NONLINEAR CONTROL APPLICATIONS FOR NONHOLONOMIC AUTONOMOUS SYSTEMS

A Thesis

Presented to

The Faculty of the Department of Mechanical Engineering

Villanova University

In Partial Fulfillment

Of the Requirements for the Degree of

Master of Science in Mechanical Engineering

by

Lucas C. McNinch

December 2009

NONLINEAR CONTROL APPLICATIONS FOR NONHOLONOMIC AUTONOMOUS SYSTEMS

by

Lucas C. McNinch

December 2009

Kenneth R. Muske, Ph.D. Thesis Advisor	Date
Hashem Ashrafiuon, Ph.D. Thesis Advisor	Date
C. Nataraj, Ph.D. Chairman, Deptartment of Mechanical Engineering	Date
Gary A. Gabriele, Ph.D. Dean, College of Engineering	Date

Acknowledgements

I would like to thank my thesis advisors Dr. Kenneth Muske and Dr. Hashem Ashrafiuon for their support throughout my time here at Villanova. They have always been available to provide advice and guidance to make my graduate studies successful. It has been a privilege to work with them and none of this work would have been possible without their vision and knowledge.

I would also like to thank my fellow graduate student, research teammate and friend Reza Soltan for sharing his knowledge and ideas. I have always valued our discussions and much of my work has been greatly enhanced by his experience and creativity.

Lucas C. McNinch

December 2009

ABSTRACT

NONLINEAR CONTROL APPLICATIONS FOR NONHOLONOMIC AUTONOMOUS SYSTEMS

by

Lucas C. McNinch

Villanova University, 2009

THESIS ADVISORS: Kenneth R. Muske, Ph.D. and Hashem Ashrafiuon, Ph.D.

This thesis presents several nonlinear control applications for nonholonomic autonomous wheeled mobile robot and unmanned surface vessel (USV) systems. First, an overview of nonlinear control and nonholonomic systems is given and sliding mode and model predictive control methods are introduced. Next, a kinematic controller is applied to an experimental wheeled mobile robot for validation of rapid prototyping techniques and new trajectory planning algorithms. Then, sliding mode controllers are developed for setpoint and trajectory tracking of USVs. Sliding mode control provides a robust control law with modest computational requirements that can be implemented on small-scale model USV systems. However, tuning the control parameters can be very non-intuitive and one set of parameters is rarely suitable for different initial conditions. A receding horizon model predictive controller is then developed for setpoint and tracking control. This controller yields open-loop optimal performance regardless of initial conditions, constraints or disturbances but the high computational demands make it challenging to apply this method to small-scale USV systems with fast dynamics. Finally, a predictive sliding mode cascade controller is developed combining the strengths of the sliding mode and model predictive control methods. Simulation and/or experimental results are presented for each application.

Contents

Acknowledgements **v**

Abstract **vi**

Contents **vii**

List of Tables **x**

List of Figures **xi**

Introduction **1**

1 Nonlinear Control and Nonholonomic Systems **4**
1.1 The Nonlinear Control Problem 5
1.2 Nonholonomic Systems 6
1.3 Sliding Mode Control 7
1.4 Model Predictive Control 8

2 Kinematic Control for Mobile Robots **10**
2.1 Mobile Robot Platform 11
2.1.1 Kinematic Model 12
2.1.2 Position Feedback 13
2.1.3 Local Tracking Control 14
2.1.4 Trajectory Tracking Performance 15
2.2 Reference Trajectory Determination 19
2.2.1 Target Tracking Trajectory 20
2.2.2 Limit Cycle Trajectory 21
2.2.3 Elliptical Limit Cycles 22
2.3 Coordinated Control 24
2.4 Obstacle Avoidance 26
2.5 Conclusions 28

3 Sliding Mode Control for Unmanned Surface Vessels **29**
3.1 USV System Model . . . 29
3.2 Experimental USV System . . . 31
3.3 Setpoint Control . . . 34
3.3.1 Partial Feedback Linearization . . . 34
3.3.2 Controllability Properties . . . 35
3.3.3 Sliding Mode Control Law . . . 36
3.3.4 Stability . . . 38
3.3.5 Results . . . 39
3.4 Trajectory Tracking Control . . . 45
3.4.1 Surge Control Law . . . 45
3.4.2 Lateral Motion Control Law . . . 47
3.4.3 Stability . . . 49
3.4.4 Results . . . 50

4 Model Predictive Control for Unmanned Surface Vessels **52**
4.1 System Dynamics . . . 53
4.2 Model Predictive Control . . . 55
4.3 Results . . . 56
4.3.1 Linear Target Trajectory Tracking . . . 58
4.3.2 Circular Target Trajectory Tracking . . . 60
4.3.3 Setpoint Control . . . 62

5 Predictive and Sliding Mode Cascade Control Structure for USVs **68**
5.1 Cascade Control Structure . . . 69
5.2 Circular Target Trajectory . . . 70
5.2.1 Minimum Tracking Error Objective . . . 71
5.2.2 Minimum Time Objective . . . 75
5.2.3 Minimum Energy Objective . . . 78
5.3 Linear Target Trajectory . . . 81
5.3.1 Minimum Tracking Error Objective . . . 81
5.3.2 Minimum Time Objective . . . 87

6 Conclusions **90**
6.1 Future Work . . . 90
6.1.1 Mobile Robots . . . 90
6.1.2 Cascade Control . . . 91
6.1.3 Trajectory Planning . . . 91

References **92**

A Vision System **97**
A.1 Hardware and MATLAB Interface 97
A.2 Image Processing 99
A.3 Calibration 101
A.3.1 2D Linear Interpolation 104

B Model USV **106**

C LEGO NXT and ECRobot **109**
C.1 Mobile Robot Embedded Software 109
C.1.1 Bluetooth Read 112
C.1.2 Bluetooth Write 113
C.2 Bluetooth Adapter 115
C.3 Host PC Communication 116
C.3.1 Serial Port Connection 116
C.3.2 MATLAB Bluetooth Communication 116
C.4 USV Motor Control Embedded Software 118

List of Tables

3.1 Model USV dimensions 32
3.2 Estimated USV system model parameters 33

5.1 Minimum error control parameters and cost: Circular target trajectory . . . 72
5.2 Minimum time control parameters and reach time: Circular target trajectory 76
5.3 Minimum energy control parameters and cost: Circular target trajectory . . 79
5.4 ITSE control parameters and cost: Linear target trajectory, unconstrained . 83
5.5 Constrained ITSE control parameters and cost: Linear target trajectory . . . 86
5.6 Minimum time control parameters and reach time: Linear target trajectory . 87

B.1 USV Motor Data 106
B.2 LEGO color code 108

List of Figures

2.1 Experimental mobile robot 12
2.2 Differential drive mobile robot schematic 13
2.3 Trajectory tracking example 16
2.4 Open-loop and closed-loop paths determined using encoder signals 16
2.5 Trajectory tracking errors 17
2.6 Closed-loop trajectory with camera position 18
2.7 Camera-based tracking errors 18
2.8 Closed-loop trajectory with slip 18
2.9 Camera-based tracking errors with slip 19
2.10 Experimental coordinated robot trajectories 25
2.11 Elliptical trajectory with camera position data 26
2.12 Experimental obstacle avoidance trajectories 27

3.1 Planar USV model schematic 30
3.2 Experimental USV 32
3.3 Experimental and simulated paths: Case 1 40
3.4 Experimental and simulated global trajectories: Case 1 41
3.5 Experimental and simulated surge force and yaw moment: Case 1 41
3.6 Simulated time history of the four surfaces: Case 1 42
3.7 Experimental and simulated paths: Case 2 43
3.8 Experimental and simulated global trajectories: Case 2 43
3.9 Experimental and simulated surge force and yaw moment for case 2 44
3.10 Simulated time history of the four surfaces: Case 2 44
3.11 Simulated USV path: trajectory tracking SMC 51
3.12 Control input: trajectory tracking SMC 51

4.1 Simulated nominal and perturbed trajectories 53
4.2 Experimental demonstration of system behavior 54
4.3 Control input vector parameterization 56
4.4 θ_d relationship 58
4.5 Simulated linear target trajectory paths 59
4.6 Linear target trajectory control input 60
4.7 Simulated circular target trajectory paths 61

4.8 Circular target trajectory control input 62
4.9 Simulated MPC setpoint path: Case 1 63
4.10 Simulated MPC setpoint closeup paths: Case 1 64
4.11 MPC setpoint control input: Case 1 65
4.12 Simulated MPC setpoint path: Case 2 66
4.13 MPC setpoint control input: Case 2 66
4.14 MPC setpoint error: Case 2 67

5.1 Controller block diagram 69
5.2 Simulated minimum error paths: Circular trajectory 73
5.3 Minimum error control input: Circular trajectory 73
5.4 Minimum error parameter evolution: Circular trajectory 74
5.5 Minimum error tracking errors: Circular trajectory 74
5.6 Minimum error normalized cost function values: Circular trajectory 74
5.7 Simulated minimum time paths: Circular trajectory 76
5.8 Minimum time control input: Circular trajectory 77
5.9 Minimum time parameter evolution: Circular trajectory 77
5.10 Minimum time tracking error: Circular trajectory 77
5.11 Simulated minimum energy paths: Circular trajectory 79
5.12 Minimum energy tracking error: Circular trajectory 80
5.13 Minimum energy parameter evolution: Circular trajectory 80
5.14 Minimum energy control input: Circular trajectory 80
5.15 Minimum energy objective function values: Circular trajectory 81
5.16 Simulated minimum error paths: Linear trajectory 82
5.17 Minimum error tracking error: Linear trajectory 82
5.18 Minimum error parameter evolution: Linear trajectory 83
5.19 Minimum error objective function values: Linear trajectory 83
5.20 Minimum error control input: Linear trajectory 84
5.21 Simulated constrained minimum error paths: Linear trajectory 85
5.22 Constrained minimum error tracking error: Linear trajectory 85
5.23 Constrained minimum error parameter evolution: Linear trajectory 86
5.24 Constrained minimum error control input: Linear trajectory 86
5.25 Constrained minimum error objective function values: Linear trajectory . . 87
5.26 Simulated minimum time paths: Linear trajectory 88
5.27 Minimum time tracking error: Linear trajectory 88
5.28 Minimum time control input: Linear trajectory 89
5.29 Minimum time parameter evolution: Linear trajectory 89

A.1 Sample camera image 100
A.2 Image processing 100
A.3 USV camera field of view 101
A.4 Mobile robot camera and calibration bar 102
A.5 USV camera and calibration lights 102

A.6 Camera calibration grid 103
A.7 x surface 103
A.8 Two dimensional interpolation: Top view 105
A.9 Two dimensional interpolation 105

B.1 Model USV internals 107
B.2 USV motor connector pin-out diagrams 108
B.3 LEGO connector pin-out diagram 108

C.1 ECRobot main model 110
C.2 'LC1' Subsystem containing function-call subsystems 111
C.3 Bluetooth Read subsystem 112
C.4 Bluetooth Write subsystems 114
C.5 Bluetooth manager control panel 115
C.6 USV motor speed control Simulink model 118
C.7 USV Motor speed control function-call subsystem 119

Introduction

Autonomous systems have received increased attention in the last decades as the need arises for mechanical systems to perform tasks in situations where it may be too difficult or dangerous for a human. The term *autonomous systems* in this context refers to mobile systems that carry out predefined tasks without human input or control, *i.e.*, mobile robots, unmanned surface or underwater vessels, unmanned air vehicles, etc. These systems are becoming more widely desired for military and industrial applications and as a result, there is a great deal of current research focused on control methodologies that allow these systems to carry out their tasks in the often unstructured and unpredictable environments in which they are used [1].

Many autonomous systems can be considered as underactuated mechanical systems [2]. Underactuated systems are defined as systems containing fewer inputs than degrees of freedom. Most of these underactuated systems are subject to first- or second-order nonholonomic constraints defined in terms of system velocities or accelerations respectively [3]. Nonholonomic systems present interesting control problems that require a variety of solutions ranging from relatively simple kinematic controllers to more sophisticated nonlinear controllers depending on the type of system, the application and the environment [4]. This thesis focuses on the control of two types of nonholonomic autonomous systems: the differential drive wheeled mobile robot and the autonomous unmanned surface vessel (USV).

Wheeled mobile robot systems have many real-world applications because they are a very useful and practical platform for both military and industrial purposes. Also, because of its simplicity, this type of system is an excellent platform for educational and research

applications. The system presented in this thesis employs a simple kinematic control law and serves as an experimental test bed in the Center for Nonlinear Dynamics and Control (CENDAC) at Villanova University. This system is used for experimental validation of new rapid-prototyping techniques and trajectory planning algorithms being developed by students and faculty in CENDAC [5].

Position control of wheeled mobile robots is a fairly well established area of research with much of the literature focused on kinematic control techniques. In general, position control of mobile robots can be divided into setpoint [6, 7] and trajectory tracking [8–12] control where the formulation of the corresponding control law can be very different depending on the intended application. A combined kinematic/torque control law developed using backstepping and applied to both trajectory tracking and setpoint stabilization is presented in [13].

Position control of autonomous USV systems is also a very active area of research which has received increased attention in the last decade with most of the research focusing on feedback linearization, backstepping, Lagrangian, and sliding mode control methods. As with control of mobile robot systems, the USV control problems can be divided into setpoint position [14–19] and trajectory tracking [20–28] control. In this thesis, several examples of nonlinear control laws for the USV setpoint and trajectory tracking control problems are presented. First, sliding mode controllers are presented for tracking [28] and setpoint control [29]. The advantage of these controllers is that they require very little computation and can be implemented on the small scale model USV system in the Unmanned Surface and Underwater Vessel (USUV) Laboratory at Villanova. A disadvantage of these controllers, however, is that tuning the control parameters can be very non-intuitive and often the optimal parameters for one initial condition can yield poor performance given different initial conditions. Next, a new model predictive control (MPC) method is applied to trajectory tracking and setpoint control [30, 31]. MPC is based on solving an open-loop optimal control problem at each sampling instant. As a result, open-loop optimal performance can be achieved regardless of initial conditions, constraints or disturbances. Another

advantage of this method is that the setpoint and trajectory tracking controllers use the same formulation. The disadvantage, however, is that the controller requires significant computation time making it challenging to implement on-line for small scale USV systems with relatively fast dynamics. Therefore, a cascade model predictive and sliding mode controller is presented which effectively combines the fast computation of the sliding mode tracking controller with the optimal performance of the model predictive controller.

The thesis is organized as follows: A general description of nonlinear control applications for nonholonomic systems is provided in the first two sections of Chapter 1. Section 1.3 gives an introduction to sliding mode control and Section 1.4 gives an overview of model predictive control. Chapter 2 presents the mobile robot platform developed for CENDAC and new trajectory planning algorithms are demonstrated experimentally. Chapter 3 presents the sliding mode setpoint and trajectory tracking controllers for USV systems and each are implemented experimentally. Chapter 4 presents the model predictive controller for USVs and Chapter 5 introduces the predictive sliding mode cascade controller. Chapter 6 provides concluding remarks and suggestions for future work. Additional documentation regarding the experimental equipment used in this work is provided in the appendix.

Chapter 1

Nonlinear Control and Nonholonomic Systems

The objective of control system design is defined by Slotine and Li as follows: *Given a physical system to be controlled and the specifications of its desired behavior, construct a feedback control law to make the closed-loop system display the desired behavior* [32]. Inherent in this definition are several basic concepts that must be considered when approaching any nonlinear control system design. First, a clear definition and control-based mathematical model of the system to be controlled must be developed. Second, the desired system behavior must be defined. In general, the desired behavior can be classified as either setpoint stabilization or trajectory tracking. Finally, the control system must be designed such that the desired system behavior can be achieved as accurately and reliably as possible. Each of these elements must be developed in congruence with the other two. That is to say, the system model must contain information about the system that is pertinent to the type of behavior that is desired and the form of the control law that is to be developed. Likewise, the control law must be designed taking into account the desired system behavior, the form of the system model and the physical properties of the system itself.

For much of the 20th century, developments in control system design were largely focused in the area of linear control. All linear control methods are based on the assumption

that the system to be controlled can be accurately described or approximated by linear, time-invariant (LTI) differential equations of the form

$$\dot{x} = Ax + Bu, \quad y = Cx \tag{1.1}$$

where $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^m$ is the control input and $y \in \mathbb{R}^p$ is the output of the system. Linear control is a very well developed subject with many relevant and successful applications for systems whose behavior in the desired operating ranges can be accurately described by these linear approximations. However, if the required operating range is large or the system is too nonlinear, linear control techniques are likely to break down as they fail to compensate for the nonlinearities of the true system. For these systems, nonlinear control techniques may provide greatly increased performance and stability. In the case of underactuated nonholonomic systems, nonlinear control methods are more appropriate because the nonholonomic constraints are inherently nonlinear.

1.1 The Nonlinear Control Problem

Many techniques have been developed for solving both nonlinear setpoint and nonlinear tracking control problems. These techniques include feedback linearization, adaptive control, gain scheduling, sliding control, model predictive control and robust control. The equation of motion for control-affine time-varying nonlinear dynamic systems is given by

$$\dot{x} = f(x,t) + g(x)u, \quad y = h(x) \tag{1.2}$$

where $x \in \mathbb{R}^n$ is the state of the system, $u \in \mathbb{R}^m$ is the control input and $y \in \mathbb{R}^p$ is the output of the system. The purpose of control system design is to define a control law $u(x,t)$ such that $x \to 0$ in the case of setpoint control, or $y \to y_d(t)$ in the case of tracking control, where $y_d(t)$ is the desired output trajectory.

Many simple, time-invariant, fully actuated problems may be solved by designing the

control law in such a way as to cancel the nonlinear dynamics of the plant, $f(x)$, leaving the closed-loop system in a linear form resembling that shown in Equation (1.1) such that linear control techniques can be applied. This type of control system design is known as *feedback linearization* and should not be confused with conventional linearization. In the case of feedback linearization, the complete nonlinear model in conjunction with state feedback is used to transform the system into a linear form, whereas the latter technique makes a linear approximation of the system about a given equilibrium point. Feedback linearization can be used to derive linear control systems for many nonlinear dynamic systems but cannot be applied universally. Some disadvantages of feedback linearization are that it requires full state feedback and it is not robust to modeling uncertainty or unmodeled dynamics. In this thesis, nonlinear sliding mode and model predictive controllers are introduced and several applications are presented.

1.2 Nonholonomic Systems

Nonholonomic systems are defined, in general, as systems that satisfy certain nonintegrable relations or constraints. Such constraints may arise as the result of physical kinematic constraints imposed on the system which are expressed in terms of the generalized velocities of the system and can not be integrated in time, and thus can not be expressed in terms of the generalized coordinates only. Constraints of this type expressed in terms of generalized velocities are considered first-order nonholonomic constraints. These systems can not be stabilized using time-invariant continuous feedback since they do not meet Brocketts necessary condition for feedback stabilization [2, 33].

Underactuated mechanical systems, *i.e.*, systems containing fewer inputs than degrees of freedom (DOF), are also examples of nonholonomic systems. Consider an n-DOF mechanical system under the action of $m < n$, $m \geq 1$ control inputs, $u \in \mathbb{R}^m$, where the set of generalized coordinates of the system is denoted by $q = (q_1, \ldots, q_n)$. The set of generalized coordinates is partitioned as $q = (q_a, q_u)$ where the vector $q_a \in \mathbb{R}^m$ represents the actuated coordinates and the vector $q_u \in \mathbb{R}^{n-m}$ represents the unactuated coordinates.

The equations of motion of the underactuated system can thus be expressed as

$$M_{11}(q)\ddot{q}_a + M_{12}(q)\ddot{q}_u + F_1(q,\dot{q}) = B(q)u \tag{1.3}$$
$$M_{21}(q)\ddot{q}_a + M_{22}(q)\ddot{q}_u + F_2(q,\dot{q}) = 0 \tag{1.4}$$

where $M_{ij}(q),\ i,j = 1,2$ represent block components of a symmetric and positive definite $n \times n$ inertia matrix. Assuming they are nonintegrable, Equations (1.4) define $n-m$ second-order nonholonomic constraints. Systems of this form are also not stabilizable using time-invariant continuous feedback due to their failure to meet Brockett's conditions. However, in both the first- and second-order cases, it can be shown that many nonholonomic systems are strongly accessible, *i.e.*, can be stabilized at all equilibrium points, and small-time locally controllable if certain conditions are met [2].

1.3 Sliding Mode Control

Sliding mode control [32] is a form of variable structure control which takes advantage of the fact that first-order systems are easier to control than general n^{th}-order systems. Like feedback linearization, sliding mode control is based on the idea of redefining the system dynamics in such a way as to create a control problem that is much easier to solve. Unlike feedback linearization however, sliding mode control can be applied to a wide range of nonlinear systems and it is very robust to modeling uncertainty, unmodeled dynamics and disturbances [34].

In the sliding mode control approach, asymptotically stable surfaces $\boldsymbol{s}$ are defined in terms of the system state variables and a nominal control law $\hat{\boldsymbol{u}}$ is defined such that all system trajectories which begin on the surfaces remain on the surfaces and slide along them until they reach the desired destination at their intersection. For system trajectories that do not begin on the surfaces, the nominal control law is modified such that system trajectories converge to the surfaces in finite time. The reaching conditions are normally established by defining $\boldsymbol{V} = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{s}$ as the Lyapunov function candidate and ensuring that

its time derivative is negative.

$$\dot{\boldsymbol{V}} = \boldsymbol{s}^T \dot{\boldsymbol{s}} < 0 \tag{1.5}$$

The sliding mode control law is derived as

$$\boldsymbol{u} = \hat{\boldsymbol{u}} - \boldsymbol{k} \circ \mathrm{sgn}(\boldsymbol{s}) \tag{1.6}$$

where the operator "$\circ$" represents an element-wise vector product. The gains $\boldsymbol{k}$ are designed to satisfy the reaching condition (1.5) given known bounds on any model uncertainties. The discontinuous switching of the control law resulting from the signum function in Equation (1.6) leads to undesirable control chatter. The signum function is therefore typically approximated by a continuous saturation or hyperbolic tangent function as necessary. Sliding mode control can be applied to nonholonomic underactuated systems [35,36]. However, these applications are not as straightforward as the fully actuated applications presented in [32]

1.4 Model Predictive Control

Model Predictive Control (MPC) is an optimization-based control algorithm that attempts to achieve a specified performance objective in an optimal manner by solving a finite horizon, open-loop optimal control problem on-line at each control interval. It has been widely applied to nonlinear systems [37] and industrial applications [38]. The main advantage to this control approach is its ability to handle constraints such as saturation limits on the control actuators, limits on states and of course, nonholonomic constraints. There are a number of other advantages to this approach. For example, common performance objectives, such as minimum tracking error and minimum time, can be easily incorporated into the predictive control framework.

MPC is primarily used in the process control industry where the system dynamics are relatively slow. Because of the relatively high computation cost of optimal control techniques and the on-line nature of MPC, it has not, as of yet, been applied to many mechani-

cal systems which typically have much faster dynamics. In this thesis, an example of MPC applied to USV systems is presented. While the work presented is simulation-based and the controller does not compute quickly enough to be applied to the real-time experimental setup, it is believed that this control methodology could have applications for autopilot systems for large sea-going vessels. Also, as the speed of microprocessors increases, the lessons learned in this work may be applied to real-time MPC for underactuated mechanical systems in the future. A summary of many of the pertinent issues pertaining to MPC and its applications is given in [37].

Chapter 2

Kinematic Control for Mobile Robots

A typical example of an underactuated system with a first-order nonholonomic constraint is the differential-drive wheeled mobile robot. There has been a great deal of research focused on control of mobile robots because of the wide range of applications for which they are useful. Mobile robot technology is widely used in military and industrial environments and the differential drive configuration is one of the most simple and widely used mobile robot platforms. Also, because this configuration is relatively simple both to manufacture, and model mathematically, it is used quite often in education and research environments to develop or demonstrate mobile robot technology. In this work, a simple, inexpensive mobile robot platform is developed for the purpose of demonstrating and evaluating new trajectory planning and obstacle avoidance methods.

Many of the trajectory tracking control methods for mobile robots require the target system trajectories to be continuous and of course feasible for the mechanical system to follow. In addition, it is usually desirable for the trajectory planning algorithm to include the capability to avoid obstacles that may be in the path of the agent, as well as the ability to coordinate multiple agents. In this chapter, a trajectory planning and obstacle avoidance approach that provides coordinated control of multiple autonomous agents is demonstrated using a simple experimental wheeled mobile robot. Local feedback control for the mobile robots is provided by a simple kinematic control law. The trajectory planning strategy is

composed of two parts consisting of a coordinated trajectory planning algorithm [39] and a real-time obstacle avoidance algorithm [40]. The main advantage of this approach is a practical, computationally efficient trajectory planning algorithm with the capability for coordination of multiple autonomous agents and real-time collision avoidance for moving obstacles. To accomplish these tasks, trajectories are defined using sets of ordinary differential equations (ODEs) of two general forms depending on the situation. If the objective is to converge to a stationary or moving target position, the trajectory is defined by a set of exponentially stable equations whose solution converges to the target trajectory. In the cases of coordinated motion and obstacle avoidance, the trajectory is defined by a set of equations whose solution is a stable limit cycle. A related approach to trajectory planning that is applicable to industrial robot manipulators is presented in [41]. Limit cycles of finite size and shape are used as a way of defining complex obstacles to be avoided. Applications of real-time navigation methods for mobile robots using limit cycles have also been addressed in [42, 43]. These approaches are expanded to dynamic limit cycles with more general elliptical geometries in [40].

2.1 Mobile Robot Platform

The experimental mobile robot system used in this work is the differential-drive wheeled robot shown in Figure 2.1 based on the LEGO Mindstorms NXT controller. This experimental platform utilizes a kinematic control law and encoder-based position feedback which provide acceptable tracking performance for the purposes of experimental validation of new trajectory planning techniques. The embedded software for coordinated trajectory planning and real-time obstacle avoidance is developed for this platform in the MATLAB/Simulink programming environment using the ECRobot rapid prototyping development tools [44]. The NXT platform provides the capability for wireless Bluetooth communication between individual robots and a host computer. The relatively low cost and high performance of this device results in a very powerful, flexible, and cost-effective mobile computing platform that can be used to implement advanced autonomous system

Figure 2.1: Experimental mobile robot

applications directly on the target hardware. Further detail on this mobile robot platform can be found in [5].

2.1.1 Kinematic Model

The differential-drive wheeled mobile robot consists of two independently actuated front drive wheels of radius r mounted on a common axis as shown in Figure 2.2. The track width l represents the length of the segment connecting the wheel centers. A passive caster supports the rear of the mobile robot. The 3-DOF planar posture of the robot is described in the inertial reference frame by the vector $\boldsymbol{p} = [x \quad y \quad \theta]^T$ and the motion of the mobile robot is subject to the first-order nonholonomic constraint

$$\dot{y}\cos\theta - \dot{x}\sin\theta = 0 \tag{2.1}$$

assuming that the wheels do not slip. The planar motion is described by the velocity vector $\boldsymbol{q} = [v \quad \omega]^T$ where v is the forward velocity of the robot in the direction orthogonal to the drive-wheel axis and ω is the angular velocity of the robot about the center of the drive-wheel axis. The kinematic equations relating the body fixed velocities $\boldsymbol{q}$ to the inertial

reference frame are written as

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{2.2}$$

The body fixed velocities are related to the angular velocities of the drive wheels by

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{l} & -\frac{r}{l} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{2.3}$$

where ω_L and ω_R are the angular velocities of the left and right wheels respectively [45].

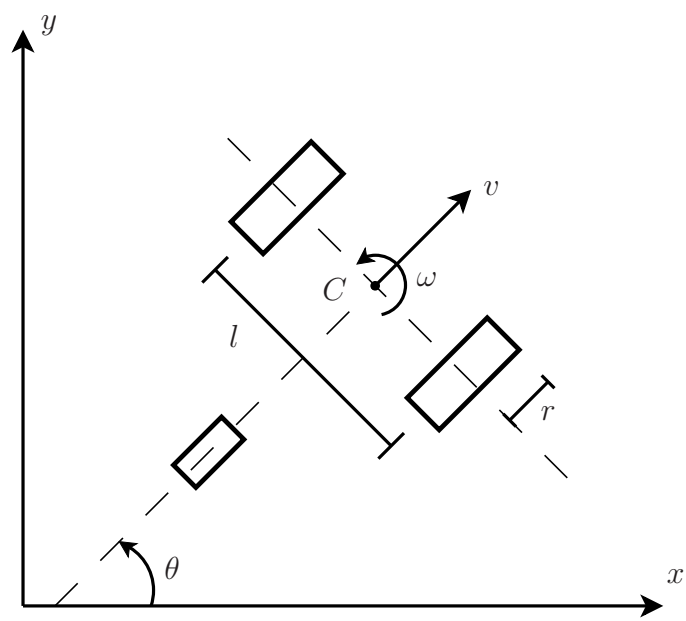


Figure 2.2: Differential drive mobile robot schematic

2.1.2 Position Feedback

Two methods of position feedback are available for the experimental mobile robots. The first uses encoder feedback from each motor to estimate the wheel velocities. The current forward velocity v and angular velocity ω are then computed using Equation (2.3) and the current position in the inertial reference frame is determined using the kinematic model

presented in Equation (2.2). A discrete-time approximation of the current pose of the robot at sample k is

$$\boldsymbol{p}_k = \boldsymbol{p}_{k-1} + \Delta\boldsymbol{p}_k = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} Tv_k \cos\left(\theta_{k-1} + \frac{T\omega_k}{2}\right) \\ Tv_k \sin\left(\theta_{k-1} + \frac{T\omega_k}{2}\right) \\ T\omega_k \end{bmatrix} \tag{2.4}$$

where T is the sample period. This method assumes that there is no wheel slip and that the mobile robot travels on a perfectly flat planar surface.

Since wheel slip and other unmeasured disturbances are typically present, a more accurate method to determine the robot position based on real-time processing of a camera image is also available. A digital black and white camera mounted a fixed position above the mobile robots is used to view the two infrared LEDs installed at the front and rear of the centerline of the robot as shown in Figure 2.1. Each camera frame provides a dead reckoning measurement of position and orientation of each robot that is determined from the pixel location of the LEDs [5]. (See Appendix A for vision system details.)

Position measurements are available at approximately 20Hz using the camera and at least 100Hz using the encoders. The advantage of using encoder feedback is that the position can be determined locally without the need to transmit a camera-based position to each robot from the host computer at the 20 Hz sample rate of the camera. The advantage of using camera feedback is that it is a dead reckoning measurement that is not subject to error from wheel slip, uneven surfaces, collisions, and other disturbances. A combination of encoder and camera feedback using multi-rate estimation schemes can be employed to obtain the benefits of each measurement while minimizing the communication requirements, however, this technique is not considered in this work.

2.1.3 Local Tracking Control

The reference posture and velocity vectors describing the desired trajectory are defined at each sample period where the reference posture $\boldsymbol{p}_k^r = [x_k^r \quad y_k^r \quad \theta_k^r]^T$ and velocity $\boldsymbol{q}_k^r = [v_k^r \quad \omega_k^r]^T$ vectors must adhere to the nonholonomic constraint Equation (2.1). The position

and orientation tracking error vector $\boldsymbol{e}_k^p = [e_k^x \quad e_k^y \quad e_k^\theta]^T$ is expressed as

$$\boldsymbol{e}_k^p = \begin{bmatrix} \cos\theta_k & \sin\theta_k & 0 \\ -\sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 1 \end{bmatrix} (\boldsymbol{p}_k^r - \boldsymbol{p}_k) \tag{2.5}$$

where the current pose of the robot, determined based on either encoder- or camera-based feedback, is given by $\boldsymbol{p}_k$. Various kinematic control laws for tracking and position control of mobile robots have been developed. In this work, the following kinematic tracking control law proposed in [8] is used

$$\begin{bmatrix} v_k^s \\ \omega_k^s \end{bmatrix} = \begin{bmatrix} v_k^r \cos e_k^\theta + k_x e_k^x \\ \omega_k^r + v_k^r (k_y e_k^y + k_\theta \sin e_k^\theta) \end{bmatrix} \tag{2.6}$$

where the tracking errors are defined in Equation (2.5), v_k^s is the corrected forward velocity setpoint, ω_k^s is the corrected angular velocity setpoint and the controller gains k_x, k_y, and k_θ are strictly positive constants. The corresponding wheel velocity setpoints which compensate for any tracking error in the reference trajectory are calculated by inverting Equation (2.3)

$$\begin{bmatrix} \omega_R^s \\ \omega_L^s \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{l}{2r} \\ \frac{1}{r} & -\frac{l}{2r} \end{bmatrix} \begin{bmatrix} v^s \\ \omega^s \end{bmatrix} \tag{2.7}$$

where the rotational speed setpoints ω_R^s and ω_L^s are maintained by individual wheel speed controllers.

2.1.4 Trajectory Tracking Performance

A demonstration of the mobile robot tracking control performance is provided by defining a reference trajectory that approaches and then follows a circular orbit with a diameter of 1 meter represented by the solid line in Figure 2.3. The mobile robot must avoid two obstacles on this path that are contained within the circles indicated by the dashed lines. These two circles represent the obstacle boundaries that should not be crossed. Figure 2.4 presents the desired reference and experimental mobile robot paths with and without tracking control.

As shown in this figure, the tracking error increases as the robot turns without encoder feedback tracking control. The maximum tracking error is reduced over five times using the closed-loop tracking control law presented in Equation (2.6). The corresponding tracking errors are presented in Figure 2.5.

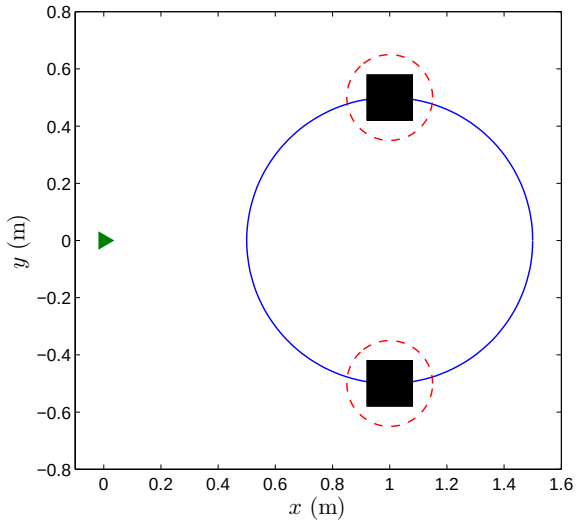


Figure 2.3: Trajectory tracking example

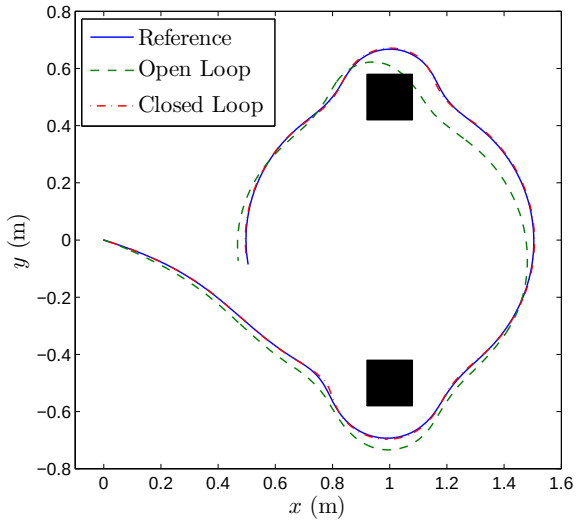


Figure 2.4: Open-loop and closed-loop paths determined using encoder signals

Figure 2.6 compares the desired reference, encoder, and camera image paths for the mobile robot under tracking control using encoder feedback on a flat surface with minimal

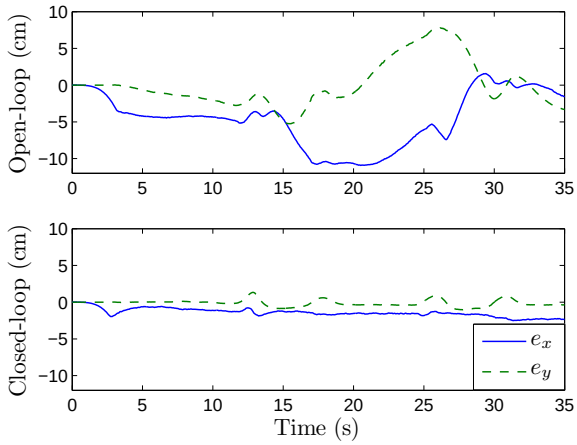


Figure 2.5: Trajectory tracking errors

wheel slip. As shown in this figure, there is very little difference between the position determined from the encoders and that determined from the camera signal. The errors that do exist result primarily from wheel slip that the encoder feedback can not account for. The tracking errors are presented in Figure 2.7. The wheel slip error accumulates as the experiment progresses, however for the experiments that are presented in this work, these errors are acceptable. Figure 2.8 presents a similar scenario except there is significant wheel slip obtained by using plastic wheels, as opposed to rubber, on a slick surface and specifying a trajectory with a smaller turning radius. A large error in the encoder position is now present under these conditions, as shown in Figure 2.9. In the sequel, the conditions in Figure 2.6 are used such that encoder feedback will provide an acceptable position measurement.

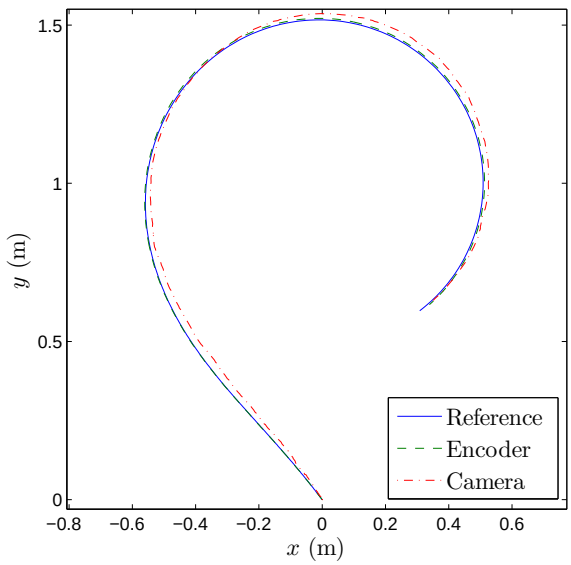


Figure 2.6: Closed-loop trajectory with camera position

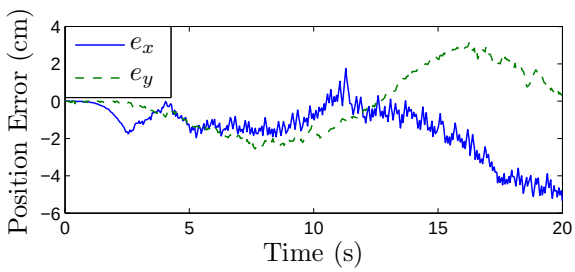


Figure 2.7: Camera-based tracking errors

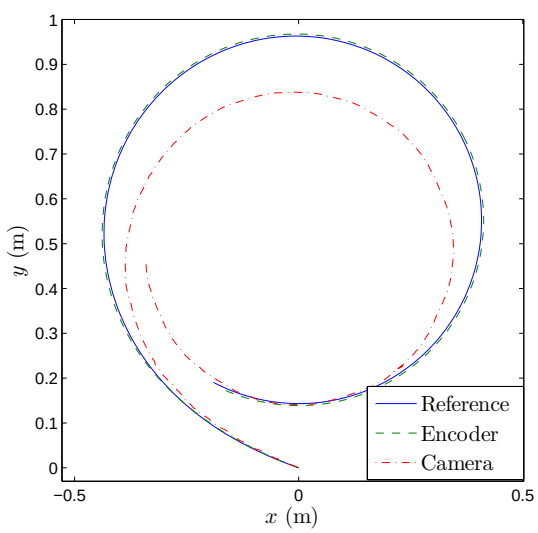


Figure 2.8: Closed-loop trajectory with slip

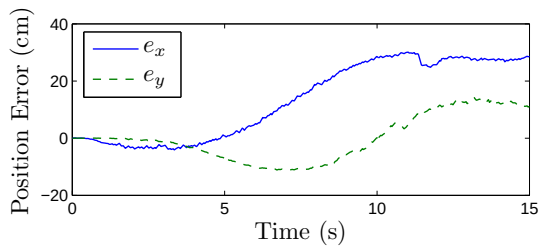


Figure 2.9: Camera-based tracking errors with slip

2.2 Reference Trajectory Determination

The reference trajectory for each mobile robot is defined using a set of two ODEs in terms of the two planar global position variables x and y

$$\dot{x}_i = f_i(x_1, x_2, t), \quad f_i : D_i \to \mathbb{R}^2, \quad i = 1, 2 \tag{2.8}$$

where x_1 and x_2 are the state variables, D_i is an open and connected subset of $\mathbb{R}^2$, and f_i is a locally Lipschitz map from D_i into $\mathbb{R}^2$. The state variables x_1 and x_2 are defined as

$$\begin{aligned} x_1 &= x^r - x^* \\ x_2 &= y^r - y^* \end{aligned} \tag{2.9}$$

where $x^r(t)$ and $y^r(t)$ represent the desired reference trajectory and $x^*(t)$ and $y^*(t)$ define the origin of the reference trajectory. The functions f_1 and f_2 in Equation (2.8) are determined based on the desired reference trajectory dynamics. The mobile robot reference posture $\boldsymbol{p}^r$ and velocity $\boldsymbol{q}^r$ are obtained from the state variables by

$$\boldsymbol{p}^r = \begin{bmatrix} x^r \\ y^r \\ \theta^r \end{bmatrix} = \begin{bmatrix} x_1 + x^* \\ x_2 + y^* \\ \arctan\left(\dfrac{\dot{x}_2 + \dot{y}^*}{\dot{x}_1 + \dot{x}^*}\right) \end{bmatrix} \tag{2.10}$$

$$\boldsymbol{q}^r = \begin{bmatrix} v^r \\ \omega^r \end{bmatrix} = \begin{bmatrix} \sqrt{(\dot{x}^r)^2 + (\dot{y}^r)^2} \\ \frac{1}{v^r}(\ddot{y}^r \cos\theta^r - \ddot{x}^r \sin\theta^r) \end{bmatrix} \tag{2.11}$$

where the expressions for θ^r and ω^r arise from the nonholonomic constraint equation (2.1).

Two general types of reference trajectory are presented in this work. The first is implemented when the mobile robot is required to reach a target position which may be either moving or stationary. This type of reference trajectory is referred to as a *Target Tracking Trajectory*. The second is implemented when the mobile robot is required to smoothly converge to and then follow a closed orbit without crossing into the orbit. This type is referred to as a *Limit Cycle Trajectory* and is useful for obstacle avoidance and coordinated control. Note that in the case of the limit cycle trajectory, the robot is not following a specific target on the closed orbit.

2.2.1 Target Tracking Trajectory

The target tracking trajectory defines a reference trajectory for which the mobile robot path converges exponentially to a stationary or moving target position. The desired reference position is denoted by $x^r(t)$ and $y^r(t)$ and the target position is denoted by $x^t(t)$ and $y^t(t)$ in the inertial reference frame. The state variables introduced in Equation (2.9) are defined as

$$\begin{aligned} x_1 &= x^r - x^t \\ x_2 &= y^r - y^t \end{aligned} \tag{2.12}$$

such that when the ODEs in Equation (2.8) are asymptotically or finite-time stable $\|\boldsymbol{x}\| \to 0$ where $\boldsymbol{x} = [x_1, x_2]^T$ which implies that $x^r \to x^t(t)$ and $y^r \to y^t(t)$.

The exponentially stable target tracking trajectory is defined by the following linear ODE system

$$\begin{aligned} \dot{x}_1 &= -k_1(t)x_1, \quad k_1(t) > 0, \\ \dot{x}_2 &= -k_2(t)x_2, \quad k_2(t) > 0, \quad t \geq t_0 \end{aligned} \tag{2.13}$$

where t_0 is the initial time and the initial conditions are determined based on the initial robot and target positions. The strictly positive parameters $k_1(t)$ and $k_2(t)$ are assumed to be functions of time in order to generate desired reference trajectories that conform to the physical constraints of the mobile robot system. In general, they are selected as monotonically increasing positive time functions that start from a small value and are defined based on actuator limitations and the distance to the target. Fifth order polynomials allow for smooth and monotonic transitions of these parameters to their final values. The following polynomial may be used to compute each parameter $k_i,\ i = 1, 2$ during the transition period $t_0 \leq t \leq t_1$

$$k_i = k_{i5}\Delta t^5 + k_{i4}\Delta t^4 + k_{i3}\Delta t^3 + k_{i2}\Delta t^2 + k_{i1}\Delta t + k_{i0} \tag{2.14}$$

where $\Delta t = t - t_0$. The following boundary conditions are selected to smoothly increase the value of k_i from 1% to 100% of its final value $\bar{k}_i$

$$\begin{aligned} &k_i(t_0) = \bar{k}_i/100, \quad k_i(t_1) = \bar{k}_i, \\ &\dot{k}_i(t_0) = \dot{k}_i(t_1) = \ddot{k}_i(t_0) = \ddot{k}_i(t_1) = 0 \end{aligned} \tag{2.15}$$

where $\dot{k}_i$ and $\ddot{k}_i$ are the 1st and 2nd time derivatives of k_i. The six polynomial coefficients k_{ij}, $j = 0, \ldots, 5$ are derived using the six boundary conditions specified in Equation (2.15). Note that $k_i = \bar{k}_i$ for $t > t_1$ and t_1 is selected manually based on the initial distance of the target from the agent taking into account the maximum velocity of the agent.

2.2.2 Limit Cycle Trajectory

The limit cycle trajectory defines a globally exponentially stable limit cycle to which the desired reference trajectory will converge. In this case, the robot does not converge to a moving target; the objective is only to avoid colliding with an obstacle or to produce coordinated motion for multiple robots. The state variables introduced in Equation (2.9)

for the limit cycle trajectory are defined as

$$\begin{aligned} x_1 &= x^r - x^o \\ x_2 &= y^r - y^o \end{aligned} \tag{2.16}$$

where $x^o(t)$ and $y^o(t)$ denote the position of the origin of the limit cycle in the inertial reference frame. The origin of the limit cycle is assumed to be a function of time to allow for dynamic obstacles. The limit cycle trajectory has the following form

$$\begin{aligned} \dot{x}_1 &= h_1(x_1, x_2, t) - k_1(t) x_1 l(x_1, x_2, t), \quad k_1(t) > 0 \\ \dot{x}_2 &= h_2(x_1, x_2, t) - k_2(t) x_2 l(x_1, x_2, t), \quad k_2(t) > 0 \end{aligned} \tag{2.17}$$

where $l(x_1, x_2, t)$ defines the limit cycle geometry which may be an explicit function of time to account for rotation of the obstacle in the x-y plane. The functions $h_1(x_1, x_2, t)$ and $h_2(x_1, x_2, t)$ represent the planar particle motion kinematics on the limit cycle; *i.e.*, when $l(x_1, x_2, t) = 0$. The solution of Equation (2.17) guarantees that any trajectory with an initial position outside of the limit cycle will converge to the limit cycle without crossing it. The positive parameters $k_1(t)$ and $k_2(t)$ are again assumed to be functions of time in order to generate realizable trajectories and are derived using Equations (2.14) and (2.15).

2.2.3 Elliptical Limit Cycles

In this work an elliptical limit cycle geometry is used. This is a very general shape which can be used for obstacle avoidance, coordinated control, aerial coverage or any number of other trajectory planning applications. The limit cycle $l(x_1, x_2, t)$ is defined by the general equation of an ellipse with semi major and semi minor axes, a and b, respectively, and origin at $(x^o(t), y^o(t))$

$$l(x_1, x_2, t) = \left[\frac{\cos\phi x_1 + \sin\phi x_2}{a}\right]^2 + \left[\frac{-\sin\phi x_1 + \cos\phi x_2}{b}\right]^2 - 1 \tag{2.18}$$

where $x_1(t)$ and $x_2(t)$ are defined in Equation (2.16) and $\phi(t)$ is the angle representing the orientation of the semi major axis relative to the global horizontal (x) axis. This angle can be time dependent if the desired limit cycle is rotating.

The functions $h_1(x_1, x_2, t)$ and $h_2(x_1, x_2, t)$ are defined based on the motion of a particle around an ellipse

$$\begin{aligned} x_1 &= a\cos\phi\cos\Omega t - b\sin\phi\sin\Omega t \\ x_2 &= a\sin\phi\cos\Omega t + b\cos\phi\sin\Omega t \end{aligned} \tag{2.19}$$

where $\Omega(t)$ is the average angular velocity of the particle on the limit cycle. The time derivative of Equation (2.19) is derived as

$$\begin{aligned} \dot{x}_1 &= -(\Omega + \dot{\Omega}t)(a\cos\phi\sin\Omega t + b\sin\phi\cos\Omega t) - x_2\dot{\phi} \\ \dot{x}_2 &= (\Omega + \dot{\Omega}t)(-a\sin\phi\sin\Omega t + b\cos\phi\cos\Omega t) + x_1\dot{\phi}. \end{aligned} \tag{2.20}$$

The functions $h_1(x_1, x_2, t)$ and $h_2(x_1, x_2, t)$ are derived by eliminating $\cos\Omega t$ and $\sin\Omega t$ from Equations (2.19) and (2.20) and are given by

$$\begin{aligned} h_1(x_1, x_2, t) &= \dot{x}_1 = -x_2\dot{\phi} + \frac{\Omega + \dot{\Omega}t}{ab}(h_{e11}x_1 - h_{e12}x_2) \\ h_2(x_1, x_2, t) &= \dot{x}_2 = +x_1\dot{\phi} + \frac{\Omega + \dot{\Omega}t}{ab}(h_{e21}x_1 - h_{e11}x_2) \end{aligned} \tag{2.21}$$

where

$$\begin{aligned} h_{e11} &= (a^2 - b^2)\sin\phi\cos\phi \\ h_{e12} &= a^2\cos^2\phi + b^2\sin^2\phi \\ h_{e21} &= b^2\cos^2\phi + a^2\sin^2\phi. \end{aligned} \tag{2.22}$$

The average angular velocity $\Omega(t)$ is assumed to be monotonically increasing in magnitude such that $\Omega + \dot{\Omega}t \neq 0$. Again, a fifth order polynomial is used to transition $\Omega(t)$ from 1%

to 100% of its final constant value $\bar{\Omega}$ in the selected transition period of $t_0 \leq t \leq t_1$

$$\Omega = \Omega_5 \Delta t^5 + \Omega_4 \Delta t^4 + \Omega_3 \Delta t^3 + \Omega_2 \Delta t^2 + \Omega_1 \Delta t + \Omega_0 \tag{2.23}$$

where $\Delta t = t - t_0$. The following boundary conditions

$$\begin{aligned} &\Omega(t_0) = \bar{\Omega}/100,\ \Omega(t_1) = \bar{\Omega}, \\ &\dot{\Omega}(t_0) = \dot{\Omega}(t_1) = \ddot{\Omega}(t_0) = \ddot{\Omega}(t_1) = 0 \end{aligned} \tag{2.24}$$

are used to derive the six polynomial coefficients $\Omega_i,\ i = 0, \ldots, 5$. Note that $\Omega_i = \bar{\Omega}_i$ for $t > t_1$ and the same transition period t_1 is used for the k_1, k_2 and Ω transitions.

2.3 Coordinated Control

An application of the limit cycle trajectory is coordinated control. The trajectories defined in Equation (2.17), which converge to stable limit cycles, are ideal for coordinated control because all paths to the limit cycle are independent and non-crossing due to the uniqueness of the solution to the ODEs.

The following experimental example considers the coordinated motion of three identical mobile robots that join an elliptical reference trajectory at spaced intervals and maintain the elliptical motion. The mobile robots are initially located at the (x, y) positions (-1,0), (0.5,0.5), and (0.5,-0.5) expressed in meters. Each robot joins the elliptical reference trajectory with semi-major and semi-minor axes of $a = 0.5$ m and $b = 0.3$ m respectively oriented with $\phi = -30$ degrees, its center at (0,0) and a desired angular velocity of $\Omega = 0.45$ rad/sec (about 4.3 rev/min). The final values of the trajectory parameters are selected as $\bar{k}_1 = \bar{k}_2 = 0.8$.

Figure 2.10 presents the experimental trajectories of the three mobile robots under tracking control using encoder feedback. The trajectories shown remain within 2 cm of the desired reference trajectory for the duration of the experiment. The markers on each trajectory represent the mobile robot positions at 2 second intervals for the first 6 seconds

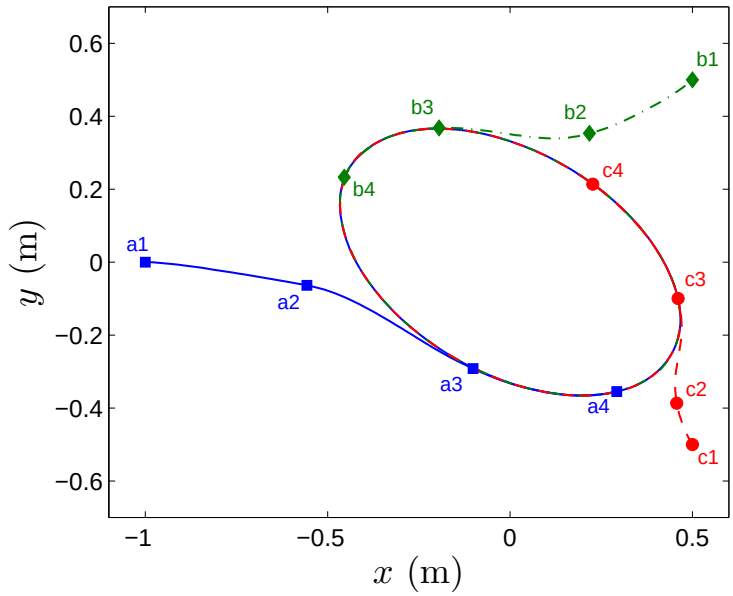


Figure 2.10: Experimental coordinated robot trajectories

of the trajectory. The velocities during the initial portion of the trajectory as the mobile robot joins the limit cycle depend on the initial distance from the closed orbit. Robots that start closer to the closed orbit proceed at a slower velocity while those that start further away must approach at a higher velocity such that all three robots converge to the limit cycle in the proper orientation. Note that these velocities are automatically generated by the ODEs in Equation (2.17).

The trajectories presented in Figure 2.10 are calculated based on the encoder feedback used by the control law. However, because of wheel slip, changes in elevation of the ground surface and/or other external disturbances, the trajectory that the robot actually follows may be different from that calculated based on encoder feedback. Figure 2.11 presents the reference and experimental trajectories of the robot which started at $(1.5, 0.5)$, including the camera image trajectory. This figure shows that under some circumstances, the actual trajectory of the mobile robot using encoder-based feedback may deviate from the desired reference trajectory. It may be necessary to include the camera image feedback in the tracking control law if this deviation becomes significant.

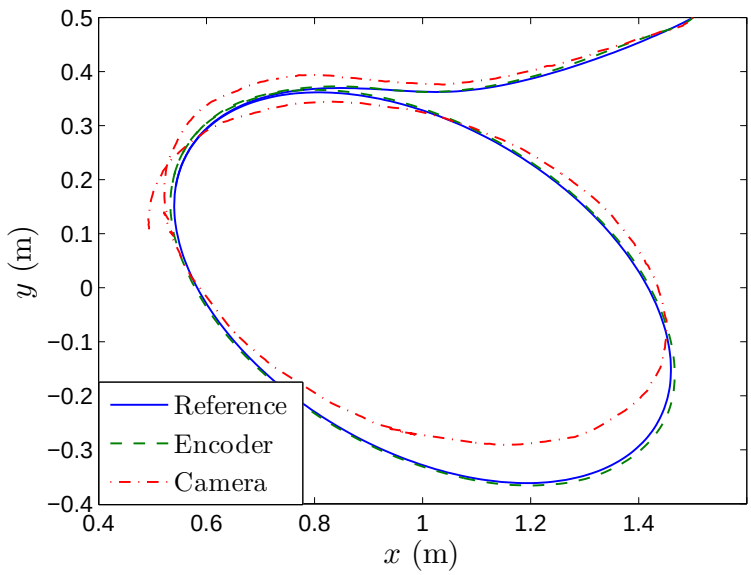


Figure 2.11: Elliptical trajectory with camera position data

2.4 Obstacle Avoidance

An application of both the limit cycle and target tracking trajectories is the problem of obstacle avoidance. In this application, a mobile robot is commanded to converge on a target position in the presence of several stationary or moving obstacles. The obstacles can be approximated by elliptical limit cycles as developed in Section 2.2.3. At each sample time, the nearest obstacles on a straight line path from the mobile robot to its target is detected. Real-time obstacle avoidance is carried out by transitioning from the target tracking trajectory to a trajectory that approaches the nearest obstacle in the original path. As soon as the mobile robot is around the obstacle, the trajectory is switched either to another limit cycle approximating and enclosing the next obstacle or to the original target tracking trajectory.

In the following experimental example the mobile robot starts at the origin and its target position is $(1.5, 0)$. Obstacles are located at $(0.5, 0)$ and $(1, -0.1)$ and are approximated by elliptical limit cycles with semi-major and semi-minor axes $a = b = r = 0.15$. Figure 2.12 shows the desired reference, encoder, and camera image paths for the robot under tracking control with encoder feedback. The numbered markers along the desired reference

trajectory represent the desired positions at four different times. Marker 1 represents the initial position of the mobile robot and marker 4 represents the target position. At the first instant of the experiment, the obstacle avoidance algorithm recognizes that obstacle 1 is the closest obstacle lying in the straight line path to the target and a limit cycle trajectory surrounding the obstacle is generated. The mobile robot approaches obstacle 1 following the limit cycle trajectory until the next obstacle is detected in its path. A new limit cycle trajectory is generated beginning at at marker 2. This trajectory is followed until marker 3 at which point the obstacle no longer lies in the direct path to the target. Finally, the target tracking trajectory defined in Section 2.2.1 is used to converge on the final target position.

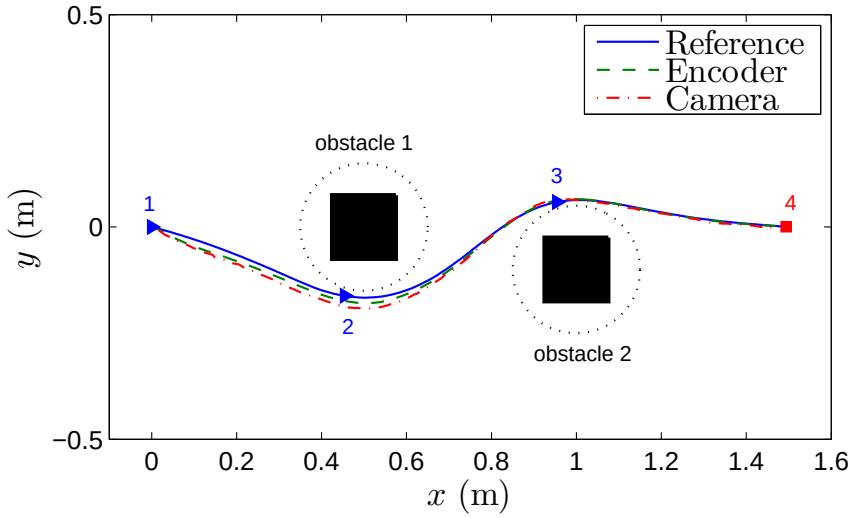


Figure 2.12: Experimental obstacle avoidance trajectories

2.5 Conclusions

The experimental mobile robot system is used to demonstrate a method combining coordinated control and real-time obstacle avoidance for autonomous systems. The method generates reference trajectories as a function of the current system trajectory and the desired target trajectory. The method can be used for coordinated control of several robots guaranteeing unique collision-free paths and coordinated operation. It can also be used for real-time obstacle avoidance and target tracking applications. A kinematic control law is applied providing good tracking performance and several methods of position feedback are available depending on the situation and the desired performance.

Chapter 3

Sliding Mode Control for Unmanned Surface Vessels

Another example of a nonholonomic system is the unmanned surface vessel system. Position control of USV systems has received increased attention in the last decade with most of the research focusing on feedback linearization, backstepping, Lagrangian, and sliding mode control methods. The control problems considered using these methods can generally be divided into setpoint position [14–19] and trajectory tracking [20–28] control. In this chapter, sliding mode setpoint and trajectory tracking controllers for USV systems are presented.

3.1 USV System Model

The 3-DOF planar model of a surface vessel shown in Figure 3.1 is considered. This model includes surge, sway, and yaw motion with two propeller force inputs f_1 and f_2. The geometrical relationship between the inertial reference frame and the vessel-based body-

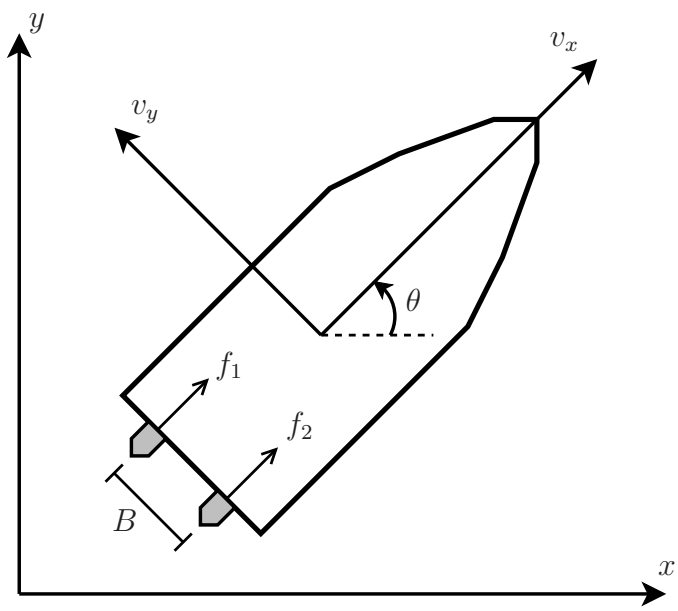


Figure 3.1: Planar USV model schematic

fixed frame is defined in terms of velocities as

$$\dot{x} = v_x \cos\theta - v_y \sin\theta \tag{3.1}$$
$$\dot{y} = v_x \sin\theta + v_y \cos\theta \tag{3.2}$$
$$\dot{\theta} = \omega \tag{3.3}$$

where x and y denote the position of the center of mass, θ is the orientation angle of the vessel in the inertial reference frame, v_x and v_y are the surge and sway velocities, respectively, and ω is the angular velocity of the vessel.

In the body-fixed frame, the nonlinear equations of motion for a simplified model of the dynamics of a surface vessel, where motion in heave, roll, and pitch are neglected, are given by

$$m_{11}\dot{v}_x - m_{22}v_y\omega + g_x(v_x) = f \tag{3.4}$$
$$m_{22}\dot{v}_y + m_{11}v_x\omega + g_y(v_y) = 0 \tag{3.5}$$
$$m_{33}\dot{\omega} + m_d v_x v_y + g_z(\omega) = T \tag{3.6}$$

where m_{ii} are the mass and inertial parameters, $m_d = m_{22} - m_{11} > 0$ and $g_{\{x,y,z\}}$ are the hydrodynamic drag forces on the vessel in each axis of motion. The m_{ii} parameters include added mass contributions that represent hydraulic pressure forces and torque due to forced harmonic motion of the vessel which are proportional to acceleration. This dynamic model is widely used in the literature and a more detailed discussion can be found in [46]. In this work, only forward vessel motion, $v_x > 0$, is considered because the forward and reverse motion dynamics are quite different. The surge control force f and the yaw control moment T are given in terms of the two propeller forces as

$$\begin{aligned} f &= f_1 + f_2 \\ T &= B(f_2 - f_1)/2 \end{aligned} \tag{3.7}$$

where B is the lateral distance between the propellers. The unactuated equation of motion (3.5) is interpreted as a second-order nonholonomic constraint.

3.2 Experimental USV System

Experimental validation of the proposed controllers is implemented using a small-scale model USV system. The experimental setup includes a boat, shown in Figure 3.2, actuated by two propellers driven by DC motors connected to a LEGO Mindstorms NXT controller with Bluetooth communication. Two light-emitting diodes (LEDs) are mounted near the front and rear of the vessel centerline. Experiments are conducted in an indoor pool with a digital color video camera mounted overhead. The motion of the two LEDs are captured by the camera and processed to determine the location and orientation of the vessel (see Appendix A for details relating to camera calibration). Since only the absolute position and orientation of the vessel are measured, the absolute velocities are numerically estimated and surge and sway velocities are calculated using the kinematic relations between inertial and body-fixed reference frames presented in Equations (3.1) – (3.3). The motor voltages are estimated through interpolation of the calculated control forces and communicated to the NXT controller via Bluetooth communication. Table 3.2 presents the physical dimensions

of the model USV where L, W, and D are the effective submerged length, width, and draft of the vessel respectively.

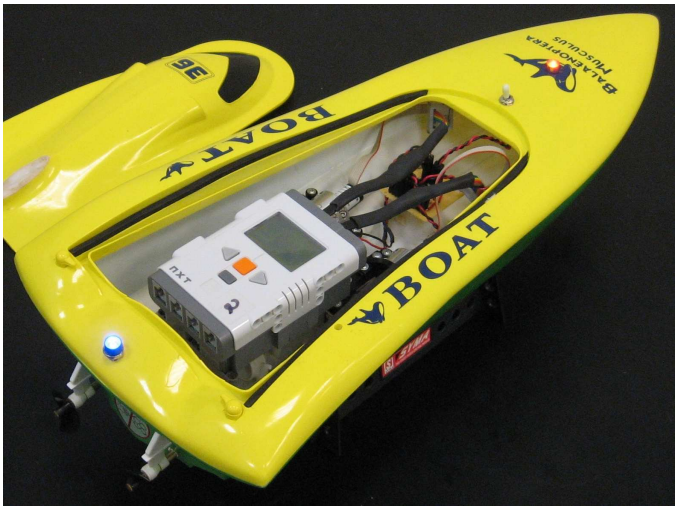


Figure 3.2: Experimental USV

Table 3.1: Model USV dimensions

Dimension	*Value (m)*
B	0.07
L	0.40
W	0.14
D	0.02

The drag forces $g_{\{x,y,z\}}$ in Equations (3.4) – (3.6) are modeled using either a linear or power law drag force model depending on the application. The linear drag force model is the more common form used in USV dynamic modeling [46–48]. This model results in stable linear trajectories, and it is appropriate for the USV system at low velocities. The nonlinear drag force is more representative at relatively high vessel velocities and results in a much more difficult system to control. In the nonlinear case, a power law velocity relationship is used to describe the hydraulic drag forces in each axis of motion with only

forward surge velocities.

$$\begin{aligned} g_x(v_x) &= d_1 v_x^{\alpha_1}, \quad v_x \geq 0 \\ g_y(v_y) &= \operatorname{sgn}(v_y)\, d_2\, |v_y|^{\alpha_2} \\ g_z(\omega) &= \operatorname{sgn}(\omega)\, d_3\, |\omega|^{\alpha_3} \end{aligned} \tag{3.8}$$

Although not typically used in dynamic surface vessel maneuvering and positioning models, this form of hydraulic drag model is consistent with the Blasius solution for viscous drag on a flat plate when $\alpha = 1.5$ [49]. Because the ratio of W to L is rather large, D is rather small, and the hull does not plane or generate a significant wake at the operating velocities of the vessel, this drag model form is reasonable for the model USV hull design. It is also consistent with the experimental parameter identification tests carried out on this vessel in [50].

The USV model parameters in Equations (3.4) – (3.6) and (3.8) are presented in Table 3.2. These mass and damping terms were determined through experimental system identification tests carried out on this vessel [50] and are assumed to remain constant.

Table 3.2: Estimated USV system model parameters

Parameter	*Value*
m_{11}	1.96 ± 0.019 kg
d_1	2.44 ± 0.023
α_1	1.51 ± 0.0075
m_{22}	2.40 ± 0.12 kg
d_2	13.0 ± 0.30
α_2	1.75 ± 0.013
m_{33}	0.0430 ± 0.0068 kg
d_3	0.0564 ± 0.00085
α_3	1.59 ± 0.0285

3.3 Setpoint Control

In this section, the setpoint control problem of autonomous surface vessels is addressed and the proposed control law is implemented experimentally. Partial feedback linearization is used to convert the system dynamics presented in the preceding sections into a nonlinear control model that is used to determine the controllability properties of the system. In this case, only the linear drag force model is considered where $\alpha_{1,2,3} = 1$ in Equation (3.8). The partial feedback linearization becomes much more difficult if the power law drag model is used. A setpoint sliding mode control law is developed using four sliding surfaces for calculation of the two propeller forces. The yaw moment control is developed by defining a sliding surface for orientation stabilization. The surge control is derived by defining three additional surfaces. The trajectories are proved to converge to the first surface at a finite time and the remaining three surfaces simultaneously at a different finite time. Finally, the sliding phase is shown to be asymptotically stable through the introduction of a Lyapunov function.

3.3.1 Partial Feedback Linearization

The two inputs of the system are redefined to be the angular and surge accelerations, $\dot{\omega} = \ddot{\theta}$ and $\dot{v}_x$, respectively

$$u_1 = \big(T - (m_{22} - m_{11})v_x v_y - d_{33}\omega\big)/m_{33} \tag{3.9}$$

$$u_2 = \big(f + m_{22}v_y\omega - d_{11}v_x\big)/m_{11} \tag{3.10}$$

where u_1 is the angular acceleration and u_2 is surge acceleration. The first three states of the model are defined as the vessel orientation angle and the body-fixed coordinates of the

vessel center of mass

$$x_1 = \theta - \theta^s \tag{3.11}$$
$$x_2 = (x - x^s)\cos\theta + (y - y^s)\sin\theta \tag{3.12}$$
$$x_3 = -(x - x^s)\sin\theta + (y - y^s)\cos\theta \tag{3.13}$$

where (x^s, y^s, θ^s) represent the desired position and orientation setpoint of the vessel. Three additional states are defined as the vessel's sway, angular, and surge velocities.

$$x_4 = v_y \tag{3.14}$$
$$x_5 = \dot{\theta} \equiv \omega \tag{3.15}$$
$$x_6 = v_x \tag{3.16}$$

Hence, the state equations can be derived as

$$\begin{aligned}
\dot{x}_1 &= x_5 \\
\dot{x}_2 &= x_6 + x_3 x_5 \\
\dot{x}_3 &= x_4 - x_2 x_5 \\
\dot{x}_4 &= -d_r x_4 - m_r x_5 x_6 \\
\dot{x}_5 &= u_1 \\
\dot{x}_6 &= u_2
\end{aligned} \tag{3.17}$$

where $d_r = \frac{d_{22}}{m_{22}}$ and $m_r = \frac{m_{11}}{m_{22}}$.

3.3.2 Controllability Properties

Defining the state vector as $\boldsymbol{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T$, Equation (3.17) can be written in vector form as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}_1 u_1 + \boldsymbol{g}_2 u_2 \tag{3.18}$$

where $\boldsymbol{g}_1 = [0,0,0,0,1,0]^T$, $\boldsymbol{g}_2 = [0,0,0,0,0,1]^T$,

$$\boldsymbol{f} = \begin{bmatrix} x_5 \\ x_6 + x_3 x_5 \\ x_4 - x_2 x_5 \\ -d_r x_4 - m_r x_5 x_6 \\ 0 \\ 0 \end{bmatrix}$$

and the system equilibrium points are defined by the manifold $\{\boldsymbol{x}_e \in \mathbb{R}^6 : x_4 = x_5 = x_6 = 0\}$.

The underactuated system represented in Equation (3.18) is not asymptotically stabilizable to an equilibrium point using time-invariant continuous feedback since it does not satisfy Brockett's necessary condition [33]. However, it can be shown that such systems are strongly accessible if certain conditions are met [2]. For the our system, the vector field $\boldsymbol{v}_f = \boldsymbol{g}_1, \boldsymbol{g}_2, [\boldsymbol{f}, \boldsymbol{g}_1], [\boldsymbol{f}, \boldsymbol{g}_2], [\boldsymbol{g}_2, [\boldsymbol{f}, \boldsymbol{g}_1]], [[\boldsymbol{f}, \boldsymbol{g}_2], [\boldsymbol{f}, \boldsymbol{g}_1]]$ may be calculated as

$$\boldsymbol{v}_f = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -x_3 & -1 & 0 & 0 \\ 0 & 0 & x_2 & 0 & 0 & -1 \\ 0 & 0 & m_r x_6 & m_r x_5 & m_r & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.19}$$

Since the determinant of the matrix $\boldsymbol{v}_f$ is m_r, the vector field spans the six-dimensional space and the strong accessibility Lie algebra rank condition is satisfied at all points. The system is also small-time locally controllable as presented in [14]. Hence, a discontinuous sliding mode control law for the underactuated vessel may be developed.

3.3.3 Sliding Mode Control Law

Since the system cannot be stabilized with a time-invariant continuous feedback, many time-variant control methods have been introduced [14, 16, 17, 19]. Discontinuous sliding

mode control laws have also been developed for similar class underactuated systems and applied to an underwater vehicle [51]. In this section, a discontinuous sliding mode control law specifically designed for unmanned surface vessels is presented. Finite time control concepts introduced in [52, 53] are used to develop a control law that guarantees system trajectories reach all sliding surfaces in finite time. Hence, the trajectories are asymptotically stabilized due to the linear stable nature of the surfaces.

The first asymptotically stable sliding surface is defined as

$$s_1 = \lambda x_1 + x_5, \quad \lambda > 0 \tag{3.20}$$

in order to stabilize the vessel orientation. The control law that pushes all trajectories to this surface can be calculated using the finite time approach

$$u_1 = -\eta_1 \operatorname{sgn}(s_1)|s_1|^{\gamma_1} - \lambda x_5, \quad \eta_1 > 0, \quad 0 \leq \gamma_1 < 1 \tag{3.21}$$

where $\gamma_1 = 0$ corresponds to the standard sliding mode control law as defined in [34]. The controller in Equation (3.21) guarantees that the trajectory reaches surface s_1 in some finite time τ_1 such that $s_1 = 0,$ for $t \geq \tau_1$ and after which slides to the origin since

$$\dot{s}_1 = \lambda \dot{x}_1 + u_1 = -\eta_1 \operatorname{sgn}(s_1)|s_1|^{\gamma_1}. \tag{3.22}$$

The following two surfaces are defined to stabilize the global position of the vessel

$$\begin{aligned} s_2 &= \beta x_2 + x_6 \\ s_3 &= d_r x_3 + x_4 \end{aligned} \tag{3.23}$$

where $\beta = \frac{d_r}{m_r} = \frac{d_{22}}{m_{11}}$. The time derivatives of s_2 and s_3 are determined as

$$\begin{aligned} \dot{s}_2 &= b\dot{x}_2 + \dot{x}_6 = \beta(x_6 + x_3 x_5) + u_2 \\ \dot{s}_3 &= d_r \dot{x}_3 + \dot{x}_4 = -m_r x_5 s_2. \end{aligned} \tag{3.24}$$

Since $\dot{s}_3$ is proportional to the product of x_5 and s_2 and x_5 approaches zero asymptotically, then deriving a control law that guarantees $\dot{s}_3 = 0$ in finite time implies that $s_2 = 0$. A new surface is defined as $s_4 = \dot{s}_3 = -m_r x_5 s_2$ and its time derivative is calculated as follows:

$$\dot{s}_4 = -m_r x_5 \beta(x_6 + x_3 x_5) - m_r x_5 u_2 - m_r s_2 u_1. \tag{3.25}$$

The control law may be derived to stabilize the dynamics of s_3 and s_4 in finite time [53]

$$\begin{aligned} u_2 &= \frac{-u_2' - m_r s_2 u_1}{m_r x_5} - \beta(x_6 + x_3 x_5) \\ u_2' &= -\eta_2 \operatorname{sgn}(\psi)|\psi|^{\frac{\gamma_2}{2-\gamma_2}} - \eta_3 \operatorname{sgn}(s_4)|s_4|^{\gamma_2} \\ \psi &= \eta_3 s_3 + \frac{1}{2-\gamma_2} \operatorname{sgn}(s_4)|s_4|^{2-\gamma_2} \end{aligned} \tag{3.26}$$

where $\eta_2, \eta_3 > 0$ and $0 < \gamma_2 < 1$.

The USV model inputs are calculated by solving for the yaw moment and surge force in Equations (3.9) and (3.10) respectively

$$T = m_{33} u_1 + (m_{22} - m_{11}) v_x v_y + d_{33}\omega \tag{3.27}$$

$$f = m_{11} u_2 - m_{22} v_y \omega + d_{11} v_x \tag{3.28}$$

where the velocities v_x, v_y and ω are estimated using camera feedback for the experimental system. Finally, the individual propeller inputs are calculated by inverting Equation (3.7).

$$\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1/2 & -1/B \\ 1/2 & 1/B \end{bmatrix} \begin{bmatrix} f \\ T \end{bmatrix} \tag{3.29}$$

3.3.4 Stability

At time $t = \tau_1$ the trajectory reaches the first surface, $s_1 = 0$, after which x_1 and x_5 approach the origin asymptotically. At time $t = \tau_2$ the trajectory reaches the third and fourth surfaces, $s_3 = s_4 = 0$. This implies that it has simultaneously reached the second

surface, $s_2 = 0$, since s_2 is proportional to the product of x_5 and s_2 and $x_5 \to 0$ only asymptotically. Hence, the trajectory reaches all surfaces in finite time after which the sliding phase starts. During the sliding phase, the dynamics can be re-derived using $s_1 = 0 \Rightarrow x_5 = -\lambda x_1$, $s_2 = 0 \Rightarrow x_6 = -\beta x_2$, and $s_3 = 0 \Rightarrow x_4 = -d_r x_3$

$$\begin{aligned}
\dot{x}_1 &= x_5 \\
\dot{x}_2 &= -bx_2 - \lambda x_1 x_3 \\
\dot{x}_3 &= -d_r x_3 + \lambda x_1 x_2 \\
\dot{x}_4 &= -d_r x_4 - d_r \lambda x_1 x_2 \\
\dot{x}_5 &= -\lambda x_5 \\
\dot{x}_6 &= -bx_6 + b\lambda x_3 x_1.
\end{aligned} \tag{3.30}$$

The sliding phase is asymptotically stable since the positive Lyapunov candidate function

$$V = \frac{1}{2}\sum_{i=1}^{5} x_i^2 + \frac{m_r^2}{2} x_6^2 > 0 \tag{3.31}$$

has a negative time derivative

$$\dot{V} = -\lambda x_1^2 - \beta x_2^2 - d_r x_3^2 - d_r x_4^2 - \lambda x_5^2 - \beta m_r^2 x_1^2 < 0. \tag{3.32}$$

A limitation of the control law presented above is that it will not be applicable if the initial and desired orientations are the same since in such cases $x_1 = x_5 = 0$ and the control law presented in Equation (3.26) does not imply $s_2 = 0$. In such cases, it is necessary to either perturb the initial conditions or use a hybrid control law such that the first controller will perturb the initial orientation.

3.3.5 Results

The control parameters are selected as $\gamma_1 = \gamma_2 = \frac{1}{2}$, $\lambda = 1$, $\eta_1 = 1$, $\eta_2 = 2$, and $\eta_3 = 1.1$. In addition, it is assumed that the boat propellers are only able to produce maximum surge force of $f_{max} = 1.18$N and yaw moment of $T_{max} = .06$Nm. These saturation limits are not

accounted for in the simulations but are present in the experiments.

The controller performance is evaluated in simulation and experimentally through two setpoint problems. In both cases the USV begins at rest at an initial position and orientation determined by the camera. For the first example, the initial position of the boat is determined to be $(0.47\text{ m}, 0.28\text{ m}, 0°)$ and the desired setpoint is $(2\text{ m}, 2\text{ m}, 90°)$. Figure 3.3 shows that the experimental and simulated USV paths are similar though with some difference. The difference becomes more pronounced when the x, y and θ time histories are plotted as in Figure 3.4. It is clear that x and θ stabilize fairly accurately but y position does not fully stabilize in the experiment due to a small forward velocity at the desired setpoint. The reason for this problem is the saturation limits which cut off the surge force significantly, as shown in Figure 3.5. The experimental error may also be partially due to camera calibration and model errors [28]. Figure 3.6 shows the simulated time history of the four surfaces where the surface reach times are $\tau_1 = 2.2$ s and $\tau_2 = 4$ s.

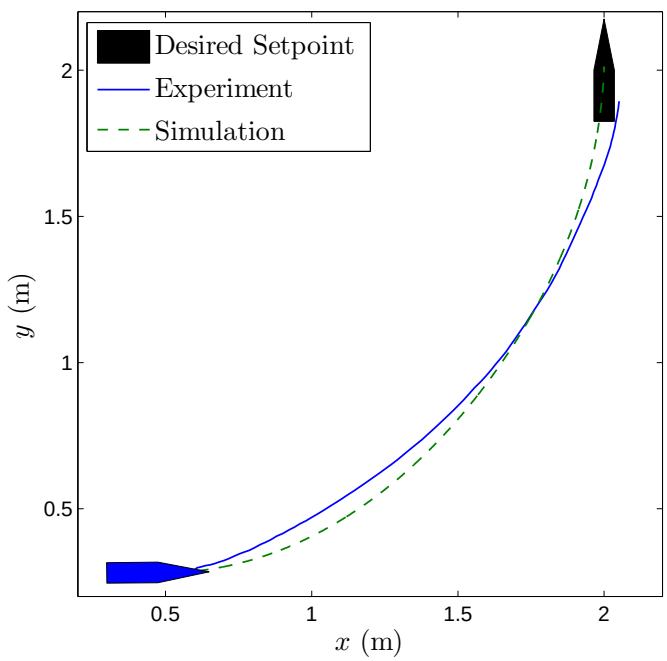


Figure 3.3: Experimental and simulated paths: Case 1

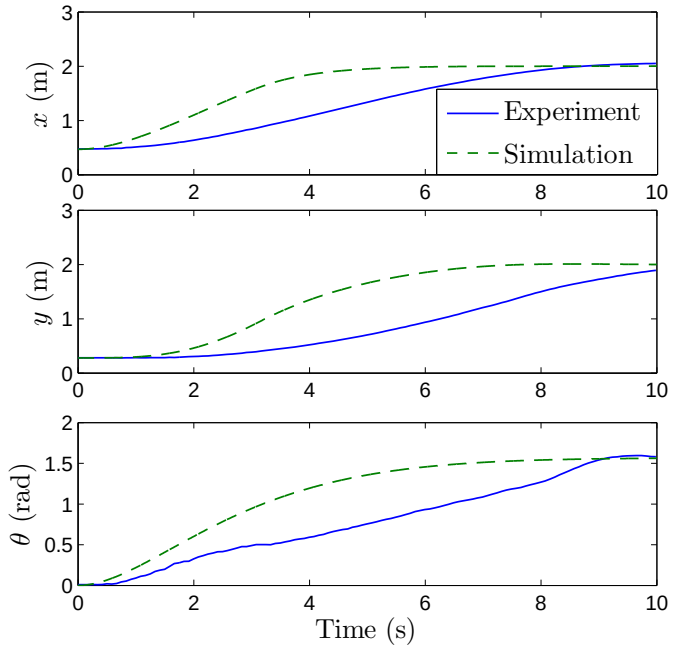


Figure 3.4: Experimental and simulated global trajectories: Case 1

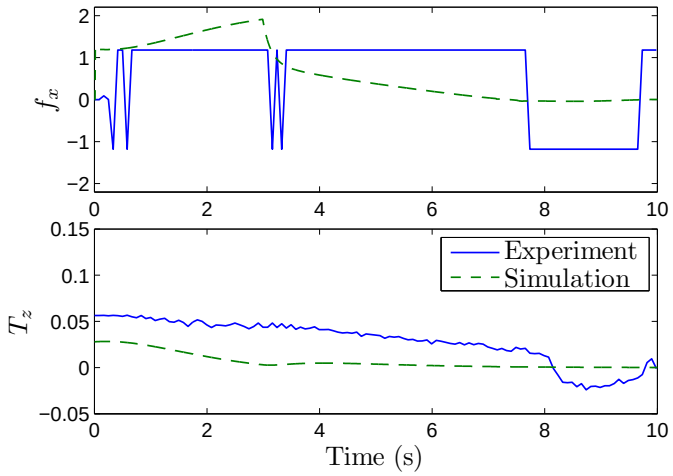


Figure 3.5: Experimental and simulated surge force and yaw moment: Case 1

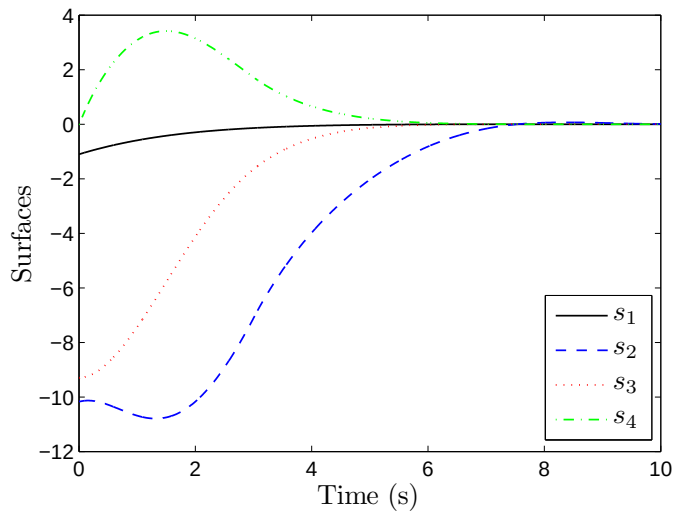


Figure 3.6: Simulated time history of the four surfaces: Case 1

For the second example, the initial position of the boat is determined as $(1.78\text{ m}, 0.28\text{ m})$ with an orientation angle $\theta = 0°$ and the desired setpoint is given as $(1\text{ m}, 2.5\text{ m}, 175°)$. In this experiment, the surge force saturation limit is increased to $f_{max} = 1.8$N. The true saturation limits of the experimental system are unclear at this point and increasing the limit for this experiment produced acceptable results. Figure 3.7 shows that the experimental boat is apparently stabilized at the setpoint having taken a similar path to that shown in simulation. The time history of x, y and θ are presented in Figure 3.8 where it is clear that a small forward velocity again persists at the setpoint causing the x position not to fully stabilize in this case. The control input is presented in Figure 3.9. Figure 3.10 shows the simulated time history of the four surfaces where in this case the surface reach times are $\tau_1 = 10$ s and $\tau_2 = 6$ s.

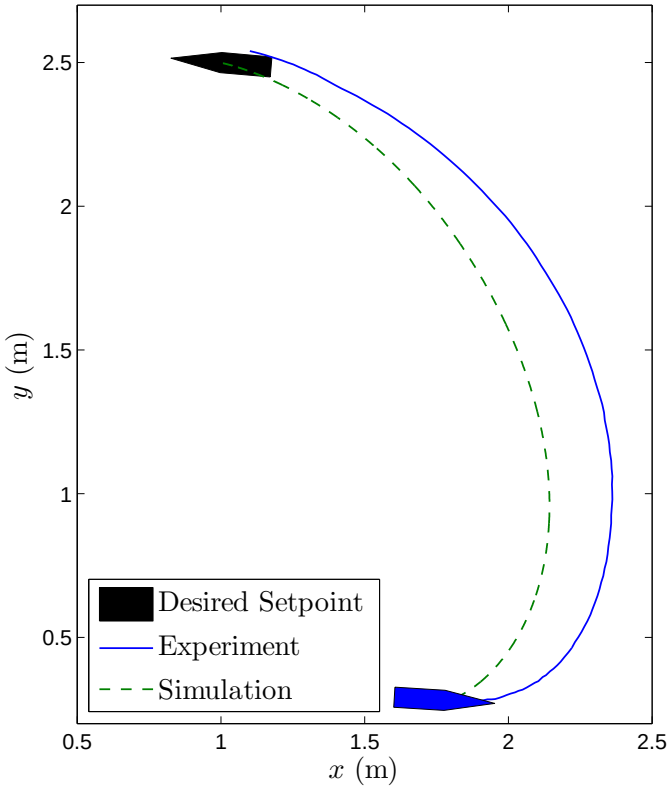


Figure 3.7: Experimental and simulated paths: Case 2

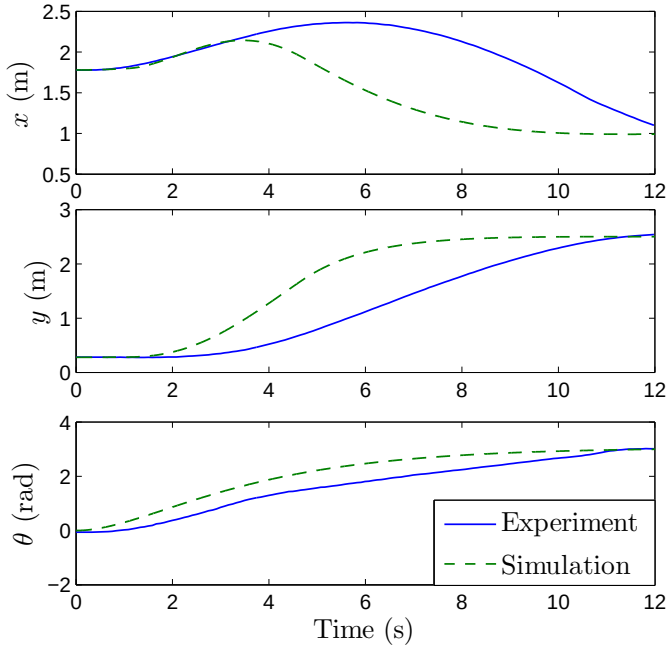


Figure 3.8: Experimental and simulated global trajectories: Case 2

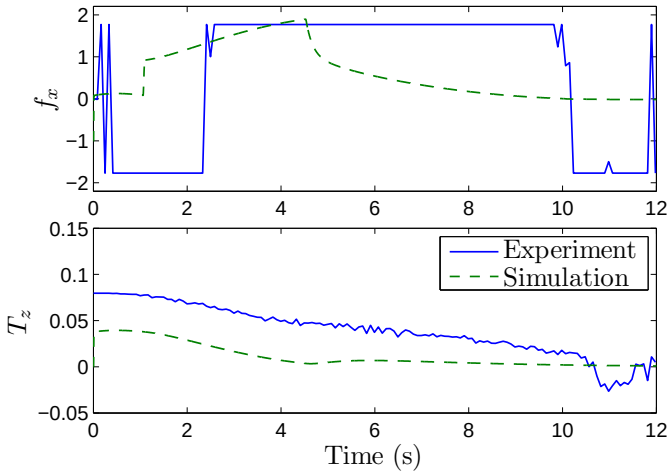


Figure 3.9: Experimental and simulated surge force and yaw moment for case 2

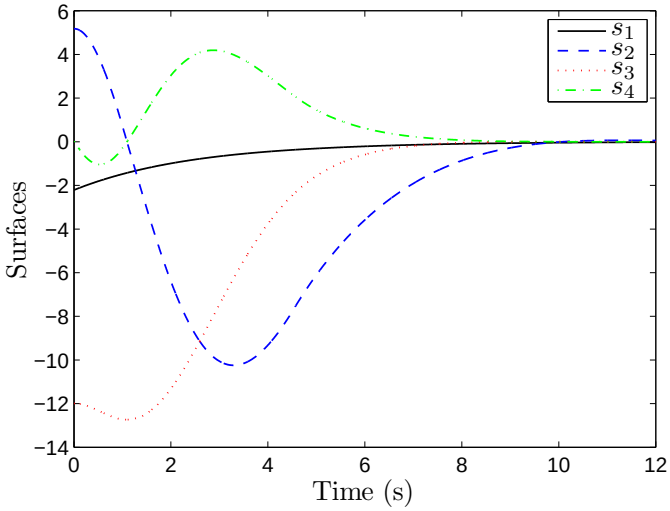


Figure 3.10: Simulated time history of the four surfaces: Case 2

3.4 Trajectory Tracking Control

In this section, a trajectory tracking sliding mode control law for autonomous surface vessels is developed using two sliding surfaces for calculation of the two propeller forces [28]. The first sliding surface is a first-order surface defined in terms of the surge motion tracking errors. The second sliding surface is a second-order surface defined in terms of the lateral motion tracking errors. It is assumed that only the absolute position and orientation of the vessel are measured and available for feedback as is the case for the experimental system presented in Section 3.2. Hence, the absolute velocities are numerically estimated, and surge and sway velocities are calculated through kinematic relations between inertial and body-fixed reference frames. The power law drag force model is considered where the values for $\alpha_{1,2,3}$ in Equation (3.8) are defined in Table 3.2.

The sliding mode controller is designed to track a target trajectory using the trajectories presented in Section 2.2. The desired state trajectories in the inertial reference frame are related to the corresponding surge and lateral velocities as follows

$$v_x^r = \dot{x}_1 \cos\theta + \dot{x}_2 \sin\theta \tag{3.33}$$

$$v_y^r = -\dot{x}_1 \sin\theta + \dot{x}_2 \cos\theta \tag{3.34}$$

where $\dot{x}_1 = \dot{x}^r - \dot{x}^t$ and $\dot{x}_2 = \dot{y}^r - \dot{y}^t$ as defined in Equation (2.12).

3.4.1 Surge Control Law

The first sliding surface is a first-order exponentially stable surface defined in terms of the vessel's surge motion tracking errors

$$s_1 = \tilde{v}_x + \lambda_1 \int_0^t \tilde{v}_x(\tau) d\tau, \quad \lambda_1 > 0 \tag{3.35}$$

where " ˜ " denotes the difference between the actual and desired values; *i.e.*, $\tilde{v}_x = v_x - v_x^r$. Note that the integral of v_x is used since position variables cannot be defined in the body-

fixed frame. Hence, the desired motion is specified in the inertial reference frame (x^r, y^r) and related to the desired surge and sway velocities and accelerations by Equations (3.33) and (3.34). The nominal surge control law for zero dynamics is determined by taking the time derivative of the surface and using Equation (3.4)

$$\dot{s}_1 = \dot{v}_x + \dot{s}_{r1} = 0, \quad \dot{s}_{r1} = -\dot{v}_x^r + \lambda_1 \tilde{v}_x \tag{3.36}$$

$$\hat{f} = -\hat{m}_{22} v_y \omega + \hat{d}_1 v_x^{\alpha_1} - \hat{m}_{11} \dot{s}_{r1} \tag{3.37}$$

where " ˆ " is used to indicate the estimated model parameters. The sliding mode control law is derived by subtracting a high-slope saturation function from the nominal control as in [54]

$$f = \hat{f} - k_1 \operatorname{sat}(s_1/\varepsilon_1) \tag{3.38}$$

$$\operatorname{sat}(s_1/\varepsilon_1) = \begin{cases} s_1/\varepsilon_1, & |s_1| \leq \varepsilon_1 \\ \operatorname{sgn}(s_1), & |s_1| > \varepsilon_1 \end{cases} \tag{3.39}$$

where ε_1 is a positive constant which defines a small boundary layer around the surface.

The coefficient k_1 is selected by first defining a Lyapunov candidate function that guarantees reaching the set $\{s_1 \in \mathbb{R} : |s_1| \leq \varepsilon_1\}$ in finite time and remain inside this set thereafter.

$$V_1 = \frac{1}{2} m_{11} s_1^2 \tag{3.40}$$

The time derivative of Equation (3.40) can be derived as presented in [28] where the following reaching condition is achieved

$$\dot{V}_1 = m_{11} s_1 \dot{s}_1 \leq -\hat{m}_{11} \eta_1 |s_1|, \quad \eta_1 > 0 \tag{3.41}$$

if k_1 is selected as

$$k_1 = M_{22} |v_y \omega| + D_1 v_x^{\alpha_1} + M_{11} |\dot{s}_{r1}| + \hat{m}_{11} \eta_1 \tag{3.42}$$

with the bounds for the model parameters, M_{ii} and D_i, defined as

$$|m_{ii} - \hat{m}_{ii}| \leq M_{ii}, \quad |d_i - \hat{d}_i| \leq D_i, \quad i = 1, 2, 3. \tag{3.43}$$

3.4.2 Lateral Motion Control Law

The second sliding surface is a second-order exponentially stable surface defined in terms of the vessel's lateral motion tracking errors

$$s_2 = \dot{\tilde{v}}_y + 2\lambda_2 \tilde{v}_y + \lambda_2^2 \int_0^2 \tilde{v}_y(\tau) d\tau, \quad \lambda_2 > 0 \tag{3.44}$$

where $\tilde{v}_y = v_y - v_y^r$ and $\dot{\tilde{v}}_y = \dot{v}_y - \dot{v}_y^r$. The nominal yaw moment control law is derived for zero dynamics by taking the time derivative of the lateral motion surface and setting it equal to zero

$$\dot{s}_2 = \ddot{v}_y - \ddot{v}_y^r + 2\lambda_2(\dot{v}_y - \dot{v}_y^r) + \lambda_2^2 \tilde{v}_y = 0 \tag{3.45}$$

where the time derivatives of Equation (3.5) and (3.34) yield the following.

$$\ddot{v}_y = -\frac{\alpha_2 d_2 \operatorname{sgn}(v_y)|v_y|^{\alpha_2 - 1}\dot{v}_y + m_{11}(\dot{v}_x \omega + v_x \dot{\omega})}{m_{22}} \tag{3.46}$$

$$\ddot{v}_y^r = v_r - v_x^r \dot{\omega} \tag{3.47}$$

$$v_r = (v_y^r \omega - 2\dot{v}_x^r)\omega - \sin\theta \dddot{x}^r + \cos\theta \dddot{y}^r \tag{3.48}$$

The nominal control law is derived by substituting (3.5), (3.6), and (3.46) – (3.48) into Equation (3.45) and solving for the yaw moment T

$$\hat{T} = \hat{h}/\hat{b} \tag{3.49}$$

where " ˆ " indicates the estimated model and

$$b = m_{22}v_x^r - m_{11}v_x \tag{3.50}$$

$$\begin{aligned} h = {} & b(m_d v_x v_y + d_3 \operatorname{sgn}(\omega) |\omega|^{\alpha_3}) \\ & - (m_{11}v_x\omega + d_2 \operatorname{sgn}(v_y) |v_y|^{\alpha_2 - 1}) \\ & + m_{33}\omega(f - d_1 v_x^{\alpha_1} + 2\lambda_2 m_{11} v_x + m_{22} v_y) \\ & + 2\lambda_2 m_{33} d_2 \operatorname{sgn}(v_y) + |v_y|^{\alpha_2} \\ & + m_{22} m_{33}(v_r + 2\lambda_2 \dot{v}_y^r - \lambda_2^2 \tilde{v}_y). \end{aligned} \tag{3.51}$$

The yaw moment sliding mode control law is defined using a high-slope saturation function in the same manner as the surge control law

$$T = [\hat{h} - k_2 \operatorname{sat}(s_2/\varepsilon_2)]/\hat{b} \tag{3.52}$$

where the nominal values of b and h are computed by Equations (3.50) and (3.51) using the estimated model parameters. Note that the lateral motion surface (3.44) must be second-order so that the yaw acceleration $\dot{\omega}$ and consequently the yaw control moment T appear in Equation (3.45). In this way, the underactuated nature of the USV system is overcome by using the yaw moment to indirectly control the sway velocity, which is otherwise unactuated. The yaw velocity ω is shown to be Bounded-Input-Bounded-Output (BIBO) stable in the following section. It is also interesting to note that the lateral motion control depends on surge force f, as one might expect intuitively.

In order to determine k_2, another Lyapunov candidate function is defined

$$V_2 = \frac{1}{2} m_{22} m_{33} s_2^2 \tag{3.53}$$

which guarantees reaching the set $\{s_2 \in \mathbb{R} : |s_2| \leq \varepsilon_2\}$ in finite time. This Lyapunov function yields the following reaching condition

$$\dot{V}_2 = m_{22} m_{33} s_2 \dot{s}_2 \leq -\hat{m}_{22}\hat{m}_{33}\eta_2 |s_2|, \quad \eta_2 > 0 \tag{3.54}$$

if k_2 is selected as

$$k_2 = \beta(H + \hat{m}_{22}\hat{m}_{33}\eta_2) + (\beta - 1)|\hat{h}| \tag{3.55}$$

with the bound for the uncertainty in h defined based on the parameter uncertainties defined in Equation (3.43)

$$|h - \hat{h}| \leq H \tag{3.56}$$

and β the bound based on the geometric mean of b [28].

In summary, the surge and yaw moment control inputs can be calculated by using Equations (3.38) and (3.52) respectively assuming that a USV model exists along with the appropriate bounds on the parameter uncertainty, and a continuously differentiable reference trajectory is provided. The individual propeller inputs are calculated using Equation (3.29). The operator must choose values for the surface parameters λ_1 and λ_2 in Equations (3.35) and (3.44) respectively, the effort parameters η_1 and η_2 in Equations (3.40) and (3.53) respectively and the boundary layer thicknesses ε_1 and ε_2 in Equations (3.38) and (3.52) respectively.

3.4.3 Stability

The surge force and yaw moment control laws in Equations (3.38) and (3.52) are derived based on the reaching conditions in Equations (3.41) and (3.54), respectively. Integration of these reaching conditions verifies that the trajectory reaches the two corresponding surfaces in a finite time of less than $\frac{m_{11}}{\hat{m}_{11}}\left(\frac{s_1(0)}{\eta_1}\right)$ and $\frac{m_{22}m_{33}}{\hat{m}_{11}\hat{m}_{33}}\left(\frac{s_2(0)}{\eta_2}\right)$ respectively. Furthermore, the two surfaces in Equations (3.35) and (3.44) are asymptotically stable. Therefore, the trajectory exponentially slides to the origin at the intersection of the two surfaces

$$\begin{aligned} \tilde{v}_x &\to 0, \ \int_0^t \tilde{v}_x(\tau)d\tau \to 0 \\ \tilde{v}_y &\to 0, \ \int_0^t \tilde{v}_y(\tau)d\tau \to 0 \end{aligned} \tag{3.57}$$

and the kinematic relations in Equations (3.1) – (3.3) guarantee trajectory tracking in the inertial reference frame.

It can be shown that ω is BIBO stable as follows. Define the Lyapunov candidate function

$$V_3 = \frac{1}{2} m_{33} \omega^2. \tag{3.58}$$

Using Equations (3.4) – (3.6), the time derivative of V_3 may be written as

$$\begin{aligned} \dot{V}_3 &= \omega \left[T - m_d v_x v_y - d_3 \operatorname{sgn}(\omega) \left| \omega \right|^{\alpha_3} \right] \\ &= \omega (T - m_d v_x v_y) - d_3 \left| \omega \right|^{1+\alpha_3} \qquad (3.59) \\ \dot{V}_3 &< 0 \;\text{ if }\; d_3 \left| \omega \right|^{\alpha_3} > \left| T - m_d v_x v_y \right| . \qquad (3.60) \end{aligned}$$

It follows from Equation (3.58) that if V_3 is a decreasing function, then $|\omega|$ is also a decreasing function. From Equation (3.60), V_3 is decreasing when $|\omega| > \frac{|T - m_d v_x v_y|}{d_3}^{\frac{1}{\alpha_3}}$ which determines an upper bound for $|\omega|$. Since T,v_x, and v_y are bounded, the upper bound for ω remains bounded.

3.4.4 Results

In the following simulation example, a target follows a circular trajectory centered at $(x = 1\text{m}, y = 2\text{m})$ with a radius of 0.5 meters and constant angular velocity of 0.2 rad/sec (a period of about 30s) beginning at $(x = 1.5\text{m}, y = 2\text{m})$. The USV must approach and track the target from an initial position at $(x = 2\text{m}, y = 1\text{m})$ represented by the boat icon in Figure 3.11. The reference trajectory $(v_x^r(t), v_y^r(t))$ is generated using the target tracking trajectory presented in Section 2.2.1 with trajectory parameters $\bar{k}_1 = \bar{k}_2 = 0.8$. In this case, no transition period is required for the trajectory parameters ($t_1 = t_0$). The control parameters are selected as $\lambda_1 = \lambda_2 = 1$, $\varepsilon_1 = \varepsilon_2 = 0.05$ and $\eta_1 = \eta_2 = 0.2$.

The target and USV paths are presented in Figure 3.11 and the control action is presented in Figure 3.12. As shown in these figures, the control action is feasible and the USV reaches the target in a reasonable amount of time. However, controller performance is a function of the control parameters and this relationship can be non-intuitive. Optimal parameter selection based on various performance objectives is discussed in Chapter 5.

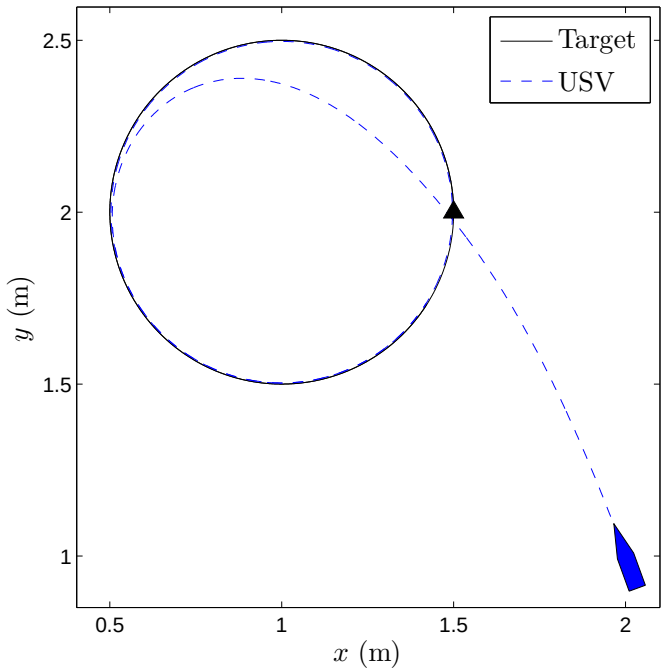


Figure 3.11: Simulated USV path: trajectory tracking SMC

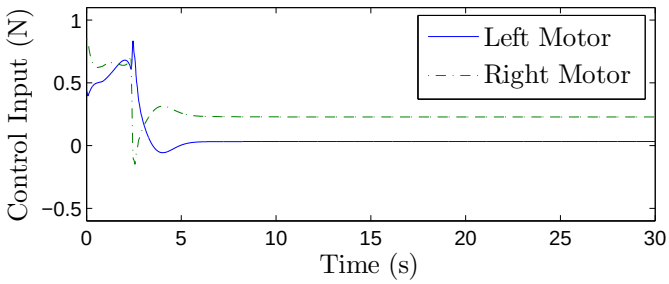


Figure 3.12: Control input: trajectory tracking SMC

Chapter 4

Model Predictive Control for Unmanned Surface Vessels

System dynamics and control input constraints are a critical factor in trajectory planning and control system design for nonholonomic autonomous systems. A disadvantage of many of the previous trajectory tracking control methods is that significant effort must go into generating target trajectories which are continuous and admit feasible control laws. Target trajectories that require excessive control action to satisfy the state equations are unlikely to be feasible or will place undue demands on the control system and actuators. System and control action constraints are explicitly included in the model predictive control formulation and therefore, the resulting controller will respect the dynamic and actuator constraints of the system. Discontinuities in the desired system trajectory can also be tolerated because predictive control will attempt to track the target trajectory as closely as possible subject to the dynamic limitations of the system and actuators. Another advantage of predictive control for this application is that both the setpoint and tracking controllers are formulated in the same manner.

4.1 System Dynamics

The dynamic behavior of the USV system model poses significant challenges for the implementation of model predictive control. As an example of this behavior, consider the minimum-time optimal control problem in which the vessel is initially at rest and must come to a stop at a fixed target 5 m away in minimum time. The surge control forces are each constrained such that $0 \leq f_{1,2} \leq 1$ N. The minimum-time optimal control law is a bang-bang controller where the maximum surge force is applied to both propellers for 4.8 sec after which the surge forces are set to zero and the vessel is allowed to drift to a stop. Figure 4.1 presents the simulated trajectories for nominal and perturbed initial conditions. In the nominal case, the minimum-time optimal control forces are applied with no external forces or disturbances acting on the vessel. For the perturbed case, the minimum-time optimal control forces are applied with an initial impulse error of 1% in the right propeller force imposed at zero time. The nominal vessel trajectory is indicated by the dashed line, the perturbed trajectory is indicated by the solid line and the circle on each trajectory rep-

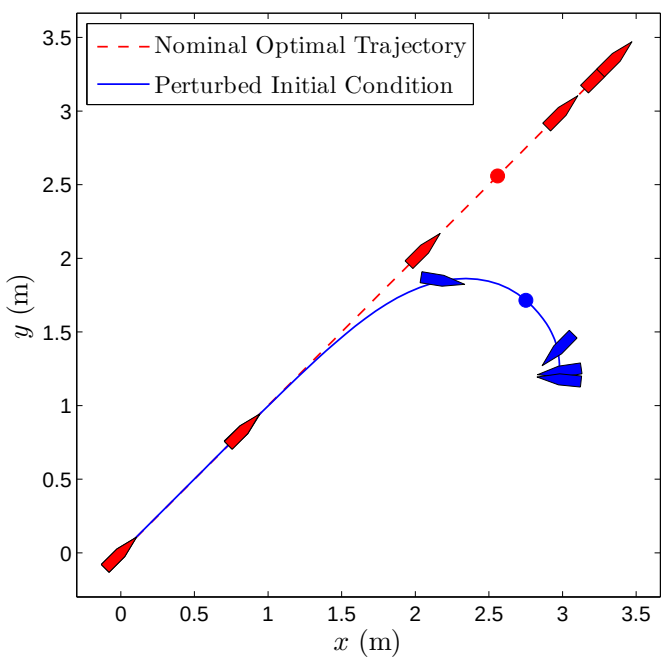


Figure 4.1: Simulated nominal and perturbed trajectories

resents the point at which the control force is set to zero. The boat-shaped icons indicate the vessel position at 2 second intervals on each trajectory. The effect of the initial impulse error in the perturbed case does not become apparent until almost halfway through the powered portion of the vessel trajectory where a circular orbit begins to emerge from the linear trajectory.

The behavior illustrated in this simulation example is also exhibited by the experimental USV system. Figure 4.2 presents an experimental example of a similar scenario in which the vessel is initially powered to provide straight line motion after which both control forces are set to zero and the vessel is allowed to drift to a stop. The origin of the experimental vessel trajectory in this figure represents the position where the control surge forces were set to zero. The experimental vessel trajectory quickly deviates from straight line motion into a circular motion before coming to a stop.

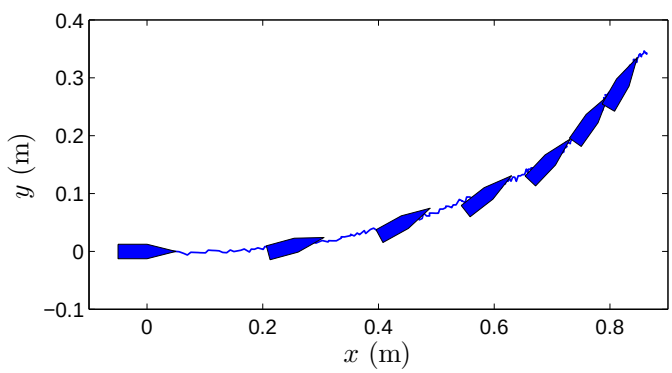


Figure 4.2: Experimental demonstration of system behavior

A stability analysis of straight line and circular motion for the USV system model presented in Equations (3.4) – (3.6) shows that for all nonlinear damping models, $\alpha > 1$, straight line motion is unstable and circular motion is always stable [55]. It is also shown that for linear damping, $\alpha = 1$, straight line motion is always stable for the relatively small propulsive forces considered in this work. Therefore, in the case of a nonlinear drag model, small perturbations to the control input or state can lead to significant future deviation in the vessel trajectory. Because disturbances in the physical system will always result in small perturbations in the sway and yaw velocities, straight line motion can never

be achieved in practice without persistent feedback control correction. The challenge for predictive control in this application is the requirement of sufficiently long prediction and control horizons that are able to capture and correct this behavior while retaining an on-line optimization problem that can be solved in real time.

4.2 Model Predictive Control

The model predictive control strategy attempts to achieve a specified performance objective in an optimal manner by solving a finite horizon, open-loop optimal control problem on-line at each control interval. The algorithm computes the control inputs $\boldsymbol{u}$ that minimize the objective function over the prediction horizon N. The first input $\boldsymbol{u}_1$ is implemented for one control interval and the process is repeated given the new initial conditions of the system. This technique is referred to as receding horizon control.

A finite horizon performance objective following the Lagrange cost function in optimal control [56] is used

$$\min_{\{\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n\}} J_k = \int_{k\Delta t}^{(k+N)\Delta t} \Phi(\boldsymbol{z}, \boldsymbol{u}, t)\, dt \tag{4.1}$$

where J_k is the optimization objective function value at sample time k, Δt is the control interval, $\Phi(\cdot)$ is the performance cost function, $\boldsymbol{z}$ is the state of the system, $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\}$ are the set of n future control inputs that determine $\boldsymbol{u}$, and n is the control horizon. This objective is minimized at each control interval subject to the constraints

$$\dot{\boldsymbol{z}} = \boldsymbol{f}\left(\boldsymbol{z}, \boldsymbol{u}, t\right) \tag{4.2}$$

$$\boldsymbol{z} \in \mathcal{Z}, \quad \{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n\} \in \mathcal{U} \tag{4.3}$$

$$\boldsymbol{0} \geq \boldsymbol{w}\left(\boldsymbol{z}, \boldsymbol{u}, t\right) \tag{4.4}$$

in which $\boldsymbol{f}(\boldsymbol{z}, \boldsymbol{u}, t)$ is the dynamic equality constraint arising from the system of nonlinear ODEs in Equations (3.1) – (3.6) that represent the dynamic behavior of the state of the vessel $\boldsymbol{z} = [x, y, \theta, v_x, v_y, \omega]^T$, $\mathcal{Z}$ is the constraint space for the state inequality constraints, $\mathcal{U}$ is the constraint space for the control inequality constraints, and $\boldsymbol{w}(\boldsymbol{z}, \boldsymbol{u}, t)$ is a general

inequality constraint on the states and control.

The control input vector $\boldsymbol{u}$ is parameterized by a series of step functions of varying duration with magnitudes $\boldsymbol{u}_1$ to $\boldsymbol{u}_n$ as shown in Figure 4.3 where the values $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ are the decision variables for the on-line optimization problem. As opposed to holding the value of each future input for a single control interval in the prediction horizon, an input or move blocking strategy [38, 57] is employed where each future input move is held constant over several control intervals within the prediction horizon. The advantage of this approach is a reduction in the number of future control moves (optimization decision variables) that must be determined such that $n << N$. Because of the dynamic behavior of the USV system illustrated in Section 4.1, a relatively long prediction horizon is required to obtain acceptable closed-loop control performance in this application. The use of input blocking allows for both a long prediction horizon and a small number of decision variables for the optimization problem in Equations (4.1) – (4.4) resulting in a significant reduction in the computation time required to solve the optimization problem.

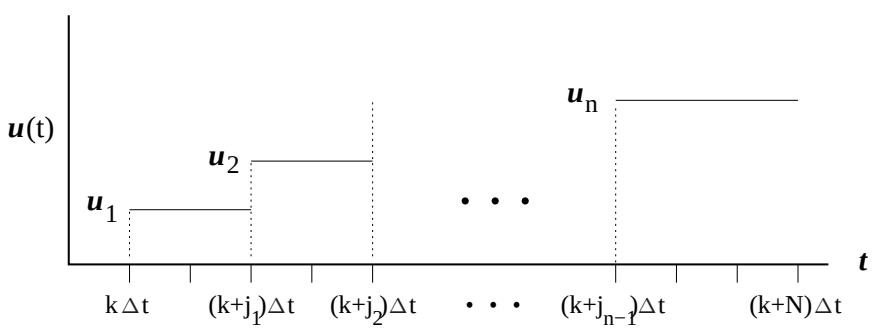


Figure 4.3: Control input vector parameterization

4.3 Results

The performance of model-based predictive control for trajectory tracking and setpoint control of the model USV system is illustrated in this section through several simulation examples. The trajectory and setpoint control examples are based on the problem of recovering an autonomous USV using a stern recovery ramp on a target vessel [58] whose

position and orientation are denoted by x_t, y_t and θ_t. The initial recovery phase, in which the USV must converge to the target vessel trajectory or position, is considered in the examples presented in this chapter.

The performance objective used in these examples is a quadratic penalty on the distance from the target trajectory and the desired orientation or heading angle

$$\Phi(\boldsymbol{z}, \boldsymbol{u}, t) = w_d d(t)^2 + w_a\big(\theta(t) - \theta_d(t, d)\big)^2 \tag{4.5}$$

$$d(t)^2 = \big(x(t) - x_t(t)\big)^2 + \big(y(t) - y_t(t)\big)^2 \tag{4.6}$$

where θ_d is the desired USV orientation and d is the distance between the USV and the target. The saturation constraint on the control inputs

$$\begin{bmatrix} -0.5 \\ -0.5 \end{bmatrix} \leq \boldsymbol{u}_i \leq \begin{bmatrix} 1.0 \\ 1.0 \end{bmatrix}, \quad i = 1, \ldots, n \tag{4.7}$$

is applied where both actuators have the same maximum and minimum constraint for each of the examples in this work. This constraint represents the approximate surge force limits on the experimental system.

Both the nonlinear drag force model presented in Equation (3.8) and a linear drag force model where $\alpha_{1,2,3} = 1$ are considered for comparison. The trajectories using the nonlinear drag force model are indicated by dashed lines in Figures 4.5, 4.7, and 4.9. The trajectories using linear drag force model are indicated by a dot-dash line in the same figures. The linear drag force is the more common form used in USV dynamic modeling [46–48] and results in stable linear trajectories, but it is only really appropriate at very low velocities. The nonlinear drag force is more representative at the vessel velocities considered in this work, but results in a much more difficult system to control as suggested by Section 4.1 and demonstrated in the following simulations.

4.3.1 Linear Target Trajectory Tracking

In this example, the recovery vessel is following a linear path with a constant velocity. The USV must first approach the recovery vessel and then track its trajectory. The following performance objective parameters are used

$$w_d = w_a = 1, \quad \theta_d(t,d) = \theta_t e^{-\beta d} + \theta_a(1 - e^{-\beta d}) \tag{4.8}$$

where the desired USV orientation angle θ_d is a function of the target vessel orientation angle θ_t and the angle of the line between the USV and the target vessel θ_a. In order to obtain reasonable tracking trajectories, the USV orientation angle should approach θ_a when it is far away from the target and approach the target orientation angle when it is close to the target. This behavior is produced by the exponential relationship in Equation (4.8) as shown in Figure 4.4 with $\beta = 8$, $\theta_t = 0°$, and $\theta_a = 90°$.

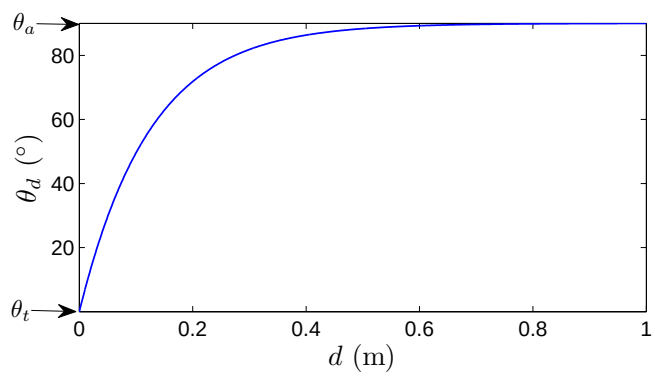


Figure 4.4: θ_d relationship

The control and prediction horizons are $n = 2$ and $N = 10$ with a control period of $\Delta t = 0.2$sec. The control move is parameterized within the prediction horizon as

$$\boldsymbol{u}(t) = \begin{cases} \boldsymbol{u}_1, & k\Delta t \leq t < (k + j_1)\Delta t \\ \boldsymbol{u}_2, & (k + j_1)\Delta t \leq t < (k + N)\Delta t \end{cases}$$

where $j_1 = 5$. In order to improve the tracking performance and prevent the USV from col-

liding with the target vessel with this short control horizon, the following state constraints

$$y - y_t - \varepsilon_y \leq 0, \quad \theta - \theta_t - \varepsilon_\theta \geq 0, \quad x - x_t - \varepsilon_x \leq 0$$

are also applied in this example where $\varepsilon_y = .01$ and $\varepsilon_\theta = -4°$ are the acceptable y and θ overshoot and $\varepsilon_x = 0$ is the stern distance bias of the target vessel. The first two constraints prevent excessive overshoot from the desired trajectory and oscillation in the USV heading angle respectively. The last constraint is used to prevent a collision. In general, the exact form of these constraints will be a function of the initial vessel positions. Figure 4.5 presents the USV trajectory with the nonlinear and linear drag forces where the target vessel trajectory is the solid line and the circles represent the location of the USV and target vessel at three second intervals. The control inputs are presented in Figure 4.6. As shown in these figures, the tracking performance with both drag force models is similar. Because the desired linear trajectory is stable with linear drag forces, the controller is able to converge to the target trajectory after approximately six seconds. With the nonlinear drag forces, however, the desired trajectory is unstable. The controller must apply significantly more control action and requires about eight seconds to converge. Improved performance is possible with a control horizon of $n = 3$ at the expense of an increase in optimization time for this example.

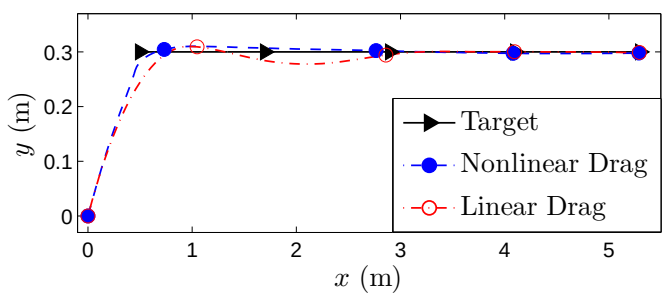


Figure 4.5: Simulated linear target trajectory paths

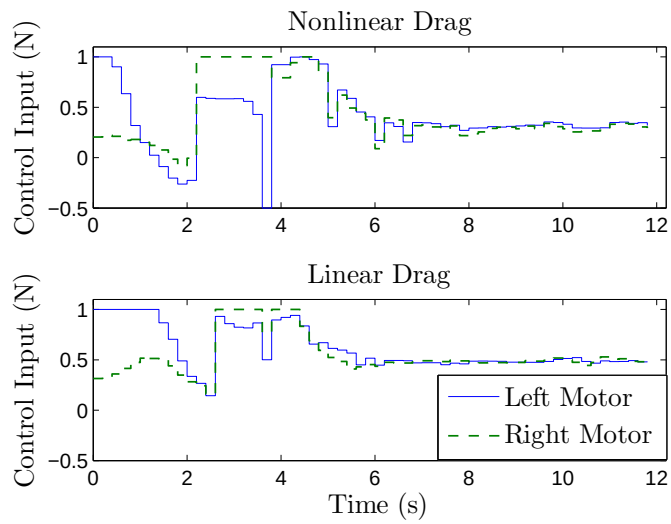


Figure 4.6: Linear target trajectory control input

4.3.2 Circular Target Trajectory Tracking

In this example, the recovery vessel is following a circular path with a constant angular velocity. The following performance objective parameters are used

$$w_d = 1, \quad w_a = 0$$

where no penalty on the USV orientation angle or inequality constraints to avoid excessive overshoot or oscillation are imposed. The control and prediction horizons are $n = 3$ and $N = 30$ with a control period of $\Delta t = 0.2$sec. The control move is parameterized within the prediction horizon as

$$\boldsymbol{u}(t) = \begin{cases} \boldsymbol{u}_1, & k\Delta t \leq t < (k + j_1)\Delta t \\ \boldsymbol{u}_2, & (k + j_1)\Delta t \leq t < (k + j_2)\Delta t \\ \boldsymbol{u}_3, & (k + j_2)\Delta t \leq t < (k + N)\Delta t \end{cases}$$

where $j_1 = 5$ and $j_2 = 10$.

Because a circular trajectory is stable for both the nonlinear and linear drag forces, the orientation angle penalty and inequality constraints imposed for the linear trajectory are

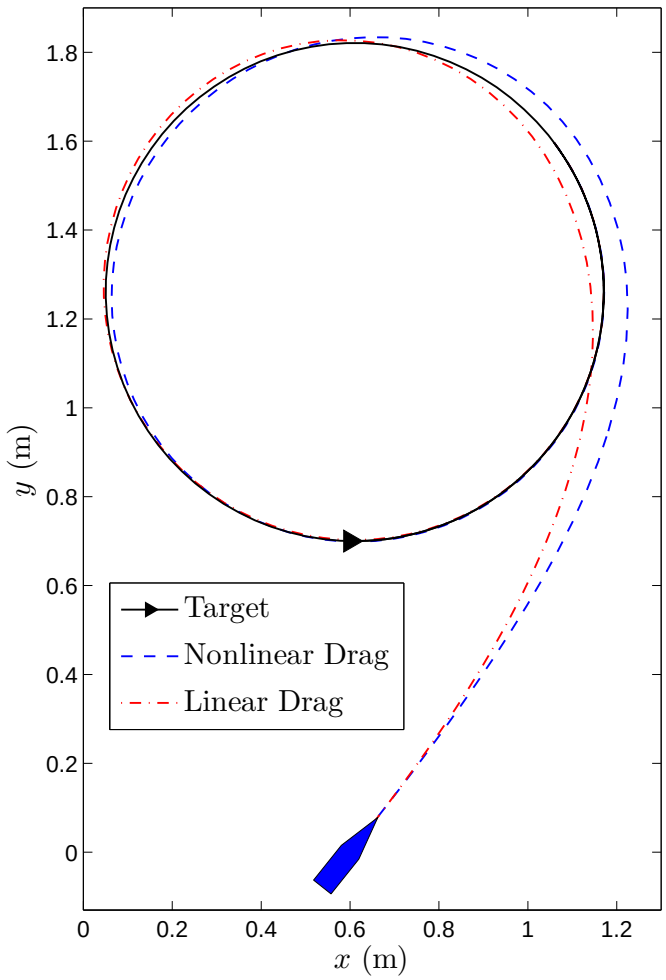


Figure 4.7: Simulated circular target trajectory paths

not necessary to obtain good tracking performance in this example. Figure 4.7 presents the USV trajectory with the nonlinear and linear drag forces where the target vessel begins at the position indicated by the triangular marker and the USV begins at the position and orientation indicated by the boat icon. The control inputs are presented in Figure 4.8. As shown in these figures, similar performance is obtained with both drag models where the USV converges onto the desired trajectory in both cases without overshoot or oscillation in the heading angle.

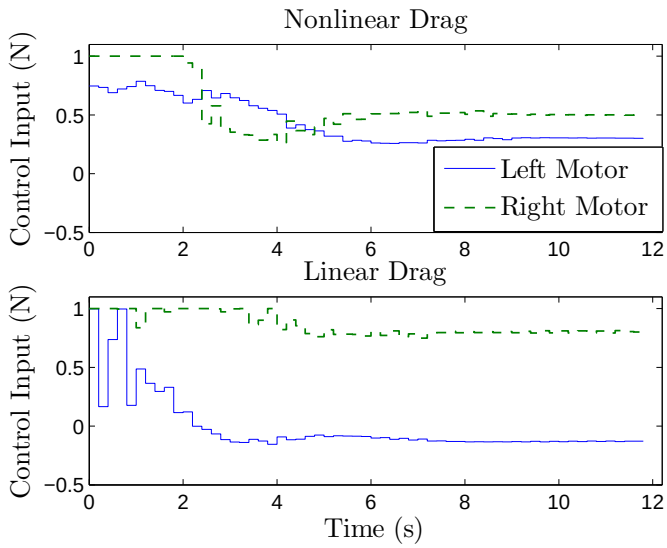


Figure 4.8: Circular target trajectory control input

4.3.3 Setpoint Control

In the following examples, the USV must approach and come to rest at a fixed target position and orientation angle. In the first example, the USV begins at the origin and the initial and desired setpoint orientations are the same. The position setpoint is $(2\text{ m}, 2\text{ m})$. The following performance objective parameters are used for this scenario

$$w_d = w_a = 1, \quad \theta_d(t, d) = \theta_i$$

where θ_i is the initial orientation angle of the USV. The control and prediction horizons are $n = 2$ and $N = 10$ with a control period of $\Delta t = 0.2$sec. The control moves are parameterized within the prediction horizon as

$$\boldsymbol{u}(t) = \begin{cases} \boldsymbol{u}_1, & k\Delta t \leq t < (k + j_1)\Delta t \\ \boldsymbol{u}_2, & (k + j_1)\Delta t \leq t < (k + N)\Delta t \end{cases}$$

where $j_1 = 1$. In order to prevent overshooting the desired position target, the following inequality constraints are also applied

$$x - x_t - \varepsilon_x \leq 0, \quad y - y_t - \varepsilon_y \leq 0$$

where $\varepsilon_x = \varepsilon_y = 0$. We note that the form of these constraints will be a function of the initial USV and target positions and orientations in general. Figure 4.9 presents the USV trajectory for both the nonlinear and linear drag forces, where the circle represents the target USV position. The control inputs for each propeller are shown in Figure 4.11. A detail view of the boxed area in Figure 4.9 is shown in Figure 4.10. The blue and red boat icons represent the USV position at 4 second intervals and the black boat icon represents the desired position and orientation. As shown in this figure, the linear drag force converges exactly to the black icon while there is approximately 2 cm of drift in the nonlinear drag force trajectory. The control corrections at approximately 5 sec were not able to completely eliminate the lateral velocity component in this case.

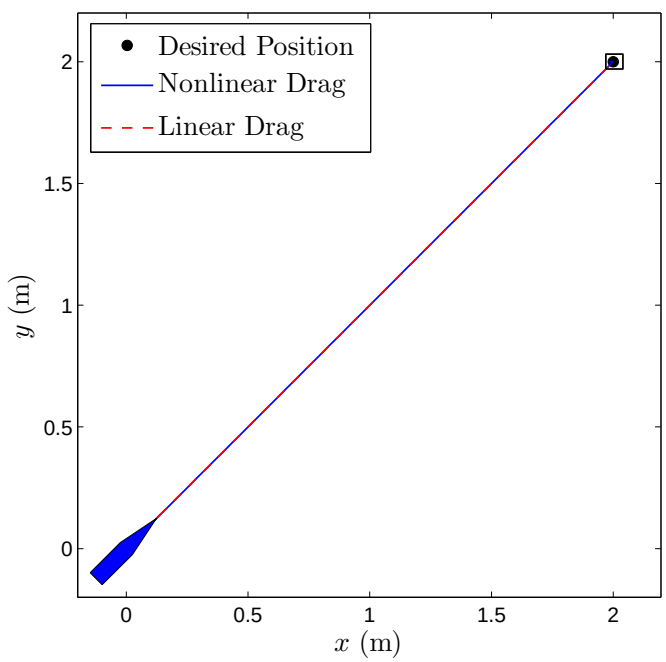


Figure 4.9: Simulated MPC setpoint path: Case 1

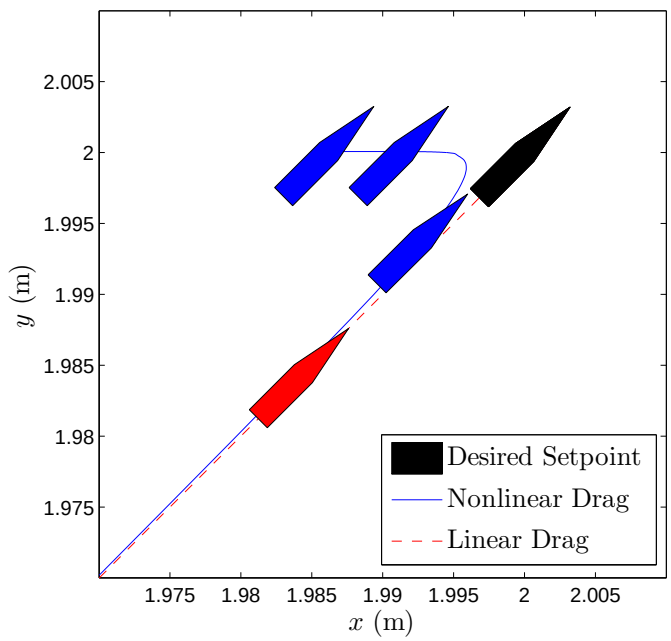


Figure 4.10: Simulated MPC setpoint closeup paths: Case 1

For the second example, the USV is initially positioned at the origin with an orientation angle of $\theta_i = 0$ degrees. The setpoint position is again $(2 \text{ m}, 2 \text{ m})$ with an orientation angle of $\theta_t = 90$ degrees. The following performance objective parameters are used for this scenario

$$w_d = 1, \quad w_a = 10, \quad \theta_d(t, d) = \theta_t e^{-\beta d_x} + \theta_i(1 - e^{-\beta d_x})$$

where θ_i is the initial orientation angle of the USV and $\beta = 2$. The control and prediction horizons are $n = 2$ and $N = 6$ with a control period of $\Delta t = 0.25$sec. The control move is parameterized within the prediction horizon as

$$\boldsymbol{u}(t) = \begin{cases} \boldsymbol{u}_1, & k\Delta t \leq t < (k + j_1)\Delta t \\ \boldsymbol{u}_2, & (k + j_1)\Delta t \leq t < (k + N)\Delta t \end{cases}$$

where $j_1 = 1$. In order to prevent overshooting the desired position target, the following

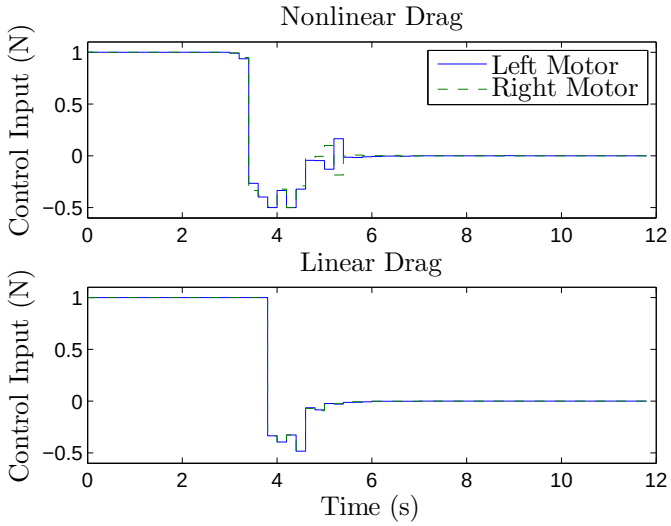


Figure 4.11: MPC setpoint control input: Case 1

inequality constraints are applied

$$x - x_t - \varepsilon_x \leq 0, \quad y - y_t - \varepsilon_y \leq 0, \quad -v_x \leq 0$$

where $\varepsilon_x = \varepsilon_y = 0.01$ m. The linear drag force model is considered in this example.

Figure 4.12 presents the USV trajectory for linear drag forces, where the black boat icon represents the USV setpoint. The control inputs for each propeller are shown in Figure 4.13 and the error trajectories are shown in Figure 4.14. As shown in this figure, the position and orientation errors converge in about 7 seconds with a final absolute position error of less than about 0.5 mm and a final orientation error of less than about 0.1 degree.

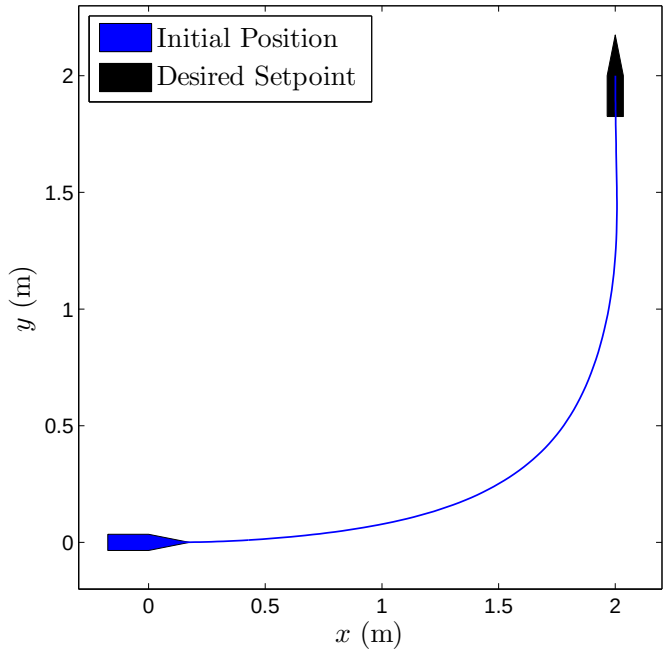


Figure 4.12: Simulated MPC setpoint path: Case 2

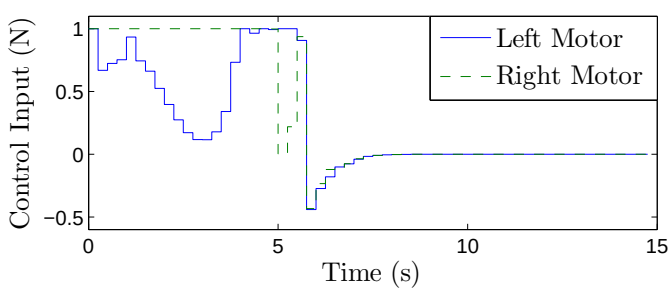


Figure 4.13: MPC setpoint control input: Case 2

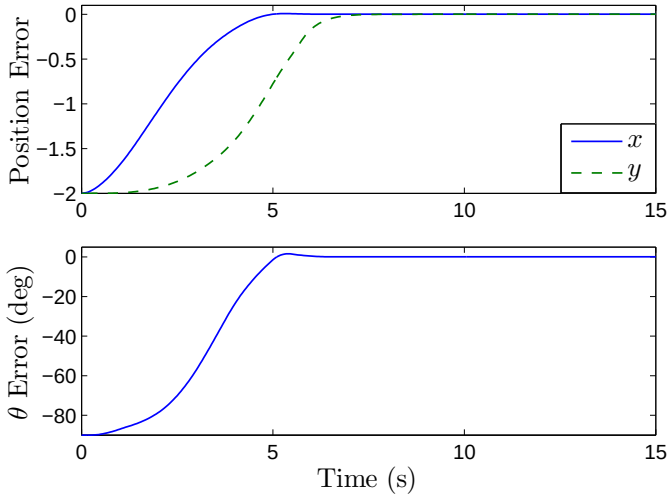


Figure 4.14: MPC setpoint error: Case 2

Chapter 5

Predictive and Sliding Mode Cascade Control Structure for Unmanned Surface Vessels

The sliding mode tracking controller presented in Section 3.4 computes very quickly but the dynamic performance of the controller is a complex function of the effort, surface, and trajectory parameters. It is difficult to manually determine parameters that yield optimal controller performance for a given scenario, initial conditions, and desired objective function without extensive simulation of the closed-loop system. Furthermore, the optimal parameters for one initial condition can yield poor or unacceptable performance for a different initial condition. A significant factor limiting the performance of the USV sliding mode control laws is input saturation. Because sliding mode control does not explicitly consider input constraints, input saturation is typically handled by clipping the control inputs to satisfy these constraints. Clipping rarely results in optimal control performance in any multi-input multi-output system. This situation is especially the case for the nonholonomic USV system since the control law calculates a surge force and yaw moment but the saturation is applied to individual propeller inputs. The resulting saturation can lead to increased control chatter that can be detrimental to mechanical actuators. In this chapter, a multi-rate cascade control structure is presented which optimally adjusts the sliding

mode controller parameters using an MPC controller. This cascade structure allows for near-optimal performance while providing a control law that can be computed in real-time and respects input saturation constraints.

5.1 Cascade Control Structure

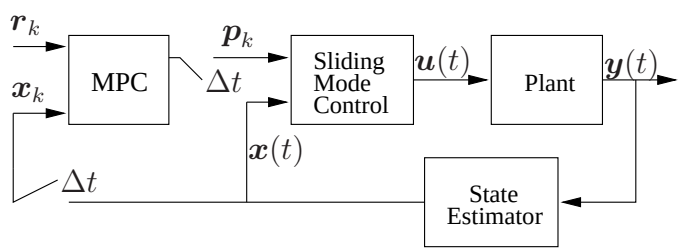


Figure 5.1: Controller block diagram

A block diagram of the cascade control structure is shown in Figure 5.1. The primary loop is a discrete-time, nonlinear model predictive controller that re-optimizes the control parameters of the sliding mode controller in the secondary loop at each control interval. The MPC controller executes at a rate that allows the on-line optimization to complete. For this application, the control interval Δt does not need to be fixed. The sliding mode controller computes the control input to the USV system at a rate sufficient to stabilize the system. The model predictive controller is based on a finite horizon performance objective following the Lagrange cost function in optimal control [56] similar to Equation (4.1)

$$\min_{\boldsymbol{p}} J_k = \int_{k\Delta t}^{(k+N)\Delta t} \Phi(\boldsymbol{z}, \boldsymbol{u}, t)\, dt \tag{5.1}$$

where J_k is the objective function value at sample time k, Δt is the control interval, N is the prediction horizon, $\Phi(\cdot)$ is the performance penalty function, $\boldsymbol{z}$ is the state of the system and $\boldsymbol{u}$ is the propeller forces determined by the sliding mode controller. This objective is minimized over the surface, effort and trajectory parameters $\boldsymbol{p}$ subject to the following

constraints

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z}, \boldsymbol{u}, t) \tag{5.2}$$

$$\boldsymbol{u} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -B/2 & B/2 \end{bmatrix}^{-1} \begin{bmatrix} f \\ T \end{bmatrix} \tag{5.3}$$

$$\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{u}) \geq \boldsymbol{0} \tag{5.4}$$

$$\boldsymbol{h}(\boldsymbol{p}) \geq \boldsymbol{0} \tag{5.5}$$

where $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ are kinematic and dynamic equality constraints arising from the USV system model Equations (3.1) – (3.6), $\boldsymbol{g}(\boldsymbol{z}, \boldsymbol{u})$ is a general inequality constraint on the states and control, and $\boldsymbol{h}(\boldsymbol{p})$ is a general inequality constraint on the controller parameters arising from the surface and trajectory derivation. Equation (5.3) determines the individual propeller forces as a function of surge force f and yaw moment T.

5.2 Circular Target Trajectory

The scenario and results presented in Section 3.4.4 are used as a base case for performance comparison with an initial optimal surface design and the MPC cascade control structure. A single optimization of the surface parameters is carried out at time $t = 0$ resulting initial optimal surface and trajectory parameters as in [59]. The target follows a circular trajectory centered at $(x = 1\text{m}, y = 2\text{m})$ with a radius of 0.5 meters beginning at $(x = 1.5\text{m}, y = 2\text{m})$. The USV must approach and track the target from an initial position at $(x = 2\text{m}, y = 1\text{m})$. The reference trajectory $(v_x^r(t), v_y^r(t))$ is generated using the target tracking trajectory presented in Section 2.2.1. For simplicity, the values of k_1 and k_2 in Equation (2.13) are defined to be equal; *i.e.*, $k_1 = k_2 = k$. Likewise, the values of the effort parameters η_1 and η_2 in Equations (3.40) and (3.53) respectively are defined to be equal; *i.e.*, $\eta_1 = \eta_2 = \eta$. The following input saturation constraints are imposed based on physical constraints of the

experimental system.

$$\boldsymbol{g}_1(\boldsymbol{u}) = \begin{bmatrix} 0.5 + f_i(t) \\ 1 - f_i(t) \end{bmatrix} \geq \mathbf{0}, \quad i = 1, 2 \tag{5.6}$$

The four control parameters $\boldsymbol{p} = [\lambda_1, \lambda_2, \eta, k]^T$ are selected as the decision variables for the optimization subject to the inequality constraints

$$\boldsymbol{h}(\boldsymbol{p}) = \boldsymbol{p} = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \eta \\ k \end{bmatrix} \geq \mathbf{0} \tag{5.7}$$

such that the sliding mode parameters are strictly positive. Minimum time, minimum tracking error and minimum energy performance objectives are considered and simulation results are presented.

5.2.1 Minimum Tracking Error Objective

The minimum tracking error objective determines the sliding mode controller parameters that minimize the integral of time multiplied by the squared error (ITSE) cost function

$$\min_{\boldsymbol{p}} J_k = \int_{k\Delta t}^{(k+N)\Delta t} d^2 t \, dt \tag{5.8}$$

subject to the kinematic and dynamic equality constraints in Equations (3.1) – (3.6), the input saturation constraints in Equation (5.6) and the parameter constraints in Equation (5.7) where d is defined as the distance between the center of mass of the USV and the target position.

$$d = \sqrt{(x - x^r)^2 + (y - y^r)^2} \tag{5.9}$$

The sliding mode control law $\boldsymbol{u}$ is updated by the model predictive controller at a control interval of $\Delta t = 1$ s and the prediction horizon is $N = 50$.

A comparison of the sliding mode control parameters, the relative ITSE cost and the

reach time for the base case, the initial optimization and the MPC cascade control structure are summarized in Table 5.1. The reach time indicates the time requred for the USV to approach and remain within $d_{tol} = 0.01$ m of the target. Note that the solution to the initial optimization problem is the same as the first control interval in the MPC cascade sequence. Figure 5.2 presents the resulting USV paths for all three cases and the control action is presented in Figure 5.3. Figure 5.4 shows the control parameter evolution over the simulation period. Due to the nature of the target tracking trajectory definition, the parameters must be relatively small when the USV is far away from the target such that the control law satisfies the input saturation constraints. However, as is the case with the initial one-time optimization, control input is quickly reduced as the USV moves closer to the target. The MPC cascade control structure updates the control law such that maximum control input can be maintained until the USV approaches the target, as illustrated by the discontinuities in the third plot in Figure 5.3. Although the path taken is similar to the initial optimization, the increased control input early in the trajectory allows the USV to reach the target more quickly, as demonstrated by the tracking errors shown in Figure 5.5 and the reach time given in Table 5.1. Figure 5.6 presents the normalized tracking error cost versus time for all three cases. The initial optimization yields a 30% performance gain over the base case and the MPC cascade structure yields another 13% cost improvement over the initial optimization.

Table 5.1: Minimum error control parameters and cost: Circular target trajectory

Parameter	Base	Initial Optimal	Cascade MPC
λ_1	1	1.33	$1.33,\ 1.77,\ 3.20,\ 3.20, \ldots$
λ_2	1	0.70	$0.70,\ 0.86,\ 1.69,\ 1.69, \ldots$
η	0.2	0.29	$0.29,\ 0.28,\ 0.14,\ 0.14, \ldots$
k	0.8	1.13	$1.13,\ 1.28,\ 1.43,\ 1.43, \ldots$
Normalized Cost	1	0.705	0.613
Reach Time (s)	7.87	5.42	5.01

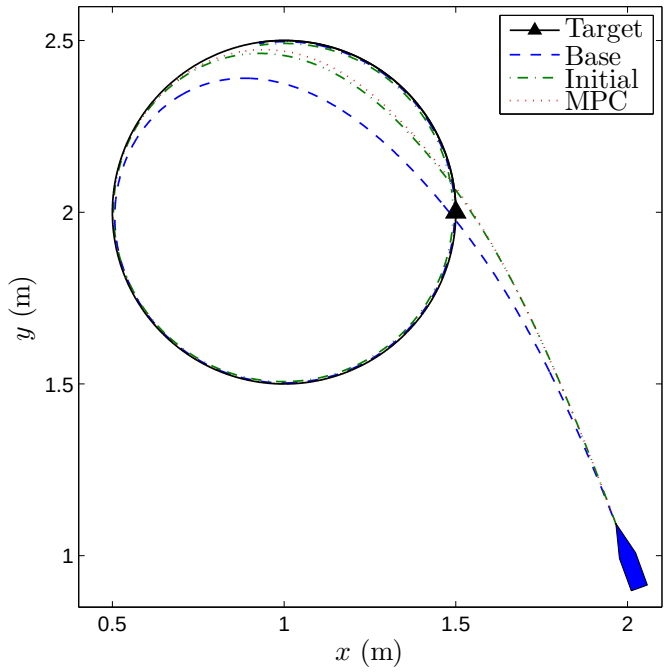


Figure 5.2: Simulated minimum error paths: Circular trajectory

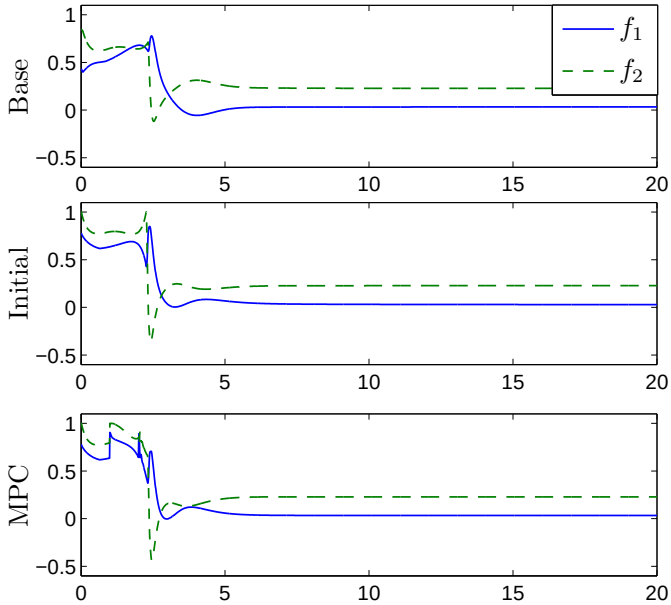


Figure 5.3: Minimum error control input: Circular trajectory

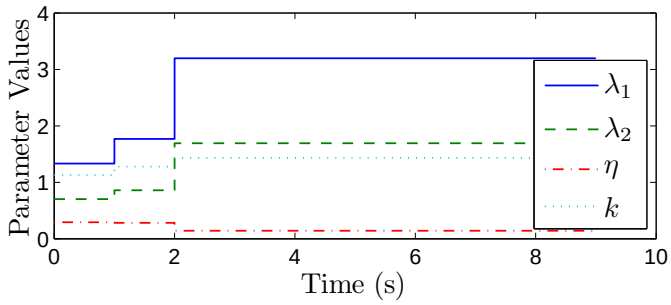


Figure 5.4: Minimum error parameter evolution: Circular trajectory

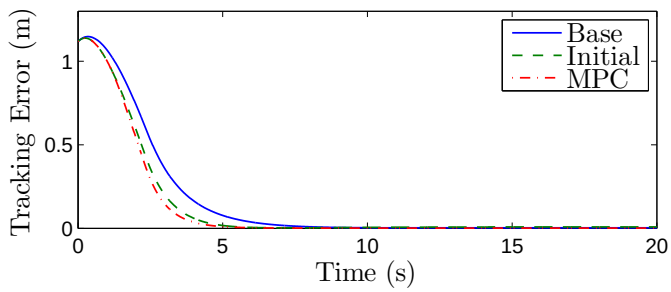


Figure 5.5: Minimum error tracking errors: Circular trajectory

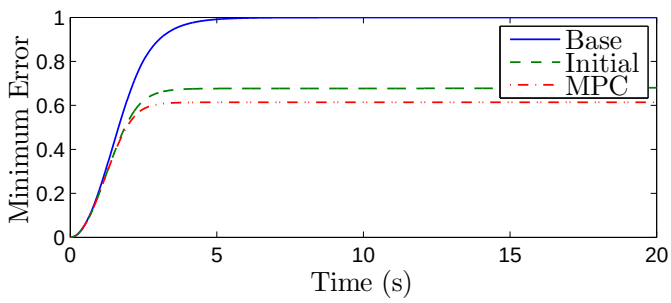


Figure 5.6: Minimum error normalized cost function values: Circular trajectory

5.2.2 Minimum Time Objective

The minimum time control objective determines the sliding mode controller parameters that minimize the reach time

$$\min_{\boldsymbol{p}} J_k = t_{f_k} \tag{5.10}$$

subject to the kinematic and dynamic equality constraints in Equations (3.1) – (3.6), the input saturation constraints in Equation (5.6) and the parameter constraints in Equation (5.7) and the following terminal constraint

$$d(\tau) \leq d_{tol}, \quad \tau \geq t_{f_k} \tag{5.11}$$

where d is the tracking error as defined in Equation (5.9) and $d_{tol} = 0.01$ m. The sliding mode control law $\boldsymbol{u}$ is updated by the model predictive controller at a control interval of $\Delta t = 1$ s and the prediction horizon is $N = 50$.

A comparison of the sliding mode control parameters and the reach time for the base case, the initial optimization and the MPC cascade control structure are summarized in Table 5.2. The initial optimization yields a 31% reduction in reach time over the base case and the MPC cascade structure yields another 12% improvement over the initial optimization. Figure 5.7 presents the resulting USV paths for all three cases and the control action is presented in Figure 5.8. Figure 5.9 shows the control parameter evolution over the simulation period. As was the case for the minimum error objective, the MPC cascade control structure updates the control law such that maximum control input can be maintained until the USV approaches the target. Figure 5.10 presents the tracking error d.

Table 5.2: Minimum time control parameters and reach time: Circular target trajectory

Parameter	Base	Initial Optimal	Cascade MPC
λ_1	1	1.33	$1.33,\ 1.77,\ 1.76,\ 1.76, \ldots$
λ_2	1	0.70	$0.70,\ 0.77,\ 0.74,\ 0.74, \ldots$
η	0.2	0.29	$0.29,\ 0.28,\ 0.27,\ 0.27, \ldots$
k	0.8	1.13	$1.13,\ 1.28,\ 1.27,\ 1.27, \ldots$
Reach Time (s)	7.87	5.42	4.78

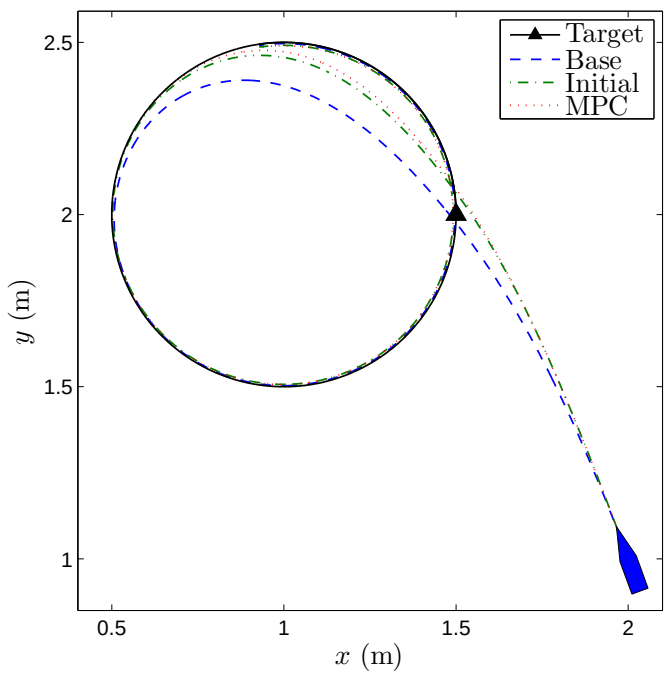


Figure 5.7: Simulated minimum time paths: Circular trajectory

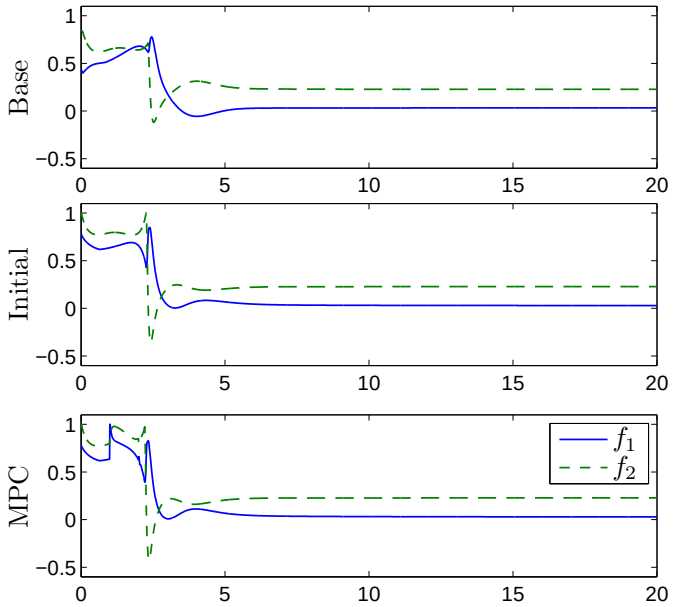


Figure 5.8: Minimum time control input: Circular trajectory

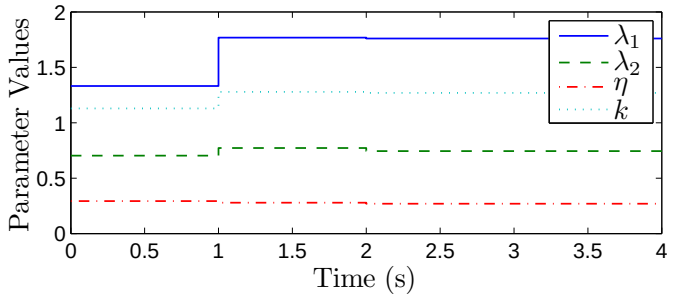


Figure 5.9: Minimum time parameter evolution: Circular trajectory

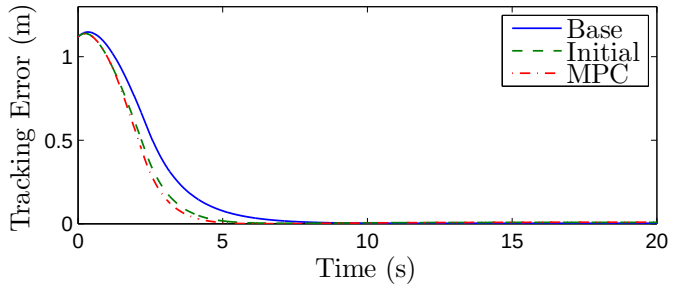


Figure 5.10: Minimum time tracking error: Circular trajectory

5.2.3 Minimum Energy Objective

The minimum energy objective determines the sliding mode controller parameters that minimize the integral squared control effort cost function

$$\min_{\boldsymbol{p}} J_k = \int_{k\Delta t}^{(k+N)\Delta t} (f_1^2 + f_2^2)\, dt \tag{5.12}$$

subject to the reaching time constraint

$$\boldsymbol{g}_2(\boldsymbol{z}) = d_{tol} - d(\tau) \geq 0, \quad \tau \geq t_r \tag{5.13}$$

where t_r is the desired reaching time, and d is the tracking error defined in Equation (5.9). The reaching time constraint is implemented to force the USV to approach the target trajectory within a specified tolerance, $d_{tol} = 0.01$ m, in a desired reaching time $t_r = 20$ s. Without this reaching time constraint, the minimum energy sliding mode controller approaches the target trajectory too slowly or it may never approach the target at all. As before, the optimization is subject to the kinematic and dynamic equality constraints in Equations (3.1) – (3.6), the input saturation constraints in Equation (5.6) and the parameter constraints in Equation (5.7). The sliding mode control law $\boldsymbol{u}$ is updated by the model predictive controller at a control interval of $\Delta t = 1$ s and the prediction horizon is $N = 30$.

Figure 5.11 presents the resulting USV paths for all three cases and the tracking error d is presented in Figure 5.12. A comparison of the sliding mode control parameters, the relative cost and the reach time for the base case, the initial optimization and the MPC cascade control structure are summarized in Table 5.3 and the control parameter evolution is presented graphically in Figure 5.13. The control action is presented in Figure 5.14 and Figure 5.15 presents the normalized energy cost versus time for all three cases. The initial optimization yields a 26% performance gain over the base case but the MPC cascade structure only yields another 4% cost improvement over the initial optimization. Since the inequality constraint on control is not active in this case, the optimal controller parameters

can be implemented from time $t = 0$ and do not change significantly as the simulation progresses. It is interesting to note, however, that the MPC cascade structure decreases the reach time as well as the integral squared effort cost. Figure 5.12 shows that the USV reaches the target within the desired reach time $t_r = 20$ s.

Table 5.3: Minimum energy control parameters and cost: Circular target trajectory

Parameter	Base	Initial Optimal	Cascade MPC
λ_1	1	0.51	$0.51,\ 0.51,\ 0,\ 0, \ldots$
λ_2	1	0.92	$0.92,\ 0.92,\ 2.41,\ 2.41, \ldots$
η	0.2	0.07	$0.07,\ 0.07,\ 0,\ 0, \ldots$
k	0.8	0.30	$0.30,\ 0.30,\ 0.29,\ 0.29, \ldots$
Normalized Cost	1	0.74	0.72
Reach Time (s)	7.87	20	19.34

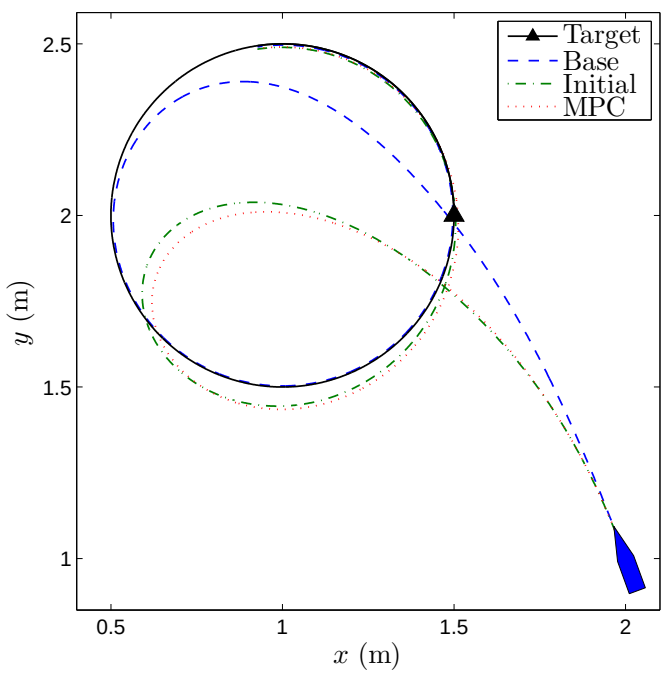


Figure 5.11: Simulated minimum energy paths: Circular trajectory

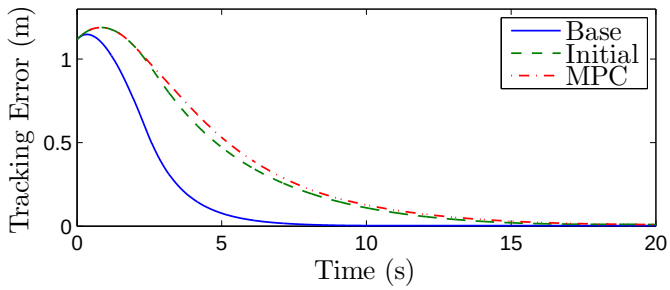


Figure 5.12: Minimum energy tracking error: Circular trajectory

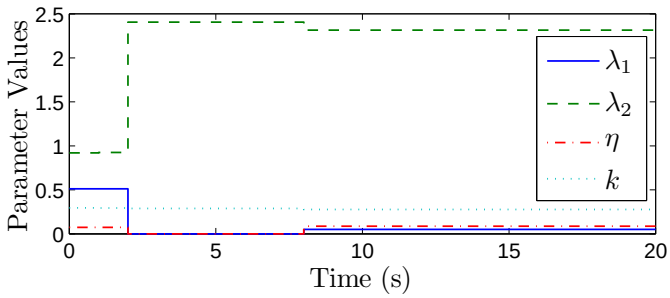


Figure 5.13: Minimum energy parameter evolution: Circular trajectory

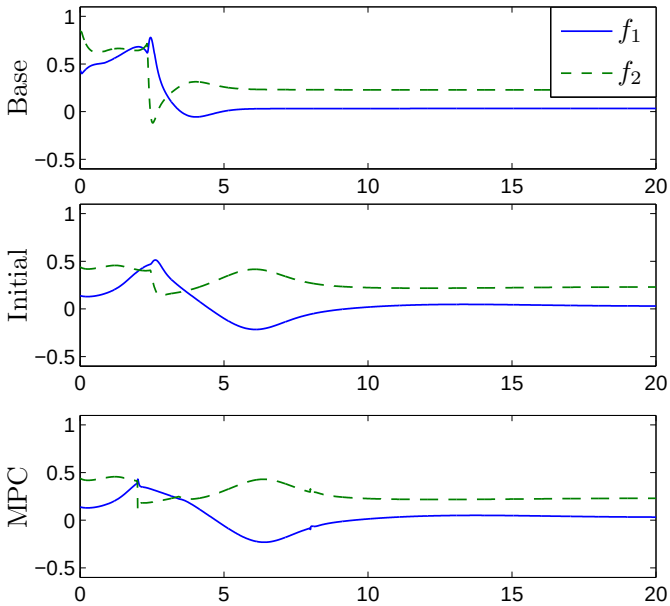


Figure 5.14: Minimum energy control input: Circular trajectory

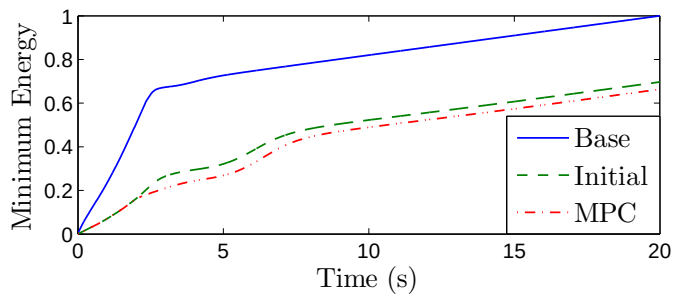


Figure 5.15: Minimum energy objective function values: Circular trajectory

5.3 Linear Target Trajectory

In this section the scenario where the target follows a linear trajectory beginning at $(x = 1.2\text{m}, y = 0.5\text{m})$ and traveling with a constant velocity of 0.2 m/s in the x-direction is considered. The USV begins at the origin and must approach and track the target. The reference trajectory $(v_x^r(t), v_y^r(t))$ is generated using the target tracking trajectory presented in Section 2.2.1. The control parameters $\boldsymbol{p} = [\lambda_1, \lambda_2, \eta, k]^T$ are again selected as the decision variables for the optimization subject to the inequality constraints defined in Equation (5.7). The parameters used for the base case are $\lambda_1 = \lambda_2 = 0.5$, $\eta = 0.1$ and $k = 0.4$.

5.3.1 Minimum Tracking Error Objective

Consider the ITSE objective presented in Equation (5.8) subject to the kinematic and dynamic equality constraints in Equations (3.1) – (3.6), the input saturation constraints in Equation (5.6) and the parameter constraints in Equation (5.7). The sliding mode control law $\boldsymbol{u}$ is updated by the model predictive controller at a control interval of $\Delta t = 1$ s and the prediction horizon is $N = 15$. Figure 5.16 shows the USV paths for the base, initial optimal and MPC cases and the x and y tracking errors are presented in Figure 5.17. A comparison of the sliding mode control parameters and the relative cost for the base case, the initial optimization and the MPC cascade control structure are summarized in Table 5.4 and the control parameter evolution is presented graphically in Figure 5.18. Figure 5.19

presents the normalized tracking error cost versus time for all three cases. The initial optimization yields a 68% performance gain over the base case and the MPC cascade structure yields another 19% cost improvement over the initial optimization. The control action is presented in Figure 5.20.

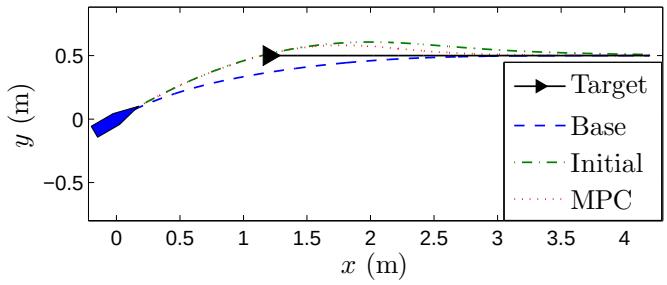


Figure 5.16: Simulated minimum error paths: Linear trajectory

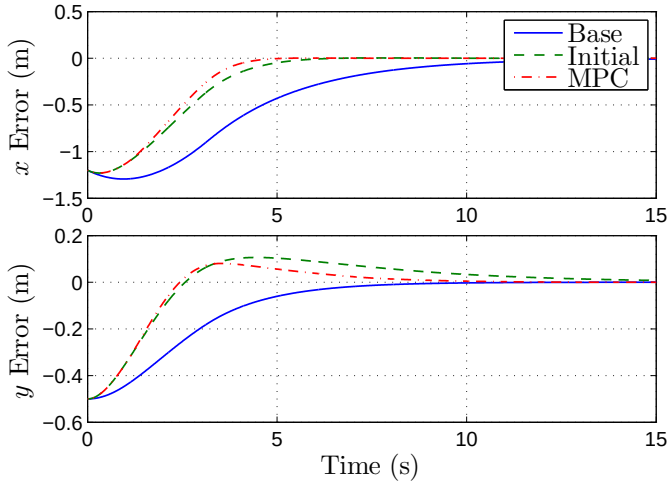


Figure 5.17: Minimum error tracking error: Linear trajectory

Table 5.4: ITSE control parameters and cost: Linear target trajectory, unconstrained

Parameter	Base	Initial Optimal	Cascade MPC
λ_1	0.5	1.38	$1.38,\ 2.06,\ 2.02,\ 1.94, \ldots$
λ_2	0.5	0.59	$0.59,\ 0.95,\ 0.92,\ 0.97, \ldots$
η	0.1	0.05	$0.047,\ 0.11,\ 0.13,\ 0, \ldots$
k	0.4	0.94	$0.94,\ 1.56,\ 1.50,\ 1.54, \ldots$
Normalized Cost	1	0.32	0.26
Reach Time (s)	14.40	14.21	8.70

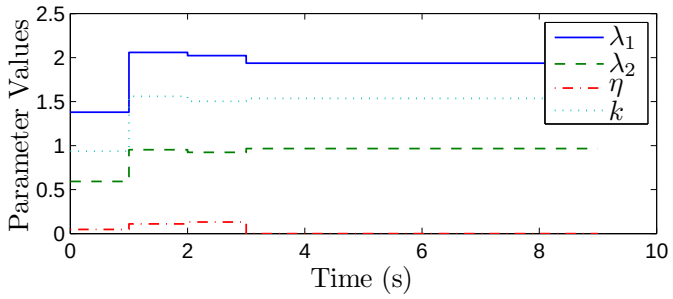


Figure 5.18: Minimum error parameter evolution: Linear trajectory

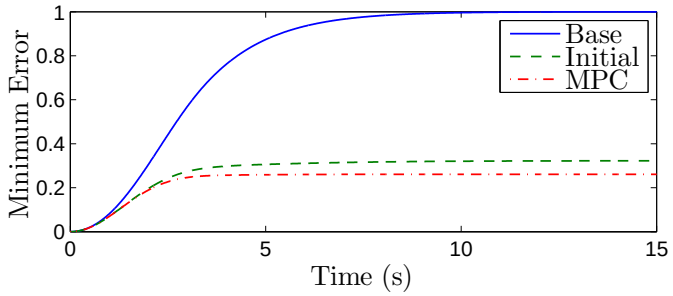


Figure 5.19: Minimum error objective function values: Linear trajectory

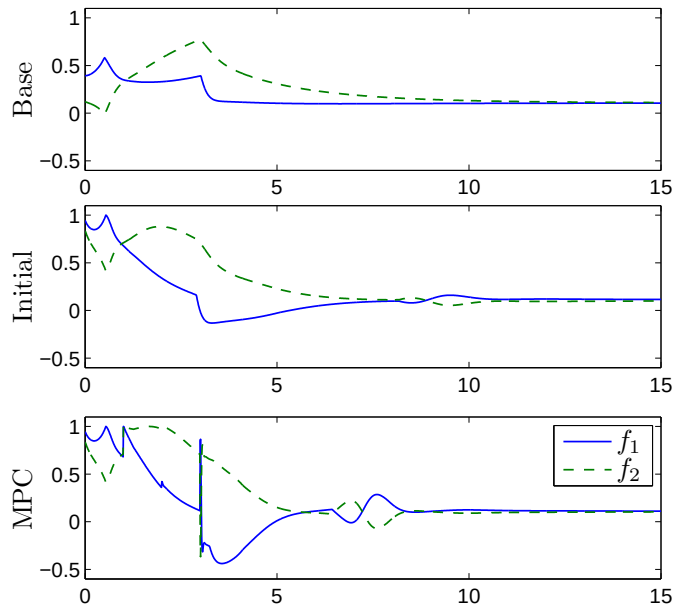


Figure 5.20: Minimum error control input: Linear trajectory

Note that in the previous example, there is overshoot in the y direction in both the initial optimal and MPC cases. This behavior is a result of the ITSE objective function. The total ITSE cost is reduced by moving as quickly as possible in the y direction. As a result, the USV overshoots the target in the y direction and eventually converges from the side. For stern-recovery operations, this behavior may be undesirable. In order to address this issue, an additional constraint is imposed on the optimization

$$\boldsymbol{g}_3(\boldsymbol{z}) = y_{tol} - y(t) \geq 0, \quad t \geq t_0 \tag{5.14}$$

where $y_{tol} = 0.01$ m. By imposing this constraint, y overshoot is eliminated and the USV will approach the target from the stern. Figure 5.21 shows the resulting USV paths and the x and y tracking errors are presented in Figure 5.22. In the initial optimal and MPC cases, the y error converges in about 4 seconds and the USV converges on the target directly from the stern several seconds later. A comparison of the sliding mode control parameters and the relative cost for the base case, the initial optimization and the MPC cascade control structure are summarized in Table 5.5. The control parameter evolution is presented graph-

ically in Figure 5.23 and the resulting control action is presented in Figure 5.24. Figure 5.25 presents the normalized tracking error cost versus time for all three cases. The initial optimization yields a 60% performance gain over the base case and the MPC cascade structure yields another 18% cost improvement over the initial optimization. Although imposing the y constraint increases the total energy cost in both the initial optimal and MPC cases, the reach time is reduced.

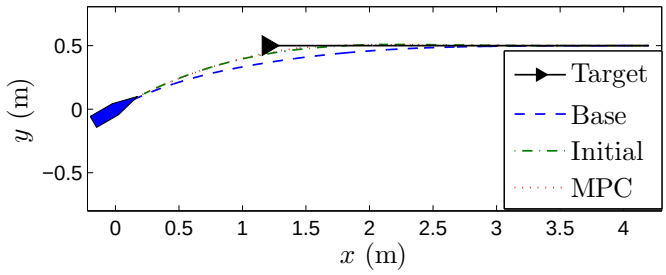


Figure 5.21: Simulated constrained minimum error paths: Linear trajectory

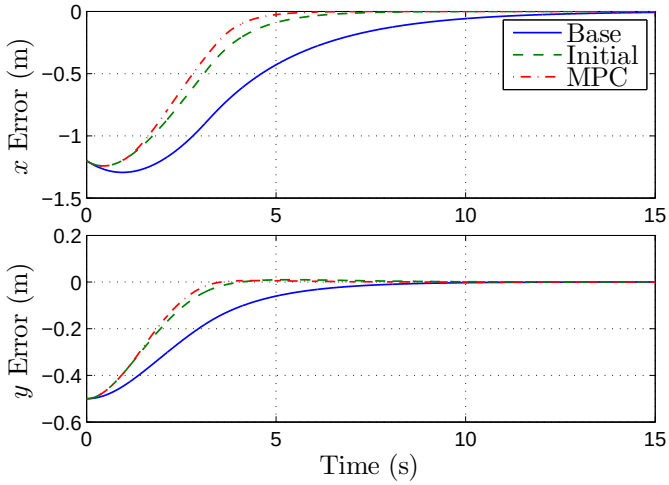


Figure 5.22: Constrained minimum error tracking error: Linear trajectory

Table 5.5: Constrained ITSE control parameters and cost: Linear target trajectory

Parameter	Base	Initial Optimal	Cascade MPC
λ_1	0.5	1.12	1.12, 1.84, 1.84, 1.84, ...
λ_2	0.5	0.80	0.80, 1.20, 1.19, 1.19, ...
η	0.1	0.13	0.13, 0.19, 0.19, 0.19, ...
k	0.4	0.91	0.91, 1.50, 1.48, 1.48, ...
Normalized Cost	1	0.40	0.33
Reach Time (s)	14.40	7.56	5.69

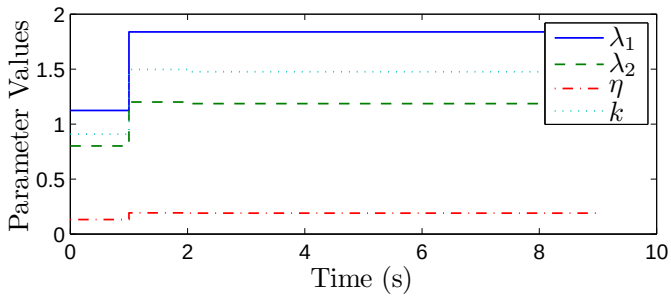


Figure 5.23: Constrained minimum error parameter evolution: Linear trajectory

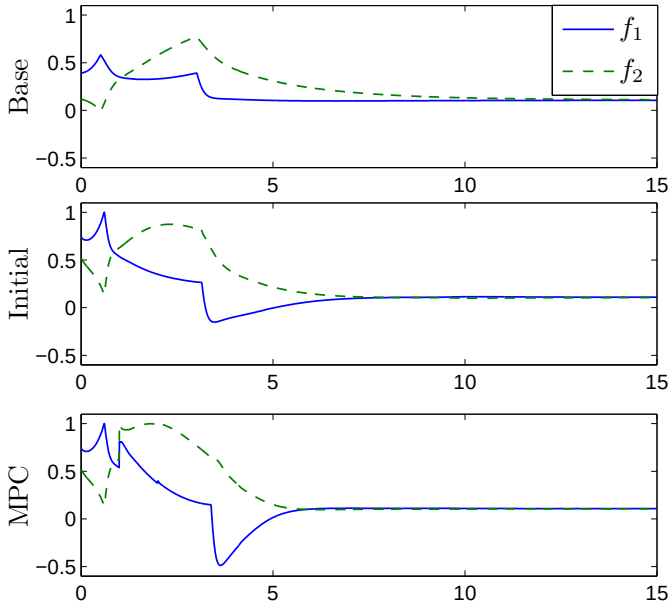


Figure 5.24: Constrained minimum error control input: Linear trajectory

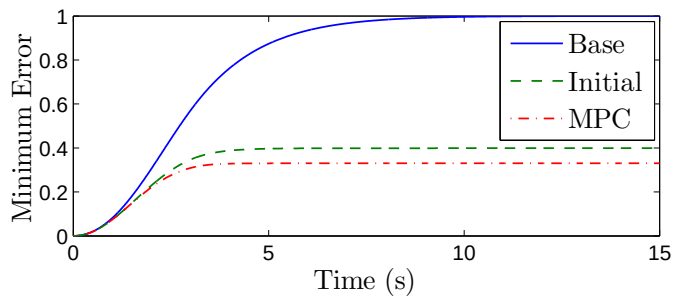


Figure 5.25: Constrained minimum error objective function values: Linear trajectory

5.3.2 Minimum Time Objective

Next, consider the minimum time objective presented in Equation (5.10) subject to the kinematic and dynamic equality constraints in Equations (3.1) – (3.6) and the inequality constraints on the control inputs and decision variables in Equations (5.6) and (5.7) respectively. Figure 5.26 presents the resulting USV paths for all three cases and the x and y tracking errors are presented in Figure 5.27. The control action is presented in Figure 5.28. In this case, it is very clear that the control input is maximized each time the control parameters are updated. A comparison of the sliding mode control parameters and the reach time for the base case, the initial optimization and the MPC cascade control structure are summarized in Table 5.6 and the control parameter evolution is presented in Figure 5.29. The initial optimization yields a 55% reduction in reach time over the base case and the MPC cascade structure yields another 18% improvement over the initial optimization.

Table 5.6: Minimum time control parameters and reach time: Linear target trajectory

Parameter	Base	Initial Optimal	Cascade MPC
λ_1	0.5	1.41	1.41, 2.00, 2.21, 3.18, ...
λ_2	0.5	1.65	1.65, 1.90, 1.99, 1.46, ...
η	0.1	0.41	0.41, 0.21, 0.15, 0.02, ...
k	0.4	1.49	1.49, 1.72, 1.73, 1.98, ...
Reach Time (s)	14.40	6.42	5.25

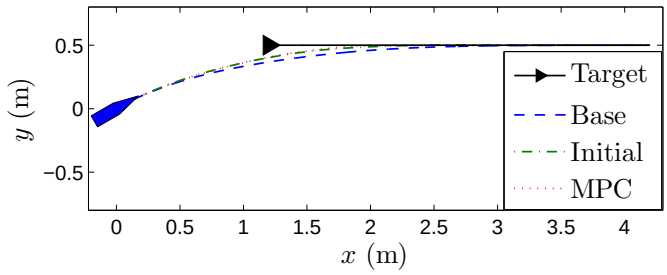


Figure 5.26: Simulated minimum time paths: Linear trajectory

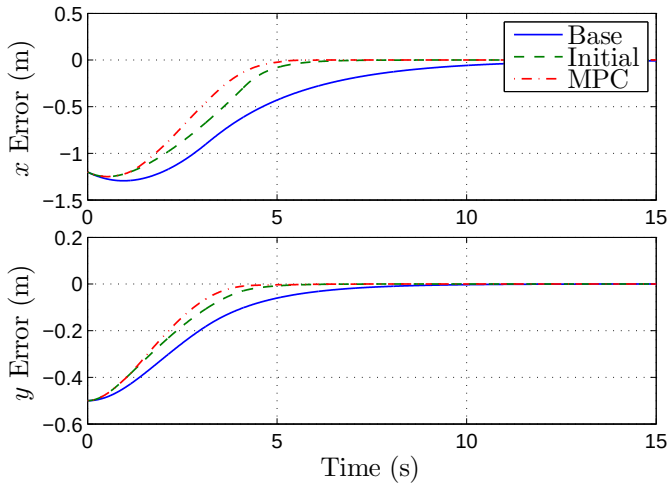


Figure 5.27: Minimum time tracking error: Linear trajectory

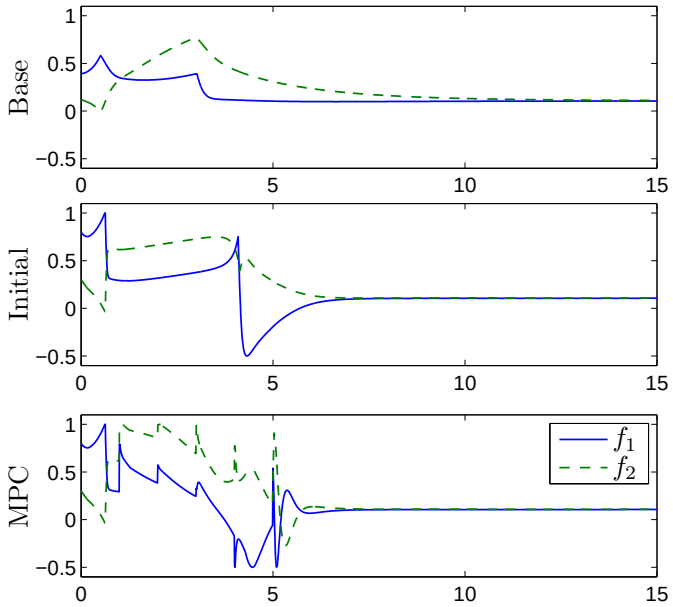


Figure 5.28: Minimum time control input: Linear trajectory

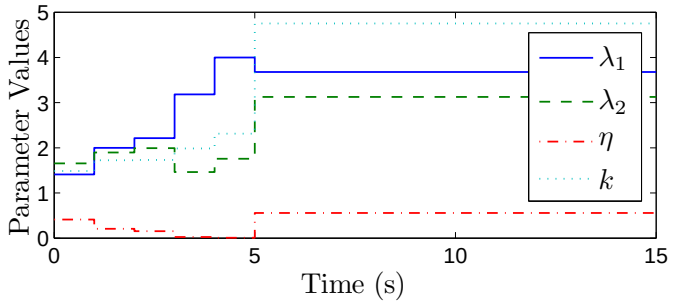


Figure 5.29: Minimum time parameter evolution: Linear trajectory

Chapter 6

Conclusions

In this thesis, several nonlinear control applications for underactuated nonholonomic systems are presented. An overview of nonlinear control and nonholonomic systems is given and sliding mode and model predictive control methods are introduced. A kinematic controller is applied to an experimental wheeled mobile robot for validation of rapid prototyping techniques and new trajectory planning algorithms. Sliding mode controllers are developed for setpoint and trajectory tracking of USVs and experimental and simulation results are presented. A receding horizon model predictive controller is developed for setpoint and tracking control and simulation examples are given. Finally, a predictive sliding mode cascade controller is developed combining the strengths of the sliding mode and model predictive control methods.

6.1 Future Work

6.1.1 Mobile Robots

The controller developed for the experimental mobile robot system in Chapter 2 does not take advantage of the vision system for the purpose of absolute position feedback. Future work in this area would involve communicating the camera-based position feedback to the

mobile robots wirelessly via Bluetooth. A combination of encoder and camera feedback using multi-rate estimation schemes can be employed. This work could also be expanded to include moving obstacles whose position would be determined in real-time by the vision system and transmitted to the mobile robots along with the robots' absolute positions.

6.1.2 Cascade Control

In the area of predictive and sliding mode cascade control, future work will include implementation of the proposed techniques on the experimental USV and mobile robot systems. At the current time, the optimization does not compute quickly enough to implement the controller in real-time. However, the cascade controller parameter sequence may be computed off-line and implemented on the experimental system open-loop.

6.1.3 Trajectory Planning

Future work in the area of trajectory planning and obstacle avoidance will involve optimizing the transition from one trajectory to another. The trajectory planning technique presented in Section 2.2 provides a means to smooth the transition from the initial condition of the mobile agent to the desired trajectory. Future work would involve optimizing these transitions in order to avoid undesirable trajectories and/or infeasible control input requirements.

References

[1] H. Durrant-Whyte. *A Critical Review of the State-of-the-Art in Autonomous Land Vehicle Systems and Technology*. Technical report, Sandia National Laboratories, Albuquerque, 2001.

[2] M. Reyhanoglu, A. van der Schaft, N.H. Mcclamroch, and I. Kolmanovsky. Dynamics and control of a class of underactuated mechanical systems. *IEEE Transactions on Automatic Control*, 44(9):1663–1671, 1999.

[3] N.P.I. Aneke. *Control of Underactuated Mechanical Systems*. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 2002.

[4] I. Kolmanovsky and N.H. McClamroch. Developments in nonholonomic control problems. *IEEE Control Systems Magazine*, 15(6):20–36, 1995.

[5] L. McNinch, R. Soltan, K. Muske, H. Ashrafiuon, and J. Peyton Jones. An experimental mobile robot platform for autonomous systems research and education. In *Proceedings of the 14th IASTED International Conference on Robotics and Applications*, pages 412–418, 2009.

[6] C. Samson. Time-varying feedback stabilization of car-like wheeled mobile robots. *The International Journal of Robotics Research*, 12(1):55–64, 1993.

[7] W. Dixon, Z.P. Jiang, and D. Dawson. Global exponential setpoint control of wheeled mobile robots: A lyapunov approach. *Automatica*, 36:1741–1746, 2000.

[8] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 384–389, 1990.

[9] Z.P. Jiang and H. Nijmeijer. Tracking control of mobile robots: a case study in backstepping. *Automatica*, 33:1393–1399, 1997.

[10] J.M. Yang and J.H. Kim. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, 15(3):578–587, 1999.

[11] Luis Gracia and Josep Tornero. Kinematic control system for car-like vehicles. In *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI*, volume 2527, pages 882–892, London, UK, 2002. Springer-Verlag.

[12] D. Chwa. Sliding-mode tracking control of nonholonomic wheeled mobile robots in polar coordinates. *IEEE Transactions on Control Systems Technology*, 12(4):637–644, 2004.

[13] R. Fierro and F. L. Lewis. Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics. *Journal of Robotic Systems*, 14(3):149–163, 1997.

[14] M. Reyhanoglu. Exponential stabilization of an underactuated autonomous surface vessel. *Automatica*, 33(12):2249–2254, 1997.

[15] T. Kim, T. Basar, and I. Ha. Asymptotic stabilization of an underactuated surface vessel via logic-based control. In *Proceedings of the 2002 American Control Conference*, volume 6, pages 4678–4683, 2002.

[16] F. Mazenc, K. Pettersen, and H. Nijmeijer. Global uniform asymptotic stabilization of an underactuated surface vessel. *IEEE Transactions on Automatic Control*, 47(10):1759–1762, 2002.

[17] D. Soro and R. Lozano. Semi-global practical stabilization of an underactuated surface vessel via nested saturation controller. In *Proceedings of the 2003 American Control Conference*, volume 3, pages 2006–2011, 2003.

[18] K. Pettersen, F. Mazenc, and H. Nijmeijer. Global uniform asymptotic stabilization of an underactuated surface vessel: experimental results. *IEEE Transactions on Control Systems Technology*, 12(6):891–903, 2004.

[19] W. Dong and Y. Guo. Global time-varying stabilization of underactuated surface vessel. *IEEE Transactions on Automatic Control*, 50(6):859–864, 2005.

[20] J. Godhavn. Nonlinear tracking of underactuated surface vessels. In *Proceedings of the 1996 IEEE Conference on Decision and Control*, pages 975–980, 1996.

[21] K. Pettersen and H. Nijmeijer. Tracking control of an underactuated surface vessel. In *Proceedings of the 37th IEEE Conference on Decision and Control*, volume 4, pages 4561–4566, 1998.

[22] G. Indiveri, M. Aicardi, and G. Casalino. Nonlinear time-invariant feedback control of an underactuated marine vehicle along a straight course. In *Proceedings of the 2000 Conference on Maneuvering and Control of Marine Craft*, pages 221–226, 2000.

[23] G. Toussaint, T. Basar, and F. Bullo. Tracking for nonlinear underactuated surface vessels with generalized forces. In *Proceedings of the 2000 IEEE Conference on Control Applications*, pages 355–360, 2000.

[24] P. Encarnacao and A. Pascoal. Combined trajectory tracking and path following: An application to the coordinated control of autonomous marine craft. In *Proceedings of the 2001 IEEE Conference on Decision and Control*, pages 964–969, 2001.

[25] A. Behal, D. Dawson, W. Dixon, and Y. Fang. Tracking and regulation control of an underactuated surface vessel with nonintegrable dynamics. *IEEE Trans. Auto. Contrl.*, 47(3):495–500, 2002.

[26] A. Aguiar and J. Hespanha. Position tracking of underactuated vehicles. In *Proceedings of the 2003 American Control Conference*, pages 1988–1993, 2003.

[27] E. Lefeber, K.Y. Pettersen, and H. Nijmeijer. Tracking control of an underactuated ship. *IEEE Transactions on Control Systems Technology*, 11(1):52–61, 2003.

[28] H. Ashrafiuon, K. Muske, L. McNinch, and R. Soltan. Sliding-mode tracking control of surface vessels. *IEEE Transactions on Industrial Electronics*, 55(11):4004–4012, 2008.

[29] L. McNinch, H. Ashrafiuon, and K. Muske. Sliding mode setpoint control of an underactuated surface vessel: Simulation and experiment. 2010 American Control Conference, submitted.

[30] L. McNinch, K. Muske, and H. Ashrafiuon. Model-based predictive control of an unmanned surface vessel. In *Proceedings of the 11th IASTED International Conference on Intelligent Systems and Control*, pages 385–390, 2008.

[31] L. McNinch, K. Muske, and H. Ashrafiuon. Model predictive control for an autonomous unmanned surface vessel. *Control and Intelligent Systems*, in press.

[32] J. Slotine and W. Li. *Applied Nonlinear Control*. Prentice-Hall, Upper Saddle River, NJ, 1991.

[33] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*, pages 181–191. Birkhauser–Verlag, 1983.

[34] V. Utkin. Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control*, 22(2):212–222, 1977.

[35] H. Ashrafiuon and R. Erwin. Sliding mode control of underactuated multibody systems and its application to shape change control. *International Journal of Control*, 81(12):1849–1858, 2008.

[36] M. Nikkhah, H. Ashrafiuon, and K. Muske. Optimal sliding mode control for underactuated systems. In *Proceedings of the 2006 American Control Conference*, pages 4688–4693, 2006.

[37] D. Mayne, J. Rawlings, C. Rao, and P. Skocart. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[38] S. Qin and T. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11:733–764, 2003.

[39] R. Soltan, H. Ashrafiuon, and K. Muske. Trajectory planning and coordinated control of robotic systems. In *Proceedings of the 33rd ASME Mechanisms and Robotics Conference*, 2009.

[40] R. Soltan, H. Ashrafiuon, and K. Muske. Real-time obstacle avoidance for underactuated unmanned surface vessels. In *Proceedings of the 33rd ASME Mechanisms and Robotics Conference*, 2009.

[41] L. P. Ellekilde and J. Perram. Tool center trajectory planning for industrial robot manipulators using dynamical systems. *The International Journal of Robotics Research*, 24(5):385–396, 2005.

[42] D.H. Kim and J.H. Kim. A real-time limit-cycle navigation method for fast mobile robots and its application to robot soccer. *Robotics and Autonomous Systems*, 42(1):17–30, 2003.

[43] R. Grech and S.G. Fabri. Trajectory tracking in the presence of obstacles using the limit cycle navigation method. In *Proceedings of the 20th IEEE International Symposium on Intelligent Control and the 13th Mediterranean Conference on Control and Automation*, pages 101–106, 2005.

[44] T. Chikamasa. Embedded coder robot NXT instruction manual. www.mathworks.com/ matlabcentral/fileexchange/13399/, 2009.

[45] G. Antonelli, S. Chiaverini, and G. Fusco. A calibration method for odometry of mobile robots based on the least-squares technique: Theory and experimental validation. *IEEE Transactions on Robotics*, 21(5):994–1004, 2005.

[46] T. Fossen. *Guidance and Control of Ocean Vehicles*. Wiley, New York, 1994.

[47] T. Fossen, S. Sagatun, and A. Sorensen. Identification of dynamically positioned ships. *Journal of Control Engineering Practice*, 4(3):369–376, 1996.

[48] R. Skjetne, O. Smogeli, and T. Fossen. A nonlinear ship maneuvering model: Identification and adaptive control with experiments for a model ship. *Modeling, Identification and Control*, 25(1):3–27, 2004.

[49] F. White. *Viscous Fluid Flow*. McGraw-Hill, New York, NY, 3rd edition, 2006.

[50] K. Muske, H. Ashrafiuon, R. McCloskey, G. Haas, and T. Flynn. Identification of a control oriented nonlinear dynamic USV model. In *Proceedings of the 2008 American Control Conference*, pages 562–567, 2008.

[51] V. Sankaranarayanan and A.D. Mahindrakar. Control of a class of underactuated mechanical systems using sliding modes. *IEEE Transactions on Robotics*, 25(2):459–467, 2009.

[52] V. T. Haimo. Finite time controllers. *SIAM Journal on Control and Optimization*, 24(4):760–770, 1986.

[53] S.P. Bhat and D.S. Bernstein. Continuous finite-time stabilization of the translational and rotational double integrators. *IEEE Transactions on Automatic Control*, 43(5):678–682, 1998.

[54] H. Khalil. *Nonlinear Systems*. Prentice-Hall, Upper Saddle River, NJ, 1996.

[55] A. Whitman, H. Ashrafiuon, and K. Muske. Analytical stability analysis of surface vessel motion for a control-oriented model. Journal of Computational and Nonlinear Dynamics, submitted.

[56] A. Bryson and Y. Ho. *Applied Optimal Control*. Taylor & Francis, Levittown, PA, revised edition, 1975.

[57] R. Cagienard, P. Grieder, E. Kerrigan, and M. Morari. Move blocking strategies in receding horizon control. *Journal of process control*, 17(6):563–570, 2007.

[58] H. Ashrafiuon and K. Muske. Trajectory tracking control for recovery of unmanned surface vessels. In *Proceedings of the 2008 American Society of Naval Engineers Launch and Recovery Symposium*, Paper H1, 2008.

[59] L. McNinch, H. Ashrafiuon, and K. Muske. Optimal specification of sliding mode control parameters for unmanned surface vessel systems. In *Proceedings of the 2009 American Control Conference*, pages 2350–2355, 2009.

Appendix A

Vision System

The vision feedback systems in this work are used to determine the location of LEDs attached to autonomous agents in order to provide position and orientation feedback for the setpoint or tracking controllers. The system is also used for trajectory data collection for system identification tests. The vision system functions by capturing an image of the experimental work area (the pool surface in the case of USV experiments or the ground in the case of mobile robot experiments). The image is then processed to determine the pixel location of LEDs mounted on the USV or mobile robot. The real-world location of each LED is then interpolated using calibration data. This vision feedback technique therefore presents several basic problems. An algorithm is required for searching the image for groups of pixels making up the LEDs. Next, a grid of known calibration points must be constructed and the pixel location of each grid point must be found. Finally, an interpolation scheme must be developed allowing pixel coordinates to be converted to real-word coordinates. First, however, the interface between MATLAB and the hardware must be established.

A.1 Hardware and MATLAB Interface

Two vision systems are used in this work. The system used for mobile robot experiments consists of a black and white digital video camera attached to a Data Translation DT3120

Frame Grabber card installed in the host PC. This system provides a 640×480 matrix of 8-bit unsigned integer pixel intensity values (`uint8` data type) ranging from 0 to 255. With an infrared filter on the lens and the iris of the lens closed down to further restrict the light entering the camera, the infrared LEDs mounted on the mobile robot are the brightest object in the image. The MATLAB video input object for this frame grabber is set up by issuing the commands shown in Listing A.1. All existing video objects are deleted, the video object `vid` is defined and several properties of the object are modified. First, the `FramesPerTrigger` option is set to 1 and the trigger configuration is set to manual. As a result, a single frame is captured each time the trigger is issued. The option `TriggerRepeat` is set to `Inf` so that trigger can be executed an unlimited number of times.

Listing A.1: Camera setup

```
delete(imaqfind);
vid = videoinput('dt',1,'RS170');
vid.FramesPerTrigger = 1;
triggerconfig(vid,'manual')
vid.TriggerRepeat = Inf;
```

The vision system used in the USUV Lab consists of an AVT Guppy F-080C color digital video camera with IEEE 1394 Firewire interface and 1032×778 pixel resolution. The image is interpreted by MATLAB in RGB format. Each image therefore consists of 3 1032×778 `uint8` matrices where each matrix corresponds to red, green or blue. The video input object for this camera is set up in MATLAB as shown in Listing A.2. The option `ReturnedColorSpace` is set to 'bayer' and the option `BayerSensorAlignment` is set to 'rggb'. These options define the configuration of the color filter on the photosensor in the AVT Guppy camera. The option `ROIPosition` defines the region of interest, including only the surface of the pool in the captured image. The trigger is configured as above and the camera gain is set manually. If the gain is not set manually, the brightness of the image will be adjusted automatically which may disrupt the contrast between the LEDs and the rest of the environment.

Listing A.2: Color camera setup

```
delete(imaqfind);
vid = videoinput('dcam');
vid.ReturnedColorSpace = 'bayer';
vid.BayerSensorAlignment = 'rggb';
vid.ROIPosition = [60 40 960 720];
vid.FramesPerTrigger = 1;
triggerconfig(vid,'manual')
vid.TriggerRepeat = Inf;
src = getselectedsource(vid); % Device Specific properties
src.Gain = 500;
```

A.2 Image Processing

This section outlines the image processing technique used for the color vision system. The technique is the same for the black and white vision system used for the mobile robot experiments except that, in the latter case, only one grayscale matrix is processed each frame as opposed to the three matrices that make up the RGB color image.

Figure A.1 shows a sample camera image containing two USVs. Each USV has a blue LED mounted at the rear and a red LED mounted near the front. (Obstacles use green LEDs.) The image processing algorithm begins by finding all red, blue and green pixels above a given intensity threshold, as shown in Figure A.2(a). Next, the algorithm isolates groups of pixels and averages their row and column positions to find the center of each group. Note that there are frequently several pixels above the red threshold in the blue LEDs and vice versa. In many cases, the center pixels of an LED may be saturated in all three colors and appear white. In this case its is not possible to determine the primary color of the individual pixel. The image processing algorithm therefore must find all groups of pixels of each color and determine the true color of each group based on the average intensities of a small area surrounding the center of the group. For example, if an $n \times n$ pixel area around a potential red point has a higher average blue intensity than red, that red point is ignored. The location of the resulting group centers are shown in Figure A.2(b)

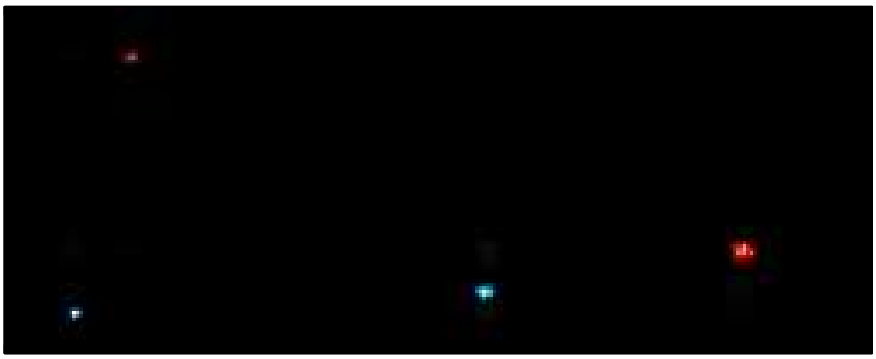
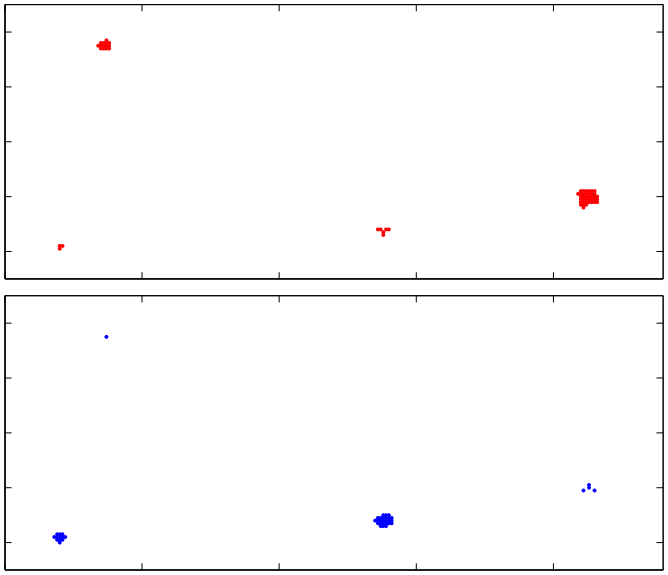
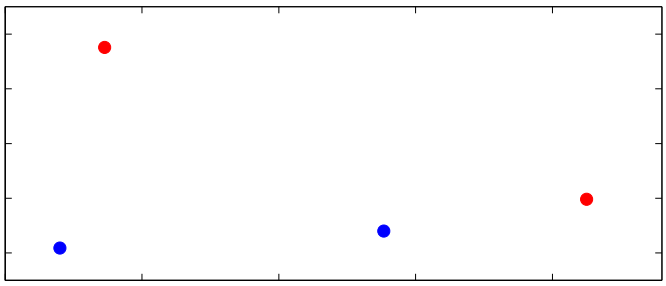


Figure A.1: Sample camera image



(a) Red and blue pixels above the threshold



(b) LED center locations

Figure A.2: Image processing

A.3 Calibration

The origin of the camera frame is the top left corner of the image with columns being denoted by i and rows denoted by j, as shown in Figure A.3. The origin of the world reference frame is near the top left corner of the pool with the x axis in the direction of increasing j and the y axis in the direction of increasing i, as shown in the figure.

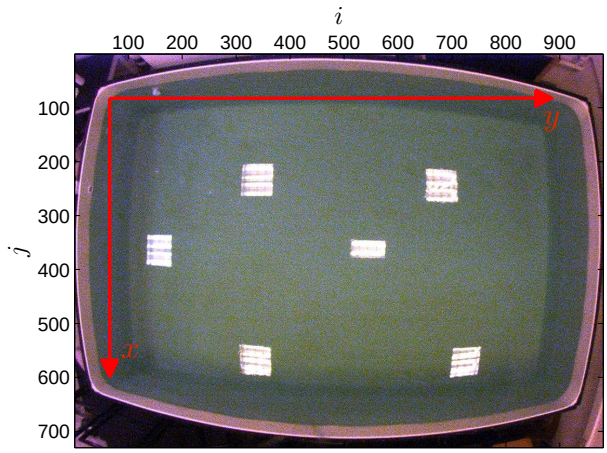


Figure A.3: USV camera field of view

Calibration is required to relate the pixel location of a point in the camera image to its actual location in a world reference frame. In order to achieve this, a grid of points with known spacing is created. For the mobile robot experimental system, the infrared filter is temporarily removed from the camera. Light colored stickers are applied at equally spaced intervals on a black bar and the bar is placed above the ground at approximately the height of the LEDs mounted on the mobile robots. An image is captured and processed using the technique presented in Section A.2. This process is repeated along the length of the field of view of the camera to create a square grid relating pixel row and column in the camera frame to known coordinates in the world reference frame. A similar process is followed for the USV experimental system. In this case, however, the line of points is created using LEDs since the background can not be blacked-out sufficiently. The camera and calibration lights are shown in Figure A.5. The resulting grid is shown in Figure A.6. Note the axial distortion imposed by the relatively wide-angle lens.

Figure A.4: Mobile robot camera and calibration bar

Figure A.5: USV camera and calibration lights

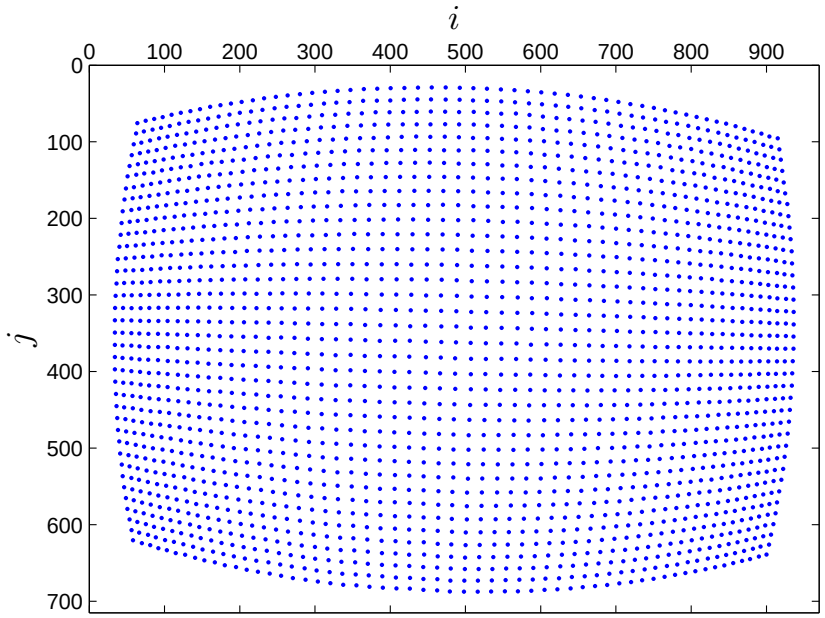


Figure A.6: Camera calibration grid

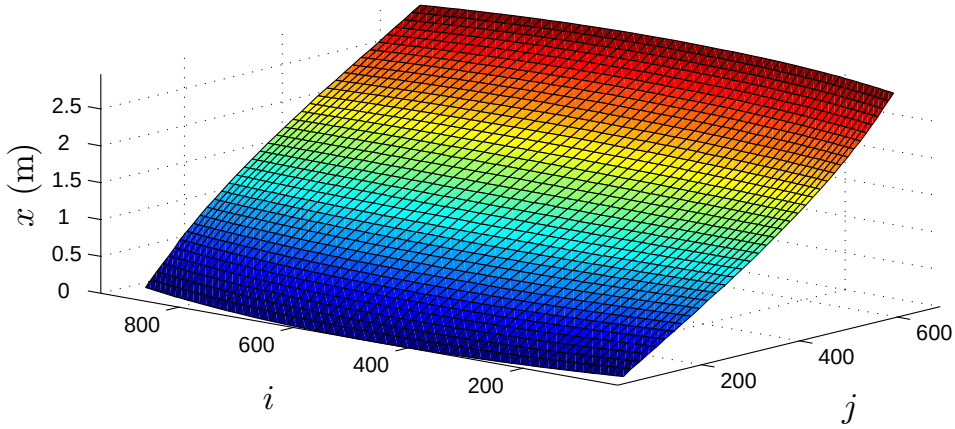


Figure A.7: x surface

A.3.1 2D Linear Interpolation

This section demonstrates the process of interpolating the x and y position of an LED in the world reference frame. The image processing algorithm presented in Section A.2 is used to determine the pixel location of each LED in the field of view. Given these pixel locations, a 2-dimensional linear interpolation is used to calculate the x and y location of each LED based on the calibration data. The surface in Figure A.7 shows the x position of the calibration grid points as a function of pixel location. A similar surface exists for the y position.

Figures A.8 and A.9 show an example interpolation of the x position of camera point $i_p = 75$, $j_p = 95$ represented by the blue $\times$ in the figures. The interpolation algorithm must first locate the four grid points that define the cell surrounding (i_p, j_p). Next, 1D interpolations in the i direction are used calculate the j position of the intermediate points x_1 and x_2. Because all x values are equal along any given row of the grid, $x_1 = x_{11} = x_{12}$ and $x_2 = x_{21} = x_{22}$. Also, the i position of both points is i_p. Hence, the j positions of these points are interpolated using Equations (A.1) and (A.2). Finally, a 1D interpolation in the j direction given by Equation (A.3) is used to calculate the x position of the point represented by the black circle. Interpolation of the y position requires a similar procedure.

$$j_1 = j_{11} + (i_p - i_{11})\frac{j_{12} - j_{11}}{i_{12} - i_{11}} \tag{A.1}$$

$$j_2 = j_{21} + (i_p - i_{21})\frac{j_{22} - j_{21}}{i_{22} - i_{21}} \tag{A.2}$$

$$x = x_1 + (j_p - j_1)\frac{x_2 - x_1}{j_2 - j_1} \tag{A.3}$$

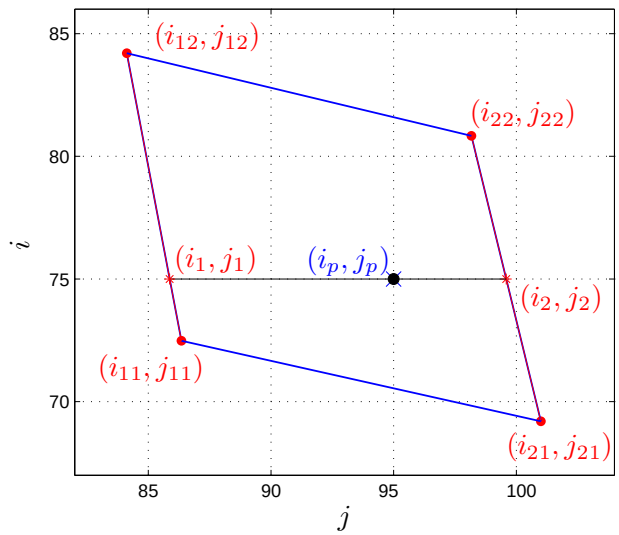


Figure A.8: Two dimensional interpolation: Top view

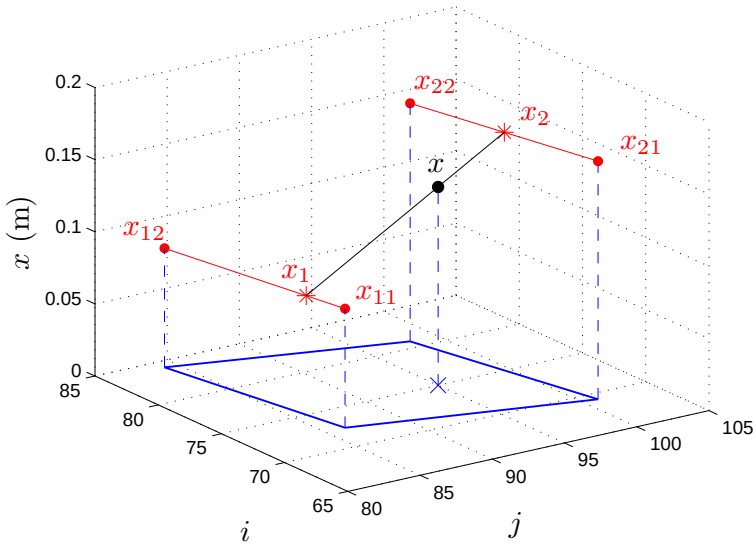


Figure A.9: Two dimensional interpolation

Appendix B

Model USV

The model USVs used in the USUV Lab are modified radio controlled boats. The original motors have been replaced with high quality DC gearmotors with integrated encoders. At the time of this writing, two types of motors are used; Maxon and MicroMo. Motor data is presented in Table B.1. The MicroMo motors are manufactured by Faulhaber. In the documentation for these motors and on the motor itself they are denoted by the name Faulhaber. The motors are controlled by LEGO NXT controllers. Development of the embedded motor control software is presented in Appendix C.

Table B.1: USV Motor Data

	Model	Encoder		Gearset	
		Model	Count	Model	Ratio
Maxon	110055	201938	32	110321	4.4:1
MicroMo	1516T009SR	IE2-512	512	16/7	3.71:1

The original pin-out diagrams for each motor/encoder are shown in Figure B.2. The 6-pin connectors on the MicroMo motors have been replaced with 10-pin connectors matching that of the Maxon, however, the pin functions given in Figure B.2(b) are still valid. Therefore, although the connectors are the same, **the pin-outs for the MicroMo and Maxon motors are different**. Pigtails which connect the 10-pin ribbon cable connectors to the LEGO motor I/O connectors have been constructed. It is important to note that

these pigtails are wired differently for each type of motor.

The LEGO wiring color code is presented in it Table B.2 along with the corresponding pin numbers for the MicroMo and Maxon motors. The pin numbers of the LEGO connector are shown in Figure B.3. Note the colors listed correspond to the colors of the conductors in the LEGO cables and do not correspond to the ribbon cable colors.

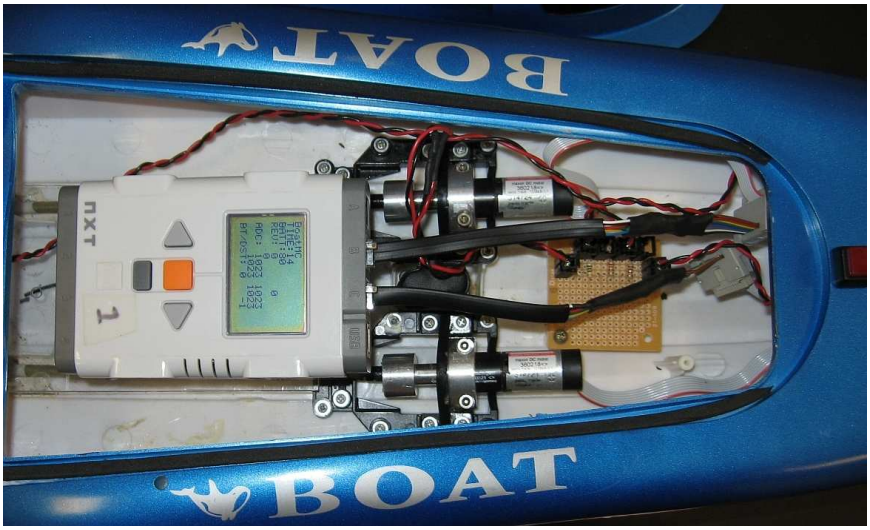
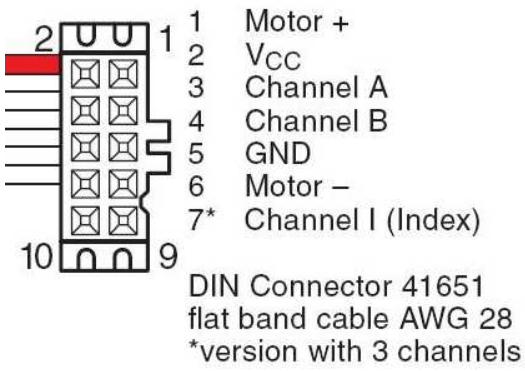
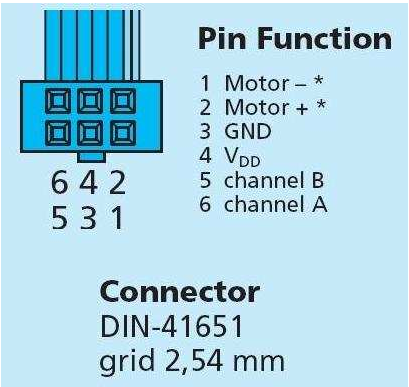


Figure B.1: Model USV internals



(a) Maxon encoder connector



(b) MicroMo encoder connector

Figure B.2: USV motor connector pin-out diagrams

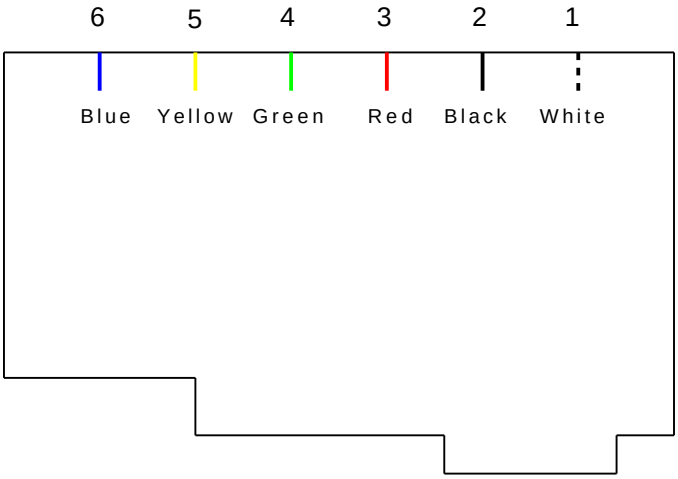


Figure B.3: LEGO connector pin-out diagram

Table B.2: LEGO color code

LEGO	Color	MicroMo	Maxon	Function
1	White	1	6	Motor −
2	Black	2	1	Motor +
3	Red	3	5	Ground
4	Green	4	2	Power
5	Yellow	5	4	Channel B
6	Blue	6	3	Channel A

Appendix C

LEGO NXT and ECRobot

Embedded software for the LEGO NXT is developed in the MATLAB/Simulink programming environment using the Embedded Coder Robot (ECRobot) rapid prototyping development tools. This tool set includes the Simulink blockset and software required to build an executable 'rxe' file which can be downloaded and executed on the NXT. This appendix presents an overview of the process of implementing embedded software on the NXT platform. More detailed instructions, including installation instructions for the various required programs, are presented in [44].

C.1 Mobile Robot Embedded Software

In this section, a Simulink model designed for trajectory tracking using encoder feedback is explained. The main model is shown in Figure C.1. At the top of the model is the Exported Function-Calls Scheduler which generates function-call events according to the user specified periodical schedule. In other words, this block defines the discrete sample rate for each function-call subsystem in the block titled 'LC1'. These subsystems, shown in Figure C.2, contain all of the necessary code for mobile robot operation. The title of this block defines the name of the executable which will be created when the model is built. The middle of the model contains interface blocks for each I/O device that is used in the

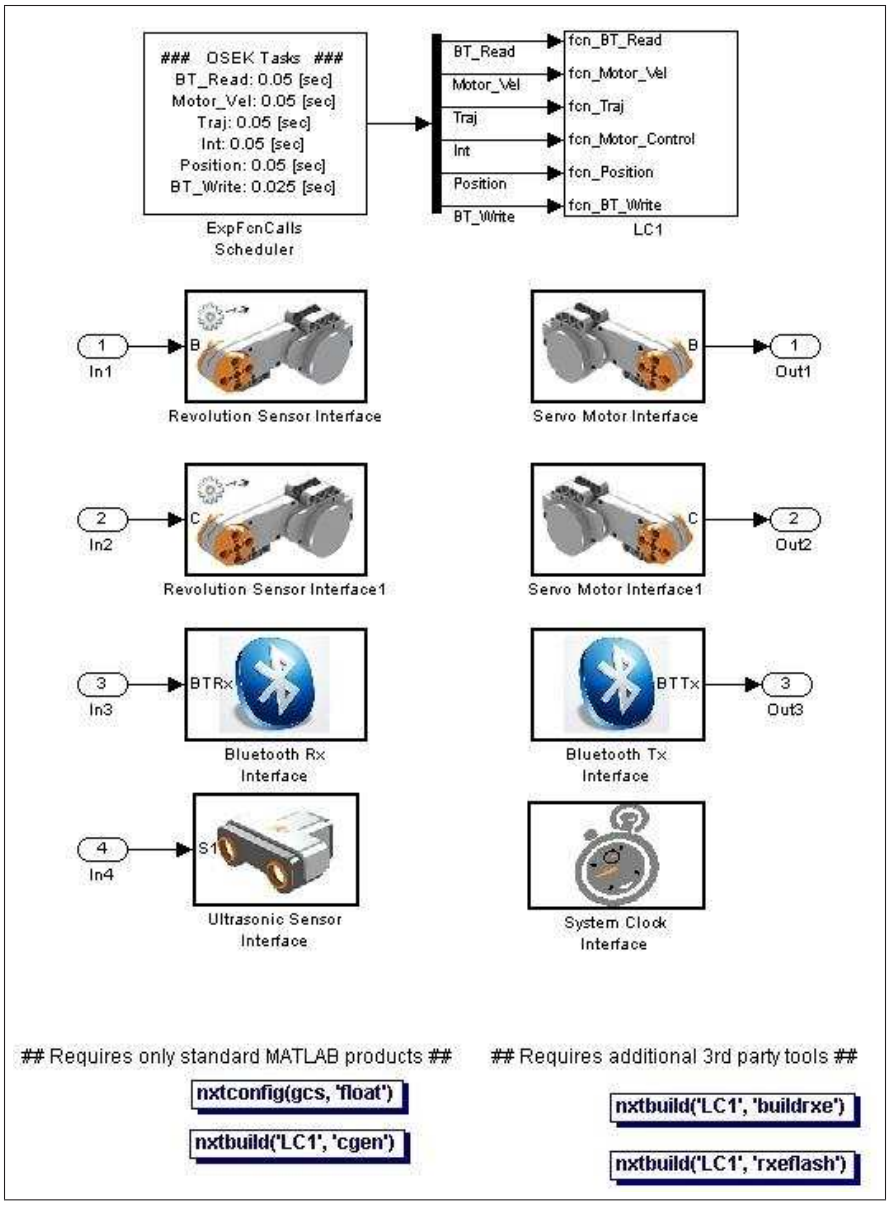


Figure C.1: ECRobot main model

function-call subsystems of the model.

The annotations at the bottom of the model contain MATLAB commands which, when clicked, execute m-files `nxtconfig` or `nxtbuild` with the given input arguments. The `nxtconfig` function simply configures the Simulink simulation configuration parameters for real-time workshop. In this model, the argument 'float' is given allowing the use of floating-point numbers which are required for the trajectory planning and feedback control

calculations. The `nxtbuild` function builds all of the necessary code and generates the ‘rxe’ executable file and downloads it to the NXT. The first argument of this function must be the name of the subsystem containing all of the code to be compiled (‘LC1’ in this case). The argument ‘buildrxe’ tells the function to build the executable file and the argument ‘rxeflash’ downloads the rxe to the NXT. The NXT must be on, running the LEGO firmware and connected to the PC via USB and the NXT USB drivers must be installed.

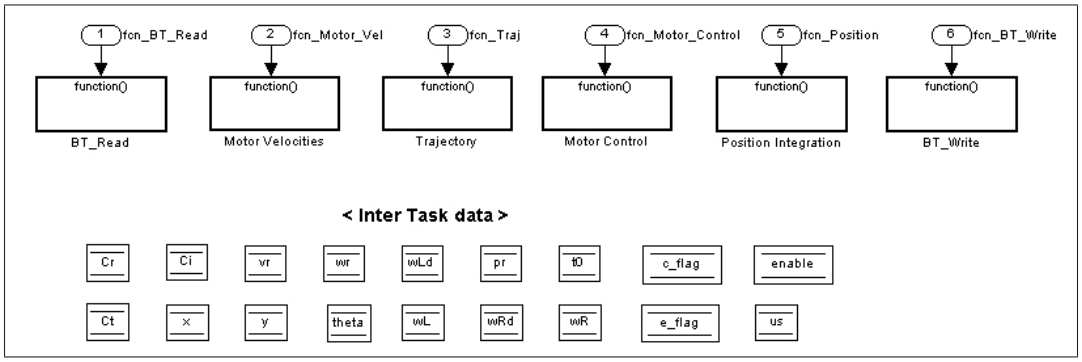


Figure C.2: ‘LC1’ Subsystem containing function-call subsystems

The main functionality of this model lies in the ‘LC1’ subsystem shown in Figure C.2. An important feature of this subsystem is the declaration of the inter-task data stores. These data stores allow data to be shared between all subsystems in the model. The function-calls generated by the Function-Calls Scheduler in the main model trigger the function-call subsystems shown in Figure C.2. The ‘Motor Velocities’ subsystem estimates the motor velocities using encoder position data. These velocities are stored in data stores to be used by other subsystems. The ‘Trajectory’ subsystem calculates the desired reference trajectory and the ‘Position Integration’ subsystem calculates the current position of the robot using the motor velocity data. The ‘Motor Control’ subsystem calculates the mobile robot kinematic control law using the reference trajectory and current position data. The output of the kinematic control law is the desired wheel speed setpoints which are maintained by PID wheel speed controllers, also in the ‘Motor Control’ subsystem. The ‘BT_Read’ and ‘BT_Write’ subsystems handle Bluetooth communication with the host PC. The following sections explain each Bluetooth subsystem in detail.

C.1.1 Bluetooth Read

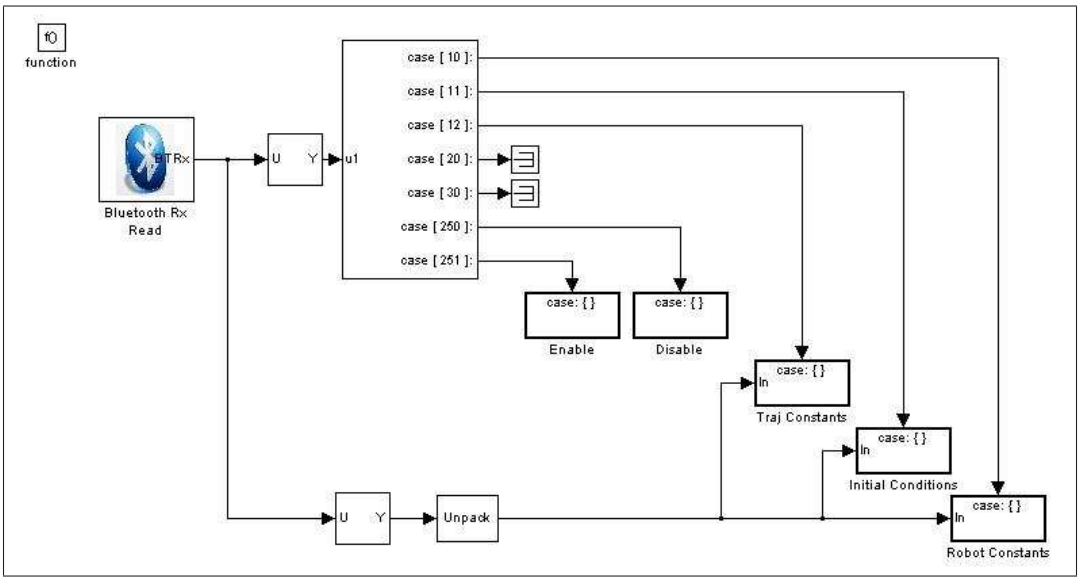


Figure C.3: Bluetooth Read subsystem

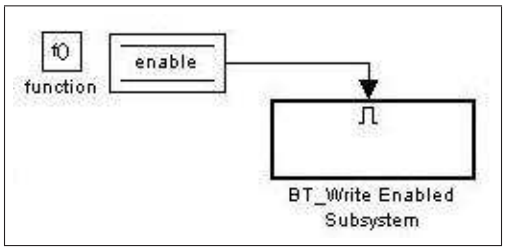
The contents of the 'BT_Read' function-call subsystem are presented in Figure C.3. This subsystem handles the Bluetooth input. Packets of data, 32 bytes each, are sent to the Bluetooth receiver in the NXT using the MATLAB command prompt (see Section C.3). The output of the 'BTRx' block contains a size 32 vector of unsigned 8-bit integers which is updated at each function-call interval. If new data is not available, the output remains the last packet that was received. The first element of the packet identifies the meaning of the remaining data in the packet and is input to the switch/case block. The remainder of the packet is sent to the appropriate case subsystem based on the switch input. Since the data is typically floating-point, it must be converted from `single` precision to `uint8` using the `typecast` command in MATLAB before transmission. Single precision variables use 4 bytes and therefore use 4 of the `uint8` packet elements. Since one of the elements must be used to identify the packet, only 7 `single` values can be transmitted in each packet. In Simulink, the data is converted back to the `single` datatype using the unpack block which is a masked Embedded MATLAB file that again uses the `typecast` command to convert the `uint8` data back to `single`. Once all of the necessary data has been transmitted to

the NXT, the ‘Enable’ data store can be set by sending 251 in the first byte of the packet. ‘Enable’ is set back to zero by sending 250.

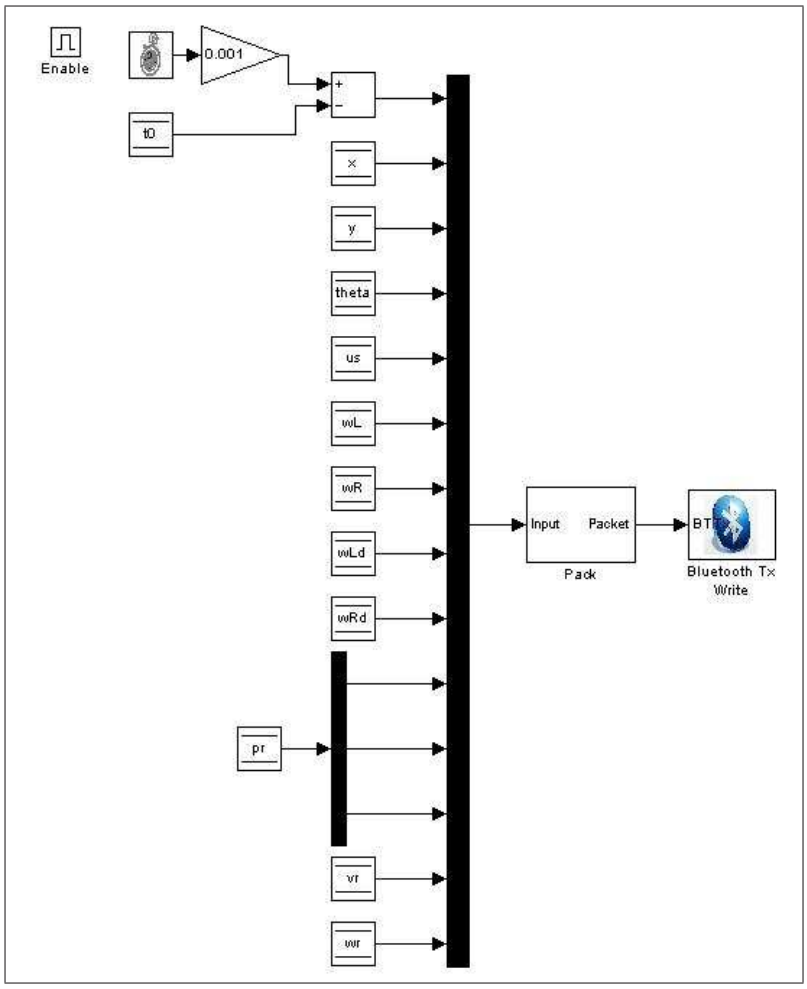
C.1.2 Bluetooth Write

The ‘BT_Write’ function-call subsystem is shown in Figure C.4(a). This subsystem transmits data from the appropriate data stores to be received by a MATLAB script executing on the host PC throughout the experiment. The function-call subsystem only contains the ‘BT_Write’ Enabled Subsystem. The enabled subsystem only functions when the value of the data store ‘Enable’ is 1. This is necessary because every time a subsystem containing the ‘BTTx’ block is triggered, a packet of data is sent to the Bluetooth memory buffer. If this subsystem is active before the experiment has begun, the buffer will fill with data which occurred before the experiment began.

The contents of the Enabled Subsystem is shown in Figure C.4(b). The ‘Pack’ block is another masked Embedded MATLAB file which sorts the input into 32 byte packets. If the input to this block is more than 32 bytes, multiple packets will be transmitted. This function-call subsystem must be triggered n-times faster than the controller and trajectory subsystems where n is the number of data packets required. In this way, all the data can be transmitted in a single sample interval of the controller.



(a) BT_Write function-call subsystem



(b) BT_Write enabled subsystem

Figure C.4: Bluetooth Write subsystems

C.2 Bluetooth Adapter

Bluetooth communication is handled on the host PC by a D-Link DBT-120 model USB Bluetooth adapter. Once the drivers and 'Bluetooth Manager' are installed, a connection profile must be set up for each NXT. The Bluetooth Manager control panel is shown in Figure C.5.

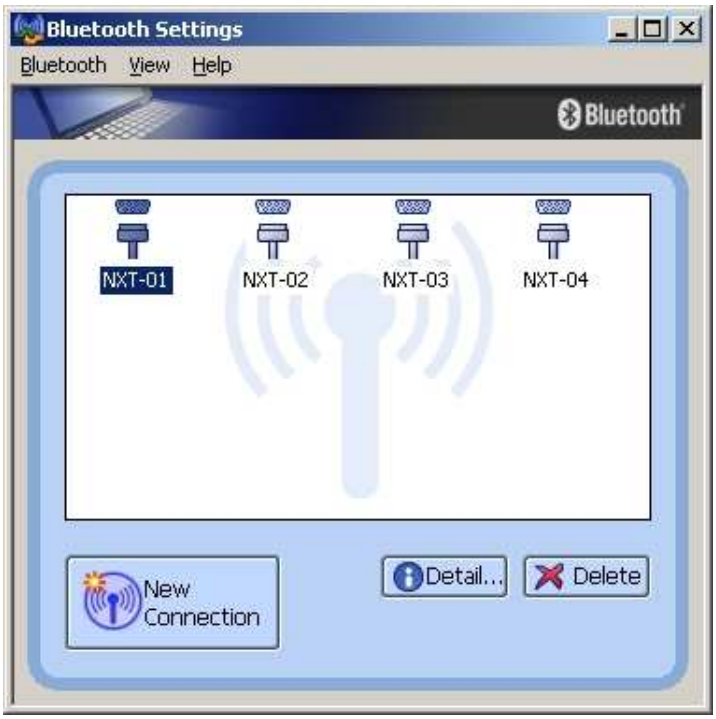


Figure C.5: Bluetooth manager control panel

In order to set up a new connection profile, click 'Add a New Connection' under the 'Bluetooth' menu. Select 'Custom Mode' and click 'Next'. Select the correct NXT device and click 'Next'. The NXT should prompt you to enter a passkey at this time. Once you enter the passkey on the NXT, the Bluetooth Manager will prompt for the same passkey. Enter the passkey and click 'Next' using the default service. Un-select 'Use default COM port' and choose the desired port number. For the NXT named NXT-01 select COM41. For NXT-02 select COM42, etc. This numbering scheme is not required but provides consistency. Click 'Next' and complete the setup.

C.3 Host PC Communication

C.3.1 Serial Port Connection

Once the Simulink model is built, the executable is downloaded to the NXT and the Bluetooth profile has been set up for the correct NXT in the Bluetooth Manager, the Bluetooth connection must be set up in the MATLAB environment. This is accomplished by creating a 'w32serial' object and using the `fopen` command to open the communication channel, as shown in Listing C.1. These commands execute m-files provided with the ECRobot package and are not standard MATLAB commands, though similar commands exist in the MATLAB library. The first argument of the `w32serial` command defines the COM port that is to be connected. Each NXT is associated with a specific port. These port associations are set up in the Bluetooth Manager as shown in Section C.2. Multiple NXT devices can be connected by changing the variable name `s` and the com port number in commands shown in Listing C.1. The variable name becomes the handle for each NXT and is used when transmitting or receiving data.

Listing C.1: MATLAB Bluetooth connection

```
s = w32serial('COM41','BaudRate',128000);
fopen(s);
```

C.3.2 MATLAB Bluetooth Communication

As stated in Section C.1.1, the NXT is capable of receiving individual 32 byte packets of data. On the MATLAB end, however, a size 34 vector of datatype `uint8` is created. The first and second elements of the vector must be 32 and 0 respectively. The remaining 32 elements make up the vector that is received by the 'BTRx' block in Simulink. Unfortunately, the transmitted packet can only be made up of 8-bit unsigned integers. If signed integers or floating-point numbers are required, the data must be typecast in the `uint8` datatype. For

most cases, single precision is acceptable. Single precision variables require four bytes and therefore will each use four elements of the 32 byte packet. Listing C.2 shows an example MATLAB script that creates the packet and sets the first transmitted element (to 23 in this case) which identifies the packet to the `BT_Read` subsystem in the Simulink model. The data to be transmitted (`data`) is converted to `single`, typecast in a four element `uint8` variable `packo`, then substituted into the 4th through the 7th elements of the data packet. The `fwrite` command is then issued. The arguments of `fwrite` are the `w32serial` handle, `s`, the 34 element packet, '`uint8`' to specify the datatype being transmitted and the total size of the packet, 34. A zero packet is then sent (lines 10 and 11). Since the output of the 'BTRx' block in the Simulink model is always the last packet received by the NXT, its output will only be non-zero for one function-call. This will prevent the case 23 subsystem from executing more than once.

Listing C.2: MATLAB Bluetooth write

```
1  packet = uint8(zeros(1,34));
2  packet(1) = 32;
3  packet_identifier = 23;
4  packet(3) = packet_identifier;
5  data = single(4.235);
6  packo = typecast(data,'uint8');
7  packet(4:7) = packo;
8  fwrite(s,packet,'uint8',34);
9
10 packet = uint8([32 zeros(1,33)]);
11 fwrite(s,packet,'uint8',34);
```

C.4 USV Motor Control Embedded Software

ECRobot is also used to generate the motor speed control software for the USV. Figure C.6 shows the main Simulink model. Like the mobile robot controller, this model contains the Exported Function-Calls Scheduler and the interface blocks for the motors and encoders.

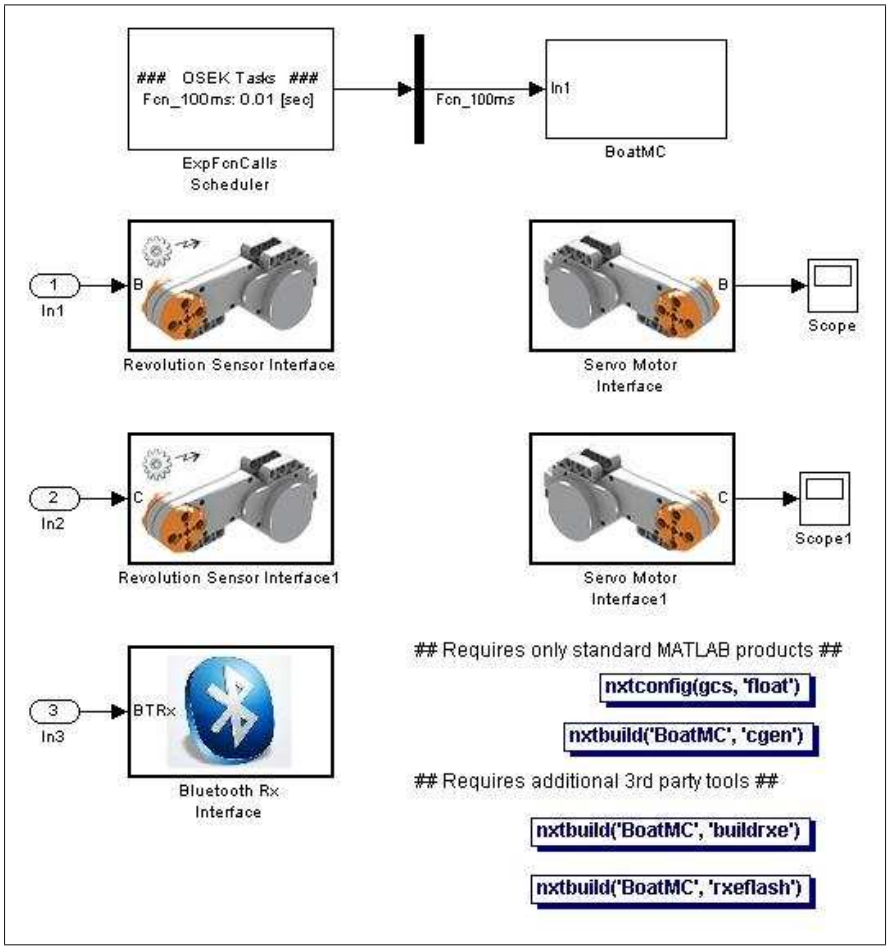


Figure C.6: USV motor speed control Simulink model

The subsystem 'BoatMC' only contains one function-call subsystem, shown in Figure C.7. The speed controller has 2 modes: 'power' mode and 'speed control' mode. In power mode, the user can set a Pulse Width Modulation (PWM) duty cycle (0–100%) for each motor. In speed control mode, encoder position data is used to estimate the propeller speed and a PID controllers use this feedback to maintain the specified angular velocity.

The Bluetooth input contains 3 elements. The first defines the mode of the speed controller. Mode ‘0’ disables both motors. Mode ‘1’ enables speed control. Mode ‘2’ enables power mode. The second and third elements of the input packet are the speed or power settings for the left and right motor respectively. See Section C.3.2 for directions regarding Bluetooth communication in MATLAB.

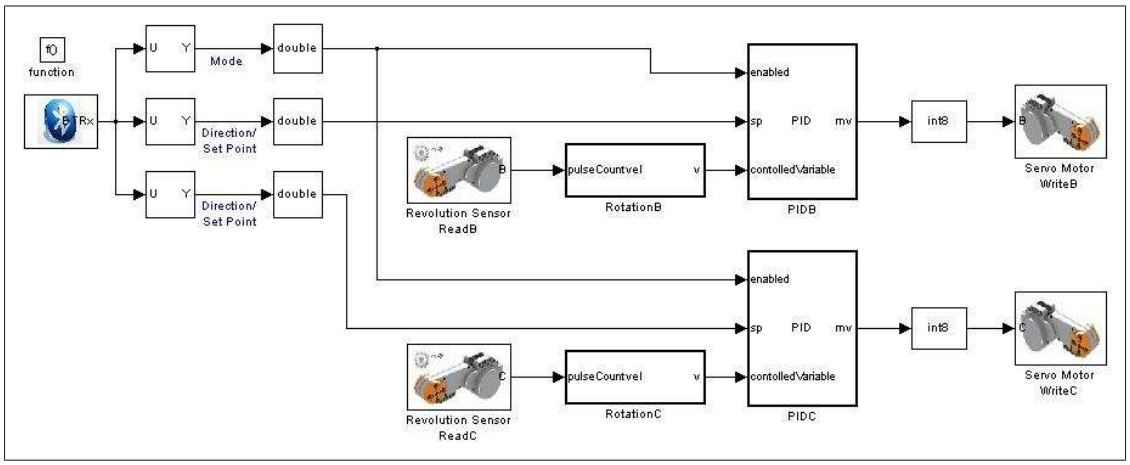


Figure C.7: USV Motor speed control function-call subsystem