x}$ to be precisely known. For this work, the input (location of the vehicle) is perturbed by the estimation error $\varepsilon_{\mathbf{x}}$. It is shown in [59] that if the measurements in a GP are accessed with inputs perturbed by a Gaussian noise, then these measurements correspond to another GP defined over perturbed inputs. These perturbed inputs lead to a warped prediction of the measurement field when using GP regression, but the robot needs to accurately predict the true function $\bar{v} = \zeta(\mathbf{x})$ for each location $\mathbf{x} \in \mathbf{X}^*$ to effectively plan new sample locations. The estimation error $\varepsilon_{\mathbf{x}}$ corresponds to variable levels of uncertainty on the position estimate $\hat{\mathbf{x}}$. The

dataset $\mathbf{z} = (\mathbf{y}, \hat{\mathbf{X}})$ can be split into separate datasets based on the level of uncertainty in $\hat{\mathbf{X}}$. This motivates the idea of using a multi-fidelity GP to create the estimate $\hat{\zeta}(\mathbf{x}; \mathbf{z})$ for all $\mathbf{x}^* \in \mathbf{X}$ of the measurement field.

A multi-fidelity GP representation of the field whose fidelity levels are dependent on the localization uncertainty is proposed. In the proposed approach, uncertainty in the location estimates is binned into $M_f + 1$ fidelity levels, where location uncertainty decreases with the fidelity level, i.e, level M_f corresponds to the lowest uncertainty. Multiple datasets $(\mathbf{y}_i, \hat{\mathbf{X}}_i)$ are constructed where measurements and input locations are assigned to a particular dataset based on the uncertainty in the input location. Each dataset will be used for prediction in a corresponding GP and these GPs are coupled through a nested structure on their means and covariance. The mean of the GP at each fidelity level is expressed as

$$\begin{aligned} \mu_i(\mathbf{X}^*) &= \rho_{i-1}\mu_{i-1}(\mathbf{X}^*) \\ &+ K_i(\mathbf{X}^*, \hat{\mathbf{X}}_i) K_i(\hat{\mathbf{X}}_i, \hat{\mathbf{X}}_i)^{-1} (\mathbf{y}_i - \rho_{i-1}\mu_{i-1}(\hat{\mathbf{X}}_i)), \end{aligned} \tag{5.7}$$

and

$$\mu_0(\mathbf{X}^*) = K_0(\mathbf{X}^*, \hat{\mathbf{X}}_0) K_0(\hat{\mathbf{X}}_0, \hat{\mathbf{X}}_0)^{-1} (\mathbf{y}_0), \tag{5.8}$$

for some fidelity level i, $0 \leq i \leq M_f$. The constants ρ_i are scaling coefficients relating the outputs $\mathbf{y}_i$ of the different fidelity levels. The variance at each fidelity level is calculated as

$$\Sigma_i(\mathbf{X}^*) = \bar{\Sigma}_i(\mathbf{X}^*) + \rho_{i-1}^2 \Sigma_{i-1}(\mathbf{X}^*), \tag{5.9}$$

and

$$\Sigma_i(\mathbf{X}^*) = \bar{\Sigma}_0(\mathbf{X}^*) \tag{5.10}$$

where $\bar{\Sigma}_i$, for $i = 0, ..., M_f$ is calculated using Eq. (5.2) and the appropriate dataset $(\mathbf{y}_i, \hat{\mathbf{X}}_i)$. The predictive mean and the variance of the multi-fidelity GP are taken as

$$\mu_{MF}(\mathbf{X}^*) = \mu_{M_f}(\mathbf{X}^*), \tag{5.11}$$

$$\Sigma_{MF}(\mathbf{X}^*) = \Sigma_{M_f}(\mathbf{X}^*) \tag{5.12}$$

with $\mu_M(\mathbf{X}^*)$ and $\Sigma_M(\mathbf{X}^*)$ calculated as in Eqs. (5.7) and (5.9). The proposed model is consistent with the recursive auto-regressive multi-fidelity model[1]

$$f_i(\mathbf{X}) = \rho_{i-1}(\mathbf{X}) f_{i-1}(\mathbf{X}) + \xi_i(\mathbf{X}), \tag{5.13}$$

presented in [83], where f_i are the means of a lower fidelity GP and ξ_i are bias terms that are independent of levels of lower fidelity. As presented in [82] and [83], the model requires that higher fidelity input data be a subset of lower fidelity input. This is required to infer the scaling parameters ρ_i. In this work, this requirement is not enforced and ρ_i is assumed to be one. This is rationalized by considering the Taylor expansion of the warped Gaussian process

$$\zeta(\mathbf{x} + \varepsilon_x) = \zeta(\mathbf{x}) + \varepsilon_x^T \frac{\partial \zeta(\mathbf{x})}{\partial \mathbf{x}} + \dots \tag{5.14}$$

as discussed in [61] and rearranging it as

$$\zeta(\mathbf{x}) = \zeta(\mathbf{x} + \varepsilon_x) - \varepsilon_x^T \frac{\partial \zeta(\mathbf{x})}{\partial \mathbf{x}} - \dots \tag{5.15}$$

[1]The original auto-regressive model presented in [82] takes $\rho_i(\mathbf{X})$ as constants that can be learned along with the covariance kernel hyperparameters. This is generalized to an input-dependent function in [83].

to approximate $\zeta(\mathbf{x})$. Then, for a two-fidelity-level model, ignoring the higher order terms, taking $\zeta(\hat{\mathbf{x}})$ as $f_{i-1}(\hat{\mathbf{x}})$, and $\xi_i(\hat{\mathbf{x}})$ as $-\varepsilon_x^T \frac{\partial \zeta(\mathbf{x})}{\partial \mathbf{x}}$ in Eq. (5.13) implies ρ_{i-1} should be one.

Intuitively, the lowest fidelity data will be used to build a warped GP model of the field due to the shift in the location of the measured data when localization error is present. Higher fidelity data, which has better localization, will correct the model at points where the higher fidelity is taken and approximate the correction at nearby input locations.

5.3 Vehicle Model

An extended version of the dynamic model presented in Section 3.1 is considered. The state vector consists of the position $b = [x, y, z]^T$ of the robot, the orientation $\Psi = [\phi, \theta, \psi]^T$ (roll, pitch, and yaw Euler angles) with respect to the inertial frame, and the body-fixed linear velocities $v_b = [v_1, v_2, v_3]^T$, the body-fixed angular velocities $\omega_b = [\omega_1, \omega_2, \omega_3]^T$ and the actuator states m_0 representing the buoyancy, r_{p1} representing the position of a movable mass from the geometric center along the body-fixed x-axis, and δ representing the tail angle of the robot with respect to the body-fixed x-axis. The state vector can be written as

$$X = [x, y, z, \phi, \theta, \psi, v_1, v_2, v_3, \omega_1, \omega_2, \omega_3, m_0, r_{p1}, \delta]^T. \tag{5.16}$$

The dynamic equations are

$$\begin{cases} \dot{b} = Rv_b \\ \dot{\Psi} = S_\omega \omega_b \\ \dot{v}_b = M^{-1}((Mv_b) \times \omega_b + m_0 g R^T k + F_{ext}) \\ \dot{\omega}_b = J^{-1}(-\dot{J}\omega_b + (J\omega_b) \times \omega_b + (Mv_b) \times v_b + T_{ext} \\ \quad + m_w g r_w \times (R^T k) + \bar{m} g r_p \times (R^T k)) \\ \dot{m}_0 = u_1 m_s \\ \dot{r}_{p1} = u_2 r_{ps} \\ \dot{\delta} = k_\delta (u_3 - \delta) \end{cases} \tag{5.17}$$

where u_1 and u_2 are normalized actuator rates and u_3 and k_δ are a reference angle and a time constant, repectively, for the tail position. R is a 3×3 rotation matrix parameterized by the Euler angles $\Psi = [\phi, \theta, \psi]^T$ following the ZYX convention. It is given as

$$R = \begin{pmatrix} c_\theta c_\psi & c_\psi s_\theta s_\phi - c_\phi s_\psi & s_\phi s_\psi + c_\phi c_\psi s_\theta \\ c_\theta s_\psi & c_\phi c_\psi + s_\theta s_\phi s_\psi & c_\phi s_\theta s_\psi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{pmatrix},$$

where c_q and s_q with $q = \phi, \theta, \psi$ represent sine and cosine of the variable in the subscript. S_ω, written as

$$S_\omega = \begin{pmatrix} 1 & \tan(\theta)\sin(\phi) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{pmatrix}$$

is a 3×3 matrix that relates the body-fixed angular velocities to Euler angle rates. $M = \text{diag}\{m_1, m_2, m_3\}$

is the total mass matrix incorporating the added-mass effect from the surrounding fluid and $J = \text{diag}\{J_1, J_2, J_3\}$ is the total inertia matrix. Note that M and J are assumed to be diagonal considering the simple and symmetric geometry of the gliding robotic fish. g is Earth's gravitational constant, $k = [0,0,1]^T$, $r_p = [r_{p1}, 0, 0]^T$, and $r_w = [0,0,r_{w3}]^T$ is the position of the center of gravity of the non-uniformly distributed mass m_w. Note that in this work the non-uniformly distributed mass m_w is assumed to be located along the body fixed z-axis of the robot.

The external hydrodynamic force F_{ext} and torque T_{ext} vectors depend on lift L, drag D, side force F_s, roll moment M_1, pitch moment M_2, and yaw moment M_3, which are taken from the model [5]. F_{ext} and T_{ext} also depend on the force F_{tail} due to moving the tail, the angle of attack $\alpha = \arctan\frac{v_3}{v_1}$ and the side-slip angle $\beta = \arcsin\frac{v_2}{\sqrt{v_1^2+v_2^2+v_3^2}}$. The lift force L is slightly different than the model used in [5]. Inspired by the data collected in [106], we add a multiplicative term in α to capture the fact that the lift force should be diminished as α approaches $\frac{\pi}{2}$. The forces and moments are given as

$$\begin{cases} D = \frac{1}{2}\rho V^2 S(C_{D0} + C_D^{\alpha}\alpha^2 + C_D^{\delta}\delta^2) \\ F_s = \frac{1}{2}\rho V^2 S(C_{F_S}^{\beta}\beta + C_{F_S}^{\delta}\delta) \\ L = \frac{1}{2}\rho V^2 S(C_{L0} + C_L^{\alpha}\alpha)\cos(\alpha) \\ F_{tail} = l_c \rho S_{\delta} l_{\delta 0} k_{\delta}^2 (\delta - u_3) \\ M_1 = \frac{1}{2}\rho V^2 S(C_{M_R}^{\beta}\beta + K_{q1}\omega_1) \\ M_2 = \frac{1}{2}\rho V^2 S(C_{M_0} + 0.5C_{M_P}^{\alpha}\sin(2\alpha) + K_{q2}\omega_2) \\ M_3 = \frac{1}{2}\rho V^2 S(C_{M_Y}^{\beta}\beta + K_{q3}\omega_3 + C_{M_Y}^{\delta}\delta) \end{cases} \tag{5.18}$$

where the parameters associated with K_q and C notations are hydrodynamic constants, ρ is the fluid density, S is the characteristic surface area of the robot, S_{δ} is the area of the tail which is

considered as a flat plate, l_c is a length corresponding to an effective volume of water with the same cross-sectional area as the tail, $l_{\delta 0}$ is the distance from the base to the center of mass of the tail, and V is the magnitude of v_b. The hydrodynamic force vector can the be given as

$$F_{ext} = R_{bv} \begin{bmatrix} -D \\ F_s \\ -L \end{bmatrix} + F_{tail} \begin{bmatrix} \sin(\delta) \\ 0 \\ 0 \end{bmatrix}$$

and

$$T_{ext} = R_{bv} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ l_{\delta 1} F_{tail} \cos(\delta) \end{bmatrix}$$

is the hydrodynamic moment vector where l_δ is the distance from the geometric center of the robot to the geometric center of the tail. R_{bv} is a 3×3 rotation matrix parameterized by the angle of attack α and the side-slip angle β that maps the hydrodynamic forces and moments from the velocity reference frame to the body-fixed frame. It is given by [5]

$$R_{bv} = \begin{pmatrix} \cos(\alpha)\cos(\beta) & -\cos(\alpha)\sin(\beta) & -\sin(\alpha) \\ \sin(\beta) & \cos(\beta) & 0 \\ \sin(\alpha)\cos(\beta) & -\sin(\alpha)\sin(\beta) & \cos(\alpha) \end{pmatrix}. \tag{5.19}$$

In addition to the motion model, a simplified energy usage model for the gliding robotic fish is proposed to use for planning under an energy budget. The model has a component d_c , which accounts for the constant energy drain due to the onboard electronics, and terms that correspond to

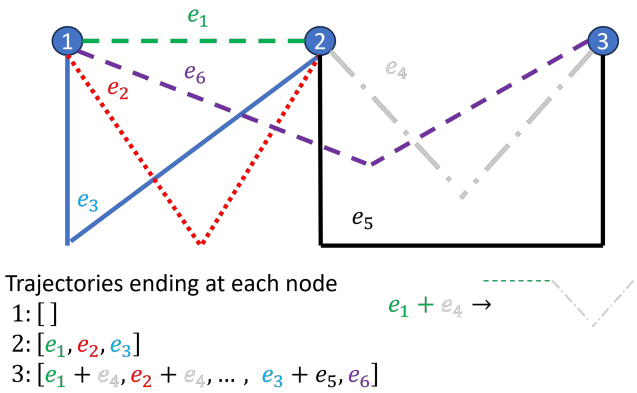


Figure 5.1: Conceptual illustration of multi-graph usage in planning algorithm depicting multi-graph with 3 nodes on a 1D space and edges that make paths in 2D space.

actuator movements. The energy usage model can be described as

$$\dot{E}_b = d_c + d_m \dot{m}_0^2 + d_r \dot{r}_{p1}^2 + d_\delta \dot{\delta}^2 \tag{5.20}$$

where d_m, d_r, and d_δ are positive constants[2] associated with the use of energy by the pump, mass position, and tail actuators, respectively.

5.4 Informative Trajectory Planner

The trajectory planner uses sampling-based motion planning inspired by the rapidly exploring information gathering (RIG) algorithms presented in [107] to explore the space of possible trajectories. The RIG algorithms use sampling-based motion planning techniques to incrementally build a graph/tree to solve the problem

$$\mathscr{X}^* = \text{argmax}_{\mathscr{X} \in \mathscr{T}} \mathscr{C}(\mathscr{X}) \text{ s.t. } S(\mathscr{X}) \leq B \tag{5.21}$$

[2] In general, d_m will depend on the pressure at specific depths and the movement direction of the actuator.

where B is a budget on a resource, $\mathscr{X}$ is a trajectory, $\mathscr{T}$ is the space of all possible trajectories, $S(\mathscr{X})$ is a monotonically increasing and additive function, and $\mathscr{C}(\mathscr{X})$ is a primary objective function to be optimized. In [107], the authors focus solely on information objectives such as mutual information, information gain, and variance reduction, but note that the algorithms may be applicable to other objectives as well. Our planner uses a modified version of the RIG algorithm that includes an additional constraint on the maximum uncertainty $\bar{\sigma}$ for any point on the trajectory and is able to incorporate the robot's ability to move from point to point using different modes of operation. The modified problem can be expressed as

$$\begin{aligned} \mathscr{X}^* = \text{argmax}_{\mathscr{X} \in \mathscr{T}}\ & \mathscr{C}(\mathscr{X}) \text{ s.t.} \\ & S(\mathscr{X}) \leq B \\ & \max(P(\mathscr{X})) \leq \bar{\sigma} \end{aligned} \tag{5.22}$$

where the function $P(\mathscr{X})$ maps the uncertainty associated with each point on the trajectory $\mathscr{X}$ to a scalar value.

It is noted that the standard RIG algorithm can be applied to plan a trajectory over the state space of the robot. However, the state space is large, and planning over it for navigation can quickly become infeasible, especially for low-cost computing platforms often employed on robots such as the gliding robotic fish. Rather than planning over the entire state space, the proposed planner builds 3D trajectories by planning 2D surfacing points for the robot and then a set of 3D trajectories between the points that satisfy the constraints. This can be represented as a multi-graph, which allows multiple edges between the same nodes in the graph. The trajectories on the edges can be generated using tools such as motion primitives, a set of controllers, or sample-based planning methods. An illustrative example of a multi-graph with 1D surfacing points and 2D

trajectories is shown in Fig. 5.1. In the following subsections, the trajectory planning algorithm is described in detail.

5.4.1 Path Planning Algorithm Description

Pseudo code for the planning algorithm is shown in Algorithm 1. It requires an initial robot pose p_0, the domain of exploration $\mathscr{D}$, a maximum uncertainty $\bar{\sigma}$, a step size Δ, a near radius R_{near}, a nearest radius R_d, a budget constraint B, and the resource function $S(\mathscr{X})$.

The planner uses a directed multi-graph $\mathscr{G}(\mathscr{V},\mathscr{E})$ with nodes $\mathscr{V}$ and edges $\mathscr{E}$ to explore the space of trajectories based on ideas from sampling-based planners. The multi-graph formulation allows for multiple directed edges between the same pair of nodes. It is assumed that a trajectory $\mathscr{X} \in \mathscr{T}$ is represented by a set of points parameterized by time and can be extracted from a set of connected nodes $\mathscr{V}$ and edges $\mathscr{E}$.

The algorithm creates a graph by starting with a node at the initial position of the robot (line 1 in Algorithm 1 and then iteratively expands the graph (lines 3-26). In each iteration, the Sample() function is used to sample a new location from a uniform distribution on the portion of $\mathscr{D}$ that represents the surface locations, termed $\bar{\mathscr{D}}$ (line 5 in Algorithm 1). For each new sampled location, the Nearest() function selects the nearest node on the graph based on a specified distance function and then extends the graph toward the sampled location (lines 6-11 in Algorithm 1). Typically this distance is taken to be the Euclidean distance, but for our specific application, a function that minimizes the distance from a ring of radius R_d centered on the sampled location, i.e.,

$$d(x_1,x_2) = |R_d - \|x_1 - x_2\||$$

is chosen to promote edges between nodes with distance R_d. This is useful for biasing the initial

Table 5.1: Motion primitives.

Primitive	Parameterization
Glide	$(\theta_g, \Delta z, \dot{z})$
Spiral	$(\Delta z, \dot{z}, \theta_s, r_s)$
Flat Dive	$(\Delta z, \dot{z})$
Swim	$(\Delta d, v_1)$

trajectories generated by the planning algorithm to have edges between nodes that are further apart which will typically lead to better energy usage of the robot. The extension toward the sampled location is carried out using the Steer(), PlanEdges(), and ExtendPaths() functions. The Steer() function returns the sampled point if it is less than some distance Δ from the nearest node on the graph. Otherwise, it returns a point at a distance Δ along the line connecting the sampled location to the nearest node on the graph. The PlanEdges() function then generates a set of edges (tuples of associated node indices, objective value, budget consumed, time taken) that connect the nearest node on the graph to the new point and satisfy the constraint $\bar{\sigma}$ on the largest uncertainty associated with the partial trajectory derived from each edge. Finally, the ExtendPaths() function extends paths from the node selected by the Nearest() function with the newly generated edges to the new point, pruning resultant trajectories that do not satisfy the budget constraint B. If all trajectories originating from a node exceed the budget constraint, it is added to a closed list and no longer considered for extension. Once a new node is added to the graph, the Near() function returns a set of nodes within a radius R_{near} of the new node based on Euclidean distance and the same procedure is used to extend each near node to the new node (lines 16-26 in Algorithm 1).

5.4.2 Edge Planner

While different methods can be used for the PlanEdges() function, the dynamics can be approximated by a simple model over relatively large distances if an appropriate controller is available.

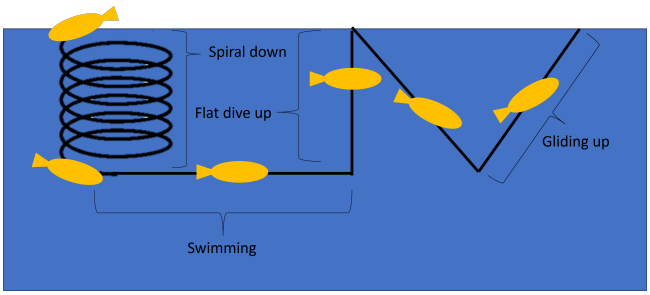


Figure 5.2: Illustration of motion primitives.

Given ongoing and previous work for modeling and control of the gliding robotic fish [5,108–111], a set of parameterized motion primitives are considered to approximate feasible partial trajectories between two nodes. The motion primitives are shown in Table 5.1 and illustrated in Fig. 5.2. They consist of steady-state glides parameterized by a glide path angle θ_g, a change in depth Δz, and the vertical speed $\dot{z}$, steady-state spirals parameterized by a change in depth Δz, vertical speed $\dot{z}$, spiral radius r_s, and spiral pitch θ_s, flat dives parameterized by a change in depth Δz and the vertical speed $\dot{z}$, and horizontal swimming parameterized by a distance Δd and velocity v_1. The spiral radius and spiral pitch can be used to define a helical spiral with curvature and torsion defined by $\frac{r_s}{\theta_s^2+r_s^2}$ and $\frac{\theta_s}{\theta_s^2+r_s^2}$, respectively, as reported in [5]. These particular motion primitives are considered because they represent steady-state motions and constant-speed swimming, respectively. Partial trajectories can be built by concatenating multiple primitives together to connect nodes in the graph.

The motion primitives also admit a fast approximation of energy used. It is assumed that when swimming, the input δ_d takes the form $\delta_d(t) = \delta_b + \delta_a \sin(2\pi f t)$ where f is the frequency of tail flapping, δ_b is the bias, and δ_a is the amplitude of the tail flapping. It is also assumed that $\delta = \delta_d$, which is reasonable as long as the frequency and amplitude respect the actuation limits of the actuator. Using these assumptions, one can calculate the energy used due to swimming by

integrating $\dot{\delta}^2 = (2\pi f \delta_a \cos(2\pi f t))^2$ over the amount of time the robot plans to swim giving

$$\frac{1}{2}\pi\delta_a^2 f(\sin(4\pi f t) + 4\pi f t)|_{t=t_0}^{t=t_1}.$$

For steady-state maneuvers (glides, spirals, and dives), the energy used by the actuators is not dependent on the time the trajectory takes. Instead, the actuators only draw energy when the robot is moving them to the desired positions. For that reason, it is assumed that each maneuver uses a separate but approximately constant amount of energy. A conservative estimate consists of using the energy required to move the actuators at full speed to the appropriate limit and back to the neutral position, which will take a constant time.

Along with the motion primitives, the position of the robot is assumed to be estimated with a Kalman filter using the model $\frac{d}{dt}[x,y,z,v_x,v_y,v_z]^T = [v_x,v_y,v_z,0,0,0]$ with a known process noise matrix and a measurement model of

$$h = \begin{cases} [x,y,z,v_x,v_y,v_z]^T & z \leq \varepsilon \\ [z,v_x,v_y,v_z]^T & z > \varepsilon \end{cases}$$

corrupted with zero mean, Gaussian noise where ε is a threshold for depth beyond which the x and y positions can no longer be used in the estimation. This allows the edge planner to calculate an expected variance for points along the partial trajectories. The depth and velocity measurements are assumed to be always known. Aquatic robots such as the gliding robotic fish typically have pressure sensors from which the depth can be derived. To satisfy this requirement for the velocities, a sensor such as a Doppler velocity log or a model-based observer such as the one presented in Chapter 3 is needed.

Algorithm 1 Planning Algorithm

Input:
Initial robot pose p_0
Domain of exploration $\mathscr{D}$
Sampling Domain $\bar{\mathscr{D}}$
Max uncertainty for position estimate $\bar{\sigma}$
Step size Δ
Near Radius R_{near}
Nearest Radius R_d
Budget constraint B
Budget Cost $S(\mathscr{X})$

$n \leftarrow \text{Node}(p_0)$, $\mathscr{V} \leftarrow \{n\}$
$\mathscr{V}_{closed} \leftarrow \{\}$, $\mathscr{E} \leftarrow \{\}$
while not terminated **do**
    %Move towards randomly sampled point
    $x_s \leftarrow \text{Sample}(\bar{\mathscr{D}})$
    $n_{nrst} \leftarrow \text{Nearest}(x_s, \mathscr{V} \setminus \mathscr{V}_{closed}, R_d)$
    $x_{feas} \leftarrow \text{Steer}(x_{nrst}, x_s, \Delta)$
    %Plan paths to new node
    $n_{feas} \leftarrow \text{CreateNode}(n_{nrst}, x_{feas})$
    $\mathscr{E}_l \leftarrow \text{PlanEdges}(n_{nrst}, n_{feas}, \bar{\sigma})$
    $\mathscr{G}_b, \mathscr{E}_l \leftarrow \text{ExtendPaths}(n_{feas}, \mathscr{E}_l, S(), B)$
    $\mathscr{V} \leftarrow \mathscr{V} \cup \{n_{feas}\}$
    $\mathscr{E} \leftarrow \mathscr{E} \cup \{\mathscr{E}_l\}$
    if $\text{Closed}(n_{feas})$ **then**
        $\mathscr{V}_{closed} \leftarrow \mathscr{V}_{closed} \cup \{n_{feas}\}$
    $N_{near} \leftarrow \text{Near}(x_{feas}, \mathscr{V} \setminus \mathscr{V}_{closed}, R_{near})$
    for $n_{near} \in N_{near}$ **do**
        %Extend near nodes toward n_{feas}
        $x_{new} \leftarrow \text{Steer}(x_{nrst}, x_{feas}, \Delta)$
        $n_{new} \leftarrow \text{CreateNode}(n_{nrst}, x_{new})$
        $\mathscr{E}_l \leftarrow \text{PlanEdges}(n_{nrst}, n_{new}, \bar{\sigma})$
        $\mathscr{G}_b, \mathscr{E}_l \leftarrow \text{ExtendPaths}(n_{new}, \mathscr{E}_l, S(), B)$
        $\mathscr{V} \leftarrow \mathscr{V} \cup \{n_{new}\}$
        $\mathscr{E} \leftarrow \mathscr{E} \cup \{\mathscr{E}_l\}$
        if $\text{Closed}(n_{new})$ **then**
            $\mathscr{V}_{closed} \leftarrow \mathscr{V}_{closed} \cup \{n_{new}\}$
return $((\mathscr{V}, \mathscr{E}), \mathscr{G}_b)$

5.4.3 Theoretical Analysis of Trajectory Planner

The RIG-Graph algorithm claims asymptotic optimality by theoretically generating all possible budget-constrained trajectories, given a stationary modular, time-varying modular, or submodular information objective. By virtue of all trajectories being generated, the optimal trajectory will be contained in the graph. The result is based on [112] which states that (1) an infinitely dense graph must be created around each point in $\mathscr{D}$ and (2) all trajectories must be of finite length as necessary properties for all feasible trajectories to be generated in a rapidly-exploring random belief tree. Similar arguments are used here and the following assumptions are declared.

Assumption 1. *Let x_a and x_b be two nodes on a multigraph and $\bar{\sigma}$ represent the largest allowable uncertainty for any point on a trajectory.* PlanEdges$(x_a, x_b, \bar{\sigma})$ *returns the set of all feasible edges between x_a and x_b.*

This assumption is required to ensure that all possible partial trajectories between two nodes are included in the graph. This may be difficult to achieve in a single call of the function in general. However, in practical application, the PlanEdges() function can be called for the same nodes on different iterations to incrementally increase the number of edges.

Assumption 2. *Let x_a, x_b, and x_c be three points within radius Δ of each other. Let E_1 be the set of partial trajectories generated by* PlanEdges(x_a, x_c)*, E_2 be generated by* PlanEdges(x_a, x_b)*, and E_3 be generated by* PlanEdges(x_b, x_c)*. For trajectory $e_1 \subset E_1$, if $x_b \in e_1 \subset E_1$, then there must be a concatenated trajectory $e_2 + e_3$ equal to e_1 that has equal cost, primary objective value, and trajectory representation with $e_2 \subset E_2$ and $e_3 \subset E_3$.*

This assumption is needed as nodes get infinitesimally close to one another as the graph is refined. It requires consistency for partial trajectories that are subsets of longer partial trajectories. It is similar to the assumption requiring consistency for the Steer() function in [107].

Assumption 3. *Let $\bar{\mathscr{D}} \subset \mathscr{D}$ be the set of points of $\mathscr{D}$ for which localization is measurable. There exists a constant $r \in \mathbb{R}_+$ such that for any point $x_a \in \bar{\mathscr{D}}$ there exists an $x_b \in \bar{\mathscr{D}}$, such that (i) the ball of radius r centered at x_a lies inside $\mathscr{D}$ and (ii) x_a lies inside the ball of radius r centered at x_b.*

This assumption is similar to the corresponding assumption in the RRG and RRT* algorithms in [113] that requires that some free space is available around the optimal trajectory to allow for convergence. Here it is only required for the surfacing locations.

Finally, the following assumption on the sampling function is required.

Assumption 4. *Points returned by the sampling function* Sample() *are i.i.d. and drawn from a uniform distribution.*

Lemma 1. *Let $\mathscr{T}^B$ denote the set of all finite length trajectories through $\mathscr{D}$ that satisfy a budget B and maximum value representing uncertainty $\bar{\sigma}$ such that for every $x \in \mathscr{X}$ and $\mathscr{X} \in \mathscr{T}^B$, $x \in \mathscr{D}$ and $\sigma(x) \leq \bar{\sigma}$. Let $\mathscr{T}_i^B$ denote the set of trajectories contained in the graph built by the trajectory planner at iteration i for budget B and uncertainty less than $\bar{\sigma}$. Then we have that the* $\lim_{i\to\infty} \mathscr{T}_i^B = \mathscr{T}^B$.

Proof. This follows from assumptions 1, 2, 3, 4, and the analysis given in the RIG, RRG, and RRBT algorithms [107, 112, 113]. According to [107], for all feasible trajectories to be produced, an infinitely dense connected graph must be created around each point in the obstacle-free exploration space and all trajectories must be of finite length. Since the budget cost function is assumed to be monotonically increasing and additive, all trajectories will be of finite length. Assumption 3 adapts the condition for an infinitely dense graph to be restricted to a subset of the exploration space that will be sampled. Similarly to the previously mentioned algorithms, 3 can be used to generate an infinitely dense graph on the sampled subset if the Near() function returns all nodes

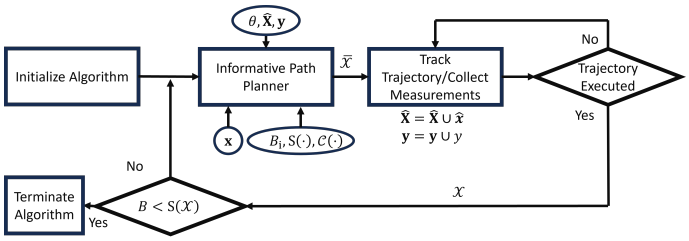


Figure 5.3: Schematic of the adaptive sampling algorithm.

within the near radius. Additionally, the algorithm requires assumption 1 to ensure all feasible edges between two nodes are generated. ■ □

In addition to the claim of asymptotic optimality, it was shown in [107] and [112] that the existence of partial ordering could be used to preserve optimality while pruning many non-useful trajectories. If such a partial ordering exists, pruning can be applied to this algorithm as well without sacrificing the asymptotic optimality.

5.4.4 Adaptive Sampling Algorithm

The informative trajectory planning algorithm is asymptotically optimal for an objective on a known, stationary field and a monotonically increasing budget cost function. However, the field is initially unknown and a model of it must be learned which means the objective is no longer stationary.

To deal with this, an adaptive sampling algorithm based on the sampling-based informative trajectory planner and Gaussian process regression models in this work is designed. A diagram depicting the algorithm is shown in Fig. 5.3. The algorithm works by iteratively planning trajectories $\bar{\mathscr{X}}$ using the proposed planning algorithm subject to partial budgets B_i. The robot executes a partial trajectory while collecting data at a constant rate and maintaining an estimate of its position $\hat{p}$. The collected data $(\tilde{v}, \hat{p})$ is assigned to an appropriate fidelity level (Algorithm 2) by compar-

ing the quantity $\eta_{\hat{p}}$ (representing the amount of uncertainty in the localization) to the user-defined thresholds $[\phi_{fM-1}, ..., \phi_{f0}]$ and placing the data in the highest fidelity if the quantity $\eta_{\hat{p}} < \phi_{fM-1}$, the lowest fidelity if $\eta_{\hat{p}} \geq \phi_{f0}$, and an appropriate middle fidelity k if $\eta_{\hat{p}}$ is between two fidelity thresholds ϕ_{fk} and ϕ_{fk-1}. The process is repeated until a total budget constraint $B = \sum B_i$ is exceeded.

Algorithm 2 Data Assignment

Input:
robot pose estimate $\hat{p}$
robot estimation covariance metric $\eta_{\hat{p}}$
Gaussian Process fidelity thresholds $[\phi_{fM-1}, ..., \phi_{f0}]$

if $\eta_{\hat{p}} \leq \phi_{fM-1}$ **then**
    $(\mathbf{y}_M, \hat{\mathbf{X}}_M) \leftarrow (\mathbf{y}_M, \hat{\mathbf{X}}_M) \cup \{(\tilde{v}, \hat{p})\}$
else if $\eta_{\hat{p}} \geq \phi_0$ **then**
    $(\mathbf{y}_0, \hat{\mathbf{X}}_0) \leftarrow (\mathbf{y}_0, \hat{\mathbf{X}}_0) \cup \{(\tilde{v}, \hat{p})\}$
else if $\phi_{fk} \leq \eta_{\hat{p}} \leq \phi_{fk-1}$ **then**
    $(\mathbf{y}_k, \hat{\mathbf{X}}_k) \leftarrow (\mathbf{y}_k, \hat{\mathbf{X}}_k) \cup \{(\tilde{v}, \hat{p})\}$

5.5 Experiments

5.5.1 Experimental Setup

Experiments are carried out with a miniaturized gliding robotic fish, Miniglider, in a 4.6 m long, 3.1 m wide, and 1.2 m deep indoor tank with 15 cm by 15 cm AprilTags [91, 92] placed on the floor and walls of the tank. The AprilTags can be used to localize the Miniglider robot with an onboard camera if the positions of the tags are known. Three overhead cameras covering an area of approximately 2.8 m by 1.7 m at the water surface and an RF Xbee radio module are connected to a "satellite" computer to serve as a positioning system for the tank. Two Apriltags were attached

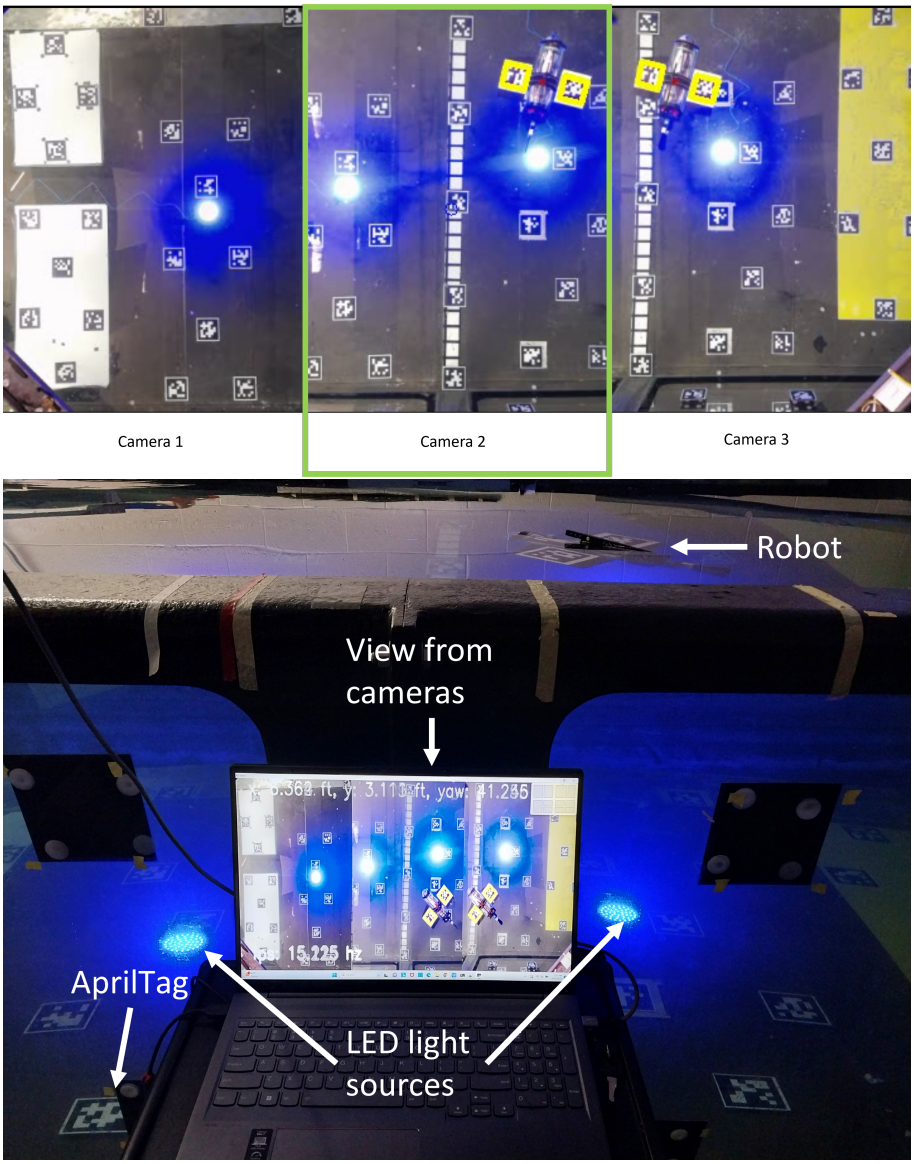


Figure 5.4: Pictures of the experimental setup from the overhead cameras (top) and side view of the tank (bottom).

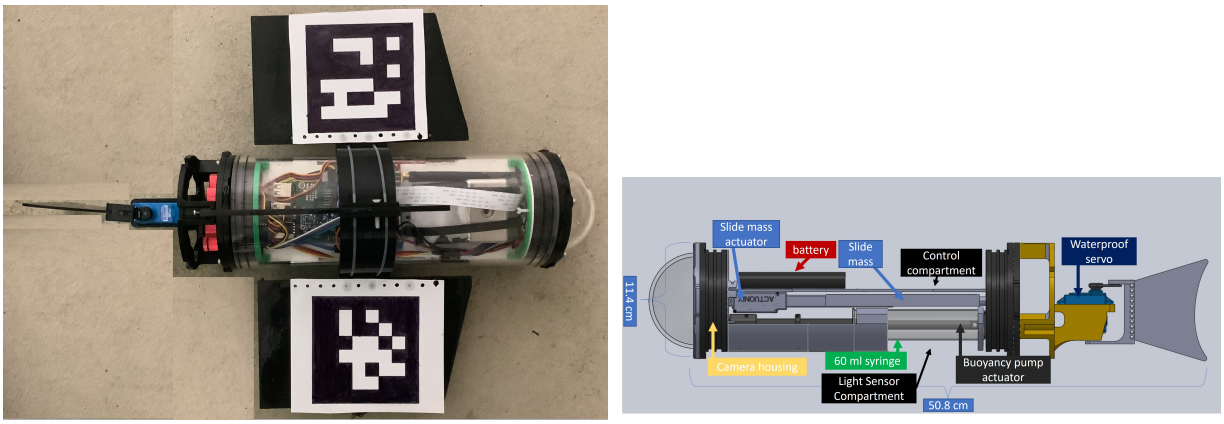


Figure 5.5: Pictures of actual (left) and SolidWorks design (right) revealing internal mechanical structure.

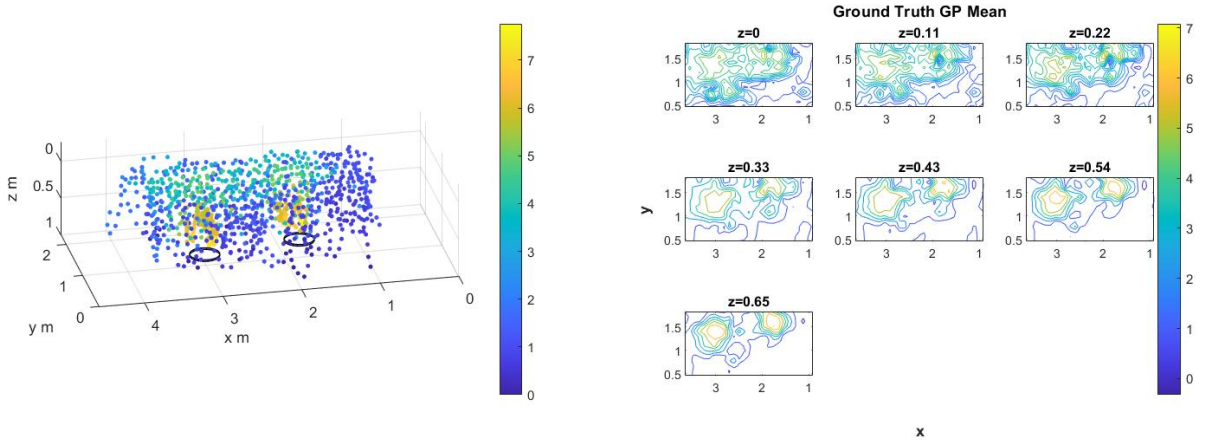


Figure 5.6: Raw data (left) collected for building the ground truth GP model and the mean (right) of the GP model at specific depths. Both color scales correspond to the log of the measured light intensity value.

to the wings of the robot for use with the positioning system. The measurement field is created using LED lights placed in the tank. The experimental setup is pictured in Fig. 5.4.

The robot, pictured in Fig. 5.5, is equipped with an Xbee RF module for communication and an Arduino BLE Sense which contains a light sensor (APDS-9960) capable of RGB color sensing. The log of the output of the 12 bit analog-to-digital converter for the blue channel is taken as the measurement for the field. Other relevant sensors include a pressure sensor, onboard camera, and an IMU. The robot is also equipped with a servo to control the tail angle δ, two linear actuators whose positions are related to the state variable r_{p1} and m_0 via the formulas $r_{p1} = (\mu - r_{p1_c})r_{p1_s}$ and $m_0 = (\mu - m_{0_c})m_{0_s}$. $\mu \in [0,1]$ is the normalized linear actuator position for the corresponding case and the subscripts c and s represent the actuator positions corresponding to the base setting $(r_{p1} = 0, m_0 = 0)$, and scaling factors, respectively. Further details about the robot can be found in [114].

A ground-truth model of the field is generated by running several experiments with the robot being teleoperated or manually moved through the large tank while collecting measurements with the light sensor and position measurements from the depth sensor and overhead cameras. In post-processing, the data from all of the experiments is aggregated and down-sampled to approximately 1100 points by limiting the density of measurements using a threshold on the Euclidean distance between points. Points with a measurement value above a threshold calculated using a weighted sum of the mean value and max value of the measurements are allowed to be more densely placed. The Matern kernel with $\nu = \frac{3}{2}$ and automatic relevance determination was selected to build the GP regression model. The kernel is given as $K(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp(1 + \sqrt{3}d) \exp(-\sqrt{3}d$ where $d = (x - x')^T L(x - x')$ and L is the inverse of the diagonal matrix of squared length scales. Figure 5.6 shows the raw, down-sampled data points and resulting Gaussian process mean function.

During experiments, the robot runs a Kalman filter with a constant velocity model that incorpo-

rates angular velocities (IMU), (x,y)-plane positions (overhead cameras, April tags), the z position (pressure sensor, April tags), and the yaw angle (overhead cameras, April tags) to estimate the positions, their velocities, and the yaw angle. Estimates from this first Kalman filter are used as input to a second Kalman filter with the assumptions discussed in Section 5.4.2. The position estimates from this second Kalman filter are used when building a Gaussian process model of the field for the planning algorithm.

5.5.2 Adaptive Sampling Algorithm Implementation

The adaptive sampling algorithm is implemented using the Python programming language. The algorithm is given a total energy budget constraint of $B = 80$ with the planner allocating a max energy budget of 20 for each planning round. The planner is restricted to a 2.75 m by 1.37 m by 0.65 m rectangular domain. Edges are created by randomly generating a set of motion primitives with the velocity parameters $\dot{z}$ and v_1 being held as predetermined constants for their respective primitives. The primitives are concatenated together and the parameters of the final primitive are changed to ensure that the edge ends at the location of the intended node. An appropriate extra primitive is added if the final primitive cannot connect to the desired node. From the edges, a set of points that are spaced uniformly in time are extracted to approximate the trajectories.

Based on the proposed energy model in Eq. (5.20) and the operating modes of the robot, energy usage is assigned to the motion primitives discussed in Section 5.4.2 for planning purposes. The energy usage due to the term d_c is taken as a constant multiple of the time duration and the energy due to swimming is calculated using the formula discussed in Section 5.4.2 multiplied by a constant. The constants are chosen as 0.2 and 0.005 for the swimming multiplier and time multiplier, respectively. The energy usage for the tail is artificially inflated to nudge the algorithm to mimic trajectories in large-scale environments where swimming across the entire field will lead to

a quicker depletion of the energy budget compared to gliding. The energy consumed for flat dives, steady-state glides, and steady-states spirals are constant and taken as 1, 1.5, and 1.5 respectively. In future work, the robot can be equipped with the ability to measure power usage and these values can be determined from experimental data.

Two options for the objective function $\mathscr{C}(\mathscr{X})$ are considered. The first is taken as the information gain for a set of test points $\mathbf{X}^*$ that sufficiently cover the measurement field $\mathscr{D}$. This can be calculated as

$$I(\mathbf{X}^*;\mathbf{y}) = H(\mathbf{X}^*) - H(\mathbf{X}^*|\mathbf{y}) = \frac{1}{2}(\log(\det(K_{prior}) - \log(\det(K_{plan})))$$

where

$$K_{prior} = K(\mathbf{X}^*,\mathbf{X}^*) - K(\mathbf{X}^*,\mathscr{X})[K(\mathscr{X},\mathscr{X}) + \sigma_n^2 I]^{-1} K(\mathscr{X},\mathbf{X}^*)$$

and

$$K_{plan} = K(\mathbf{X}^*,\mathbf{X}^*) - K(\mathbf{X}^*,\mathscr{X}\cup\mathscr{X}_{plan})[K(\mathscr{X}\cup\mathscr{X}_{plan},\mathscr{X}\cup\mathscr{X}_{plan}) + \sigma_n^2 I]^{-1} K(\mathscr{X}\cup\mathscr{X}_{plan},\mathbf{X}^*).$$

Here, K_{prior} and K_{plan} are the posterior covariance for the GP model before and after the planned trajectory $\mathscr{X}_{plan}$ is concatenated with the trajectory $\mathscr{X}$ that the robot has already executed and σ_n is the measurement noise.

The second objective function is based on the ergodic metric presented in [115]. The ergodic metric can be thought of as a tool for matching time-averaged statistics of a trajectory to a given probability distribution. It does so by converting a trajectory into a probability distribution and minimizing the difference between the trajectory-based distribution and the target distribution. Subsequent works have proposed different methods to quantify the ergodic metric [116–118].

In this work, the negative of a measure employing the KL divergence to compare the distributions is used as proposed in [116] and [118]. The measure can be calculated as $D_{KL}(p,q) = \sum_{x\in\mathbf{X}^*} p(x)\log(\frac{p(x)}{q(x)})$ where $\mathbf{X}^*$ is again taken as a set of test points used to approximate $\mathscr{D}$, $p(x)$ is a target spatial distribution, and $q(x) = \frac{1}{t_1-t_0}\int_{t_0}^{t_1}\frac{1}{\eta}\exp(-\frac{1}{2}[x-\mathscr{X}_{plan}(t)]^T\Sigma^{-1}[x-\mathscr{X}_{plan}(t)])$ is a Gaussian function with a covariance Σ and a normalizing factor η. Only the planned trajectory is considered for the ergodic objective in this work, but in general, the framework also allows for the historical trajectory $\mathscr{X}$ to be incorporated for the calculation. A brief overview of the ergodic metric and this measure is given in the appendixD and the reader is referred to references [115, 116, 118] for more details. The target distribution for the ergodic metric is taken as $p(x) = \frac{\exp(\Phi(x))}{\sum_{z\in\mathscr{D}}\exp(\Phi(z))}$ where $\Phi(x) = \alpha\mu(x) + (1-\alpha)\sigma(x)$ is a weighted sum of the posterior mean and standard deviation of the GP model for a point $x \in \mathscr{D}$ and $\alpha \in [0,1]$ is user-defined. The target distribution is a normalization of an upper confidence bound-like function using the softmax function. The tuning parameter α, which can be tuned to be more exploitative of the GP mean or more exploratory, is added because an optimally ergodic trajectory does not seek to find the maximum of $p(\cdot)$ but instead seeks to be statistically similar to $p(\cdot)$. Choosing $\alpha = 1$ would make $p(\cdot)$ solely dependent on the GP mean function while choosing $\alpha = 0$ would make $p(\cdot)$ solely dependent on the uncertainty of the GP model.

The planner is allotted approximately 45 seconds to generate the graph and select a trajectory. To reduce the computational burden, $\mathscr{C}(\mathscr{X})$ is only evaluated if the budget exceeds 90% of the allocated planning budget. This heuristic will not degrade the performance when using the information gain as $\mathscr{C}(\mathscr{X})$ since a trajectory will always have at least as much information as a truncation of that same trajectory. When using the ergodic measure as $\mathscr{C}(\mathscr{X})$, this heuristic also provided promising results numerically.

For both choices of objective $\mathscr{C}(\mathscr{X})$, the sampling algorithm is used with the proposed multi-

fidelity GP regression model. The proposed approach is compared to using a single-fidelity GP regression model that incorporates all of the data without considering the localization error. This makes a total of four scenarios: multi-fidelity GP model with the ergodic measure as $\mathscr{C}(\mathscr{X})$ (MFE), multi-fidelity GP model with information gain as $\mathscr{C}(\mathscr{X})$ (MFIG), single-fidelity GP model with the ergodic measure (SFE), and single-fidelity GP model with information gain (SFIG). Six trials of each scenario were run where the robot collected the estimated position and light measurement to use for the GP model at 0.5 Hz. When calculating the information gain for the multi-fidelity GP, the information gain for a single-fidelity model is used as a heuristic. The single-fidelity and multi-fidelity GP regression models were implemented using the GPy [119] and Emukit [120], respectively. The multi-fidelity GP model was taken to have 3 fidelities (i.e., $M = 2$). Based on Eqs. (5.7) and (5.9), the predictive mean and variance at the highest fidelity level become

$$\begin{aligned}\mu_2(\mathbf{X}^*) = \rho_1\rho_0[K_0(\mathbf{X}^*,\hat{\mathbf{X}}_0)[K_0(\hat{\mathbf{X}}_0,\hat{\mathbf{X}}_0)+\sigma_n^2 I]^{-1}(\mathbf{y}_0)] \\ +\rho_1[K_1(\mathbf{X}^*,\hat{\mathbf{X}}_1)[K_1(\hat{\mathbf{X}}_1,\hat{\mathbf{X}}_1)+\sigma_n^2 I]^{-1}(\mathbf{y}_1-\rho_0\mu_0(\hat{\mathbf{X}}_1))] \\ +K_2(\mathbf{X}^*,\hat{\mathbf{X}}_2)[K_2(\hat{\mathbf{X}}_2,\hat{\mathbf{X}}_2)+\sigma_n^2 I]^{-1}(\mathbf{y}_2-\rho_1\mu_1(\hat{\mathbf{X}}_2))\end{aligned}$$

and

$$\begin{aligned}\Sigma_2(\mathbf{X}^*) = K_2(\mathbf{X}^*,\mathbf{X}^*)-K_2(\mathbf{X}^*,\hat{\mathbf{X}}_2)[K_2(\hat{\mathbf{X}}_2,\hat{\mathbf{X}}_2)+\sigma_n^2 I]^{-1}K_2(\hat{\mathbf{X}}_2,\mathbf{X}^*) \\ +\rho_1^2[K_1(\mathbf{X}^*,\mathbf{X}^*)-K_1(\mathbf{X}^*,\hat{\mathbf{X}}_1)[K_1(\hat{\mathbf{X}}_1,\hat{\mathbf{X}}_1)+\sigma_n^2 I]^{-1}K_1(\hat{\mathbf{X}}_1,\mathbf{X}^*)] \\ +\rho_1^2\rho_0^2[K_0(\mathbf{X}^*,\mathbf{X}^*)-K_0(\mathbf{X}^*,\hat{\mathbf{X}}_0)[K_0(\hat{\mathbf{X}}_0,\hat{\mathbf{X}}_0)+\sigma_n^2 I]^{-1}K_0(\hat{\mathbf{X}}_0,\mathbf{X}^*)]\end{aligned}$$

respectively, where K_i, $\hat{\mathbf{X}}_i$ and $\mathbf{y}_i$ are the kernel function, estimated position, and measurements associated with fidelity i and $\mathbf{X}^*$ is the set of prediction points.The Matern32 covariance kernel is chosen for both GP models and the hyperparameters are assumed to be known. The single-fidelity models are given the hyperparameters of the ground-truth model while the multi-fidelity model

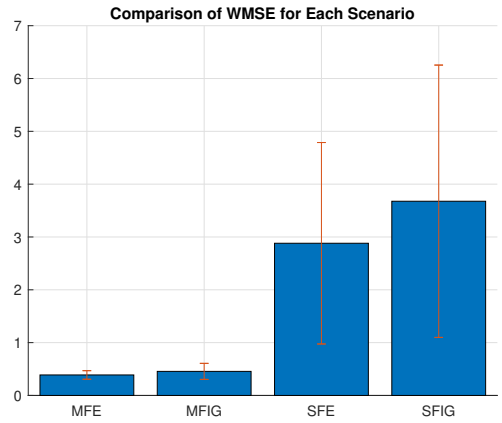


Figure 5.7: Average and standard deviation of WMSE for each of the four scenarios: multi-fidelity GP model with ergodic measure as planning objective (MFE), multi-fidelity GP model with information gain as planning objective (MFIG), single fidelity GP model with ergodic measure (SFE), and single fidelity GPmodel with information gain (SFIG) .

hyperparameters are obtained by training using an estimated trajectory calculated during one of the manual data collection experiments and the trajectory's associated field measurements.

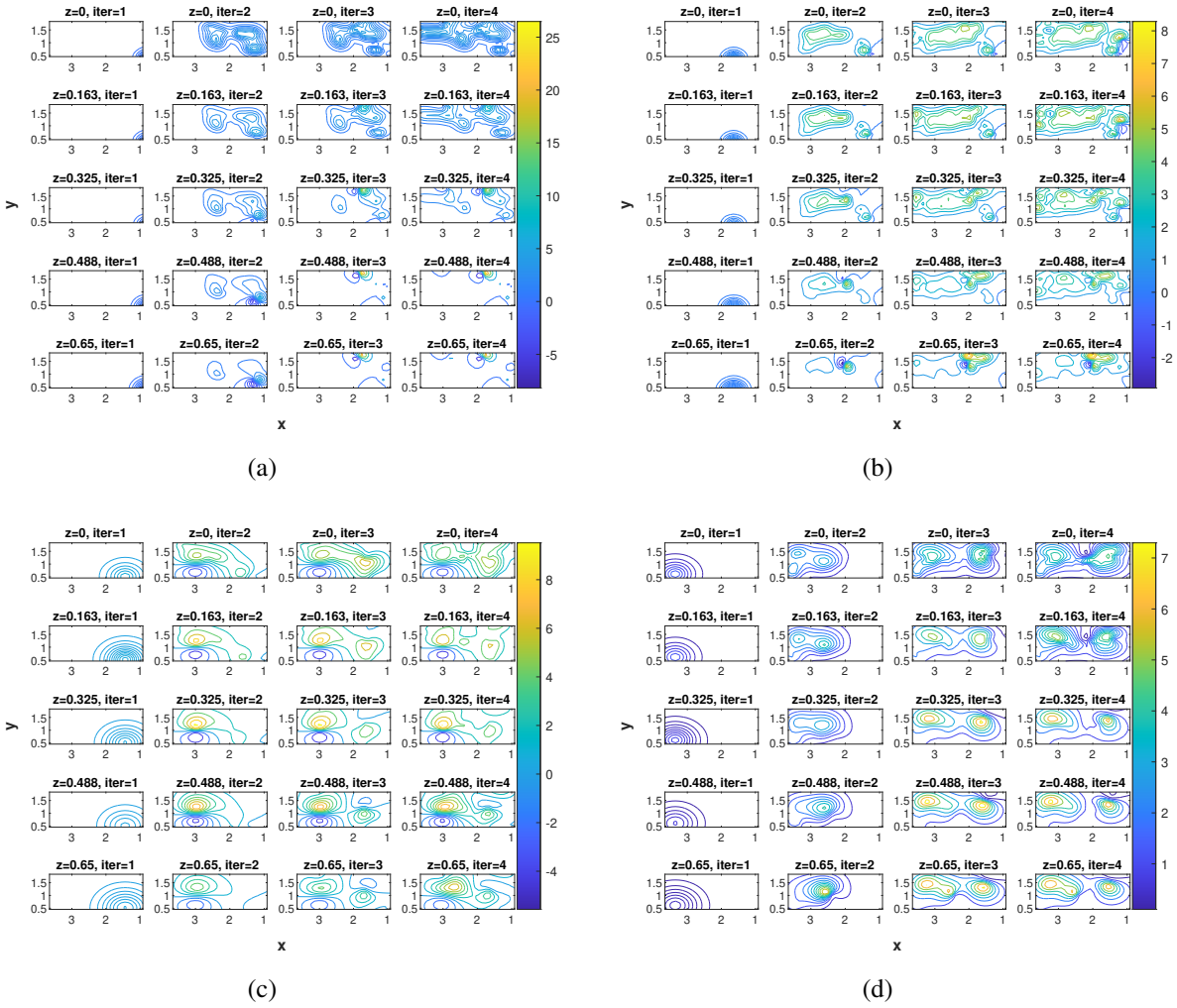
During each planning round, the GP model is updated and the target distribution is recalculated before the planning starts when basing $\mathscr{C}(\mathscr{X})$on the ergodic measure. It is noted that, with information gain as $\mathscr{C}(\mathscr{X})$, the planning can be done once in advance if the hyperparameters are known. However, the planning algorithm is still implemented in rounds to account for the mismatch between the planned trajectory and the estimated trajectory. It also mimics the case in which hyperparameters are learned during the experiment which is a subject of future work.

5.5.3 Results

To compare performance, the weighted mean squared error (WMSE) $\frac{1}{n}e^T\Sigma^{-1}e$ is used, where e is the difference between the value predicted by the GP model and the ground-truth value (predicted using the model from Section 5.5.1) at a set of test points X^* approximating $\mathscr{D}$, n is the number of test points, and Σ is the covariance matrix associated with the prediction of the GP at the test points.

Fig. 5.7 shows the average WMSE for each scenario and error bars representing the standard deviation. The average WMSEs are 0.3879 for MFE, 0.4556 for MFIG, 2.8812 for SFE, and 3.7766 for SFIG. For both primary objectives, the WMSE is smaller for the multi-fidelity model-based methods, which also results in smaller standard deviations. This implies that the multi-fidelity model better predicts the field while appropriately placing less confidence where localization error is high. Because the single-fidelity model does not take localization into account and treats all of the measurements as equally informative, it inaccurately places high confidence in predictions for all locations where measurements are reported. For the single-fidelity model, the ergodic measure produces a WMSE that is approximately 75% of the WMSE for using information gain, while the ergodic measure has a WMSE that is approximately 85% of the WMSE for information gain when using the multi-fidelity model.

Fig. 5.8 shows the predictive mean of the GP model for a single trial of each of the four scenarios. Each column corresponds to the mean immediately before the robot starts planning. Initially, each GP model predicts a near-uniform field, having only a measurement taken at the starting location. After completing the partial trajectory generated by the first planning iteration, the single-fidelity model with information gain as the planning objective (Fig. 5.8 (a)) starts to show distinctive blobs near the two sources. However, it experiences the nugget effect after the second planning iteration. The nugget effect causes the model to predict large values or negative values (despite having only positive measurements) and is due to inconsistent field values at nearby locations caused by the localization error. This is typical, but to a lesser extent, among all runs where the single-fidelity GP model is used. The effect is reduced in the case of single-fidelity GP with ergodic measure (Fig. 5.8 (b)) and the model predicts appropriate values near one of the light sources. While the multi-fidelity GP model also experiences the nugget effect, if higher fidelity measurements become available nearby, the prediction can be corrected. In the absence of the



(a) (b)

(c) (d)

Figure 5.8: Plots showing the predictive mean of the GP model at specific depths prior to the planning iterations: (a) single fidelity GP model with information gain as planning objective, (b) single fidelity GP model with the ergodic measure as planning objective, and (c) multi-fidelity GP model with information gain as planning objective, and (d) multi-fidelity GP model with the ergodic measure as planning objective.

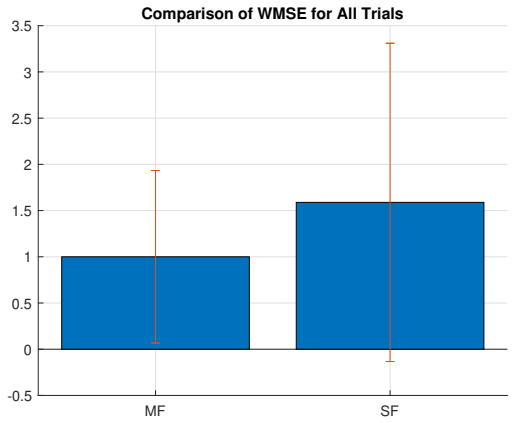


Figure 5.9: The average and standard deviation of WMSE for training single and multi-fidelity GP models on each of the individual datasets.

higher fidelity measurements, the variance reduction is smaller. For both the multi-fidelity and single-fidelity models using the ergodic measure with $\alpha = 0.2$, the robot focuses more time near the light sources.

After all of the experiments were completed, one multi-fidelity model and one single-fideity model were trained per experiment using the data collected in the respective experiments. This is done to remove the effect of the trajectory planning algorithm and more directly compare the multi-fidelity model and single-fidelity model by using the same sets of data when computing the WMSE. The mean and standard deviation for the WMSEs of all these datasets are shown in Fig 5.9 for the multi-fidelity model and the single-fidelity model. The average WMSE of the multi-fidelity model increases but still has better prediction performance than the single-fidelity model. The decrease in the single-fidelity model WMSE and increase in the multi-fidelity model WMSE also imply that the data generated by using the multi-fidelity model with the planner is of better quality.

5.6 Conclusion and Future Work

In this work, using multi-fidelity Gaussian process regression was proposed to account for localization uncertainty while modeling spatial fields. An informative trajectory planning algorithm was then developed and paired with the multi-fidelity GP model to form the basis of an adaptive sampling algorithm. Through experimental validation on a miniature gliding robotic fish, the approach was shown to improve field reconstruction in terms of the weighted mean squared error compared to an approach that ignores localization error and puts all measurements into a standard single-fidelity GP model.

In future work, online hyperparameter optimization can be addressed and techniques such as motion tomography [121] or smoothing may be applied to enhance the trajectory estimate based on the surfacing locations before adding new data to the Gaussian process. It is also of interest to study if there is an optimal way to select the number of fidelity levels and what data should be associated with each.

Another interesting direction would be to study how to split the budget. For the ergodic-based method, switching between exploration ($\alpha = 0$) and exploration $(\alpha = 1)$ similar to the ideas in [122] provides an interesting direction for this line of work. In addition, analysis of the heuristic used for the trajectory selection may be of interest.

Chapter 6

Summary and Future Work

6.1 Summary

This dissertation discussed the gliding robotic fish, GRACE, developed nonlinear control approaches for the robot, and developed an algorithm for autonomous exploration under localization uncertainty. A history of the development of the robot was provided along with possible future improvements. The most recent upgrades allow for quicker manufacturing and repair. They also enabled the robot to implement more complex algorithms, facilitated faster development of code for the robot, and enhanced the mechanical design of the robot. In addition, a miniature gliding robotic fish called Miniglider was developed along with an experimental test bed that allows for rapid testing of algorithms and functionality in a controlled environment.

Following the discussion of GRACE, a model-based nonlinear controller was developed that enables the robot to follow gliding-like trajectories. The controller takes advantage of the robot's natural dynamics to converge onto the desired time-varying position and pitch angle. Time-scale analysis was used to prove theoretically that the controller will drive the error between the actual and desired position and pitch angle of the robot to a region around the origin and this was demonstrated via simulation and experiments with the Miniglider robot.

Motivated by the application of tracking fish using range measurements, control barrier functions were utilized to enforce the nonlinear observability rank condition for the problem of target

tracking using range measurements. In both simulation and experiments, the approach was shown to be able to track a moving target while maintaining an acceptable estimate of relative position by modifying the behavior of a baseline tracking controller. The approach provides a promising general approach to addressing output feedback control for nonlinear systems.

Finally, a strategy for autonomous exploration of the robot in the presence of localization uncertainty was presented. Leveraging multi-fidelity Gaussian process regression and sampling-based path planning, the algorithm is able to generate informative trajectories that satisfy budget constraints such as time or energy. Localization uncertainty is addressed by using the multi-fidelity Gaussian process model and assigning different levels of fidelity to field measurements based on localization uncertainty. Experiments showed that the approach offers improvements over using standard Gaussian process regression where the localization uncertainty is ignored.

6.2 Future Work

In future work, GRACE can be further improved. The key areas of improvement are manufacturing, operational interfacing, sensing capabilities, and computational processing power. While great strides were made in the mechanical design to reduce the manufacturing time and reduce the need to build custom parts, further improvements can be made. While custom parts will probably not be totally eliminated, there are still a few major components of the robot that are currently custom-built but can be streamlined by using more commercially available parts. The current operational interface of the robot is rudimentary and only exposes the basic capabilities and teleoperation of the robot. An operational interface that allows non-technical users to employ the advanced capabilities developed in this dissertation and in the future is also needed to make the robot more widely useful in aquatic environments. Sensing and processing capabilities of the robot should

continuously be improved as new technology becomes available, but care should be taken to manage energy consumption so that the robot maintains the ability to execute long-term autonomous operations.

In addition, further modeling is needed to improve the non-steady state control of the robot and state estimation. The model used in Chapter 3 primarily considers steady-state operation of the robot. While the model in Chapter 5 is more representative, it still does not sufficiently capture the qualities of the hydrodynamics for non-steady state operation of the robot. Given the continual convergence of control and machine learning [123–129], both data-driven modeling and control would be an interesting direction to explore for this purpose. Along with this, the theory in Chapter 4 is a promising path to improve state estimation for nonlinear systems with partial state feedback and the underlying theory should be further developed. Lastly, networked control for a gliding robotic fish fleet is also a natural next step.

The exploration algorithm in Chapter 5 can also be expanded upon. Specifically, studying alternative edge-generation strategies for the trajectory planner and using other representations of the explored field are of interest. Better edge generation is helpful so that the robot can more closely execute the planned trajectory, especially in smaller environments where the robot dynamics are not negligible. Using other representations to model the field such as occupancy maps may have some advantages or worthwhile trade-offs compared to Gaussian process regression. Developing and studying other optimization objectives for the planner and characterizing the actual energy usage of the robot for planning purposes are also of interest. Finally, this algorithm should be implemented on the larger robot and its performance should be studied in larger-scale environments.

BIBLIOGRAPHY

[1] Henry Stommel. The Slocum Mission. *Oceanography*, 2(1):22–25, 1989.

[2] Jeff Sherman, Russ E Davis, WB Owens, and J Valdes. The autonomous underwater glider "Spray". *IEEE Journal of Oceanic Engineering*, 26(4):437–446, 2001.

[3] Jan Sliwka, Benoît Clement, and Irvin Probst. Sea glider guidance around a circle using distance measurements to a drifting acoustic source. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 94–99. IEEE, 2012.

[4] Jun Yuan, Zhengxing Wu, Junzhi Yu, and Min Tan. Sliding mode observer-based heading control for a gliding robotic dolphin. *IEEE Transactions on Industrial Electronics*, 64(8):6815–6824, 2017.

[5] Feitian Zhang. *Modeling, Design and Control of Gliding Robotic Fish*. Dissertation, Michigan State University. Electrical Engineering, 2014.

[6] Osama N. Ennasr. *Gliding Robotic Fish: Design, Collaborative Estimation, and Application to Underwater Sensing*. PhD thesis, Michigan State University, 2020.

[7] Jianxun Wang and Xiaobo Tan. A dynamic model for tail-actuated robotic fish with drag coefficient adaptation. *Mechatronics*, 23(6):659–668, 2013.

[8] Osama Ennasr, Christopher Holbrook, Darryl W Hondorp, Charles C Krueger, Demetris Coleman, Pratap Solanki, John Thon, and Xiaobo Tan. Characterization of acoustic detection efficiency using a gliding robotic fish as a mobile receiver platform. *Animal Biotelemetry*, 8(1):1–13, 2020.

[9] Feitian Zhang, Osama Ennasr, Elena Litchman, and Xiaobo Tan. Autonomous sampling of water columns using gliding robotic fish: Algorithms and harmful-algae-sampling experiments. *IEEE Systems Journal*, 10(3):1271–1281, 2016.

[10] Daniel L Rudnick, Russ E Davis, Charles C Eriksen, David M Fratantoni, and Mary Jane Perry. Underwater gliders for ocean research. *Marine Technology Society Journal*, 38(2):73–84, 2004.

[11] Pierre Testor, Brad De Young, Daniel L Rudnick, Scott Glenn, Daniel Hayes, Craig M Lee, Charitha Pattiaratchi, Katherine Hill, Emma Heslop, Victor Turpin, et al. Oceangliders: a component of the integrated goos. *Frontiers in Marine Science*, 6:422, 2019.

[12] Joshua G Graver and Naomi Ehrich Leonard. Underwater glider dynamics and control. In *12th International Symposium on Unmanned Untethered Submersible Technology*, pages 1742–1710, 2001.

[13] Naomi Ehrich Leonard and Joshua G Graver. Model-based feedback control of autonomous underwater gliders. *IEEE Journal of Oceanic Engineering*, 26(4):633–645, 2001.

[14] Khalid Isa and Mohd Rizal Arshad. Neural network control of buoyancy-driven autonomous underwater glider. In *Recent Advances in Robotics and Automation*, pages 15–35. Springer, 2013.

[15] Jian Wang, Zhengxing Wu, Min Tan, and Junzhi Yu. Model predictive control-based depth control in gliding motion of a gliding robotic dolphin. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 51(9):5466–5477, 2019.

[16] Enzeng Dong, Shuxiang Guo, Xichuan Lin, Xiaoqiong Li, and Yunliang Wang. A neural network-based self-tuning PID controller of an autonomous underwater vehicle. In *the Proceedings of the International Conference on Mechatronics and Automation, Chengdu, China*, pages 5–8, 2012.

[17] Anirban Nag, Surendra Singh Patel, and SA Akbar. Fuzzy logic based depth control of an autonomous underwater vehicle. In *Automation, Computing, Communication, Control and Compressed Sensing (iMac4s), 2013 International Multi-Conference on*, pages 117–123. IEEE, 2013.

[18] Nina Mahmoudian and Craig Woolsey. Underwater glider motion control. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 552–557. IEEE, 2008.

[19] Feitian Zhang, Xiaobo Tan, and Hassan K Khalil. Passivity-based controller design for stablization of underwater gliders. In *American Control Conference (ACC), 2012*, pages 5408–5413. IEEE, 2012.

[20] Maria L Castaño and Xiaobo Tan. Simultaneous stabilization of pitch and yaw of a gliding robotic fish using sliding mode control. In *Dynamic Systems and Control Conference*, volume 57243, page V001T04A004. American Society of Mechanical Engineers, 2015.

[21] Hai Yang and Jie Ma. Sliding mode tracking control of an autonomous underwater glider. In *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, volume 4, pages V4–555. IEEE, 2010.

[22] Maziyah Mat-Noh, MR Arshad, Rosmiwati Mohd-Mokhtar, and Qudrat Khan. Control of an autonomous anderwater glider using integral super-twisting sliding mode control (ISTSMC). In *Underwater System Technology: Theory and Applications (USYS), 2017 IEEE 7th International Conference on*, pages 1–6. IEEE, 2017.

[23] Barkat Ullah, Mark Ovinis, Masri B Baharom, MY Javaid, and SS Izhar. Underwater gliders control strategies: A review. In *Control Conference (ASCC), 2015 10th Asian*, pages 1–6. IEEE, 2015.

[24] G Ciccarella, M Dalla Mora, and Alfredo Germani. A Luenberger-like observer for nonlinear systems. *International Journal of Control*, 57(3):537–556, 1993.

[25] Hassan K Khalil. *Nonlinear Systems*. Prentice Hall, Upper Saddle River, NJ, 3 edition, 2002.

[26] Sarah K Spurgeon. Sliding mode observers: A survey. *International Journal of Systems Science*, 39(8):751–764, 2008.

[27] Frank Allgöwer, Thomas A Badgwell, Joe S Qin, James B Rawlings, and Steven J Wright. Nonlinear predictive control and moving horizon estimation—An introductory overview. *Advances in Control*, pages 391–449, 1999.

[28] Greg Welch and Gary Bishop. An introduction to the Kalman filter. 1995.

[29] Fredrik Gustafsson. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010.

[30] He Bai and Clark N Taylor. Future uncertainty-based control for relative navigation in GPS-denied environments. *IEEE Transactions on Aerospace and Electronic Systems*, 56(5):3491–3501, 2020.

[31] Kristoffer M Frey, Ted J Steiner, and Jonathan P How. Towards online observability-aware trajectory optimization for landmark-based estimators. *arXiv preprint arXiv:1908.03790*, 2019.

[32] Mohammadhussein Rafieisakhaei, Suman Chakravorty, and PR Kumar. On the use of the observability gramian for partially observed robotic path planning problems. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 1523–1528. IEEE, 2017.

[33] Gianluca Antonelli, Filippo Arrichiello, Stefano Chiaverini, and Gaurav S Sukhatme. Observability analysis of relative localization for AUVs based on ranging and depth measurements. In *2010 IEEE International Conference on Robotics and Automation*, pages 4276–4281. IEEE, 2010.

[34] Filippo Arrichiello, Gianluca Antonelli, Antonio Pedro Aguiar, and Antonio Pascoal. An observability metric for underwater vehicle localization using range measurements. *Sensors*, 13(12):16191–16215, 2013.

[35] Christopher Grebe, Emmett Wise, and Jonathan Kelly. Observability-aware trajectory optimization: Theory, viability, and state of the art. In *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 1–8. IEEE, 2021.

[36] Arthur J Krener and Kayo Ide. Measures of unobservability. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6401–6406. IEEE, 2009.

[37] Jane Wu, Russell C Bingham, Samantha Ting, Kolton Yager, Zoë J Wood, Timmy Gambin, and Christopher M Clark. Multi-AUV motion planning for archeological site mapping and photogrammetric reconstruction. *Journal of Field Robotics*, 36(7):1250–1269, 2019.

[38] Stefan B Williams, Oscar Pizarro, Michael Jakuba, and Neville Barrett. AUV benthic habitat mapping in South Eastern Tasmania. In *Field and Service Robotics*, pages 275–284. Springer, 2010.

[39] Woods Hole Oceanographic Institution: Acoustic Micromodem. https://acomms.whoi.edu/micro-modem/, 2021.

[40] Liam Paull, Sajad Saeedi, Mae Seto, and Howard Li. AUV navigation and localization: A review. *IEEE Journal of Oceanic Engineering*, 39(1):131–149, 2013.

[41] Aditya S Gadre and Daniel J Stilwell. A complete solution to underwater navigation in the presence of unknown currents based on range measurements from a single location. In *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1420–1425. IEEE, 2005.

[42] Qizhu Chen, Keyou You, and Shiji Song. Cooperative localization for autonomous underwater vehicles using parallel projection. In *2017 13th IEEE International Conference on Control & Automation (ICCA)*, pages 788–793. IEEE, 2017.

[43] Alexander Bahr, John J Leonard, and Maurice F Fallon. Cooperative localization for autonomous underwater vehicles. *The International Journal of Robotics Research*, 28(6):714–728, 2009.

[44] Yulong Huang, Yonggang Zhang, Bo Xu, Zhemin Wu, and Jonathon A Chambers. A new adaptive extended Kalman filter for cooperative localization. *IEEE Transactions on Aerospace and Electronic Systems*, 54(1):353–368, 2017.

[45] Maurice F Fallon, Georgios Papadopoulos, John J Leonard, and Nicholas M Patrikalakis. Cooperative AUV navigation using a single maneuvering surface craft. *The International Journal of Robotics Research*, 29(12):1461–1474, 2010.

[46] Philippe Baccou and Bruno Jouvencel. Homing and navigation using one transponder for AUV, postprocessing comparisons results with long base-line navigation. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 4004–4009. IEEE, 2002.

[47] Brian T Hinson, Michael K Binder, and Kristi A Morgansen. Path planning to optimize observability in a planar uniform flow field. In *2013 American Control Conference*, pages 1392–1399. IEEE, 2013.

[48] Andrew Ross and Jérôme Jouffroy. Remarks on the observability of single beacon underwater navigation. In *Proc. Intl. Symp. Unmanned Unteth. Subm. Tech*, 2005.

[49] Bruno Ferreira, Aníbal Matos, and Nuno Cruz. Single beacon navigation: Localization and control of the MARES AUV. In *OCEANS 2010 MTS/IEEE SEATTLE*, pages 1–9. IEEE, 2010.

[50] Jake D Quenzer and Kristi A Morgansen. Observability based control in range-only underwater vehicle localization. In *2014 American Control Conference*, pages 4702–4707. IEEE, 2014.

[51] AJ Murphy, MJ Landamore, and RW Birmingham. The role of autonomous underwater vehicles for marine search and rescue operations. *Underwater Technology*, 27(4):195–205, 2008.

[52] Lai Wei, Xiaobo Tan, and Vaibhav Srivastava. Expedited multi-target search with guaranteed performance via multi-fidelity gaussian processes. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7095–7100. IEEE, 2020.

[53] Osama Ennasr, Giorgos Mamakoukas, Todd Murphey, and Xiaobo Tan. Ergodic exploration for adaptive sampling of water columns using gliding robotic fish. In *Dynamic Systems and Control Conference*, volume 51913, page V003T32A016. American Society of Mechanical Engineers, 2018.

[54] Franco Hidalgo and Thomas Bräunl. Review of underwater SLAM techniques. In *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, pages 306–311. IEEE, 2015.

[55] Mahdi Jadaliha, Yunfei Xu, Jongeun Choi, Nicholas S Johnson, and Weiming Li. Gaussian process regression for sensor networks under localization uncertainty. *IEEE Transactions on Signal Processing*, 61(2):223–237, 2012.

[56] Sungjoon Choi, Mahdi Jadaliha, Jongeun Choi, and Songhwai Oh. Distributed Gaussian process regression under localization uncertainty. *Journal of Dynamic Systems, Measurement, and Control*, 137(3):031007, 2015.

[57] Patrick Dallaire, Camille Besse, and Brahim Chaib-Draa. An approximate inference with Gaussian process to latent functions from uncertain data. *Neurocomputing*, 74(11):1945–1955, 2011.

[58] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Warped Gaussian processes occupancy mapping with uncertain inputs. *IEEE Robotics and Automation Letters*, 2(2):680–687, 2017.

[59] Daniel Cervone and Natesh S Pillai. Gaussian process regression with location errors. *arXiv preprint arXiv:1506.08256*, 2015.

[60] Hildo Bijl, Thomas B Schön, Jan-Willem van Wingerden, and Michel Verhaegen. Online sparse Gaussian process training with input noise. *ArXiv*, abs/1601.08068, 2016.

[61] Andrew McHutchon and Carl Rasmussen. Gaussian process training with input noise. *Advances in Neural Information Processing Systems*, 24, 2011.

[62] Rafael Oliveira, Lionel Ott, and Fabio Ramos. Bayesian optimisation under uncertain inputs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1177–1184. PMLR, 2019.

[63] Feitian Zhang, Jianxun Wang, John Thon, Cody Thon, Elena Litchman, and Xiaobo Tan. Gliding robotic fish for mobile sampling of aquatic environments. In *Networking, Sensing and Control (ICNSC), 2014 IEEE 11th International Conference on*, pages 167–172. IEEE, 2014.

[64] A Pedro Aguiar and Joao Pedro Hespanha. Position tracking of underactuated vehicles. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 3, pages 1988–1993. IEEE, 2003.

[65] K Duc Do, J Pan, and Zhong-Ping Jiang. Robust and adaptive path following for underactuated autonomous underwater vehicles. *Ocean Engineering*, 31(16):1967–1997, 2004.

[66] Khac Duc Do and Jie Pan. *Control of Ships and Underwater Vehicles: Design for Underactuated and Nonlinear Marine Systems*, chapter 12-13, pages 109–124. Springer Science & Business Media, 2009.

[67] F Rezazadegan, K Shojaei, F Sheikholeslam, and A Chatraei. A novel approach to 6-DOF adaptive trajectory tracking control of an AUV in the presence of parameter uncertainties. *Ocean Engineering*, 107:246–258, 2015.

[68] Mansour Karkoub, Hsiu-Ming Wu, and Chih-Lyang Hwang. Nonlinear trajectory-tracking control of an autonomous underwater vehicle. *Ocean Engineering*, 145:188–198, 2017.

[69] Jinqiang Wang, Cong Wang, Yingjie Wei, and Chengju Zhang. Command filter based adaptive neural trajectory tracking control of an underactuated underwater vehicle in three-dimensional space. *Ocean Engineering*, 180:175–186, 2019.

[70] Zewei Zheng, Linping Ruan, and Ming Zhu. Output-constrained tracking control of an underactuated autonomous underwater vehicle with uncertainties. *Ocean Engineering*, 175:241–250, 2019.

[71] Ruikun Xu, Guoyuan Tang, Lijun Han, and De Xie. Trajectory tracking control for a CMG-based underwater vehicle with input saturation in 3D space. *Ocean Engineering*, 173:587–598, 2019.

[72] Junjun Cao, Junliang Cao, Zheng Zeng, and Lian Lian. Nonlinear multiple-input-multiple-output adaptive backstepping control of underwater glider systems. *International Journal of Advanced Robotic Systems*, 13(6):1729881416669484, 2016.

[73] Aaron D Ames, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control*, pages 6271–6278. IEEE, 2014.

[74] Urs Borrmann, Li Wang, Aaron D Ames, and Magnus Egerstedt. Control barrier certificates for safe swarm behavior. *IFAC-PapersOnLine*, 48(27):68–73, 2015.

[75] Aaron D Ames, Xiangru Xu, Jessy W Grizzle, and Paulo Tabuada. Control barrier function based quadratic programs for safety critical systems. *IEEE Transactions on Automatic Control*, 62(8):3861–3876, 2016.

[76] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431. IEEE, 2019.

[77] Dimitra Panagou, Dušan M Stipanovič, and Petros G Voulgaris. Multi-objective control for multi-agent systems using lyapunov-like barrier functions. In *52nd IEEE Conference on Decision and Control*, pages 1478–1483. IEEE, 2013.

[78] Dimitra Panagou, Dušan M Stipanović, and Petros G Voulgaris. Distributed coordination control for multi-robot networks using Lyapunov-like barrier functions. *IEEE Transactions on Automatic Control*, 61(3):617–632, 2015.

[79] Li Wang, Aaron D Ames, and Magnus Egerstedt. Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 2659–2664. IEEE, 2016.

[80] Paul Glotfelter, Jorge Cortés, and Magnus Egerstedt. Nonsmooth barrier functions with applications to multi-robot systems. *IEEE Control Systems Letters*, 1(2):310–315, 2017.

[81] Demetris Coleman, Shaunak D Bopardikar, Vaibhav Srivastava, and Xiaobo Tan. Exploration of unknown scalar fields with multifidelity gaussian processes under localization uncertainty. In *2023 American Control Conference (ACC)*, pages 3296–3303. IEEE, 2023.

[82] Marc C Kennedy and Anthony O'Hagan. Predicting the output from a complex computer code when fast approximations are available. *Biometrika*, 87(1):1–13, 2000.

[83] Loic Le Gratiet and Josselin Garnier. Recursive co-kriging model for design of computer experiments with multiple levels of fidelity. *International Journal for Uncertainty Quantification*, 4(5), 2014.

[84] Loïc Brevault, Mathieu Balesdent, and Ali Hebbal. Overview of Gaussian process based multi-fidelity techniques with variable relationship between fidelities, application to aerospace systems. *Aerospace Science and Technology*, 107:106339, 2020.

[85] Simone Ficini, Umberto Iemma, Riccardo Pellegrini, Andrea Serani, and Matteo Diez. Assessing the performance of an adaptive multi-fidelity gaussian process with noisy training data: A statistical analysis. In *AIAA AVIATION 2021 FORUM*, page 3098, 2021.

[86] S Ficini, R Pellegrini, A Odetti, A Serani, U Iemma, M Caccia, and M Diez. Uncertainty quantification of an autonomous surface vehicle by multi-fidelity surrogate models. In *9th*

edition of the International Conference on Computational Methods for Coupled Problems in Science and Engineering (COUPLED PROBLEMS 2021), 2021.

[87] Yoonchang Sung, Deeksha Dixit, and Pratap Tokekar. Environmental hotspot identification in limited time with a UAV equipped with a downward-facing camera. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13264–13270. IEEE, 2021.

[88] Maria L Castaño and Xiaobo Tan. Rapid maneuvering control of pectoral fin-actuated robotic fish. In *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, pages 705–712. IEEE, 2021.

[89] Maria L Castaño and Xiaobo Tan. Model predictive control-based path-following for tail-actuated robotic fish. *Journal of Dynamic Systems, Measurement, and Control*, 141(7), 2019.

[90] Feitian Zhang, Osama Ennasr, Elena Litchman, and Xiaobo Tan. Autonomous sampling of water columns using gliding robotic fish: Algorithms and harmful-algae-sampling experiments. *IEEE Systems Journal*, 10(3):1271–1281, 2015.

[91] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.

[92] Maximilian Krogius, Acshi Haggenmiller, and Edwin Olson. Flexible layouts for fiducial tags. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1898–1903. IEEE, 2019.

[93] Petar Kokotovic, Hassan K Khali, and John O'reilly. *Singular Perturbation Methods in Control: Analysis and Design*, volume 25. SIAM, 1999.

[94] S Esteban, F Gordillo, and J Aracil. Three-time scale singular perturbation control and stability analysis for an autonomous helicopter on a platform. *International Journal of Robust and Nonlinear Control*, 23(12):1360–1392, 2013.

[95] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[96] Shauying R Kou, David L Elliott, and Tzyh Jong Tarn. Observability of nonlinear systems. *Information and Control*, 22(1):89–99, 1973.

[97] Robert Hermann and Arthur Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, 22(5):728–740, 1977.

[98] Aaron D Ames, Gennaro Notomista, Yorai Wardi, and Magnus Egerstedt. Integral control barrier functions for dynamically defined control laws. *IEEE Control Systems Letters*, 5(3):887–892, 2020.

[99] Nathan D Powel and Kristi A Morgansen. Empirical observability gramian rank condition for weak observability of nonlinear systems with control. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 6342–6348. IEEE, 2015.

[100] Nathan Powel. *Noise-Enabled Observability of Nonlinear Dynamic Systems Using the Empirical Observability Gramian*. PhD thesis, 2016.

[101] Quan Nguyen and Koushil Sreenath. L 1 adaptive control barrier functions for nonlinear underactuated systems. In *2022 American Control Conference (ACC)*, pages 721–728. IEEE, 2022.

[102] Axton Isaly, Omkar Sudhir Patil, Ricardo G Sanfelice, and Warren E Dixon. Adaptive safety with multiple barrier functions using integral concurrent learning. In *2021 American Control Conference (ACC)*, pages 3719–3724. IEEE, 2021.

[103] Demetris Coleman, Shaunak D Bopardikar, and Xiaobo Tan. Observability-aware target tracking with range only measurements. In *2021 American Control Conference (ACC)*, pages 4217–4224. IEEE, 2021.

[104] MathWorks. Matlab optimization toolbox, 2020. The MathWorks, Natick, MA, USA.

[105] Christopher K Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[106] Robert E Sheldahl and Paul C Klimas. Aerodynamic characteristics of seven symmetrical airfoil sections through 180-degree angle of attack for use in aerodynamic analysis of vertical axis wind turbines. Technical report, Sandia National Labs., Albuquerque, NM (USA), 1981.

[107] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[108] Demetris Coleman, Maria Castaño, Osama Ennasr, and Xiaobo Tan. Backstepping-based trajectory tracking for underwater gliders. In *Dynamic Systems and Control Conference*, volume 59162, page V003T19A005. American Society of Mechanical Engineers, 2019.

[109] Osama Ennasr, Giorgos Mamakoukas, Maria Castaño, Demetris Coleman, Todd Murphey, and Xiaobo Tan. Adaptive single action control policies for linearly parameterized systems. In *Dynamic Systems and Control Conference*, volume 59148, page V001T02A005. American Society of Mechanical Engineers, 2019.

[110] Demetris Coleman and Xiaobo Tan. Backstepping control of gliding robotic fish for trajectory tracking in 3D space. In *2020 American Control Conference (ACC)*, pages 3730–3736. IEEE, 2020.

[111] Maria L Castaño. *Nonlinear Control of Robotic Fish*. PhD thesis, Michigan State University, 2021.

[112] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation*, pages 723–730. IEEE, 2011.

[113] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[114] Demetris Coleman, Maria Castaño, and Xiaobo Tan. Backstepping control of gliding robotic fish for pitch and 3D trajectory tracking. *Control Engineering Practice*, 129:105350, 2022.

[115] George Mathew and Igor Mezić. Metrics for ergodicity and design of ergodic dynamics for multi-agent systems. *Physica D: Nonlinear Phenomena*, 240(4-5):432–442, 2011.

[116] Elif Ayvali, Hadi Salman, and Howie Choset. Ergodic coverage in constrained environments using stochastic trajectory optimization. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5204–5210. IEEE, 2017.

[117] Rabiul Hasan Kabir and Kooktae Lee. Receding-horizon ergodic exploration planning using optimal transport theory. In *2020 American Control Conference (ACC)*, pages 1447–1452. IEEE, 2020.

[118] Ian Abraham, Ahalya Prabhakar, and Todd D Murphey. An ergodic measure for active learning from equilibrium. *IEEE Transactions on Automation Science and Engineering*, 18(3):917–931, 2021.

[119] GPy GPy. A gaussian process framework in python, 2012.

[120] Andrei Paleyes, Mark Pullin, Maren Mahsereci, Cliff McCollum, Neil D Lawrence, and Javier González. Emulation of physical processes with emukit. *arXiv preprint arXiv:2110.13293*, 2021.

[121] Dongsik Chang, Wencen Wu, Catherine R Edwards, and Fumin Zhang. Motion tomography: Mapping flow fields using autonomous underwater vehicles. *The International Journal of Robotics Research*, 36(3):320–336, 2017.

[122] Piyush Gupta and Vaibhav Srivastava. Deterministic sequencing of exploration and exploitation for reinforcement learning. In *2022 IEEE 61st Conference on Decision and Control (CDC)*, pages 2313–2318. IEEE, 2022.

[123] Jeremy Coulson, John Lygeros, and Florian Dörfler. Data-enabled predictive control: In the shallows of the deepc. In *2019 18th European Control Conference (ECC)*, pages 307–312. IEEE, 2019.

[124] Giorgos Mamakoukas, Maria L Castano, Xiaobo Tan, and Todd D Murphey. Derivative-based koopman operators for real-time control of robotic systems. *IEEE Transactions on Robotics*, 37(6):2173–2192, 2021.

[125] Thomas Beckers and Sandra Hirche. Prediction with approximated gaussian process dynamical models. *IEEE Transactions on Automatic Control*, 67(12):6460–6473, 2021.

[126] Urban Fasel, Eurika Kaiser, J Nathan Kutz, Bingni W Brunton, and Steven L Brunton. Sindy with control: A tutorial. In *2021 60th IEEE Conference on Decision and Control (CDC)*, pages 16–21. IEEE, 2021.

[127] Charles Vorbach, Ramin Hasani, Alexander Amini, Mathias Lechner, and Daniela Rus. Causal navigation by continuous-time neural networks. *Advances in Neural Information Processing Systems*, 34:12425–12440, 2021.

[128] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Mathias Lechner, Yutong Ban, Chuang Gan, and Daniela Rus. On the forward invariance of neural odes. In *International conference on machine learning*, pages 38100–38124. PMLR, 2023.

[129] Wei Xiao, Tsun-Hsuan Wang, Ramin Hasani, Makram Chahine, Alexander Amini, Xiao Li, and Daniela Rus. Barriernet: Differentiable control barrier functions for learning of safe robot control. *IEEE Transactions on Robotics*, 2023.

APPENDIX A: TWO-TIME-SCALE ANALYSIS OF SINGULARLY PERTURBED SYSTEMS

A standard model for a two-time-scale system is given by

$$\begin{bmatrix} \dot{x} \\ \varepsilon \dot{z} \end{bmatrix} = \begin{bmatrix} f(t,x,z,\varepsilon) \\ g(t,x,z,\varepsilon) \end{bmatrix}, \quad \begin{matrix} x \in R^n \\ z \in R^m \end{matrix} \tag{A.1}$$

Time-scale analysis requires five conditions to be satisfied for all $(t,x,z,\varepsilon) \in [t_0,\infty) \times B_x \times B_z \times [0,\varepsilon_1]$ so that a composite Lyapunov function can be constructed to establish the asymptotic stability of the system (A.1), where $B_x \subset R^n$ and $B_z \subset R^m$ are closed sets. This is done by studying the properties of the reduced system $\dot{x} = f(t,x,z,\varepsilon)$ evolving on a manifold given by $z = h(t,x)$, the boundary-layer system $\frac{dz}{d\tau} = g(t,x,z,\varepsilon)$ with $\tau = \frac{t}{\varepsilon}$ and x treated as a fixed parameter, and their interconnections. The assumptions are given as follows [25, 93].

Assumption 5. *There exists an isolated equilibrium point at the origin ($x = 0, z = 0$) for the system (A.1) such that*

$$f(t,0,0,\varepsilon) = g(t,0,0,\varepsilon) = 0.$$

In addition, for a given x, $z = h(t,x)$ is a unique root of $g(t,x,z,0)$, such that $g(x,h(t,x),0) = 0$ and there exists a class κ function ρ such that $||h(x)|| \leq \rho(||x||)$.

Assumption 6. *$x = 0$ is an asymptotically stable equilibrium for the reduced-order system; namely, there exists some Lyapunov function candidate $V(t,x)$ such that*

$$0 < q_1(||x||) \leq V(t,x) \leq q_2(||x||)$$

for some class κ functions q_1 and q_2, and the following holds:

$$\frac{\partial V(t,x)}{\partial t} + \frac{\partial V(t,x)}{\partial x} f(t,x,h(t,x),0) \leq -\gamma_1 \psi_1^2(x)$$

where γ_1 is a positive scalar and $\psi_1(x)$ is a continuous scalar function of x that vanishes only when x is 0.

Assumption 7. *There exists a Lyapunov function candidate $W(t,x,z)$ satisfying*

$$0 < q_3(||z-h(t,x)||) \leq W(t,x,z) \leq q_4(||z-h(t,x)||)$$

$$W(t,x,z) > 0, \forall z \neq h(t,x), W(t,x,h(t,x)) = 0$$

for some class κ functions $q_3(\cdot)$ and $q_4(\cdot)$ and

$$\frac{\partial W(t,x,z)}{\partial z} g(t,x,z,0) \leq -\gamma_2 \psi_2^2(z-h(t,x)), \gamma_2 > 0$$

where γ_2 is a positive constant, $\psi_2(\cdot) > 0$ is a scalar function that vanishes only when its argument is 0, and x is treated as a fixed parameter.

Assumption 8.

$$\frac{\partial V(t,x)}{\partial x}[f(t,x,z,\varepsilon) - f(t,x,h(t,x),0)]$$
$$\leq \beta_1 \psi_1(x) \psi_2(z-h(t,x)) + \varepsilon \alpha_1 \psi_1^2(x)$$

for some non-negative constants α_1 and β_1.

Assumption 9.

$$\frac{\partial W(t,x,z)}{\partial z}[g(t,x,z,\varepsilon) - g(t,x,h(t,x),0)] \leq$$
$$\varepsilon \alpha_2 \psi_2^2(z-h(t,x)) + \beta_2 \psi_1(x) \psi_2(z-h(t,x))$$

$$\frac{\partial W}{\partial t} + \frac{\partial W}{\partial x} f(t,x,z,\varepsilon) \leq \gamma_3 \psi_2^2(z - h(t,x)) + \alpha_3 \psi_1(x) \psi_2(z - h(t,x))$$

for non-negative constants α_2, β_2, γ_3, *and* α_3.

Conditions 5-7 guarantee asymptotic stability of the reduced and boundary-layer systems. The fourth and fifth conditions handle the interconnection between the reduced model and the boundary-layer system by looking at the derivatives of Lyapunov candidate functions $V(t,x)$ and $W(x,z)$, taking $g(t,x,z,\varepsilon) - g(t,x,h(t,x),0)$, $f(t,x,z,\varepsilon) - f(t,x,h(t,x),0)$, and $z - h(t,x)$ as perturbations, and imposing conditions on the growth of those perturbations.

Theorem 2. *Consider the singularly perturbed system (A.1) that satisfies assumptions 5-9. Then there exists an* $\varepsilon^* > 0$ *such that the equilibrium of (A.1) is asymptotically stable for all* $\varepsilon < \varepsilon^*$. *Furthermore, a candidate composite Lyapunov function for the system can be constructed .from the weighted sum*

$$V_c(t,x,z) = (1-d)V(t,x) + dW(t,x,z) \tag{A.2}$$

where $0 < d < 1$.

APPENDIX B: PARTICLE SWARM PARAMETER ESTIMATION

Let

$$\begin{cases} \dot{X} = f(X,U,p), \\ Y = h(X) \end{cases} \tag{B.1}$$

be a nonlinear system with state $X \in \mathbb{R}^n$, input $U \in \mathbb{R}^m$, parameters $p \in \mathbb{R}^q$, and outputs $Y \in \mathbb{R}^o$. Given a data set containing the control trajectory $U(t)$ and measured state trajectories $Y(t)$ taken at discrete times $t \in [t_0, t_1]$, p can be estimated using particle swarm optimization (PSO). PSO optimizes a cost $J(\cdot)$ by considering a swarm of particles that explore a space. Taking the parameter vector as the state space of the particles in the PSO algorithm, the particle swarm optimization paradigm allows a particle p_i to travel across the parameter search space according to

$$\begin{aligned} v_i^{k+1} &= w v_i^k + a_1 rand()(p_{b_i} - p_i^k) + a_2 rand()(g_b - p_i^k) \\ p_i^{k+1} &= p_i^k + v_i^{k+1} \\ p_i^{k+1} &\in [l_b, u_b] \end{aligned}$$

where $rand() \in [0,1]$, $[l_b, u_b]$ defines box constraint boundaries on the parameter search space, v_i^k is the velocity of particle i at time k, w is a momentum term on the previous velocity, and a_1 and a_2 scale the acceleration caused by the differences between current state p_i and the personal best p_{b_i} and global best g_b optimizers of $J(p_i)$. Typically a user-selected number of particles are generated from a uniform distribution across the search space defined by $[l_b, u_b]$ but the search can be biased by selecting particles based on prior knowledge of the parameters. The cost for the PSO algorithm

is posed as

$$p^* = \text{argmin}_p \quad J(p) = \sum_{j=0}^{o} \alpha_j ||h_j(\hat{X}(t)) - Y_j(t)||^2$$

subject to

$$\begin{aligned} \dot{\hat{X}}(t) &= f(\hat{X}(t), U(t), p) \\ \hat{X}(0) &= X(0) \\ t &\in [t_0, t_1] \end{aligned} \tag{B.2}$$

where $\hat{X}$ is the simulated state generated from the dynamics in Eq. (B.1) parameterized by the model parameter vector p with control input $U(t)$, p^* is the optimal parameter vector, $X(0)$ corresponds to the initial system state for the optimization horizon, and α_j is a weight corresponding to a specific measurement. The cost function penalizes a weighted sum of the norm of the errors between the measurements from the true system and the measurements calculated using the state simulated from the dynamic model with a specific set of parameters. The can be expanded to account for multiple sets of data by summing the cost for each set of data.

APPENDIX C: ABLATION STUDY FOR CBF-ENFORCED OBSERVABILITY

For the system from Section 4.3 and the associated controller, the plots in Fig.C.1 show the effect of using the baseline controller and the effect of different functions for $\alpha(\cdot)$ in the CBF constraint in Eq. (4.9) under state feedback. Some of the functions venture into the territory of barrier certificates [74] by aggressively pushing the state toward the interior of the set. The constraint is not always able to be met for these cases. When this happens, the nominal control is implemented.

Changing the function $\alpha(\cdot)$ changes how aggressive (large vs tight) the orbiting behavior is. At close distances, the orbiting behavior seen in Section 4.4 tends to persist, but as the distance gets larger there are noticeable differences. Tuning $\alpha(\cdot)$ to aggressively push the state toward the interior (using functions such as $\alpha(B(x)) = -B(x)$) of the set results in larger orbits that tend to have better observability earlier on but take longer to approach the target. Choosing $\alpha(\cdot)$ such that it allows the state to get closer to the boundary of the set (functions such as $B(x)$ and $B(x)^3$) results in the unicycle moving almost directly toward the target before starting the orbiting behavior.

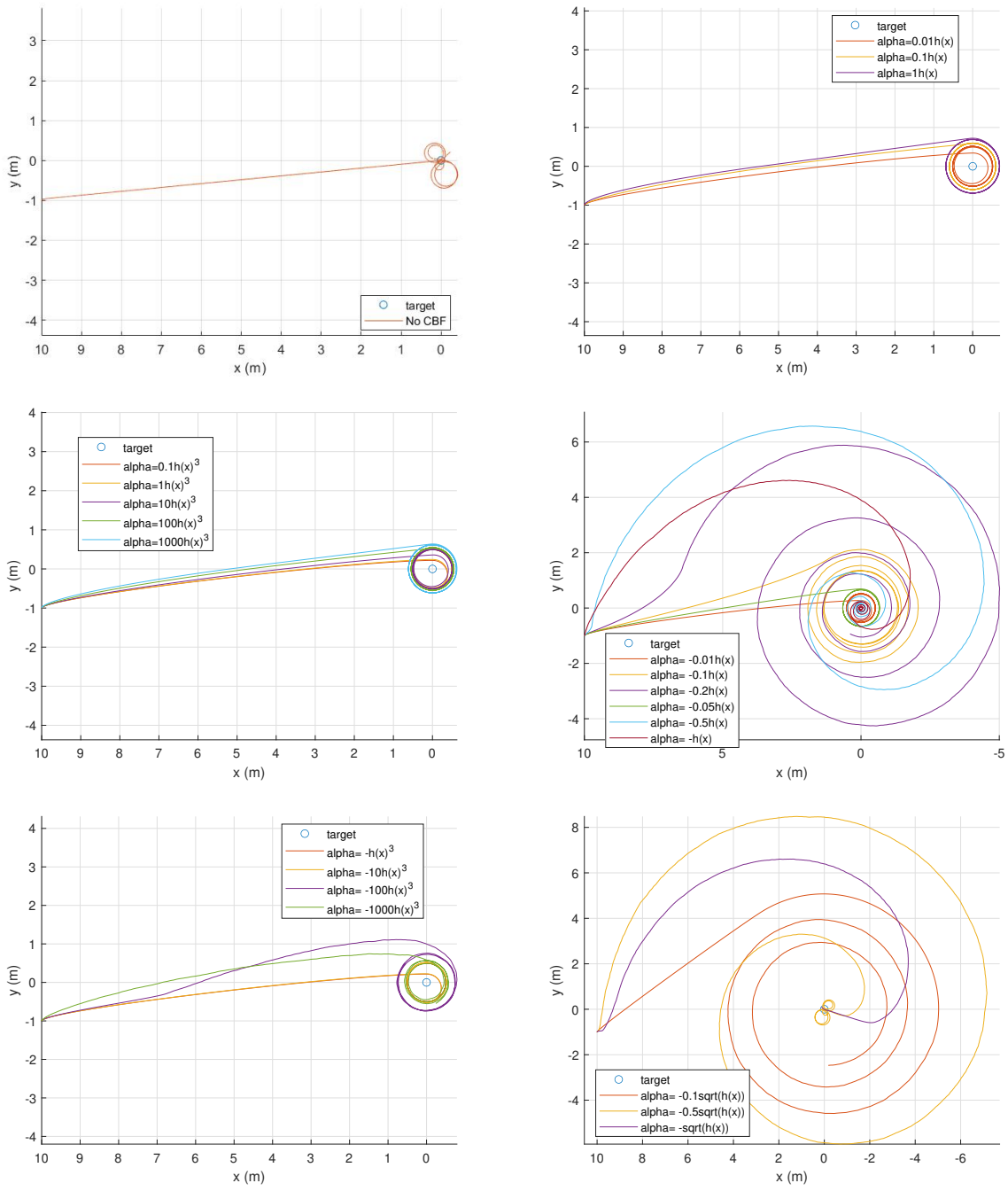


Figure C.1: Effect of $\alpha(\cdot)$ on behavior for a unicycle model tracking a static target. Here $h(x)$ is the barrier function.

APPENDIX D: ERGODIC METRIC AND MEASURE

For a rectangular domain $U \subset \mathbb{R}^n$ and a target probability distribution μ, the ergodic metric is given by [115]

$$E^2(t) = \int_0^R \int_U (d^t(x,r) - \bar{\mu}(x,r))^2 dx dr,\ R > 0. \tag{D.1}$$

Here $\bar{\mu}(x,r) = \int_U \mu(y)\chi_{(x,r)}(y)dy$ is the measure of a spherical set $B(x,r) = \{y : \|y-x\| \leq r\}$ with a corresponding indicator function $\chi_{(x,r)}(y)$ for the target probability distribution μ and $d^t(x,r) = \frac{1}{t}\int_0^t \chi_{(x,r)}(\bar{x}(\tau))d\tau$ is the fraction of time spent in the set $B(x,r)$ by an agent with trajectory $\bar{x} : [0,t] \to \mathbb{R}^n$. The metric is motivated by the condition for ergodicity,

$$\lim_{t\to\infty} d^t(x,r) = \bar{\mu}(x,r).$$

It is worth noting that, for a single agent, the spatial statistics at a point x, given by

$$d^t(x,r) = \frac{1}{t_f - t_0}\int_{t_0}^{t_f} \chi_{(x,r)}(\bar{x}(\tau))d\tau,$$

can be computed for partial time periods as $d^t(x,r) = \frac{1}{t_f - t_0}(\int_{t_0}^{t_1} \chi_{(x,r)}(\bar{x}(\tau))d\tau + \int_{t_1}^{t_f} \chi_{(x,r)}(\bar{x}(\tau))d\tau)$.

It is noted in [118] and [116] that KL-divergence is suitable as a measure of ergodicity. It can be used by approximating $d^t(x,r)$ as $q(x|\bar{x}) = \frac{1}{t_f - t_0}\int_{t_0}^{t_f} \frac{1}{\eta}\exp(-\frac{1}{2}[x-\bar{x}(t)]^T\Sigma^{-1}[x-\bar{x}(t)])$ where Σ and η are the covariance and normalizing factor, respectively, of a Gaussian function. Using this, the KL-divergence between the time-averaged statistics of the robot and the target distribution μ is given as $D_{KL}(\mu,q) = \int_U \mu(x)\log(\frac{\mu(x)}{q(x|\bar{x})})dx$. Practically, the domain U can be approximated using

a grid of points that sufficiently cover U. The ergodic measure can then be expressed as

$$\begin{aligned} D_{KL}(\mu, q) &= \sum_{x \in U} \mu(x) \log(\frac{\mu(x)}{q(x|\bar{x})}) \\ &= \sum_{x \in U} \mu(x) \log(\mu(x)) - \sum_{x \in U} \mu(x) \log(q(x|\bar{x})) \\ &= \sum_{x \in U} \mu(x) \log(\mu(x)) + \sum_{x \in U} \mu(x) \log(t_f - t_0) \\ &\quad - \sum_{x \in U} \mu(x) \log(\int_{t_0}^{t_f} \frac{1}{\eta} \exp(-\frac{1}{2} \|x - \bar{x}(t)\|^2_{\Sigma^{-1}}) \end{aligned}$$

where $\|x - \bar{x}(t)\|^2_{\Sigma^{-1}} = [x - \bar{x}(t)]^T \Sigma^{-1} [x - \bar{x}(t)])$