Decentralized Receding Horizon Control with Application to Multiple Vehicle Systems

Yan Zhao

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Mechanical Engineering) at

Concordia University

Montreal, Quebec, Canada

August 2008

Your file Votre référence
ISBN: 978-0-494-45535-7
Our file Notre référence
ISBN: 978-0-494-45535-7

ABSTRACT

Decentralized Receding Horizon Control with Application to Multiple Vehicle Systems

Yan Zhao

Receding horizon control (RHC) has been one of the most popular control approaches recently due to its capability to achieve optimal performance in the presence of saturation constraints. There have been numerous new research results for RHC (also referred to as model predictive control) in the process control community. However, due to the high computational cost, associated with the numerical optimization problem, RHC has not often been successfully implemented on multiple vehicle systems with fast dynamics.

Decentralized receding horizon control (DRHC) is a new promising approach to reduce the computational burden of RHC. It allows the division of the computation problem into smaller parts which are solved using a group of computational nodes. This results in a substantial reduction in the computational time required for RHC. This thesis involves modeling of wheeled and hovercraft vehicles including actuator dynamics. It then applies the DRHC approach to the vehicles and implements the DRHC systems in virtual reality simulations and an experimental setup. Together, these results establish a new and useful framework for applying RHC to multiple vehicle problems.

Acknowledgements

I would like to take this opportunity to express a great gratitude to my supervisor, Dr. Brandon Gordon, who led me into the world of scientific field, inspired my interest and passion for research, gave me the clues whenever I was lost, and pointed out my weaknesses in order to improve myself. Without his insightful guidance and advice, this thesis would never be accomplished.

I would like to thank my friend Jiande Li and his family for all their help, support and comforting throughout these years.

I would also like to thank Ali Azimi, who not only helped me from the academic point of view, but also extended my views of the world that we are living in.

I would also like to express my gratitude to my colleagues, Reza Pedrami, Hojjat Izadi, Sivaram Wijenddra, Hongan Wang and Fei Yang, who built a friendly atmosphere in our office and lab.

Last but not least, I would like to thank my wife, Xiaoou, and my parents, whose love, support, understanding and encouragement pushed me into moving forward and overcome the difficulties on the way.

To Xiaoou, Mom and Dad

Table of Contents

List of Figures

List of Tables

Nomenclature

RHC	Receding Horizon Control
DRHC	Decentralized Receding Horizon Control
MPC	Model Predictive Control
node	a computer in a distributed computing system
IC	Initial Condition
SQP	Sequential Quadratic Programming
$\mathbf{x}$	state vector
$\mathbf{u}$	input vector
$\mathbf{u}^*$	optimal input vector
$\mathbf{x}_i$	state vector of the i^{th} vehicle
$\mathbf{u}_i$	input vector of the i^{th} vehicle
$\mathbf{u}_i^*$	optimal input vector of the i^{th} vehicle
$\mathbf{x}_{i,j}$	state vector of the i^{th} vehicle estimated on the j^{th} vehicle
$\mathbf{u}_{i,j}$	input vector of the i^{th} vehicle estimated on the j^{th} vehicle
$\mathbf{P},\mathbf{Q},\mathbf{R},\mathbf{K},\mathbf{C}$	weighting matrices
J	cost function
x_c, y_c	coordinates of the vehicle
u_c	longitudinal velocity of the vehicle
v_c	latitudinal velocity of the vehicle

θ_c	yaw angle of the vehicle
r_c	yaw angular velocity of the vehicle
R_{wheel}	the radius of the wheeled vehicle's wheel
ω_{wm}	angular velocity of the wheeled vehicle's motor
l_w	distance between the wheeled vehicle's wheels
J_{wm}	moment of inertia of the wheeled vehicle's motor
K_{bwm}	linear friction coefficient of the wheeled vehicle's motor
μ_{wm}	Coulomb friction coefficient of the wheeled vehicle's motor
N_v	the number of the vehicle in a multiple vehicle system
(X_B, Y_B)	body attached coordinate system
(X_G, Y_G)	global coordinate system
U_{wm}	input voltage of the wheeled vehicle motor
m_h	mass of the hovercraft vehicle
a_{hM}, a_{hT}	coefficients of the linear relationship between average voltage and motor
F_{hM}, F_{hT}	thrust generated by the main and tail rotors of the hovercraft
l_{hM}	distance between the two main rotors
l_{hT}	distance between the tail rotor and the center of the hovercraft
J_h	moment of inertia of the hovercraft
b_{hu}, b_{hv}, b_{hr}	viscous friction coefficients of the hovercraft along X_B, Y_B, Z_B
J_i^{fl}	cost function for leader's trajectory following

r_{ij}	nominal distance between i^{th} and j^{th} vehicle
J_i^{fl}	cost function for a non-leading vehicle's formation control
R_O	radius of the obstacle
x_O, y_O	coordinate of the obstacle
P, K	weighting scalars
J_1^{av}	cost function for leader's obstacle avoiding
φ	communication delay
ζ	computation time
$\mathbf{x}_D$	vector of desired states
(x_D, y_D)	desired position coordinate
θ_D	desired angle
$\omega_{D,L}$	desired angle velocity of the left motor of the wheeled vehicle
$\omega_{D,R}$	desired angle velocity of the right motor of the wheeled vehicle
U	set of admissible input
X	set of admissible state
t_s	time when optimization step starts
t_{c1}	time when computation starts
t_{c2}	time when computation ends
t_c	computation time
t_a	actuation time

$[t_1, t_2]$	time period from t_1 to t_2
A	set of multi-vehicle system
A_i	set of neighbours of the i^{th} vehicle
J_i	cost function of the i^{th} vehicle
H1	hovercraft 1
H2	hovercraft 2
$\mathbf{z}$	flat output vector
$\hat{\mathbf{x}}$	state vector of the multi-vehicle system
$\hat{\mathbf{u}}$	input vector of the multi-vehicle system
$\hat{\mathbf{x}}_i$	state vector of the neighbours of the i^{th} vehicle
$\hat{\mathbf{u}}_i$	input vector of the neighbours of the i^{th} vehicle
θ_r	reference angle
J_i	cost function of the i^{th} vehicle
TPBVP	Two Point Boundary value Problem
H	Hamiltonian function
t_f	end time of optimization
N_i	number of interpolation points
N_c	number of control points
N_p	number of unknown parameters
$\mathbf{A}_{ls}$	matrix of states for parameter identification
$\mathbf{x}_{ls}$	vector of unknown parameters

$\mathbf{b}_{ls}$	vector of outputs for parameter identification
U_{level}	voltage level on the servo amplifier of the hovercraft
K_{ratio}	duty cycle ratio
U_{hMotor}	average voltage on the motor
U_{hT}	input voltage for the tail motor of the hovercraft
U_{hM}	input voltage for the main motors of the hovercraft
U_{h}	input voltage for the any motor of the hovercraft
t_{SD}	sensor delay from the vision system
T_{SD}	upper bound of the sensor delay from the vision system
σ	synchronization offset among subsystems

1. Introduction

The control of multiple vehicle systems has been a popular topic in both the scientific and engineering world in recent years. The most commonly researched aspects are the online strategies and controller design suitable for multiple vehicle systems in different environments. These strategies and controllers have to guarantee a desired cooperative performance among the members of the systems. Many fruitful theoretical algorithms are created and their implementations can be found in many journals and conferences.

Among those methods, Receding Horizon Control, also known as Model Predictive Control (MPC), stands out due to its ability of yielding a superior tracking performance [9]. Since its introduction in the process control world, in the early eighties [1][2], it has attracted attention of many researchers, and has been successfully applied to industrial processes [3][5]. Thus, it is natural to advance a step further by applying RHC to the formation control of multi-agent systems.

However, RHC is also well known for its high computational expenses of solving numerical optimization problems involved with it [37], which make it difficult to be implemented on fast and/or complex dynamical systems. In addition, for the problems involving some subsystems, like formation control of multi-vehicle systems, the commonly used method was in centralized fashion, in which one controller had full control of the system and calculated all the control inputs for each member in such system [47]. This method significantly increased the dimensionality of the optimization problem and the computation burden, as a result, which made it nearly impossible to be implemented in real-time systems.

Thanks to the advent of decentralized RHC, the formation control of multiple agent systems becomes possible because of the concept of solving problems among a group of solvers. Furthermore, due to the recent development in computer industry, faster and more reliable calculation capacities in personal computers and the mass production of multi-core CPU and high speed network, solving complicated large-scale numerical optimization problems does not anymore rely on extremely powerful computers, which makes the study of RHC easier and more affordable.

1.1. Literature Review

The literature review is presented in this subsection; however, it is divided into different subsections for readers' convenience. Firstly, application of single RHC is reviewed. After that, distributed computing and decentralized RHC are considered and some of the articles related to the current work are presented.

1.1.1. Receding Horizon Control and its Implementation

Receding Horizon Control is essentially a repeated on-line solution of a finite horizon open-loop optimal control problem [64]. Based on the current states, the controller predicts the states of the system over a period, called *optimization horizon*, and achieves the admissible inputs by solving the cost function associated with the actual control problem. However, only a fraction of the calculated inputs will be applied to the actual system during a period called *execution horizon*. Then the process is repeated.

This control scheme is capable of controlling linear or nonlinear systems, as long as the model of the system is accurate enough to depict the system's behaviour. In addition, it can handle the constraints of the system, such as input saturations and state constraints, by modifying the cost function associated with the control problem. Furthermore, changing the mission of the controller can also be done simply by modifying the cost function, and the modification can be done in an online fashion according to the mission and environment.

On the other hand, there are some disadvantages of RHC that holds back the researchers from applying it to the fast dynamic systems. The first one is the foresaid computation cost. The high computational demand of RHC has created a challenging obstruction that makes the employment of RHC to fast dynamic systems, such as aerospace or aviation, extremely difficult. The other drawback is in theoretical field. It is difficult to deal with the stability and feasibility of RHC, and the stability of its usage in decentralized fashion is still left undone. However, these two disadvantages evoke the researchers to challenge the problems and improve the performance of RHC.

Several researchers have already conducted intensive surveys on RHC, for example, in [4] the author provided a tutorial for its mathematical background; in [5] the authors not only concentrated on RHC theories, but delivered a comprehensive comparison among the most commonly used RHC structures; while in [6] the authors discussed more about the robustness of this control method. The authors in [37], similar to [4], provided a systematic explanation of RHC, and a different numerical optimization solver for nonlinear systems with perturbation.

Besides the above articles, the efforts of improving RHC usually can be divided into three categories: one is to improve the stability of the controller with respect to nonlinear systems with uncertainties, such as noise, model uncertainty and delays; another one is to reduce the computation time by using different optimization solvers; and the last one is to improve the performance of RHC in different applications.

A. Stability

In [6], a receding horizon controller for constrained linear time-invariant systems with additive uncertainty was introduced. This controller presented better performance in terms of robustness and the ability to handle cases with large computational complexity. In that method, the control algorithm took the optimization horizon as a tunable parameter, which allowed a tradeoff between the performance and the complexity.

In [9], a robust receding horizon controller for linear systems with model uncertainty was proposed. This method was differed from the method in [6], since they sought the worst case scenario for the cost function and its upper bound. In addition, they extended their method into solving arbitrary reference tracking problems. The authors in [12] proposed a relatively simple method to determine the feedback control inputs for both linear and nonlinear systems. However, because of the computation complexity, this method is only good for slow nonlinear systems.

To ensure the stability, some basic controllers were embedded into the RHC controller. For example, the authors in [8] brought linear quadratic controller into RHC for the cases of finite input constraint sets, and proved asymptotical stability. In [10] and [11], the authors provided a robust dual-mode receding horizon controller for a wide class

of nonlinear systems with state and control constrains and model errors. In these two papers, the control inputs were obtained from two algorithms; an optimal control algorithm and a P controller. The optimal algorithm was applied when the plant was stable, or the states were within a predefined region, while the other was applied when the plant was considered unstable, or the states were out of this region.

In [42], the authors proposed a combination between adaptive control and receding horizon control method for nonlinear systems in order to stabilize the plant with control constraints. The adaptive controller was used to adjust the model in case of modeling errors and/or perturbations in the system. However, a different solution was presented in [38] to solve receding horizon control problems for nonlinear systems by finding a global Control Lyapunov Function.

B. Optimization Solvers

Different solvers were applied and tested to reduce the computation time associated with RHC or MPC problems. Usually, the goal is achieved by decreasing the number of iterations in an optimization step.

A Newton's based optimization method was proposed in [20]. It is used for online optimization of nonlinear model predictive method. In this method, Newton-type iteration is performed per sampling interval, and it provides faster convergence and shorter computation time, which is helpful for controlling fast nonlinear systems.

In [21], the authors proposed a novel method for RHC systems. This method is a computational approach to real-time trajectory generation. It uses spline interpolation and sequential quadratic programming (SQP). By upgrading this method with the Non-

Monotone Line Search approaches in [25] and [26], it resulted in faster optimization solver, ideal for trajectory tracking problems.

The computational expenses can be further reduced by using flat outputs, which can lower the dimension of an optimal control problem. Based on its definition, if the states and control inputs can be recovered by using a set of system outputs and/or derivative of the outputs, then we could call the system a flat system, and the set of outputs flat outputs [77]. In the rest of the thesis, all the optimal problems are solved by using the flat output method.

C. Implementations of RHC to Systems with Fast Dynamics

Because of the efforts mentioned briefly in the previous subsection, RHC has been successfully applied to some fast dynamic systems, such as an indoor vectored thrust flight stabilization experiment [53], simulation results for formation control of Unmanned Aerial Vehicles (UAV) [54], and roll control of delta wing vortex-coupled systems [73].

The use of RHC control method can also be found in other fields, such as, solving Markov Games [14] in the area of mathematics, controlling of production plants [39] in industrial engineering, controlling of supply chain [40] in logistics, and mine exploration planning [41] in oil industry.

1.1.2. Distributed Computing Systems

Distributed computing system is a sub-branch of parallel computing systems, which means simultaneous executions of single and/or multiple computing instructions and data on multiple processors in order to obtain results faster. A processor refers to the CPU of a computer. In this thesis, each computer has one processor, and the computer is called a *node* in the distributed computing system.

The most commonly accepted classification of parallel computing system was proposed by Flynn in [57] and Y. Censor and S. A. Zenois in [58]. There are four categories based on the interaction between instruction and data streams:

- Single instruction stream, single data stream (SISD)
- Single instruction steam, multiple data streams (SIMD)
- Multiple instruction steams, single data streams (MISD)
- Multiple instruction steams, multiple data streams (MIMD)

In this definition, the instruction streams denote the programs that are running on the computer in the network, and data streams denote the data exchange among those computers. The distributed computing system falls into the MIMD category, which refers to the systems where different parts of a program run simultaneously on two or more computers that are communicating with each other through a network. Literally, any computer could join in this network and contribute to computation. An example of this application is the Screen Saver Science (SSS) [55]. Usually, the members in the network are assumed to have same specification, i.e. CPU, RAM, etc, in order to balance the computation burden among every node to achieve the most efficient computation [13]. An example of this structure is a computer cluster [56].

In [33], an optimization method for data exchange scheme is proposed for parallel computers with distributed memory. In [30], the authors proposed an advanced dynamic programming method which is especially suitable for parallel computation, implemented on distributed memory computers, while [23] contains an example of parallel computation method providing facilities for dynamic formation on mobile robots.

In [32] a parallel asynchronous particle swarm optimization algorithm is proposed to dynamically adjust the workload assigned to each processor in a PC cluster, while in [35], the authors used parallel computation to solve a similar problem. In [34], the authors introduced an application of parallel computation on robot drives. Similar to the current popular parallel algorithms, this method estimates states on other computation nodes. In [30] a method of dynamic programming is proposed for a general-purpose cluster.

1.1.3. Decentralized RHC Formation Control

Although, only centralized solutions can theoretically guarantee asymptotic stability in many multi-vehicle applications [51][66], the computation cost makes the centralized method impractical, if not impossible, to be applied to the control of multi-agent systems [37]. On the other hand, the decentralized scheme has become popular due to lower computational burden associated with it [70][71]. It breaks the large-scale optimization problem into small pieces of individual subproblems for each member in the system. Therefore, splits the computation burden from one computer to several, and reduces the computation requirement.

It is a common assumption in most of the Decentralized RHC (DRHC) studies for multi-vehicle systems to assume that the subsystems are dynamically decoupled. However, they have coupling effects from their cooperative objective and interaction constraints.

In DRHC, the states of the plants should be communicated within the computation nodes, or at least with the ones that have coupled objectives. Some researchers have suggested that each system should provide its most updated trajectory to the other systems, so that the solver on each node could compute according to the most up-to-date information. For example, in [48], [50], and [51], the authors proved the stability of distributed formation control problems with coupled cooperative cost functions on dynamically decoupled subsystems, by using synchronous updating and exchanging the most recent optimal control trajectories between the coupled subsystems.

Others suggest a method that involves an estimator/predictor at each node to estimate the states of other nodes, and correct their estimation only at the beginning of optimizing iteration. And this method guarantees feasibility as long as the mismatch between the estimation and actual cost is within a certain range [60].

In [45], the authors investigate the stability of the DRHC controller by studying the local variables, costs and constraints of a subsystem and the ones who have direct interaction with it. Through an estimator, during every sampling time, the subsystem not only solves its own optimization problem, but calculates the states of its neighbors from the data received at that time. The authors propose in [46] an algorithm that is able to partition a distributed control system into manageable subsystems. Contrary to the other articles in this section, the authors in this paper do not concentrate on the control of a

group of subsystems and their autonomous control strategies, but provide a division method. It has some similarities to the parallel RHC implementations, but differs in the way that, in this paper, a p-step prediction algorithm is used to estimate the states on the other nodes to reduce the effect from the delay inherited within distributed systems. Similar strategies can be found in [29], [47], [49], and [52].

In the aspect of improving DRHC performance, the authors in [72] and [75] propose an interesting theory that the communication among the computation nodes plays an important role in the performance of controller as well. The main concept of their theory is to improve the behaviour of the group by manipulating the communication bandwidth in order to reduce the mismatch between the estimated and actual trajectory of a specific member [74][76][43][44].

Another problem in single and distributed RHC systems is delays. In addition to the computation delays for a single RHC, the structure of the distributed RHC systems adds more delays to the problem, since these systems require time to solve the optimization problems and exchange information from one computing node to another.

In [24] and [63], the authors proposed a new algorithm on real-time RHC computation in order to reduce the instability caused by the computation delay inherited in the RHC formulations. The solver only needs to solve the premature cost according to the introduced criterion, thus the overall computation time is reduced.

In [27], the authors presented some studies on time-delay systems. This made an overview on different types of time-delay systems, and proposed some modifications. In [36] and [63], the authors propose a RHC method for constrained linear systems with uncertain delays by using a novel artificial Lyapunov function.

Even though, the focus of this thesis is discussion and implementation of distributed receding horizon control, mentioning some backgrounds and implementations of the parallel RHC, enables the reader to compare and distinguish the differences of these two approaches.

The authors in [15] propose a method for evaluating the optimal-control problems by using iterative method of dynamic programming. In this paper, the authors decomposed the plant model, and assigned each node a decomposed part. By solving the cost function of each local optimal problem, the controller integrates the solution of all decomposed parts, and finds an optimal solution to the plant.

In [16], the authors provide a solution to large-scale convex optimal control problems in a different aspect from [15]. Instead of model decomposition, their method is based on time decomposition. The optimal problem is dispatched to several computing nodes. However, this method could not be easily implemented with the presence of time delay in the environment of network.

The authors, in [17] , use similar decomposition method as in [15], a hierarchical decomposition method. This method focuses on the problem structure, decomposes the large problem into small subproblems.

In [19], the author proposes a hardware implementable parallel computing algorithm for general minimum-time control, by using time decomposition technique. In addition, this method is applied on hardware setup, based on Very-large-scale integration VLSI) array processor technology. In [18], this method is extended to solve receding horizon control for constrained nonlinear systems on the basis of VLSI technology.

Authors, in [28], propose an on-line task assignment solver for multi-vehicle distributed control. The solver is based on a trajectory primitive decomposition approach, which could be categorized as time decomposition approach. Before presenting their method, the authors also compare several different methods and conduct several simulations on Cornell's RoboFlag environment. The author evaluates in [31] not only several different programming procedures and algorithms for MPC on real-time multiprocessing computing, but the task structures/computation model as well, such as: linear array, tree, and mesh.

1.2. Thesis Objectives and Contributions

In this thesis, the decentralized receding horizon control method is investigated through numerous simulations and experiments. New algorithms and methods for trajectory following and formation control of multiple vehicle systems are evaluated and compared. Accurate models of both wheeled robots and hovercraft vehicles are developed, experimentally identified, and tested. Decentralized RHC is then applied to both types of vehicles through simulations and experiments. A virtual reality simulation system with a 6 DOF cockpit is combined with the experiments to provide a higher level of capability to study more advanced DRHC problems. Together, these results provide a new and useful framework for simulation and experimental testing of new decentralized RHC algorithms and other types of nonlinear control methods for multi-vehicle systems.

The remaining parts of this thesis are organized as follows. Chapter 2 reviews the receding horizon control method and decentralized RHC in detail. Chapter 3 and Chapter 4 develop and experimentally test the models for the wheeled and hovercraft vehicles,

respectively. Chapter 5 provides a set of simulations for decentralized RHC of wheeled vehicles and also provides experimental testing. Chapter 6 presents simulations for DRHC of hovercraft vehicles and then develops an upgrade to the system by adding a virtual reality system with a 6 DOF cockpit. Conclusions and future work are discussed in Chapter 7.

2. Overview of RHC and Decentralized RHC

Basic theoretical background of RHC and DRHC are presented in this chapter. Since RHC can be categorized as an optimal control problem, the concept of optimal control will be firstly discussed. Then the concept of flat outputs is explained, followed by the review of RHC and DRHC. Lastly, an example of angle regulator of two hovercrafts is presented as a simple tutorial of how to form cost function for RHC and DRHC methods.

2.1. Optimal Control

Suppose to have a system with state equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^n$ is the vector of state variables of the system for $\forall t \geq 0$ and $\mathbf{u}(t) \in \mathbb{R}^m$ is the vector of input variables for $\forall t \geq 0$, and they both satisfy the following constraints

$$\begin{aligned} \mathbf{u}(t) &\in U \\ \mathbf{x}(t) &\in X, \end{aligned} \tag{2}$$

where U denotes the allowable set of inputs, X is a set of admissible states, and $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \rightarrow \mathbb{R}^n$. It also has an initial condition

$$\mathbf{x}(0) = \mathbf{x}_0. \tag{3}$$

An optimal control problem is to find a control input $\mathbf{u}^*(t)$, so that minimizes the following cost function of the system

$$J(\mathbf{x}(t),\mathbf{u}(t),t_f)=\int_0^{t_f} q(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)d\tau + V(\mathbf{x}(t_f),t_f). \tag{4}$$

where t_f denotes the time when the optimization process finishes, q is usually a quadratic cost function, which is responsible for the performance of the system, and V is called *terminal cost*, which is important to ensure the stability of the controller [83].

A standard method is to bring in a vector of co-state variables $\boldsymbol{\lambda}(t)\in\mathbb{R}^n$, and generate the Hamiltonian of the system as following [79]:

$$H(\mathbf{x}(t),\mathbf{u}(t),\boldsymbol{\lambda}(t),t)=q(\mathbf{x}(t),\mathbf{u}(t),t)+\lambda^T f(\mathbf{x}(t),\mathbf{u}(t),t). \tag{5}$$

The optimal input can be obtained by solving the following equations:

$$\begin{aligned}\dot{\mathbf{x}}(t)&=\mathbf{f}(\mathbf{x}(t),\mathbf{u}(t),t)\\ \dot{\boldsymbol{\lambda}}&=-\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T\boldsymbol{\lambda}-\left(\frac{\partial q}{\partial \mathbf{x}}\right)^T\end{aligned} \tag{6}$$

where the input can be obtained from

$$\frac{\partial H}{\partial \mathbf{u}}=\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T\boldsymbol{\lambda}+\left(\frac{\partial q}{\partial \mathbf{x}}\right)^T=0 \tag{7}$$

subject to

$$\begin{aligned}\mathbf{x}(0)&=\mathbf{x}_0\\ \lambda(t_f)&=\left(\frac{\partial V}{\partial \mathbf{x}}\right)^T_{t=t_f}\end{aligned} \tag{8}$$

The above problem is also called Two Point Boundary value Problem (TPBVP), and there have been many articles about solving this kind of problems, the interested readers are referred to [79][85] for detailed information.

2.2. Flat Outputs

The flat outputs help increase the speed of solving the optimization problem, associated with some optimal control problems by reducing the dimension of the problem. The definition of flat outputs is as follows [77]: for a dynamic system, if there exists output $\mathbf{z}$, where

$$\mathbf{z} = \mathbf{g}(\mathbf{x}, \mathbf{u}) \tag{9}$$

and $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ is the input vector, and $\mathbf{g}: \mathbb{R}^m \times \mathbb{R}^n$, such that the states and inputs can be recovered by a function $\mathbf{h}(\cdot)$ using $\mathbf{z}$ and/or its derivatives as below:

$$(\mathbf{x}, \mathbf{u}) = \mathbf{h}(\mathbf{z}, \dot{\mathbf{z}}, \cdots, \mathbf{z}^{(r)}) \tag{10}$$

where $\mathbf{z}^{(r)}$ denotes the r^{th} time derivative of $\mathbf{z}$. Then, the system is called a flat system. Therefore, in a flat system, the states and inputs of the system can be recovered by finite number of flat outputs and their derivatives, but no integration by the flat outputs [77].

2.3. Receding Horizon Control

Receding Horizon Control is essentially a repeated on-line solution of a finite horizon open-loop optimal control problem [64]. Its scheme is shown in Fig.1. Based on the states at time t_s, the controller predicts the states of the system over *optimization horizon* T, and achieves the admissible inputs $\mathbf{u}^*$ by solving the cost function associated with the actual control problem. Only the first part of the inputs will be applied to the actual system during *execution horizon* δ. Then the process is repeated. The process is

illustrated in Fig.1, where the thick curve indicates the actual state of the system, and the light curves denote the computed or predicted state of the system by the controller, based on the model of the system.

According to the figure, the above procedure can be further explained as follows: at time t, the controller samples the state of the system (point 1), and based on the sampled state the controller predicts the future state of the system over the optimization horizon T (line a), and based on the prediction it obtains the optimal input for the system. But only the first part of the input will be applied to the system during the execution horizon δ, and the rest will be discarded. Then at time $t+\delta$, the new state of the system is sampled and used to predict next trajectory (line b) for optimization. Then the process is repeated until the system meets the goal.

According to the above explanation, the procedure of an RHC controller can be summarized as three steps:

- Form the problem
- Solve the problem
- Apply the inputs

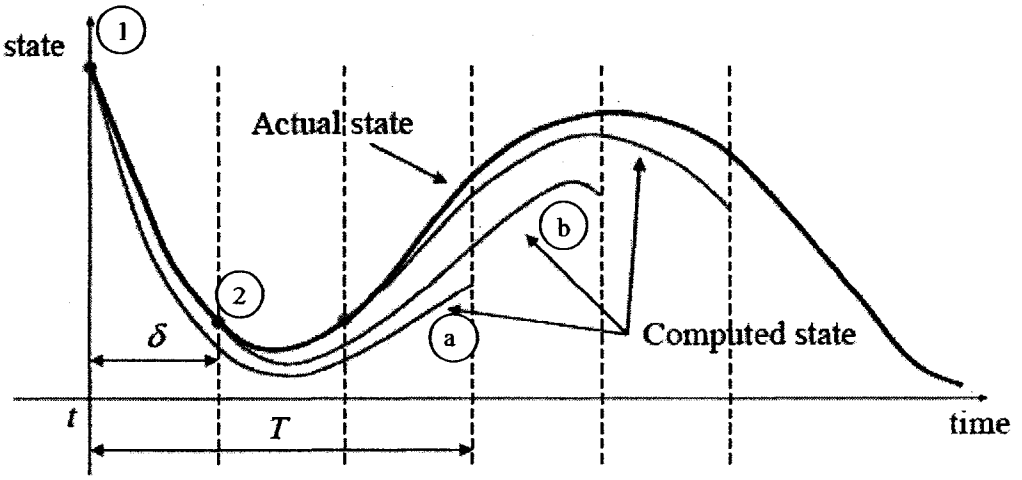

Fig.1. Illustration of RHC trajectory generation [83]

2.3.1. Form the Problem

The ease of using RHC for a control problem is that the objective of a mission can be explicitly and solely formed in a cost function. After that, the controller will be able to drive the system to the desired states, provided that the model of the system is accurate enough and the sensors are working properly. Furthermore, the cost function can be changed during the control process if there is any modification in the objective. Thus, the purpose of this section is to illustrate how to generate a cost function according to the objective of the control mission.

Suppose we have a system with state equation:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \ \mathbf{x}(0) = \mathbf{0} \tag{11}$$

where as stated earlier, $\mathbf{x}(t) \in \mathbb{R}^n$ is the vector of state variables of the system for $\forall t \geq 0$ and $\mathbf{u}(t) \in \mathbb{R}^m$ is the vector of input variables for $\forall t \geq 0$, and they satisfy the constraints in (2), and $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \to \mathbb{R}^n$. Also define $X \subseteq \mathbb{R}^n$ the set of admissible states and $U \subseteq \mathbb{R}^m$ the set of admissible inputs of the system respectively:

$$\mathbf{x}(t) \in X, \ \mathbf{u}(t) \in U \text{ for } t > 0 \tag{12}$$

In addition, consider the assumptions A1-A3 in [65] are also satisfied, where:

- $\mathbf{f}$ is twice differentiable;
- U is compact and convex;
- System (11) has a unique solution at any given initial condition.

The first assumption is provided to ensure continuity of the cost function. The second assumption ensures the optimization region admits a well defined locally optimal solution.

Then the cost function for the system (11) over prediction horizon T is defined as follows [79]:

$$J(\mathbf{x}(t),\mathbf{u}(\cdot),T)=\int_{t}^{t+T}\left(\|\mathbf{x}(\tau;\mathbf{x}(t))\|_{\mathbf{Q}}^{2}+\|\mathbf{u}(\tau)\|_{\mathbf{R}}^{2}\right)d\tau+\|\mathbf{x}(t+T;\mathbf{x}(t))\|_{\mathbf{P}}^{2} \tag{13}$$

where $\mathbf{P}\in\mathbb{R}^{n\times n}$, $\mathbf{Q}\in\mathbb{R}^{n\times n}$, and $\mathbf{R}\in\mathbb{R}^{m\times m}$ are positive definite weighting matrices, and $\mathbf{x}(\tau;\mathbf{x}(t))$ denotes the states of the system at time τ resulted from the input $\mathbf{u}(\cdot)$ when the initial condition is $\mathbf{x}(t)$; T is a finite optimization horizon, the weighted norms in (13) are defined as $\|\mathbf{x}\|_{\mathbf{P}}^{2}=\mathbf{x}^{T}\mathbf{P}\mathbf{x}$. Therefore, the resulted $J(\mathbf{x}(t),\mathbf{u}(\cdot),T)$ is a scalar variable denoting the cost of the system.

Ideally, the choice of the terminal cost is $\|\mathbf{x}(\infty;\mathbf{x}(t))\|_{\mathbf{P}}^{2}$ such that the mismatch between the optimal finite cost and the infinite cost is zero; however, this situation will never happen and the nature of the problem is to reduce the mismatch [83].

2.3.2. Solve the Problem

The optimization problem is to find an input $\mathbf{u}$, so that the following equation holds:

$$J^{*}(\mathbf{x}(t),T)=\min_{\mathbf{u}(\cdot)}J(\mathbf{x}(t),\mathbf{u}(\cdot),T) \tag{14}$$

subject to

$$\left.\begin{array}{c}\dot{\mathbf{x}}(\tau)=\mathbf{f}(\mathbf{x}(\tau),\mathbf{u}(\tau),\tau)\\ \mathbf{u}(\tau)\in U\\ \mathbf{x}(\tau;\mathbf{x}(t))\in X\end{array}\right\}\tau\in[t,t+T] \tag{15}$$

where $J^{*}(\mathbf{x}(t),T)$ denotes the optimal cost based on the optimal input.

The approach to solve the open-loop optimal control problem in this thesis is based on the method introduced in [21].

Firstly, check whether the system is a flat system according to the definition described in section 2.2, and if the system satisfies the definition, some of the system outputs will be selected as the flat outputs in the hope of lowering the dimension of the optimal problem; but this step can be skipped if the system is not, however, the time consumed to solve the problem is expected to be longer.

Then an interpolation method is used to characterize and simplify the optimization problem. The cubic spline interpolation method is employed in this thesis because it is an effective approach that is more simple and computationally inexpensive compared to other methods such as B-splines. The degree of each spline is defined by setting the *control points*. This results in a continuous curve and is divided into discrete pieces by adding points on the curve. These points are called *interpolation points*, which should be selected close enough to be able to present the behaviour of the curve. Then the optimization problem is modified to find a set of inputs that minimizes the cost function of the system at the interpolation points.

The scheme of the above interpolation method is shown in Fig.2, where N_i denotes the total number of interpolation points over an optimization horizon T, and N_c denotes the total number of control points. Using more control points results in more optimization parameters and thus increases the computation time. In addition, more interpolation points result in a smoother cure and increase computation time, as well. Several interpolation methods could be used in this section to parameterize the flat outputs (if they exist) selected in the last section, such as linear interpolation method and

B-Spline method. However, in this thesis, the cubic spline method is chosen, as stated earlier. The interested readers are referred to [86] for detailed explanation of how this method works.

Lastly, the resulted optimization problem can be solved by the numerical optimization solvers, such as Sequential Quadratic Programming (SQP) [21], Powell's Method [86], and other optimization packages such as SNOPT [87].

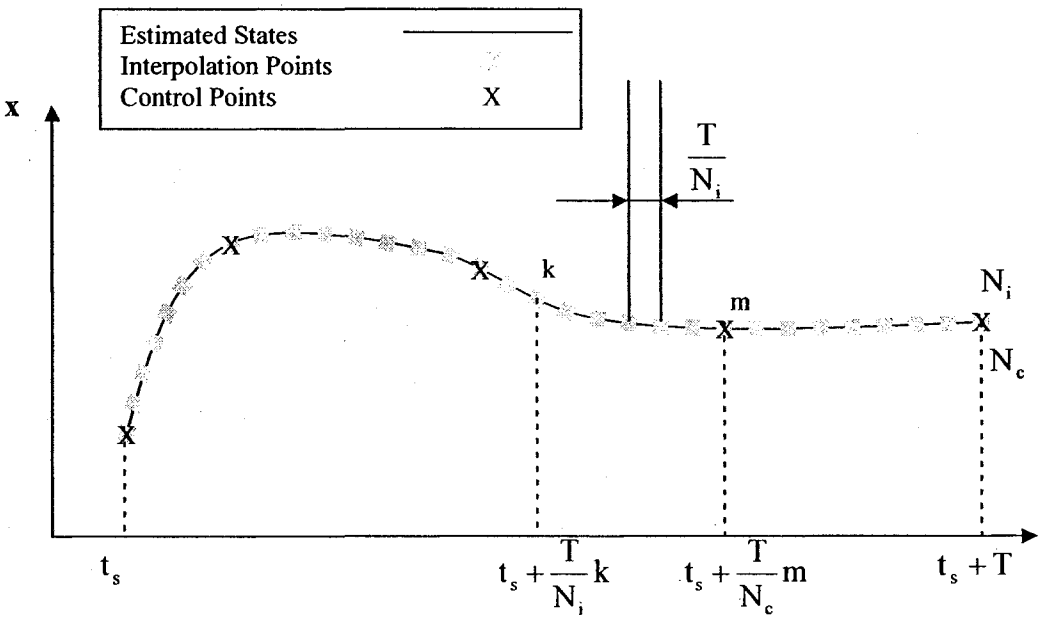

Fig.2. Interpolation scheme.

2.3.3. Apply the Input

Suppose the solution to (14) is obtained as

$$\mathbf{u}^*(\tau) = \mathbf{u}^*(\tau; \mathbf{x}(t)), \tag{16}$$

then during a period of time $\tau \in (t, t+\delta]$, the optimal input is applied to the plant, where δ denotes execution horizon, and $0 < \tau \leq T$. After applying the control to the system, the

resulted states of the system becomes the initial condition for the optimization problem of the next step.

Moreover, the choice of the values of optimization and execution horizon is vital to the performance of the controller. Usually, for the stability of the system, the execution horizon is chosen much smaller comparing to the optimization horizon [83], because in this way, the mismatch between the predicted and actual trajectory in Fig.1 is small thus the performance can be guaranteed.

However, the above concept has a crucial constraint in the practical implementation. Before we discuss that constraint, we need to define some parameters in the first place [84]:

- *Step start time*: t_s. This is the time when an optimization step starts. This is also the time when the controller starts sampling the state of the system. Since the time used in sampling is considerably less than the following time periods, we assume that the time used to sample is zero, and the controller obtains the states at time t_s.
- *Computation start time*: t_{c1}. This is the time when the optimization procedure starts (for one step).
- *Computation finish time*: t_{c2}. This is the time when the optimization procedure finishes (for one step).
- *Computation time*: $t_c = t_{c2} - t_{c1}$. This is the time period for how long the optimization step takes.
- *Actuation time*: t_a. This is the time when the calculated input is applied to the actuator. This input could be either a newly computed input or an input that has been obtained in advance.

- *Actuation latency*: l_a This is the delay generated in the actuation of the system after starting optimization step or sampling the states.

An illustration of the above parameters is shown in Fig.3.

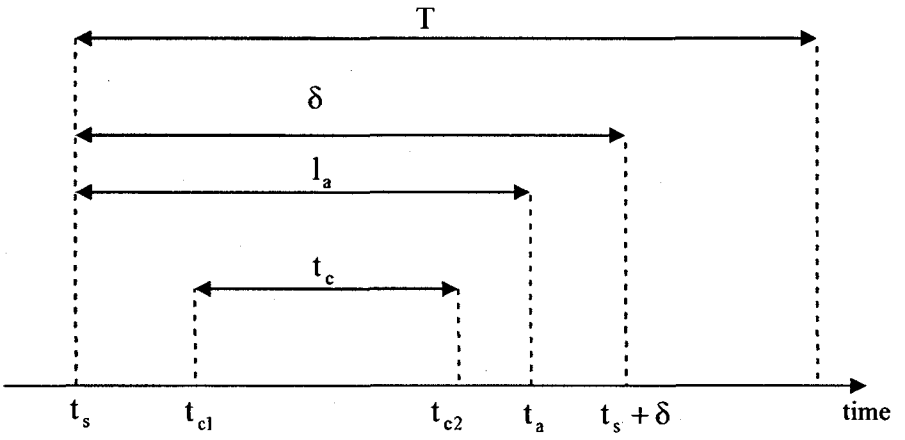

Fig.3. RHC time parameters.

Typically in theoretical discussions, the computation time t_c is assumed to be zero, but in the implementations in the real world, t_c is non-negligible, and rather plays an important role in the application. That is because the existence of the non-zero computation time prevent us from choosing small execution horizon δ, that is, δ must be more or equal to the computation time t_c. Because of this constraint, the controller is unable to apply the input as soon as the current state of the system is sampled, and must wait until the input is calculated, which is where the actuation latency l_a comes from. Two methods are created to tackle this problem: *Retarded Actuation* Method and *On-the-Fly Computation* method.

The Retarded Actuation method, as its name indicates, solves the optimization problem for the next step in advance and applies the input at the beginning of that step [53]. This method gives the controller sufficient time to finish the calculation. Its scheme

is shown in Fig.4. The solid lines in the diagram denote the input is applied to the system, the dashed lines denote the input is generated but not applied, and $[t_s, t_s + \delta]$ denotes the time interval from t_s to $t_s + \delta$, so $\mathbf{u}^*[t_s, t_s + \delta]$ is the optimal input generated for the time interval $[t_s, t_s + \delta]$.

There are two methods to generate the input. In order to better explain the difference between these two methods, we will take how the optimal input signal for interval $[t_s + \delta, t_s + 2\delta]$ is obtained as an example.

In the first method, the process takes the states of the system at time t_s, $\mathbf{x}(t_s)$, as an initial condition. Based on this initial condition, the controller generated the input for $[t_s + 0, t_s + \delta]$ and $[t_s + \delta, t_s + 2\delta]$, and only the input for the interval $[t_s + \delta, t_s + 2\delta]$ will be applied to the system at time $[t_s + \delta, t_s + 2\delta]$; in the second method, instead of using $\mathbf{x}(t)$, the controller firstly predicts the system states $\mathbf{x}(t_s + \delta)$, then uses these states as an initial condition, and applies the corresponding input for the interval $[t_s' + 0, t_s' + \delta]$, where $t_s' = t_s + \delta$.

For Retarded Actuation method, the following equations should be satisfied:

$$\begin{aligned} t_s &= t_{c1} \\ t_{c2} &\le t_s + \delta \\ t_a &= t_s + \delta \\ t_c &\le \delta \\ l_a &= \delta \end{aligned} \qquad (17)$$

The other method is called *On-the-Fly Computation* method. In this method, there is no prediction involved, and the actuation latency is smaller. Let us take the interval $[t_s + 0, t_s + \delta]$ for an example.

The controller will start the optimization process at time t_s when the states are sampled, and apply the inputs as soon as they are available. However, because of the existence of t_c in each step, there will be a time interval $[t_s, t_s + t_c]$, in which no optimal input is available (The input for the last step has finished, and the input for this step is not yet available). To solve this problem, instead of applying the last step input till t_s, the system continues using the input calculated for the last step for the interval $[t_s, t_s + t_c]$, until the new optimal input is available. When the new input is available, the controller will switch to apply the new input to the system at t_s+t_c. The scheme of this method is shown in Fig.5.

Unlike the Retarded Actuation method, the On-the-Fly Computation method does not involve a variation that requires predicting the states of the system $[t_s, t_s + t_c]$ and uses that states as the initial condition. Because the computation time t_c of the next step is unknown until the optimization is finished and the new input is obtained.

For On-the-Fly Computation method, the following equations should be satisfied:

$$\begin{aligned} t_s &= t_{c1} \\ t_{c2} &\le t_s + \delta \\ t_a &\le t_s + \delta \\ t_c &\le \delta \\ l_a &= t_c \end{aligned} \tag{18}$$

The whole process of a complete RHC implementation is shown in the flowchart in Fig.6. And the RHC computations in this thesis heavily rely on Receding Horizon Control Object-Oriented Library (RHCOOL) in [7].

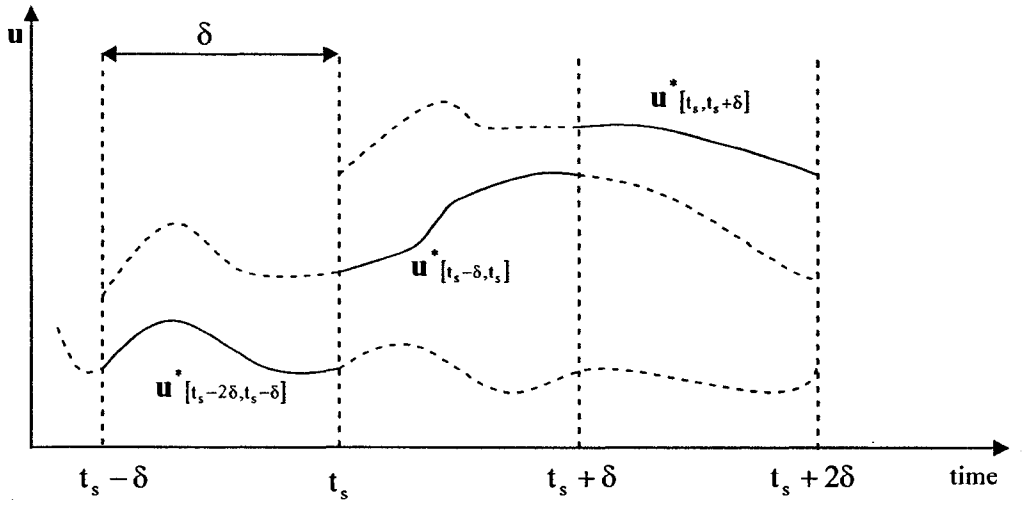

Fig.4. Control signal for retarded actuation method.

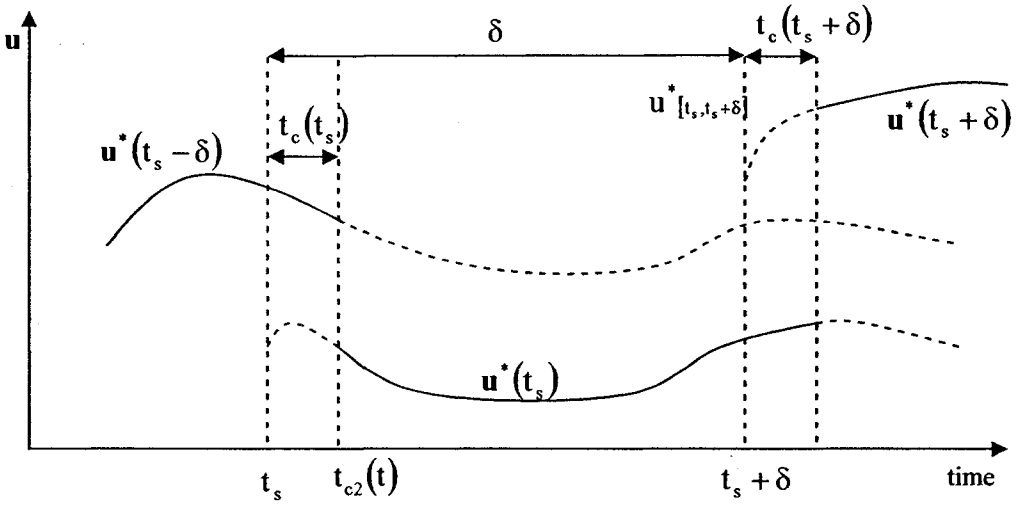

Fig.5. Control signal for on-the-fly computation method.

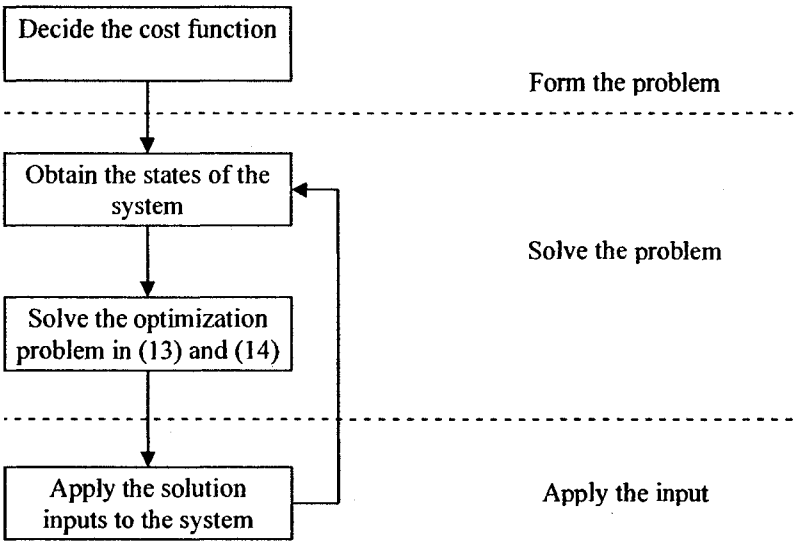

Fig.6. RHC flowchart.

2.4. Decentralized Receding Horizon Control

Suppose there is a set A with N_v vehicles, which forms a formation, then for the i^{th} vehicle in the system, there is a set A_i containing the neighbours of the i^{th} vehicle, thus named the set of neighbours of i^{th} vehicle. The definition of neighbour can be found in [60] and [74]. For example, in the following six vehicle system (Fig.7), vehicle No.1 has No. 2 and No. 3 as its neighbours, while No. 3 has No. 2, No. 5, No. 6, and No. 1 as its neighbours.

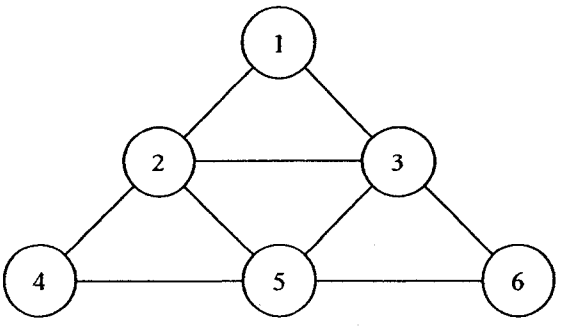

Fig.7. Six-vehicle system.

There are two popular approaches in current decentralized RHC formation control area [43]. In the first approach, the agent will only estimate its own states, but estimated trajectory of each agent will be exchanged among the agents [61]. The second involves using the most available states of the agent's neighbour, and calculating the optimal cost of that agent by estimating the states of both its neighbours and itself [60].

A. The First Approach

Suppose the following state equation is of the i^{th} vehicle:

$$\dot{\mathbf{x}}_i(t) = \mathbf{f}_i(\mathbf{x}_i(t), \mathbf{u}_i(t), t) \tag{19}$$

where $\mathbf{x}_i(t) \in \mathbb{R}^{n_i}$ is the vector of state variables of the i^{th} system and $\mathbf{u}_i(t) \in \mathbb{R}^{m_i}$ is the vector of the input variables of the i^{th} system for $\forall t \geq 0$. Also define $X_i \subseteq \mathbb{R}^{n_i}$ the set of admissible states and $U_i \subseteq \mathbb{R}^{m_i}$ the set of admissible inputs of the system respectively:

$$\mathbf{x}_i(t) \in X_i,\ \mathbf{u}_i(t) \in U_i \text{ for } t > 0 \tag{20}$$

Also let $\hat{\mathbf{x}}(t) \in \mathbb{R}^{\hat{n}}$ and $\hat{\mathbf{u}}(t) \in \mathbb{R}^{\hat{m}}$ be the vectors that store the states and inputs of the whole system at time t, where $\hat{n} = \sum_{i=1}^{N_v} n_i$ and $\hat{m} = \sum_{m=1}^{N_v} m_i$, and the state equation for the whole system can be obtained as:

$$\dot{\hat{\mathbf{x}}}(t) = \mathbf{f}(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t). \tag{21}$$

Therefore, the cost function for the whole system is given as:

$$J(\hat{\mathbf{x}}(t), \hat{\mathbf{u}}(t), t) = \sum_{i=1}^{N_v} J_i(\mathbf{x}_i(t), \mathbf{u}_i(t), \hat{\mathbf{x}}_i(t), \hat{\mathbf{u}}_i(t)) \tag{22}$$

where J_i denotes the cost function for the i^{th} vehicle. This cost depends on the behaviour of the i^{th} vehicle, as well as the interactive relation to its neighbours' states and inputs, which are presented by $\hat{\mathbf{x}}_i(t)$ and $\hat{\mathbf{u}}_i(t)$.

Furthermore, J_i in (22) can be achieved by:

$$\begin{aligned} &J_i\left(\mathbf{x}_i(t),\mathbf{u}_i(t),\mathbf{x}_j(t),\mathbf{u}_j(t)\right) \\ &= J_{i,i}\left(\mathbf{x}_i(t),\mathbf{u}_i(t)\right)+\sum_{i,j\in A} J_{i,j}\left(\mathbf{x}_i(t),\mathbf{u}_i(t),\mathbf{x}_j(t),\mathbf{u}_j(t)\right) \end{aligned} \tag{23}$$

where $J_i(\cdot)$ denotes the cost function for the i^{th} vehicle. The admissible input $\mathbf{u}^*_{i,j}(t)$ is obtained by solving

$$J^*_{i,j}\left(\mathbf{x}_{i,i}(t),\mathbf{x}_{i,j}(t),T\right)=\min_{\mathbf{u}_{i,i}(\cdot),\mathbf{u}_{i,j}(\cdot)} J_i\left(\mathbf{x}_{i,i}(t),\mathbf{u}_{i,i}(\cdot),\mathbf{x}_{i,j}(t),\mathbf{u}_{i,j}(\cdot),T\right) \tag{24}$$

where the first subscript of $\mathbf{x}$ and $\mathbf{u}$ indicates the state and input belong to the i^{th} vehicle, the second one indicates the location where the state or input is calculated or estimated. Let us take $\mathbf{x}_{2,1}(t)$ for an example, $\mathbf{x}_{2,1}(t)$ stands for the states of the 2^{nd} vehicle estimated on the 1^{st} vehicle. The input $\mathbf{u}_{i,j}(\cdot)$ will be applied to the i^{th} vehicle at each execution horizon. Afterwards, the actual states and inputs of each agent will be exchanged among the whole system for the next optimization step.

The problem certainly can be solved by using the centralized fashion [60]. In order to solve the optimization problem associated with the i^{th} vehicle in a decentralized way, the i^{th} vehicle at least needs to know its current states and its neighbours' current states. Based on the states, it is possible to predict its optimal inputs and its neighbours' optimal inputs. Its own inputs will be applied to the system, while the inputs to its neighbours', however, will only be used to predict the neighbours' trajectories, and then discarded. So the procedure of DRHC is shown in Fig.8.

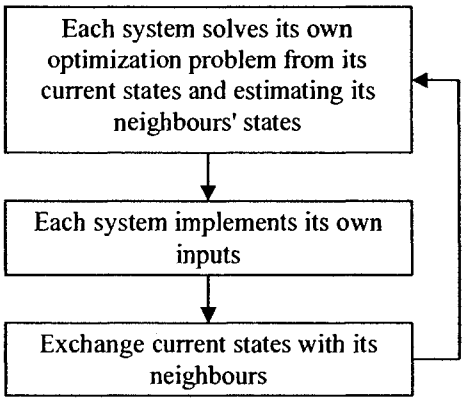

Fig.8. DRHC flowchart, first approach.

It is commonly known that the stability of DRHC is not ensured, because the prediction of the j^{th} system on i^{th} vehicle is independent from the actual j^{th} vehicle's behaviour, and the mismatch between these two values usually causes problems.

The authors in [60] proposed a solution stating that if the mismatch is within a range, then the system is asymptotically stable. Suppose mismatch between the i^{th} system and i^{th} system's prediction on j^{th} system is given as:

$$\varepsilon_{i,j} = \int_{t}^{t+T} \left(2\left\|\mathbf{x}_{j,j}(\tau) - \mathbf{x}_{j,i}(\tau)\right\|_{\mathbf{Q}}^{p} + \left\|\mathbf{u}_{j,j}(\tau) - \mathbf{u}_{j,i}(\tau)\right\|_{\mathbf{R}}^{p} \right) d\tau \tag{25}$$

then the mismatch for the whole system, ε, is obtained as:

$$\varepsilon = \sum_{j=1, j\in A_j}^{6} \varepsilon_{i,j} = \sum_{j=1, j\in A_j}^{6} \int_{t}^{t+T} \left(2\left\|\mathbf{x}_{j,j}(\tau) - \mathbf{x}_{j,i}(\tau)\right\|_{\mathbf{Q}}^{p} + \left\|\mathbf{u}_{j,j}(\tau) - \mathbf{u}_{j,i}(\tau)\right\|_{\mathbf{R}}^{p} \right) d\tau \tag{26}$$

If the following relation holds, then the system is asymptotically stable:

$$\varepsilon_{i,j} \le \left\|\mathbf{x}_{i}(t)\right\|_{\mathbf{Q}}^{p} + \left\|\mathbf{x}_{j}(t)\right\|_{\mathbf{Q}}^{p} + \left\|\mathbf{x}_{i}(t) - \mathbf{x}_{j}(t)\right\|_{\mathbf{Q}}^{p} + \left\|\mathbf{u}_{i,i}(t)\right\|_{\mathbf{R}}^{p} + \left\|\mathbf{u}_{j,i}(t)\right\|_{\mathbf{R}}^{p} \tag{27}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are positive definite matrices if $p = 2$, and $\mathbf{Q}$ and $\mathbf{R}$ are full rank matrices if $p = 1, \infty$. The proof of this theory can be found in [60].

This approach delivers outstanding performance in formation control, but will add some computation burden to each node since they have to do extra calculation to achieve the estimation of the states of their neighbours. Therefore, if being applied into the real world, the retarded actuation method is recommended, since that method gives the controller sufficient time to predict the states and trajectory of the other vehicles.

B. The Second Approach

In this approach, let us still assume (19) is the state equation for the i^{th} vehicle, which satisfies the constraints in (20). The resulted cost function, on the other hand, will be written in the following form:

$$J_i\left(\mathbf{x}_i(t),\mathbf{u}_i(t),\mathbf{x}_j(t)\right)=\sum_{i=1,j\in A_i}^{N_v} J_i\left(\mathbf{x}_i(t),\mathbf{u}_i(t),\mathbf{x}_j(t)\right) \tag{28}$$

where $J_i(\cdot)$ denotes the overall cost function for the i^{th} vehicle. Please note that there is no other extra subscript associated with the states and inputs except for the one that indicates the number of the agent. The cost function generated by i^{th} and j^{th} vehicle can be obtained as [61]:

$$J_{i,j}\left(\mathbf{x}_i(t),\mathbf{u}_i(t),\mathbf{x}_j(t)\right)=\int_t^{t+T}\left\|\left(\mathbf{x}_i(\tau)-\mathbf{x}_j(\tau)\right)\right\|_{\mathbf{P}}^2+\left\|\mathbf{u}_i(\tau)\right\|_{\mathbf{Q}}^2+\left\|\mathbf{x}_i(\tau)\right\|_{\mathbf{R}}^2 d\tau\,. \tag{29}$$

where $J_{i,j}(\cdot)$ denotes the cost function caused by the formation between i^{th} and j^{th} vehicle. The admissible inputs of the i^{th} agent $\mathbf{u}_i^*(t)$ is obtained by the following:

$$J_i^*\left(\mathbf{x}_i(t),\mathbf{x}_j(t)\right)=\min_{u_i(\cdot)} J_i\left(\mathbf{x}_i(t),\mathbf{u}_i(t),\mathbf{x}_j(t)\right) \tag{30}$$

and the input will be applied to the vehicle at each execution horizon. Then the actual states and inputs of each agent will be exchanged among the whole system for the next optimization step. The flowchart of this approach is shown in Fig.9.

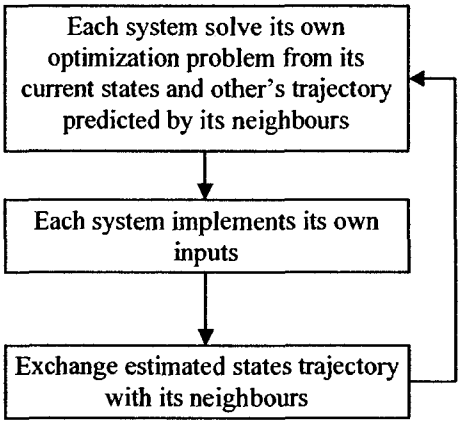

Fig.9. DRHC flowchart, second approach.

This method is not as computationally expensive as the first approach since the nodes do not estimate any states other than their own. It is also obvious that, in most cases, the i^{th} agent is unable to obtain its input at time t, as the states of its neighbour j^{th} agent may not be available at that time. Thus when being used in the real practice, the On-the-Fly Computation method is recommended. In the following chapters of this thesis, the second approach will be considered, and a method to cancel the effect of delays will be introduced.

2.5. Cooperative Control Example

In this section, an example of completing a DRHC control of two hovercrafts is explained. The example basically involves an angle regulation and tracking of both vehicles. A reference angle θ_r is set for hovercraft 1 (H1), so that H1 will point to that

direction. Besides, hovercraft 2 (H2) will follow H1's step and point to that direction as well (Fig.10).

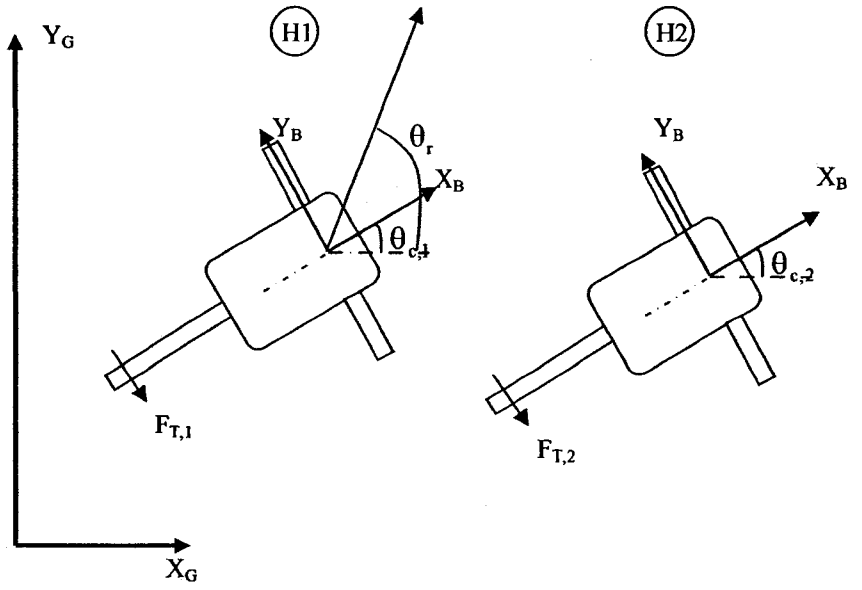

Fig.10. Angular regulation example of DRHC

The dynamic models for both vehicles' angle is shown below

$$\begin{aligned} \dot{r}_{c,1} &= a_1 F_{T,1} - b_1 r_{c,1} \\ \dot{r}_{c,2} &= a_2 F_{T,2} - b_2 r_{c,2} \\ \dot{\theta}_{c,1} &= r_{c,1} \\ \dot{\theta}_{c,2} &= r_{c,2} \end{aligned} \tag{31}$$

where $r_{c,1}$ and $r_{c,2}$ are the angular velocity of H1 and H2, $\theta_{c,1}$ and $\theta_{c,2}$ are angle of H1 and H2 respectively, a_1, a_2, b_1, and b_2 are the parameters associated with hovercrafts rotation, and $F_{T,1}$ and $F_{T,2}$ denote the input applied to the motor of H1 and H2 respectively.

Then the cost function for H1 can be formed as:

$$J_{c,1}(\theta_{c,1}(t),T)=\int_{t}^{t+T}(\theta_{c,1}(\tau)-\theta_r)^2\,d\tau+(\theta_{c,1}(t+T)-\theta_r)^2 \tag{32}$$

and the cost function H2 can be formed as:

$$\begin{aligned}&J_{c,2}(\theta_{c,1}(t),\theta_{c,2}(t),T)\\&=\int_{t}^{t+T}(\theta_{c,1}(\tau)-\theta_{c,2}(\tau)-\theta_{1,2})^2\,d\tau+(\theta_{c,1}(t+T)-\theta_{c,2}(t)-\theta_{1,2})^2\end{aligned} \tag{33}$$

where $\theta_{1,2}$ denote the desired angle between H1 and H2. $\theta_{1,2}$ can be set as any number, and in this case, it is set to be zero. After these two definitions, the DRHC controller is able to follow the procedure discussed in the previous sections and finish the mission. It should be noted that the computation time is assumed to be zero in this example.

In order to simplify the optimization process, the flat outputs method discussed in section 2.2 can be used here. From (31), the flat outputs of the system can be selected as:

$$\begin{aligned}z_1&=\theta_{c,1}\\z_2&=\theta_{c,2}\end{aligned} \tag{34}$$

where z_1 and z_2 denote the flat output of H1 and H2, respectively. Using the selected flat outputs, the remaining states and inputs of the system can be obtained as:

$$\begin{aligned}r_{c,1}&=\dot{z}_1\\r_{c,2}&=\dot{z}_2\\F_{T,1}&=\frac{\ddot{z}_1+b_1\dot{z}_1}{a_1}\\F_{T,2}&=\frac{\ddot{z}_2+b_2\dot{z}_2}{a_2}\end{aligned} \tag{35}$$

The following parameters are selected for the RHC controller:

$$
\begin{aligned}
N_c &= 3 \\
N_i &= 50 \\
\delta &= 0.1 \text{ s} \\
T &= 1.0 \text{ s}
\end{aligned}
\tag{36}
$$

and the initial conditions for H1 and H2 are:

$$
\begin{aligned}
r_{c,1}(0) &= 0 \\
r_{c,2}(0) &= 0 \\
\theta_{c,1}(0) &= 0.5 \text{ rad} \\
\theta_{c,2}(0) &= 1.0 \text{ rad}
\end{aligned}
\tag{37}
$$

and the reference angle is set to:

$$
\theta_r = \frac{\pi}{2} \text{ rad} \tag{38}
$$

Thereby, the problem is well set up and the simulation result of the above problem is shown in Fig.11.

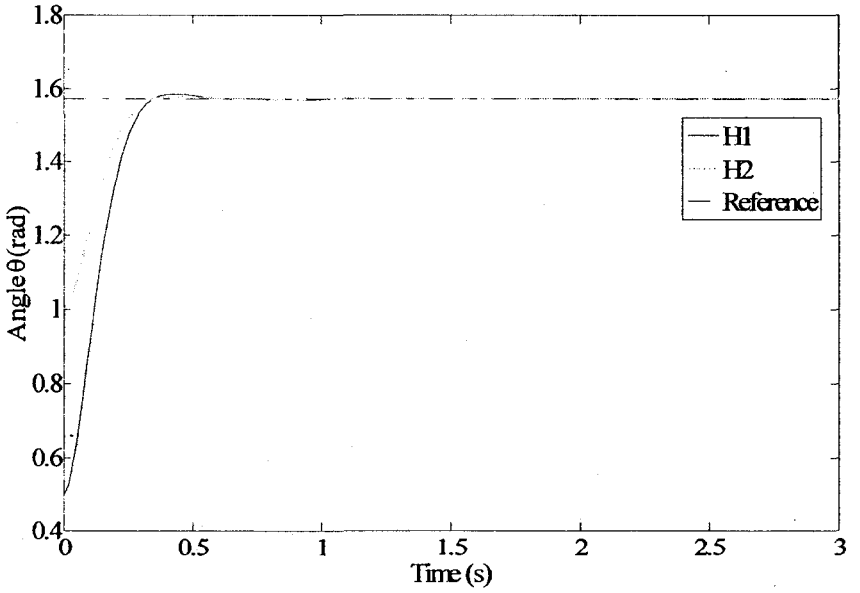

Fig.11. Simulation result of the angle regulation and tracking example

3. Modeling and Identification of Wheeled Vehicles

A successful receding horizon control implementation is based on the prediction of the system's states over the optimization horizon, and reducing the mismatch between the predicted states and the actual states (Fig.1) is crucial. A good prediction is primarily based on the accuracy of the system model. Thus, in the following two chapters, the modeling and identification of the vehicles is discussed in detail.

3.1. Wheeled Vehicle Model

The kinematic model of the wheeled vehicle and the dynamic model of the actuators are presented in the following subsections.

3.1.1. Kinematic Model of the Wheeled Vehicle

The configuration of the wheeled vehicle is illustrated in Fig.12 to Fig.15. Although both dynamic models and kinematic models can be used for wheeled vehicles, the kinematic model is adopted. Since the vehicle does not move fast and the wheels do not slip much the kinematic model is able to accurately describe the motion of the system. Also the kinematic model is more computationally simple which is helpful when solving the RHC optimization problem. Furthermore, kinematic models have been successfully used in similar experiments [67].

The kinematic equations for each vehicle expressed in the body attached frame (X_B, Y_B) are given as follows [67] (see Fig.16):

$$\dot{x}_c = \frac{1}{2}\left(\omega_{wm,R} R_{wheel} + \omega_{wm,L} R_{wheel}\right)\cos\theta_c$$
$$\dot{y}_c = \frac{1}{2}\left(\omega_{wm,R} R_{wheel} + \omega_{wm,L} R_{wheel}\right)\sin\theta_c \qquad (39)$$
$$\dot{\theta}_c = \frac{1}{2l_w}\left(\omega_{wm,R} R_{wheel} - \omega_{wm,L} R_{wheel}\right)$$

where R_{wheel} is the radius of each wheel, $\omega_{wm,R}$ and $\omega_{wm,L}$ denotes the angular velocity of the right and left wheel respectively, and l_w denotes the distance between the two wheels, (x_c, y_c) denote the coordinate of the vehicle in the global frame, and θ_c denotes the angle between the global and the body attached frame.

Fig.12. The wheeled vehicle side view

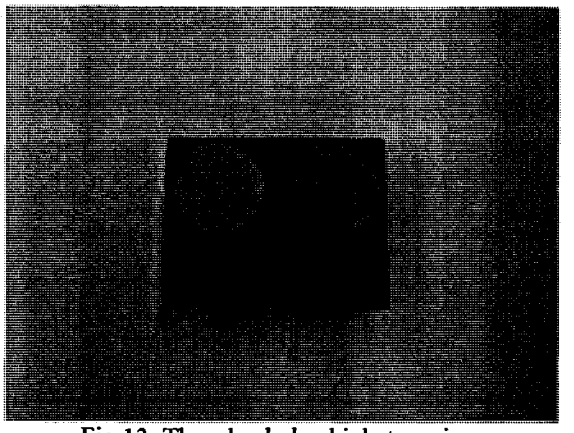

Fig.13. The wheeled vehicle top view

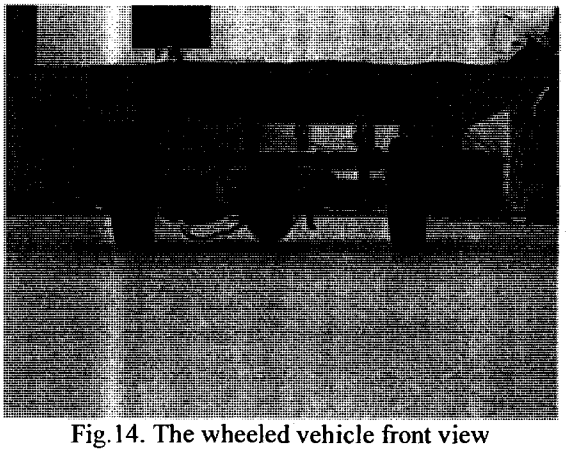

Fig.14. The wheeled vehicle front view

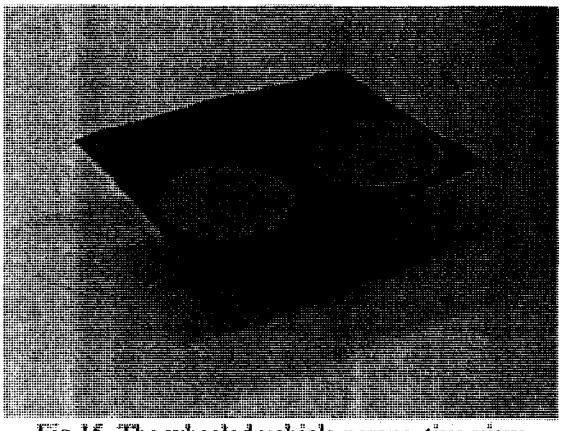

Fig.15. The wheeled vehicle perspective view

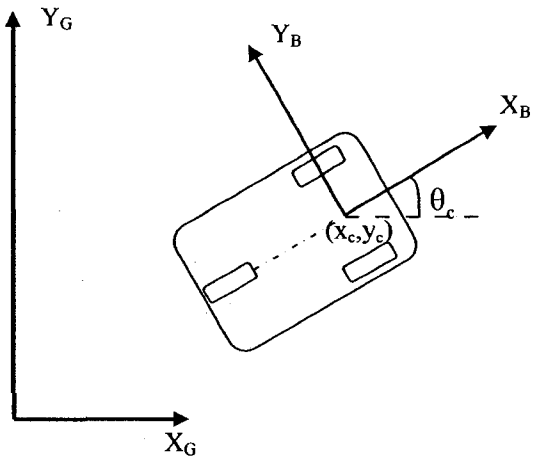

Fig.16. The wheeled vehicle's schematic model

3.1.2. Dynamic Model of the Vehicle Actuators

The 3 degree of freedom motion of a wheeled vehicle is controlled by two servo motors, which are controlled remotely by a computer via wireless FM radio communication links.

The dynamic equation of each motor is given by the following:

$$J_{wm}\dot{\omega}_{wm} = K_{1wm}U_{wm} - \eta_{wm}\omega_{wm} - K_{bwm}\omega_{wm} - \mu_{wm}\,\mathrm{sgn}(\omega_{wm}) \qquad (40)$$

where $\dot{\omega}_{wm}$ and ω_{wm} represent the angular acceleration and angular velocity of the wheel respectively, U_{wm} represents the voltage applied on the motor, J_{wm} represents moment of inertia of the motor, K_{1wm} and η_{wm} are constant parameters associated with the motor, K_{bwm} denotes the linear friction coefficient of the motor, and μ_{wm} denotes the Coulomb friction coefficient of the motor.

3.1.3. Sensor Dynamics and Noises

All vehicles are placed and controlled under a 9-camera overhead vision system (Fig.17). The vision system is able to track the color targets placed on the vehicles (as shown in Fig.13 and Fig.19) at a sampling rate of 25Hz. Note that the angular velocity and acceleration terms are obtained by using center finite-difference approximations of the values of the targets.

However, the vision system, like most of the other tracking systems, has a sensor delay, which alters the performance of controller. The delay is mainly caused by the nature of the vision system, which will be fully explained in Chapter 5. In this section, only the pattern of the sensor delays (41) will be discussed.

In the experimental tests performed to investigate the delay, a LED flash light bulb was placed under the vision system. It flashed on and off at a constant frequency. A timing computer recorded the time when the bulb was turned on, and the vision system sent a signal back to the timing computer immediately after capturing the light. The timing computer recorded the time at the moment of receiving the signal. Thus, the delays were obtained by comparing the two times on the timing computer. The result is shown in Fig.18. Although in Fig.18, only a small portion of number of samples are shown, the figure is adequate enough to show the pattern of the delay, since throughout the experiment, the delay has never exceeded the maximum value in Fig.18. Therefore, the upper bound of the delay can be found as:

$$t_{SD} \leq 0.06 \text{ s} = T_{SD} \tag{41}$$

where t_{SD} denotes the sensor delay and T_{SD} denotes the upper bound of the sensor delay. This information will be used in the DRHC implementation in Chapter 6.

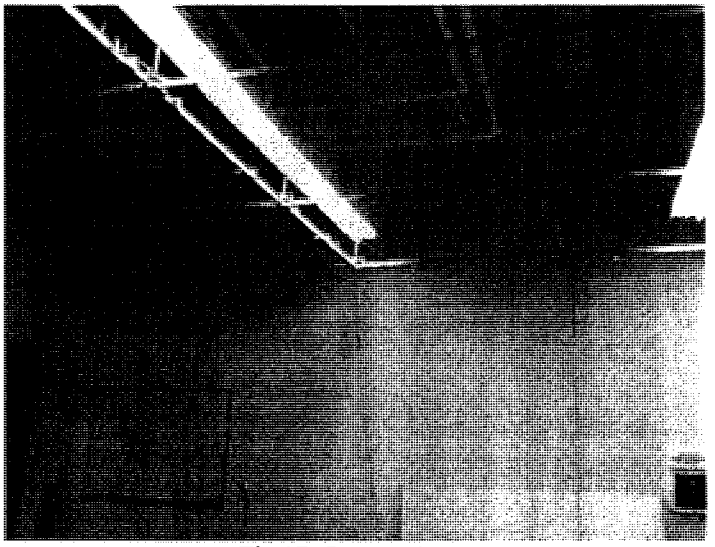

Fig.17. Camera Array

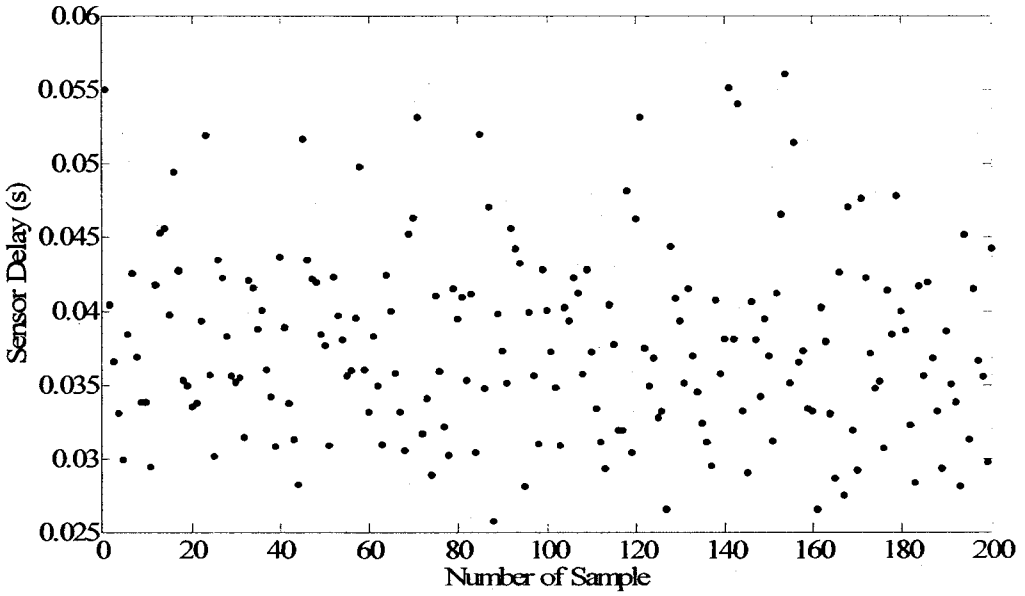

Fig.18. Pattern of the sensor delays

Sensor noise can be observed in the sensor related diagrams in the coming sections. This noise mainly comes from the following three sources:

- incorrect time measurement for the vision sample, because we are unable to control the sampling time, but only put the frequency to its highest possible level;
- noise from the cameras themselves causes noise in position data;

- finite difference error for derivative calculations, which is caused by the above two factors combined.

Although filters, such as low-pass filter, can be used to handle the noise, no filter was used in any of our experiments. This is mainly because in the process of parameter identification, the curve fit method (to be discussed in the next section) averages the data to some extent; while in feedback control, the phase lag from low-pass filters caused the system to become less stable.

3.1.4. State Equations of the Wheeled Vehicles

The state equations of the wheeled vehicle used in this thesis are obtained in (42) by combining the equations in the first two sections. Please note that the subscripts R and L indicate the right and left motor on the vehicles respectively.

$$
\begin{aligned}
\dot{x}_c &= \frac{1}{2}\left(\omega_{wm,R} R_{wheel} + \omega_{wm,L} R_{wheel}\right)\cos\theta_c \\
\dot{y}_c &= \frac{1}{2}\left(\omega_{wm,R} R_{wheel} + \omega_{wm,L} R_{wheel}\right)\sin\theta_c \\
\dot{\theta}_c &= \frac{1}{2l_{wm}}\left(\omega_{wm,R} R_{wheel} - \omega_{wm,L} R_{wheel}\right) \\
\dot{\omega}_{wm,L} &= \frac{1}{J_{wm,L}}\left(K_{Iwm,L} U_{wm,L} - \eta_{wm,L}\omega_{wm,L} - K_{bwm,L}\omega_{wm,L} - \mu_{wm,L}\,\mathrm{sgn}(\omega_{wm,L})\right) \\
\dot{\omega}_{wm,R} &= \frac{1}{J_{wm,R}}\left(K_{Iwm,R} U_{wm,R} - \eta_{wm,R}\omega_{wm,R} - K_{bwm,R}\omega_{wm,R} - \mu_{wm,R}\,\mathrm{sgn}(\omega_{wm,R})\right)
\end{aligned}
\tag{42}
$$

3.2. Parameter Identification

This subsection will introduce the procedure of how parameters in (42) are identified in details.

3.2.1. Parameter Identifications of the Actuators

A cross bar with two coloured targets attached is installed on the wheel (Fig.19), to measure the angular velocity and angular acceleration of the motor. The center of the bar was precisely placed on the center of the wheel to ensure the targets were mounted with same distance from the center. Before estimation, the motors were balanced by a leveller to ensure the accuracy of the measured data.

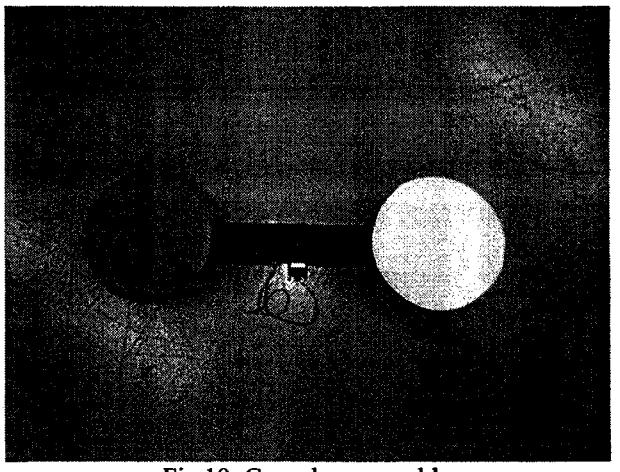

Fig.19. Cross bar assembly

For the sake of minimizing the number of parameters to be identified, the equation in (40) is rearranged as follow:

$$\dot{\omega}_{wm} = a_1 U_{wm} - a_2 \omega_{wm} - a_3 \, \mathrm{sgn}(\omega_{wm}) \tag{43}$$

where $a_1 = \frac{K_{Iwm}}{J_{wm}}$, $a_2 = \frac{\eta_{wm} + K_{bwm}}{J_{wm}}$, $a_3 = \frac{\mu_{wm}}{J_{wm}}$

In order to solve (43), it is rearranged into the following equation:

$$\left[U_{wm} \quad -\omega_{wm} \quad -\mathrm{sgn}(\omega_{wm})\right]\cdot\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \left[\dot{\omega}_{wm}\right] \tag{44}$$

The parameters in (44) can be solved by performing a least squares curve fit to the sets of experimental data. The least squares identification problem can then be formulated as an over determined linear system as below:

$$\begin{aligned} &\mathbf{A}_{ls}\mathbf{x}_{ls} = \mathbf{b}_{ls} \\ &\mathbf{A}_{ls} = \begin{bmatrix} \mathbf{A}(t_1) \\ \mathbf{A}(t_2) \\ \vdots \\ \mathbf{A}(t_{Np}) \end{bmatrix} \qquad \mathbf{b}_{ls} = \begin{bmatrix} \mathbf{b}(t_1) \\ \mathbf{b}(t_2) \\ \vdots \\ \mathbf{b}(t_{Np}) \end{bmatrix} \end{aligned} \tag{45}$$

where N_p is the number of points in a given experimental data set. In this case, the experimental data sets include step input responses with different magnitudes. In this case, the parameters of (45) can be presented in the following and the problem can be solved by using pseudo-inverse approach.

$$\begin{aligned} &\mathbf{A}_{ls} = \left[U_{wm} \quad -\omega_{wm} \quad -\mathrm{sgn}(\omega_{wm})\right] \\ &\mathbf{x}_{ls} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \\ &\mathbf{b}_{ls} = \left[\dot{\omega}_{wm}\right] \end{aligned} \tag{46}$$

Two sets of experimental data were used in the parameter identification process (IC#1 and IC#2). They included step inputs with different magnitudes and similar initial conditions of $\omega_{wm}(t) = 0.0\mathrm{rad/s}$ and $U_{wm}(t) = 0.0\mathrm{V}$ for all $t \le 5\mathrm{s}$. IC#1 had a step input of $U_{wm}(t) = 0.2\mathrm{V}$, and IC#2 had a step input of $U_{wm}(t) = 0.3\mathrm{V}$ for all $t > 5\mathrm{s}$.

The identified parameters for left and right motors are listed in Table 1 and Table 2, respectively, along with the nominal parameters obtained from the average numerical

values of the identified parameters from IC#1 and IC#2. The estimated error bounds for the motor parameters were obtained through completing the identification procedure by using other data sets but same parameters, and computing the maximum deviation with the estimated parameters.

Fig.20 shows the time history of the angular acceleration response of the left motor for IC#1, and Fig.21 shows the corresponding simulation obtained from the identified parameters for IC#1. In addition, the results of the left motor for IC#2 are shown in Fig.22 and Fig.23; the results of the right motor for IC#1are presented in Fig.24 and Fig.25; and the results of the right motor for IC#2 are shown in Fig.26 and Fig.27.

Parameters	IC#1	error bounds	IC#2	error bounds
a_1	60.0045	±0.2055	61.0320	±0.3083
a_2	5.348	±0.8258	5.02	±1.073
a_3	0.062	±0.003	0.059	±0.003

Table 1. Estimated motor parameters from linear least square approximation of the left motor

Parameters	IC#1	error bounds	IC#2	error bounds
a_1	61.128	±0.1982	62.119	±0.2973
a_2	5.01	±0.195	4.932	±0.2535
a_3	0.093	±0.032	0.061	±0.032

Table 2. Estimated motor parameters from linear least square approximation of the right motor

The comparisons in Fig.20 - Fig.24 illustrate that the identified parameters from linear least squares method are accurate enough to depict the behaviour of the motors under different inputs, although with a small latency in some occasions.

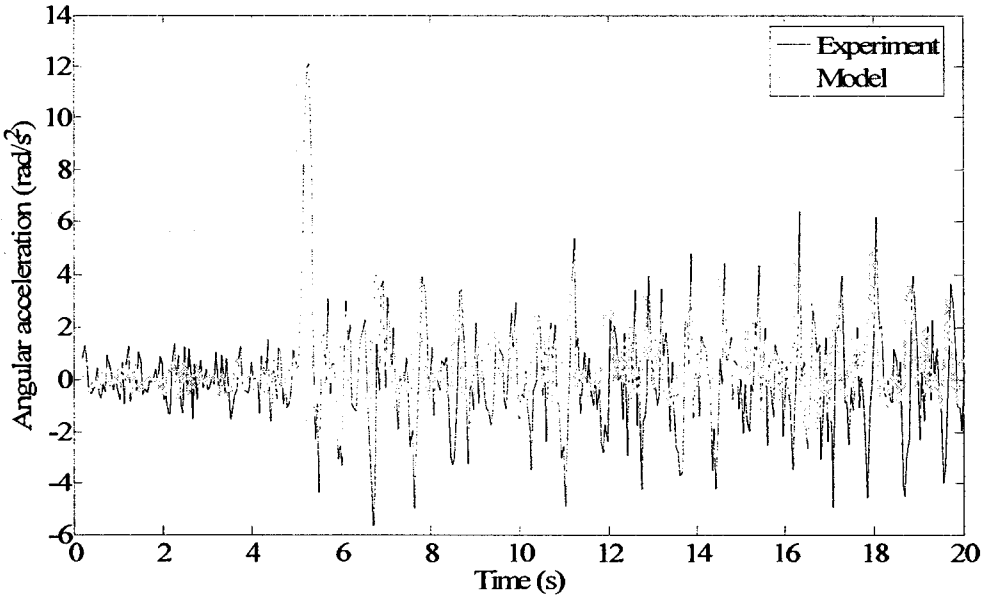

Fig.20. Linear square approximation of angular acceleration of the left motor (IC#1)

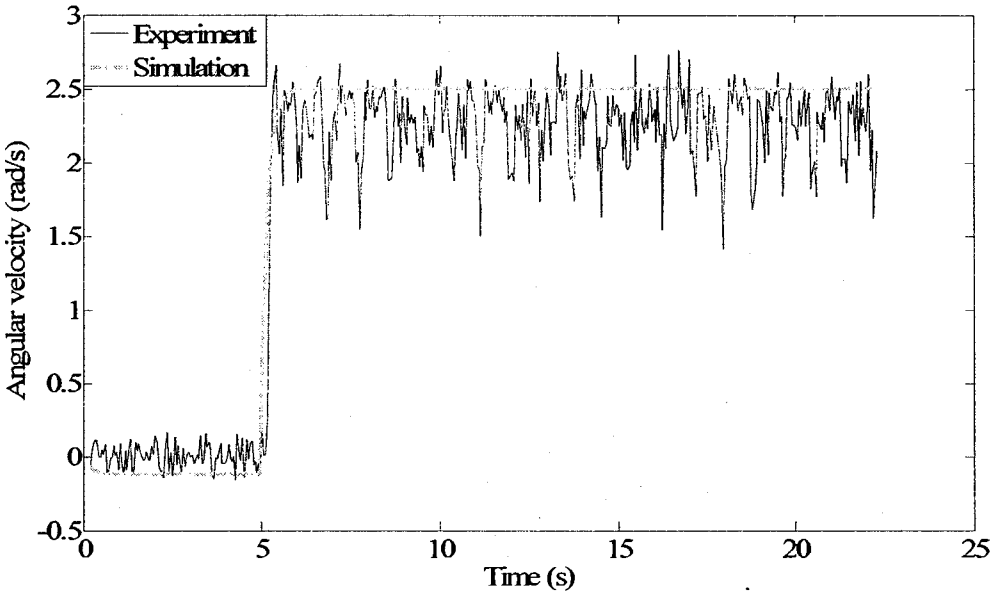

Fig.21. Left motor angular velocity response using linear square approximation (IC#1)

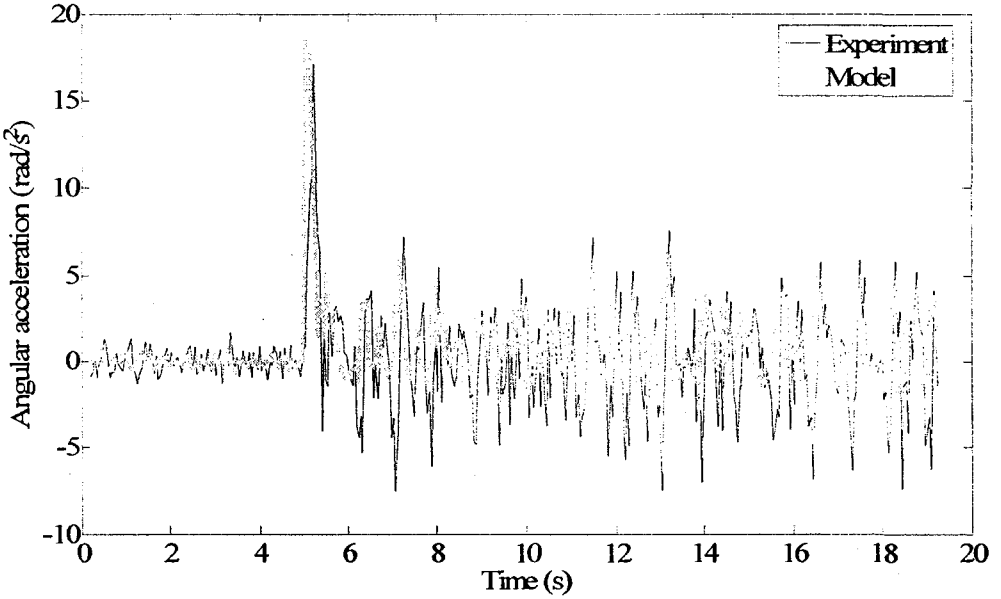

Fig.22. Linear square approximation of angular acceleration of the left motor (IC#2)

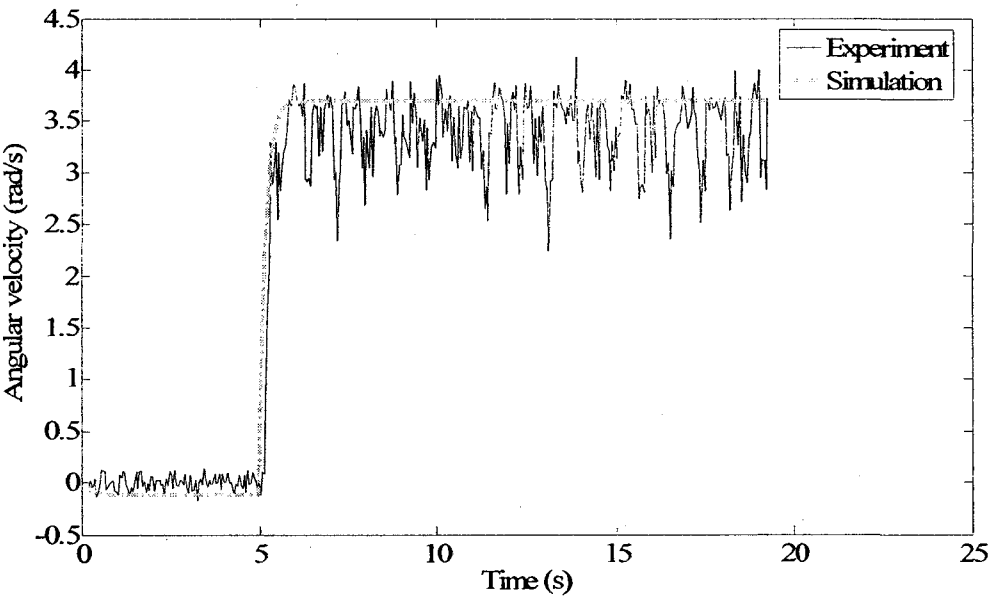

Fig.23. Left motor angular velocity response using linear square approximation (IC#2)

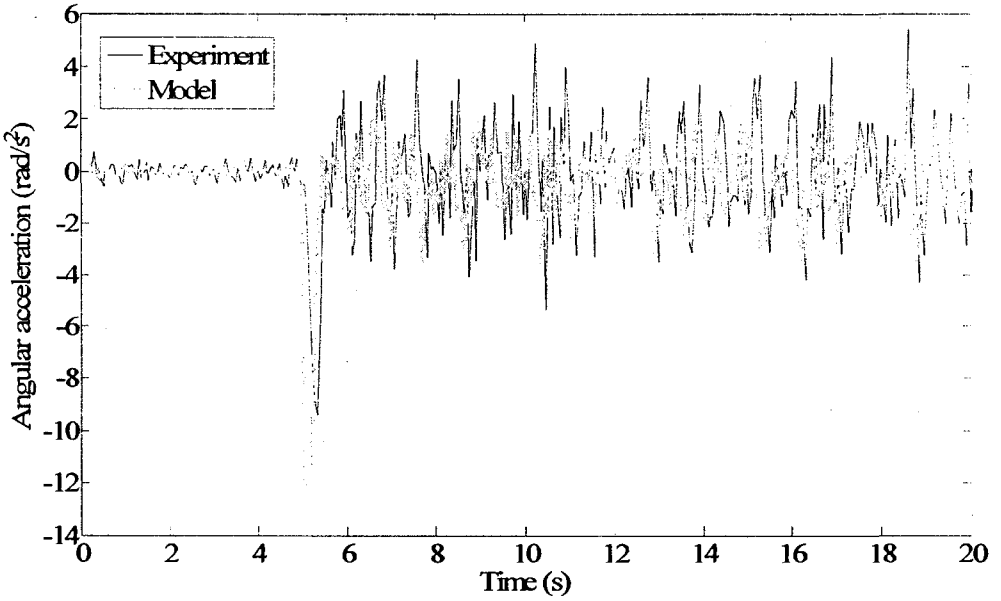

Fig.24. Linear square approximation of angular acceleration of the right motor (IC#1)

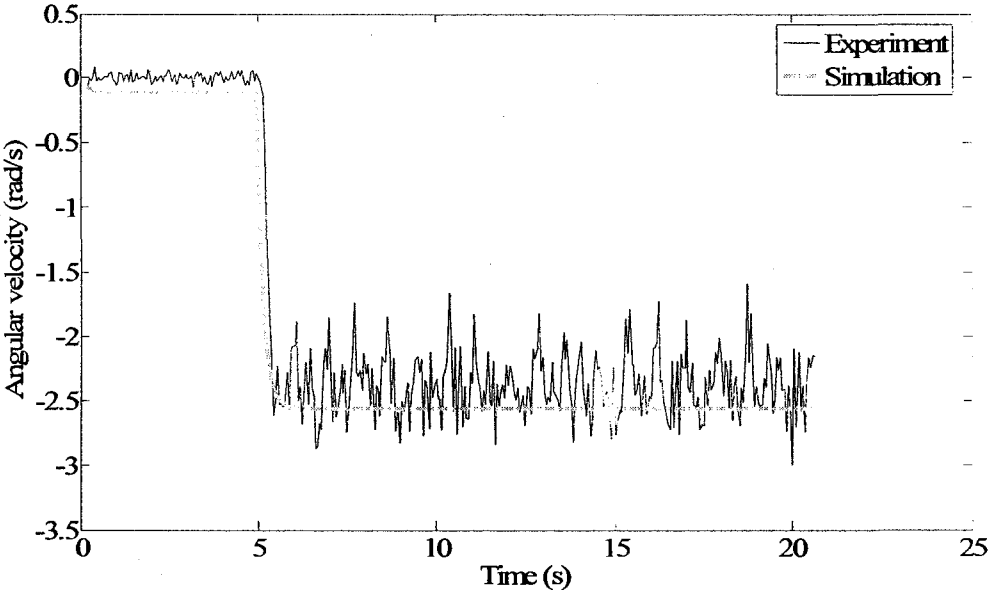

Fig.25. Right motor angular velocity response using linear square approximation (IC#1)

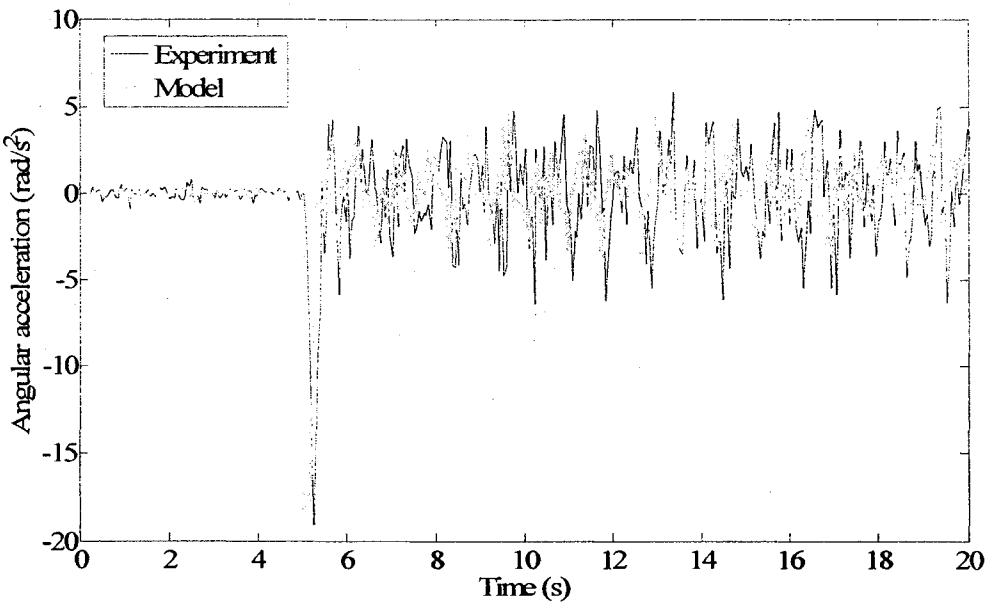

Fig.26. Linear square approximation of angular acceleration of the right motor (IC#2)

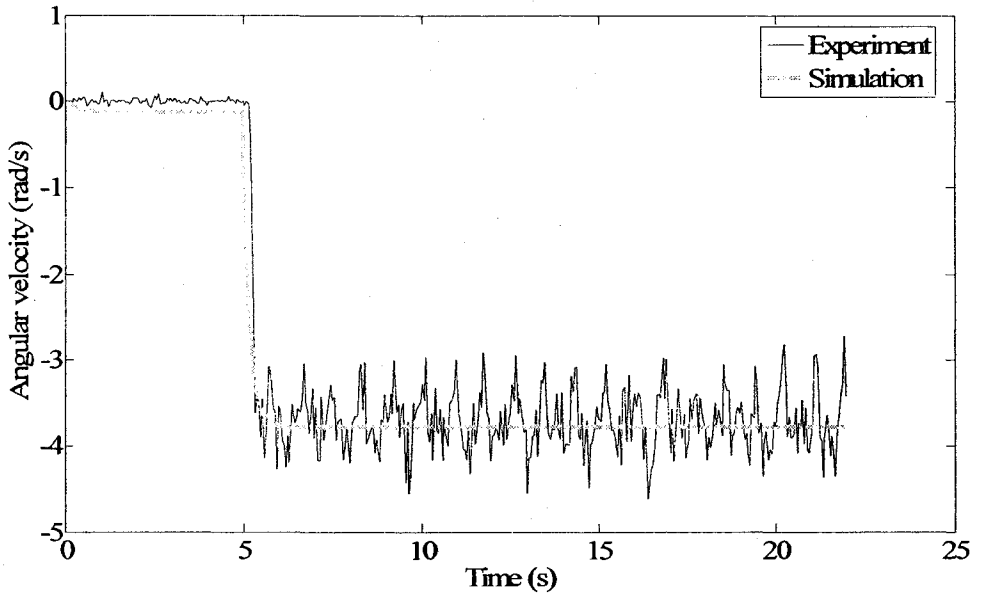

Fig.27. Left motor angular velocity response (IC#2)

3.2.2. Parameter Identification of Wheeled Vehicles

Though, the parameters of motor kinematic equations were identified in the last section, when the motors are installed on the vehicle, the motor model parameters will

need to be adjusted. Because the inertia will increase due to the weight of the vehicle's body and the friction will also increase due to rolling resistance.

Another two sets of experimental data were used in this process of the parameters identification of the wheeled vehicle. The first data set (IC#3) had an initial condition, where $\theta_c = 1.55\text{rad}$, and $U_{wm,L}(t) = U_{wm,R}(t) = 0\text{V}$ ($U_{wm,R}$ and $U_{wm,L}$ denote the voltage applied on the right and left motor respectively) for all $t \le 5\text{s}$, and had a step magnitude of $U_{wm,R}(t) = U_{wm,L}(t) = 0.4\text{V}$ when $t > 5$. The other set of step input (IC#4) consisted of an initial condition of $\theta_c = 0.45\text{rad}$ and $U_{wm,L}(t) = U_{wm,R}(t) = 0\text{V}$ for all $t \le 5\text{s}$, and a step magnitude of $U_{wm,R}(t) = -0.4\text{V}, U_{wm,L}(t) = 0.4\text{V}$ for all $t > 5\text{s}$. The first set mainly produced translational movement of the wheeled vehicle, while the latter caused rotational movement. The parameter identification process would combine the results from both types of movements as shown in Table 3. Please note that the first subscript refers to the number of parameter in (43), and the second one refers to the parameters belong to either left or right motor of the vehicle.

The simulation results are compared with experiment data in diagrams from Fig.28 to Fig.38. It can be seen that the simulation results are close to the experimental data in both translational and rotational movements.

Parameters	IC#3	error bounds	IC#4	error bounds
$a_{1,L}$	58.15	±0.946	60.515	±0.946
$a_{2,L}$	20.348	±0.237	22.718	±3.792
$a_{3,L}$	0.0093	±0.0002	0.0095	±0.0002
$a_{1,R}$	56.37	±2.9932	48.887	±2.9932
$a_{2,R}$	21.008	±0.101	22.107	±1.758
$a_{3,R}$	0.0097	±0.0001	0.0096	±0.0001

Table 3. Estimated wheeled vehicle parameters

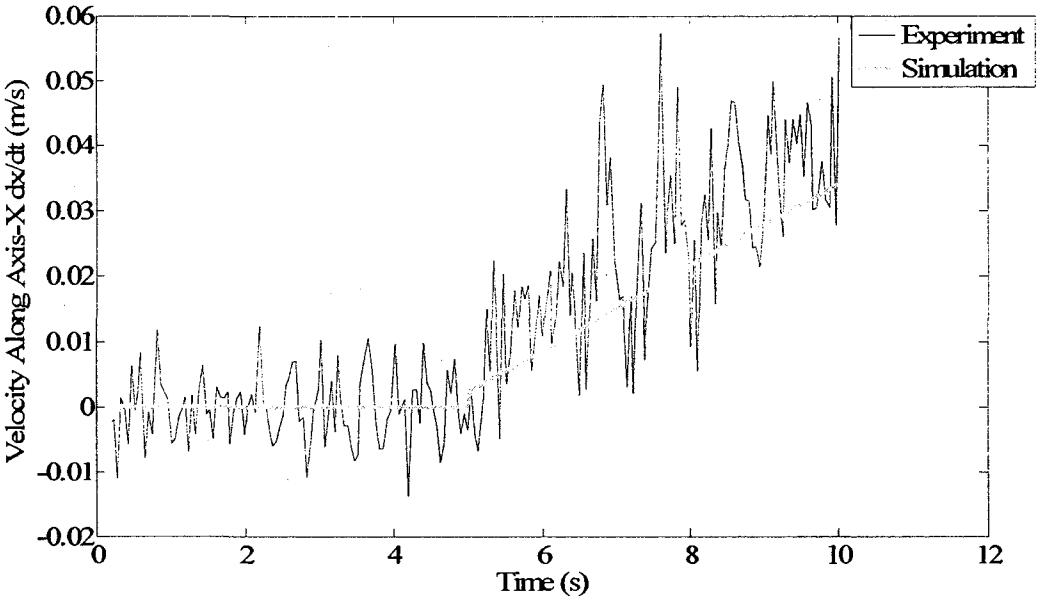

Fig.28. Velocity along X-axis (IC#3)

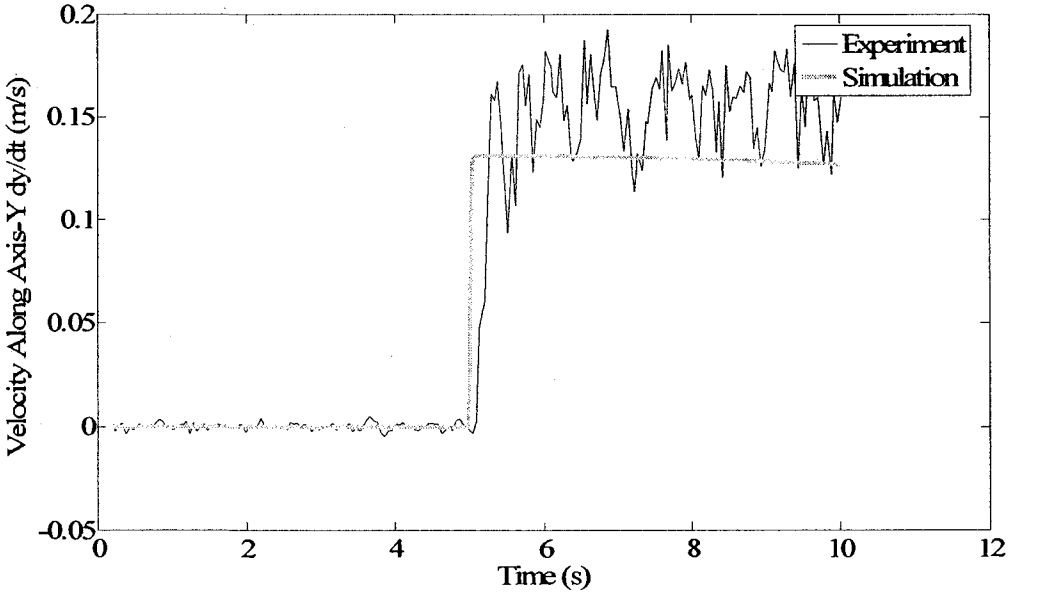

Fig.29. Velocity along Y-axis (IC#3)

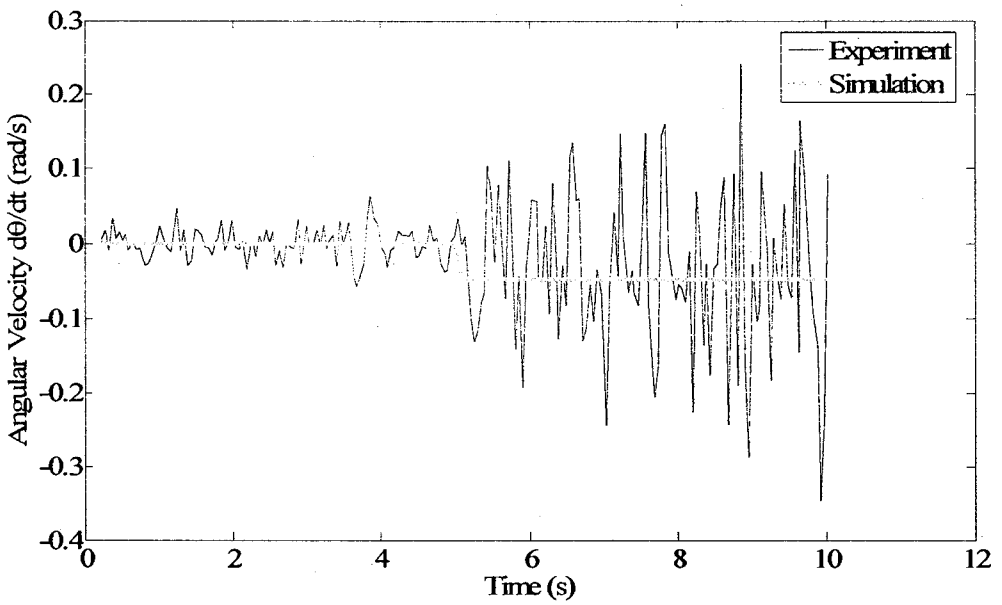

Fig.30. Angular velocity (IC#3)

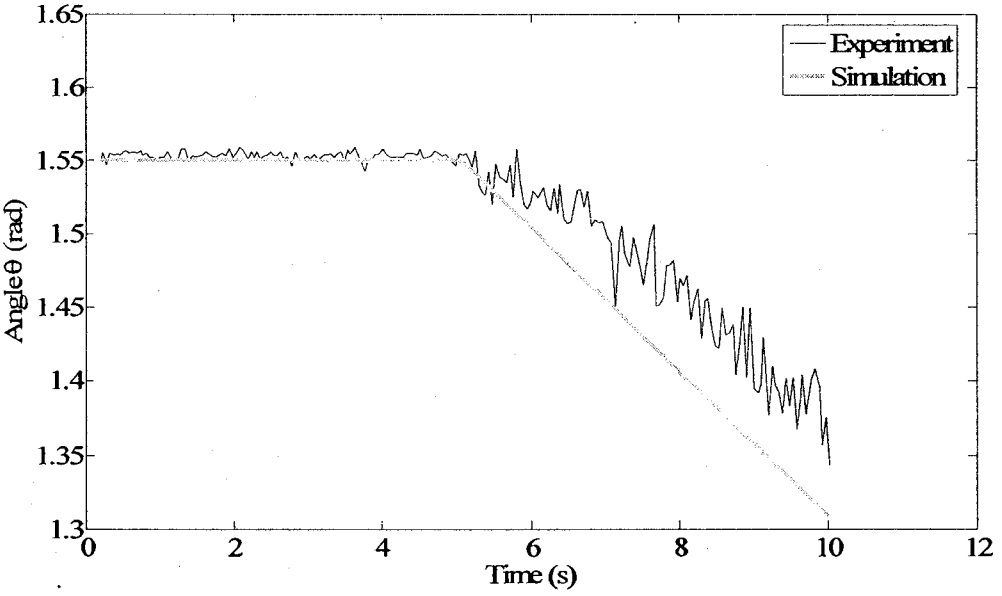

Fig.31. Angle vs. time (IC#3)

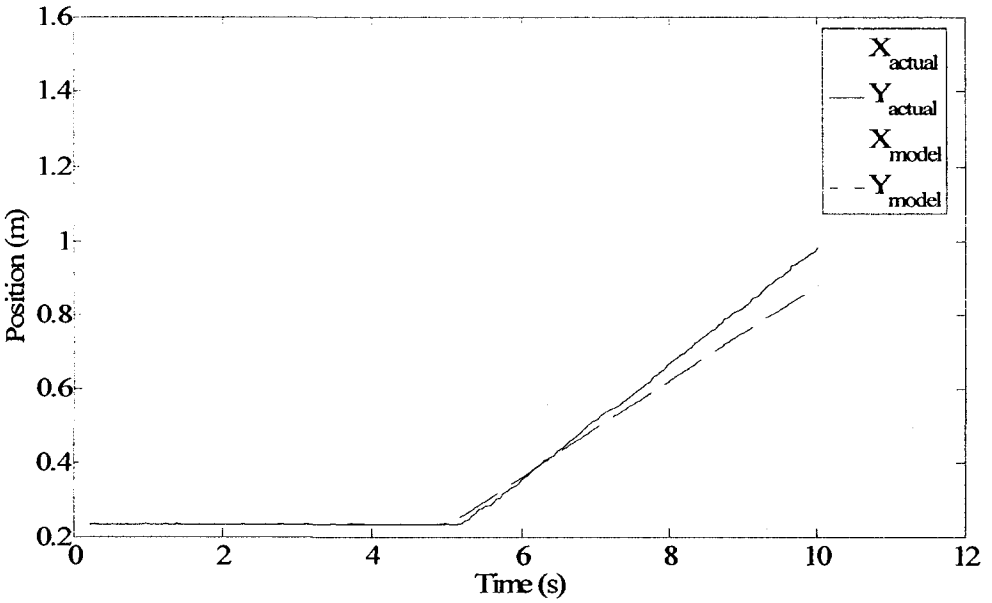

Fig.32. Position vs. time (IC#3)

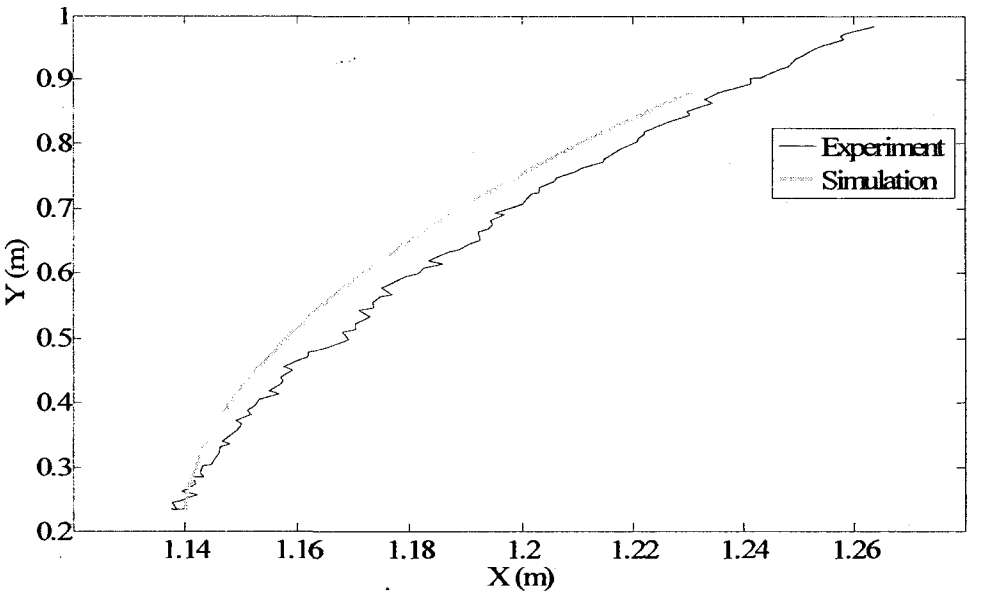

Fig.33. Path (IC#3)

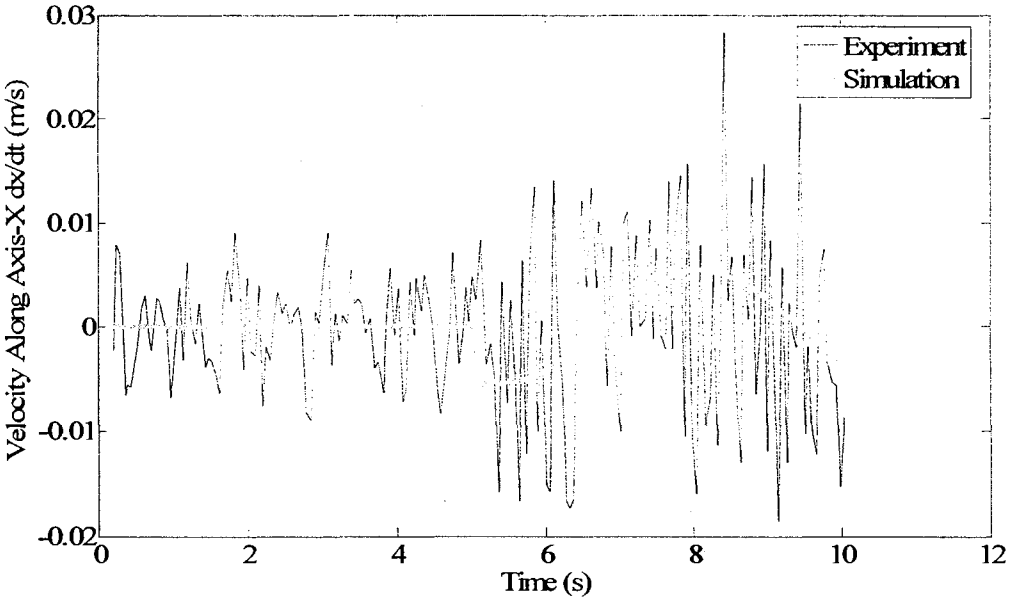

Fig.34. Velocity along X-axis (IC#4)

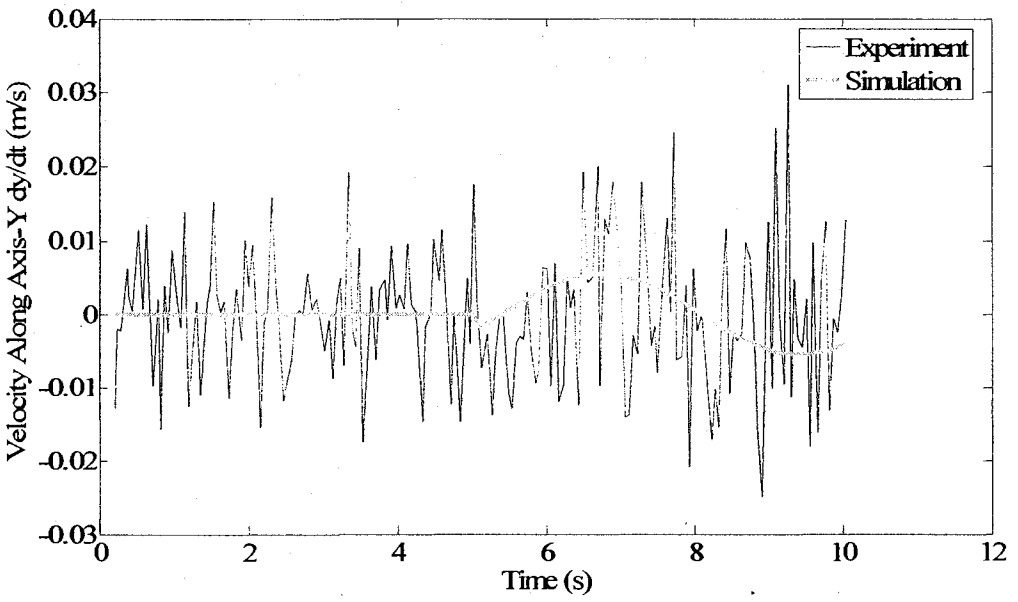

Fig.35. Velocity along Y-axis (IC#4)

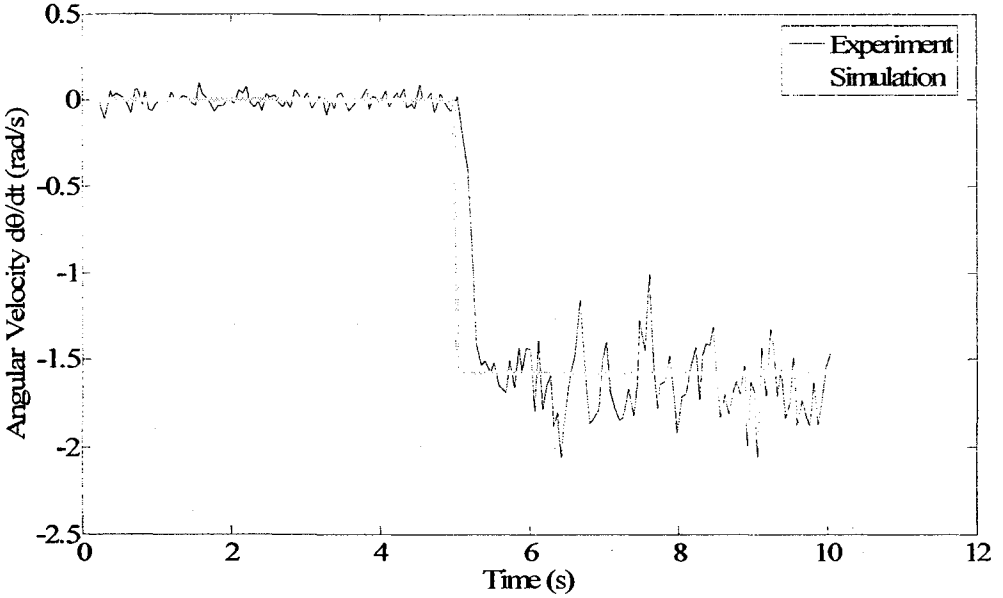

Fig.36. Angular velocity (IC#4)

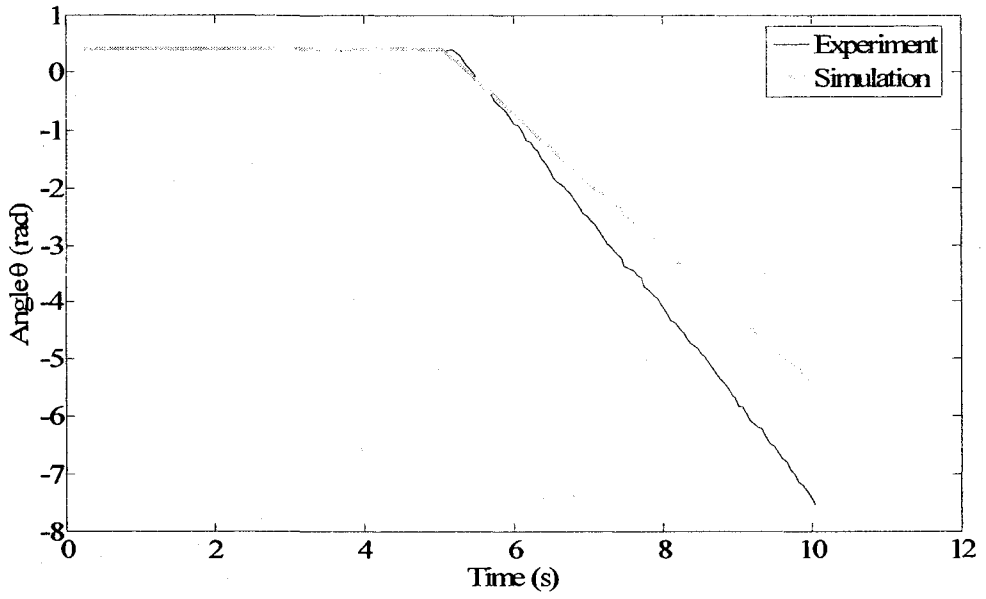

Fig.37. Angle vs. time (IC#4)

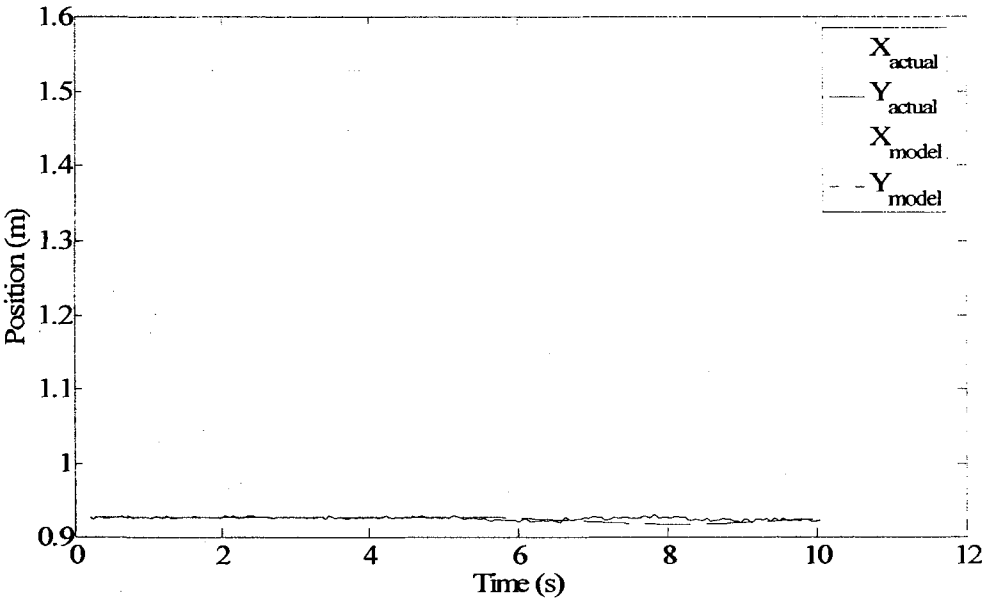

Fig.38. Position vs. time (IC#4)

3.3. Model Verification

The proposed actuator model, identified by the linear least square method, was validated by comparing the actual output to the simulation output of the model for a different data set IC#5, which was not used in the parameter identification process. IC#5 had initial conditions of $\omega_{wm,L}(t) = \omega_{wm,R}(t) = 0.0$ and $U_{wm}(t) = 0.0$ for all $t \leq 0$ and a sinusoidal input of $U_{wm}(t) = 0.5\sin(0.3t)$ V for all $t > 0$. It was adopted in the validation process because it is a standard input and its magnitude is not too big to create saturations. It was applied on both left and right motors. The simulations were performed by using Euler's method with a step size of 0.05 seconds. Fig.39 and Fig.40 show the angular velocity responses of left and right motors versus time, respectively. It is evident that the overall response of the linear model is close to the experimental data, except for some small deviations.

Same as the motors, the parameters of the vehicle were validated by another set of input (IC#6), which was not used in the process of parameter identification, either. IC#6 had an initial condition where $\theta_c = 1.6\,\text{rad}$ and $U_{wm,R}(t) = U_{wm,L}(t) = 0$ for all $t \leq 5\,\text{s}$ with a step magnitude of $U_{wm,R}(t) = 0.5\,\text{V}$ and $U_{wm,L}(t) = 0.7\,\text{V}$ when $t > 5\,\text{s}$. The simulated results are shown and compared with the experiment data in diagrams from Fig.41 to Fig.46. As it is shown in Fig.31, Fig.32, Fig.37, Fig.38, Fig.44 and Fig.45, the simulated angle and position value have relatively large deviation from the experimental data especially when the simulations were approaching to the end. However, the RHC controller used in this thesis does not require a very long optimization horizon (normally one second), hence, the model is adequate for our RHC experiment in the latter chapters.

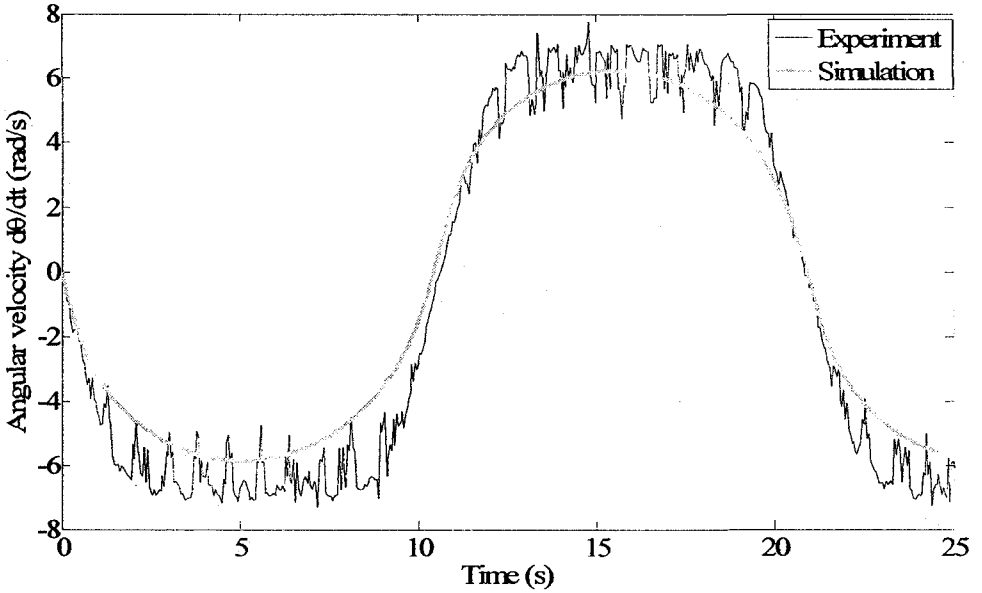

Fig.39. Right motor angular velocity response (IC#5)

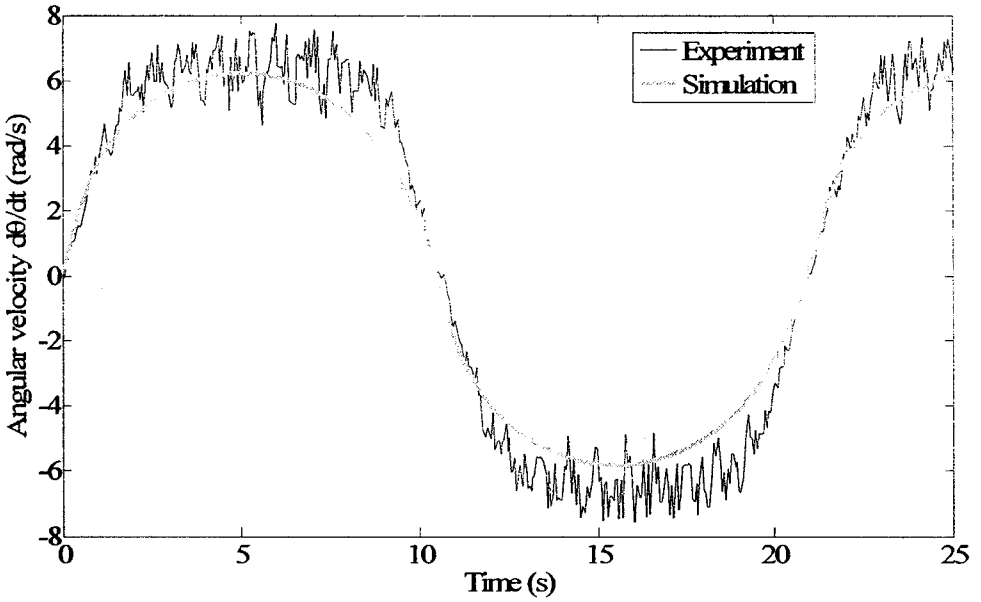

Fig.40. Left motor angular velocity response (IC#5)

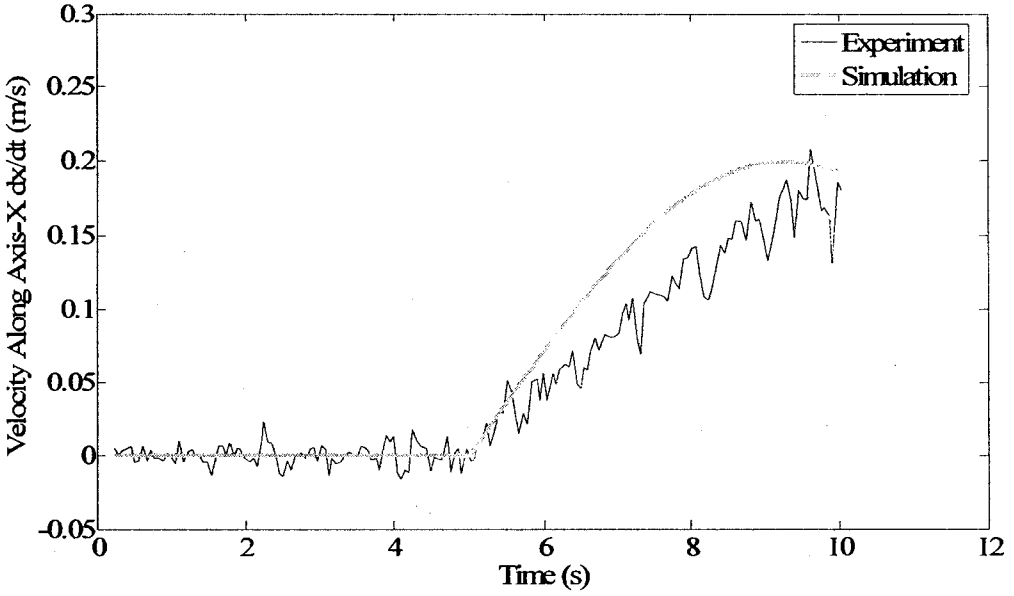

Fig.41. Velocity along X-axis (IC#6)

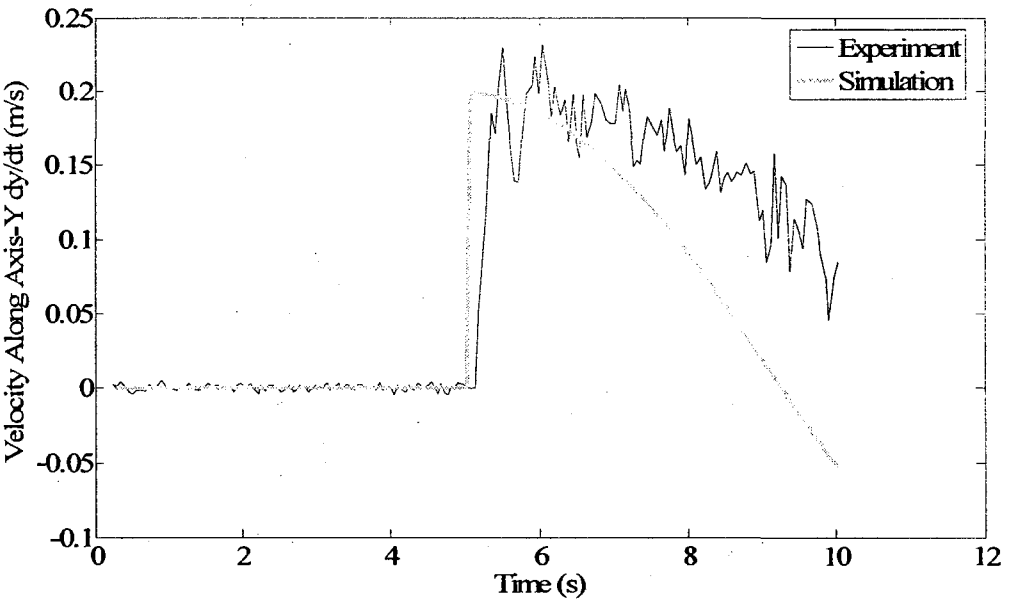

Fig.42. Velocity along Y-axis (IC#6)

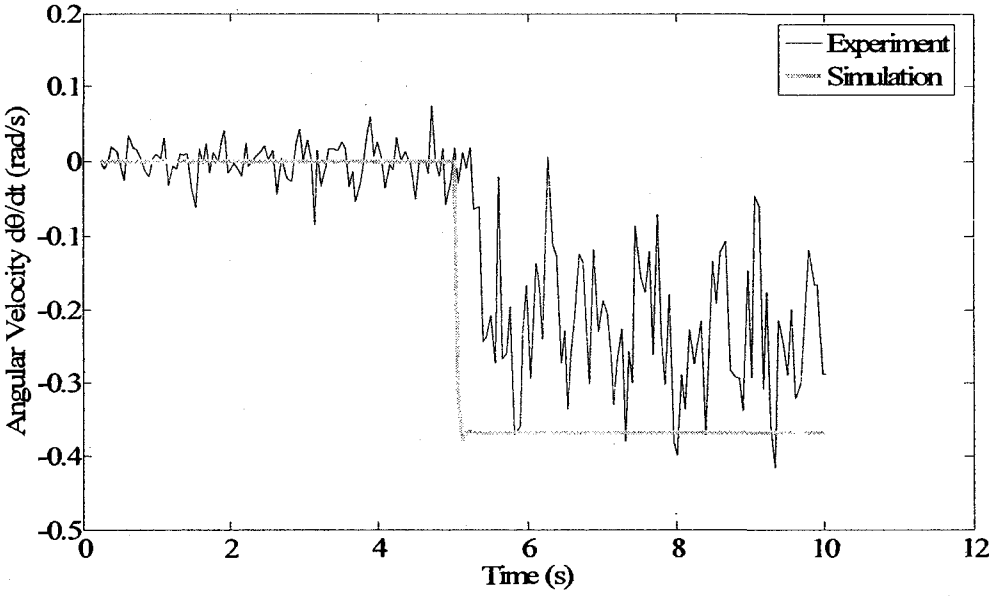

Fig.43. Angular velocity (IC#6)

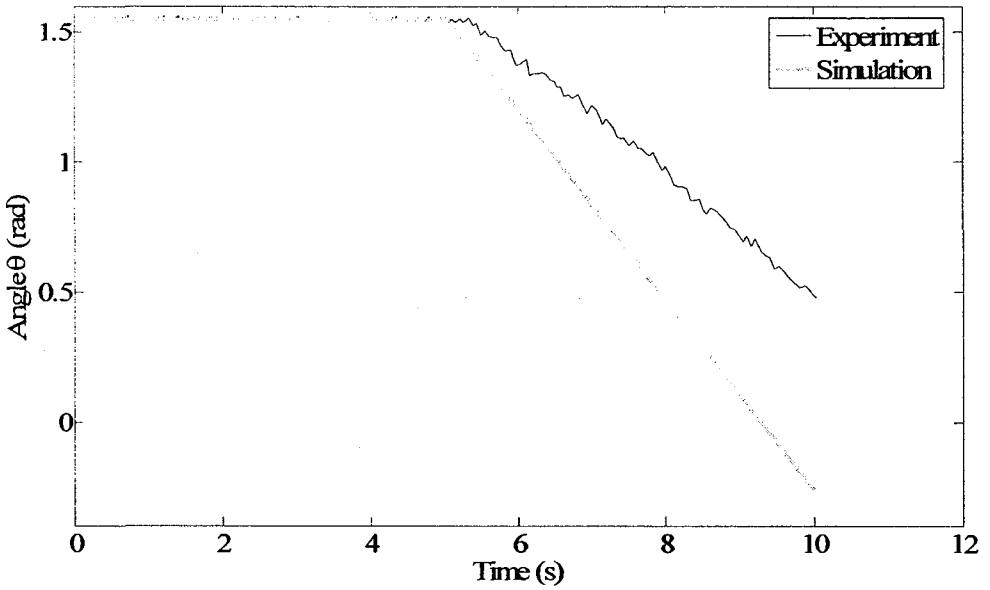

Fig.44. Angle vs. time (IC#6)

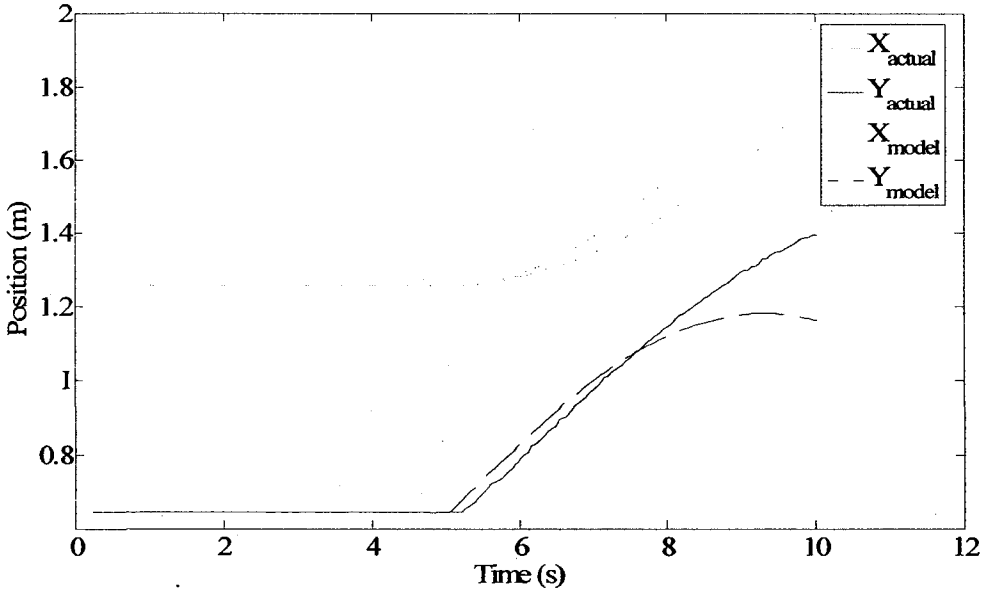

Fig.45. Position vs. time (IC#6)

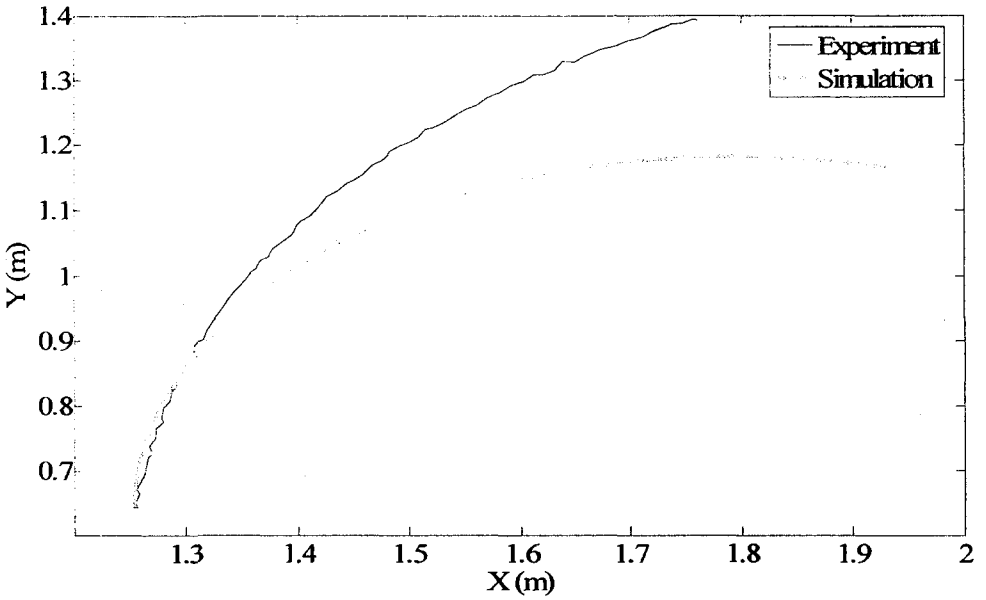

Fig.46. Path (IC#6)

4. Modeling and Identification of Hovercraft Vehicles

The hovercraft vehicles used in the experiments are modified from radio controlled (RC) hovercrafts. By adding two powerful ducted fans on both sides, and an extra fan on the tail, the new hovercraft becomes dynamically similar to a helicopter in 2D environment. This will be a preparatory stage for our future research on applying RHC to miniature helicopters.

4.1. Hovercraft Vehicle Model

The dynamic model of the foresaid modified hovercraft and its actuators are presented in this subsection. The pictures of the hovercraft are shown in Fig.47 and Fig.48.

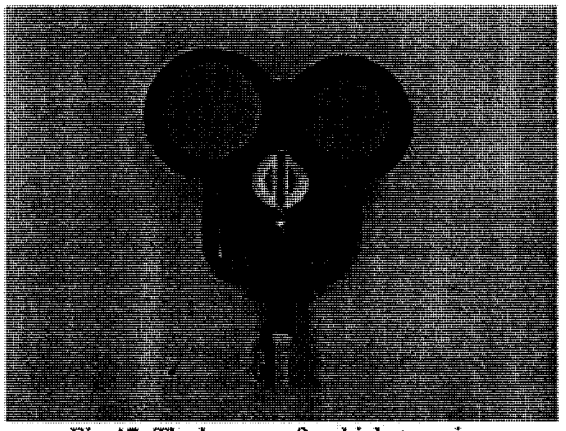

Fig.47. The hovercraft vehicle top view

Fig.48. The hovercraft vehicle perspective view

4.1.1. Dynamic Model of the Hovercraft Actuators

The 3 degree of freedom motion of a hovercraft vehicle is obtained by three powerful ducted fans, which are connected to a servo amplifier. Similar to the wheeled

vehicles, the hovercraft is controlled remotely by a computer via wireless FM radio communication links.

The servo amplifier is able to transfer the voltage applied to the FM transmitter into pulses, changes the pulse width when different voltages applied on the FM radio controller, and uses the different *duty cycle ratio* to change the average voltage applied to the fan's motor; thus manipulates the thrust generated by the motor. Duty cycle ratio is equal to pulse width divided by the period of the pulse.

The average voltage on the motor is obtained by multiplying the voltage level and duty cycle ratio as:

$$U_{level} \cdot K_{ratio} = U_{hMotor} \tag{47}$$

where U_{level} is the voltage level, K_{ratio} is the duty cycle ratio and U_{hMotor} is the resulted average voltage the motor sees.

During the experiments, it is found out that the generated thrust is weaker when positive voltage applied on the motor than the thrust when negative voltage is applied. This is because the amplifier produces different set of pulse width when positive and negative voltages are applied. This discovery can be proved by the following diagram Fig.49.

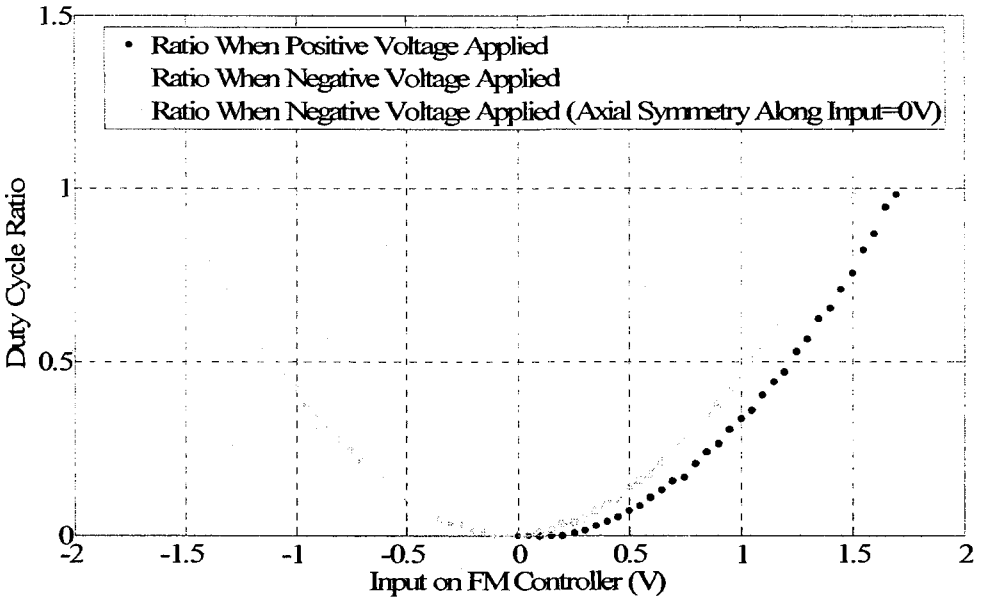

Fig.49. Duty cycle ratio vs. input voltage

It is obvious that when negative voltage applied, the duty cycle ratio increases faster than the case when positive voltage applied, accordingly, the average voltage grows faster. However, the thrust generated by the motor is unable to be measured, therefore in this thesis it is assumed that the thrust is proportional to the average voltage. Therefore, the relationship between the input voltage and the thrust can be found, if the relationship between the input voltage and the duty cycle ratio is established. By using 5^{th} order polynomial method, the following equation is able to depict this relationship:

$$K_{ratio} = \sum_{i=1}^{5} b_i U_h^i \tag{48}$$

where b_i denotes the parameter to be identified, and U_h denotes the input voltage from the FM controller. By combining (47) and (48), we can have:

$$U_{hMotor} = U_{level} \cdot \sum_{i=1}^{5} b_i U_h^i \tag{49}$$

4.1.2. Dynamic Model of the Hovercraft

The dynamic model of the hovercraft is different from the wheeled vehicles in a number of respects. This is mainly because of the difference in their actuators and friction mechanisms. The hovercraft vehicles are actuated by thrust forces from their fans, so the motion is more complicated and unpredictable than the motion of the wheeled vehicles. Furthermore, the frictional force is mostly viscous since the vehicle floats on a cushion of air. Finally, there are no kinematic constraints due to the low friction air cushion.

The equations for hovercrafts are also expressed in the body attached frame (X_B, Y_B) as shown in Fig.50, which are given by the following equations [68]:

$$
\begin{aligned}
F_{hM} &= a_{hM} \cdot U_{level} \cdot \sum_{i=1}^{5} b_i U_{hM}^i \\
F_{hT} &= a_{hT} \cdot U_{level} \cdot \sum_{i=1}^{5} b_i U_{hT}^i \\
\dot{u}_c &= \frac{1}{m_h}(F_{hM} - b_{hu} u) + v_c r_c \\
\dot{v}_c &= -\frac{1}{m_h} b_{hv} v_c - u_c r_c \\
\dot{r}_c &= \frac{l_{hT}}{J_h}(F_{hT} - b_{hr} r_c) - \frac{l_{hM}}{J_h} F_{hM}
\end{aligned}
\tag{50}
$$

where l_{hT} denotes the distance between the block in which tail rotor is installed and the center of the hovercraft, l_{hM} denotes the distance between the two blocks in which main rotors are installed, m_h denotes the mass of the hovercraft, J_h is the mass moment of inertia of the hovercraft, u_c and v_c are the velocity along X_B and Y_B axes, respectively, b_{hu}, b_{hv}, and b_{hr} denote the viscous friction coefficients along X_B, Y_B, and Z_B (pointing out from the diagram, not shown in Fig.50) directions, respectively. a_{hM} and

a_{hT} denote the coefficients of the linear relationship between the average voltage on the main and tail motors and the produced thrust by the ducted fans, F_{hM} and F_{hT} represents the thrust generated by the main and tail rotors respectively, and U_{hM} and U_{hT} represent the input voltage applied to the main and tail motors, respectively.

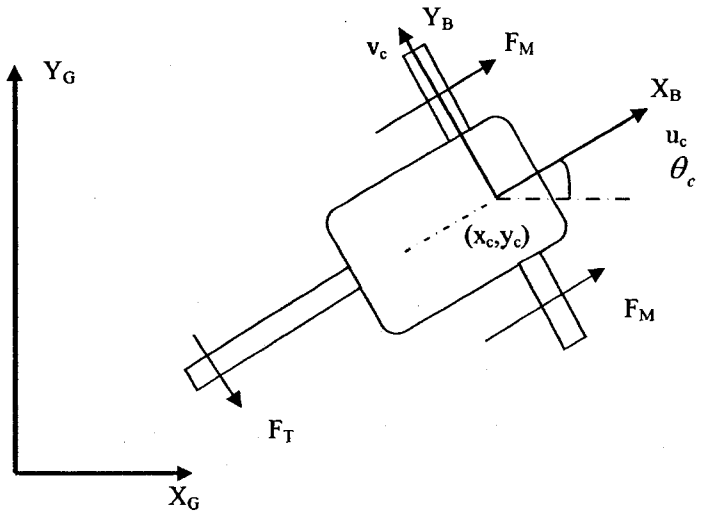

Fig.50. The hovercraft vehicle's schematic model

4.1.3. State Equations of the Hovercraft Vehicles

The state equations of the hovercraft vehicles are obtained by combining the dynamic equations of the actuators, and the kinematic equations of the hovercraft.

$$
\begin{aligned}
F_{hM} &= a_{hM} \cdot U_{level} \cdot \sum_{i=1}^{5} b_i U_{hM}^i \\
F_{hT} &= a_{hT} \cdot U_{level} \cdot \sum_{i=1}^{5} b_i U_{hT}^i \\
\dot{u}_c &= \frac{1}{m_h}(F_{hM} - b_{hu} u) + v_c r_c \\
\dot{v}_c &= -\frac{1}{m_h} b_{hv} v_c - u_c r_c \\
\dot{r}_c &= \frac{l_{hT}}{J_h}(F_{hT} - b_{hr} r_c) - \frac{l_{hM}}{J_h} F_{hM} \\
\dot{x}_c &= u_c \cos\theta_c - v_c \sin\theta_c \\
\dot{y}_c &= u_c \sin\theta_c + v_c \cos\theta_c \\
\dot{\theta}_c &= r_c
\end{aligned}
\tag{51}
$$

4.2. Parameter Identification

In this section, the procedure for parameter identification of the hovercraft vehicle and its actuators is presented.

4.2.1. Parameter Identification of the Actuators

Similar to the process of parameter identification of the actuators of the wheeled vehicle, this part will use (45) to find the parameters associated with the motors, and by combining (47) and (48), we can change (46) into the following equations:

$$
\begin{aligned}
\mathbf{A}_{ls} &= \begin{bmatrix} U_h & U_h^2 & U_h^3 & U_h^4 & U_h^5 \end{bmatrix} \\
\mathbf{x}_{ls} &= \begin{bmatrix} b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix}^T \\
\mathbf{b}_{ls} &= \begin{bmatrix} \frac{U_{hMotor}}{U_{level}} \end{bmatrix}
\end{aligned}
\tag{52}
$$

and through experiment, it is discovered that

$$U_{level} = +7.0\,V \text{ , if } U_{hMotor} > 0.0\,V$$
$$U_{level} = -7.0\,V \text{ , if } U_{hMotor} < 0.0\,V \tag{53}$$

Although the identification process is similar to the ones described in previous sections, the parameters of the main and tail motors should be calculated by different data sets. This is because, in the experiment, both positive and negative inputs will be applied to the tail motor for rotational motion of the hovercraft, but only negative voltage will be applied to the main motor, since the hovercraft only needs to move forward and the thrust is stronger when negative voltage is applied. Also, please note that, for the purpose of convenience and tradition, the input for the main motors will multiple -1 after being used, so that we could say the hovercraft moves forward when positive voltage is applied. This modification is only for the habit of the author, and will not change the dynamics of the system.

There were two sets of data chosen for the tail motor. These two data sets (IC#7 and IC#8) had the same initial conditions of $U_{hMotor}(t) = 0.0$ and $U_{wh}(t) = 0.0$ for all $t \le 5s$. IC#7 had a step input of $U_{wh}(t) = 1.25\,V$, and IC#8 had a step input of $U_{wm}(t) = -1.25\,V$ for all when $t > 5s$. Also for the main motors, there were two sets of data employed. One of them was IC#7 and the other one was IC#9, which had the same initial conditions of $U_{hMotor}(t) = 0.0$ and $U_{wh}(t) = 0.0$ for all $t \le 5s$, and a step input of $U_{wh}(t) = 0.45\,V$ for all when $t > 5s$.

The identified parameters for the tail and main motors are listed in Table 4 and Table 5 respectively, along with the nominal parameters obtained from the average

numerical values of the identified parameters from IC#7, IC#8 and IC#9, and the estimated error bounds.

Parameters	IC#7	error bounds	IC#8	error bounds
b_5	-0.2958	±0.06265	-0.31633	±0.0.06265
b_4	0.09188	±0.0457	0.07315	±0.0475
b_3	2.032	±0.0973	2.0818	±0.0973
b_2	-0.4095	±0.229	-0.26236	±0.229
b_1	0.856	±0.02907	0.8337	±0.02907

Table 4. Estimated motor parameters for the hovercraft tail motor

Fig.51 shows the time history of the average voltage applied on the tail motor for IC#7, and Fig.52 shows the result for IC#8. The corresponding cases for the main motor for IC#7 and IC#8 are shown in Fig.52 and Fig.53, respectively. In addition, the input-output relationship of the tail and main motors are obtained by using the identified parameters, and illustrated in Fig.55 and Fig.56, respectively. Furthermore, there is no need to validate the parameters, since Fig.55 and Fig.56 have proved the accuracy of the model.

Parameters	IC#7	error bounds	IC#9	error bounds
b_5	-0.2981	±0.0595	-0.3176	±0.00036
b_4	1.32	±0.3173	1.45	±0.00533
b_3	1.986	±0.2753	1.845	±0.0128
b_2	3.892	±1.5391	2.907	±0.1995
b_1	-0.07508	±0.0074	-0.081	±0.00266

Table 5. Estimated motor parameters for the hovercraft main motor

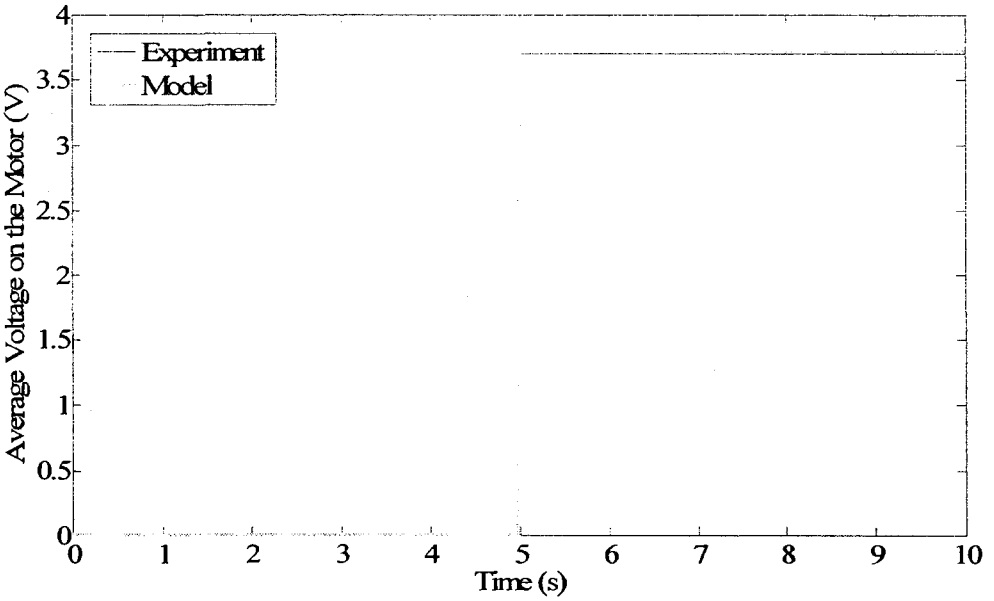

Fig.51. Average Voltage on the tail motor (IC#7)

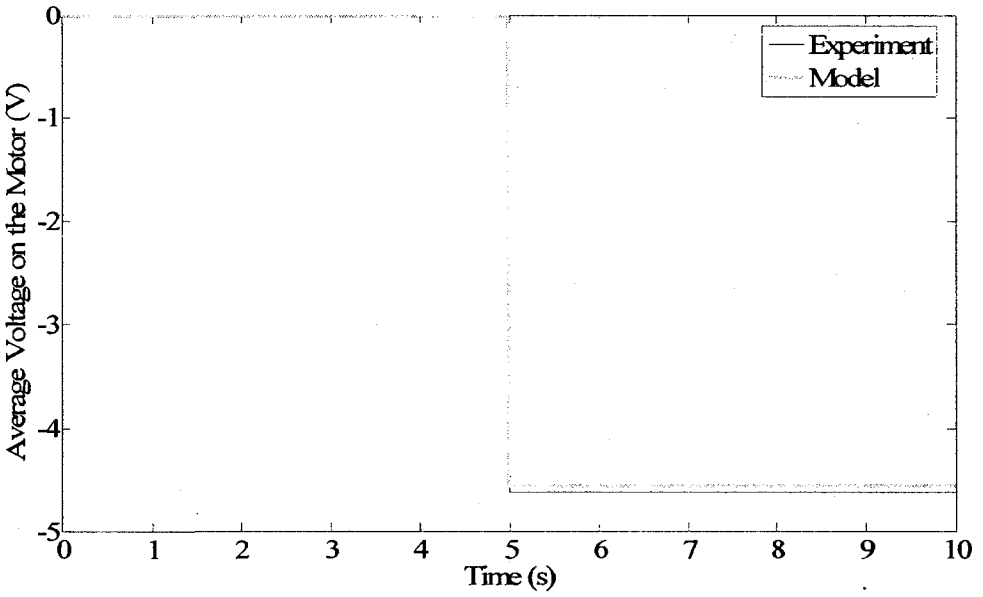

Fig.52. Average Voltage on the tail motor (IC#8)

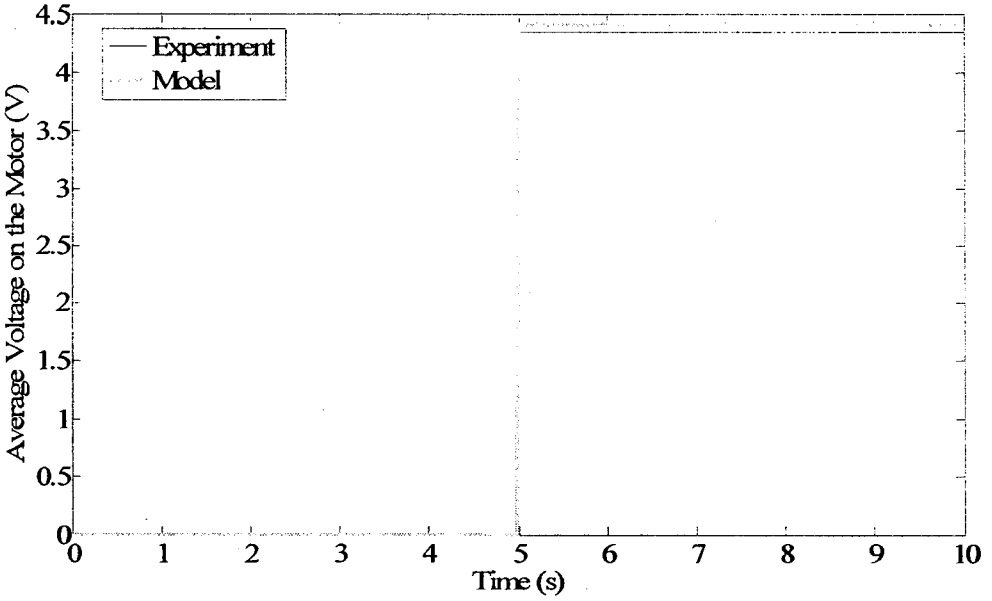

Fig.53. Average Voltage on the main motor (IC#7)

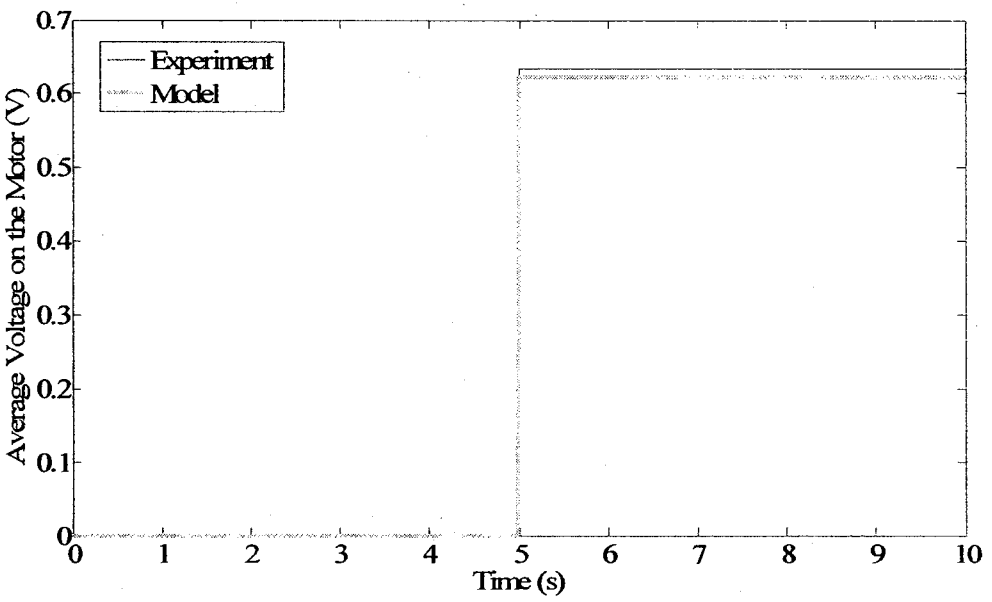

Fig.54. Average Voltage on the main motor (IC#9)

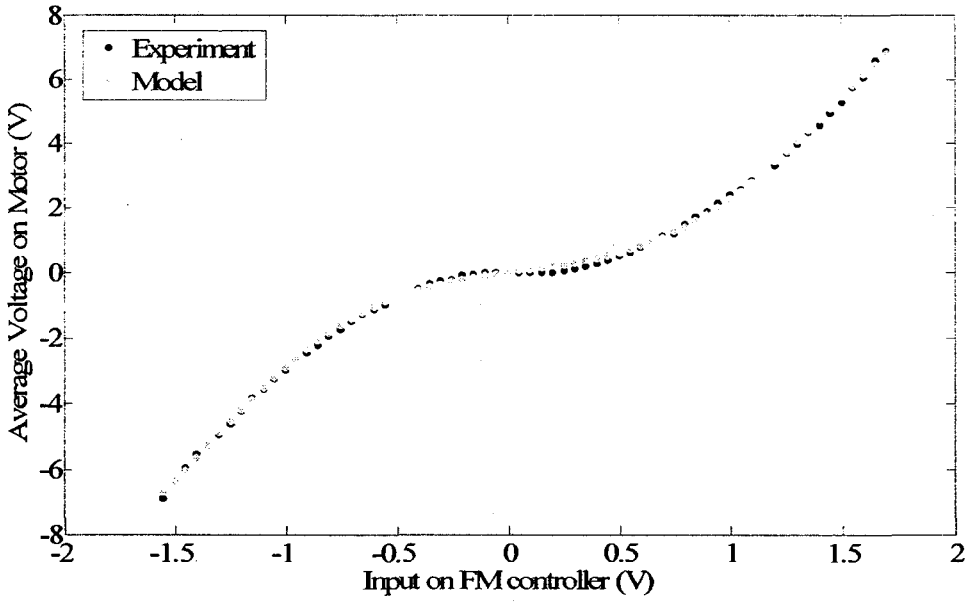

Fig.55. Average Voltage on tail motor vs. input voltage on FM controller

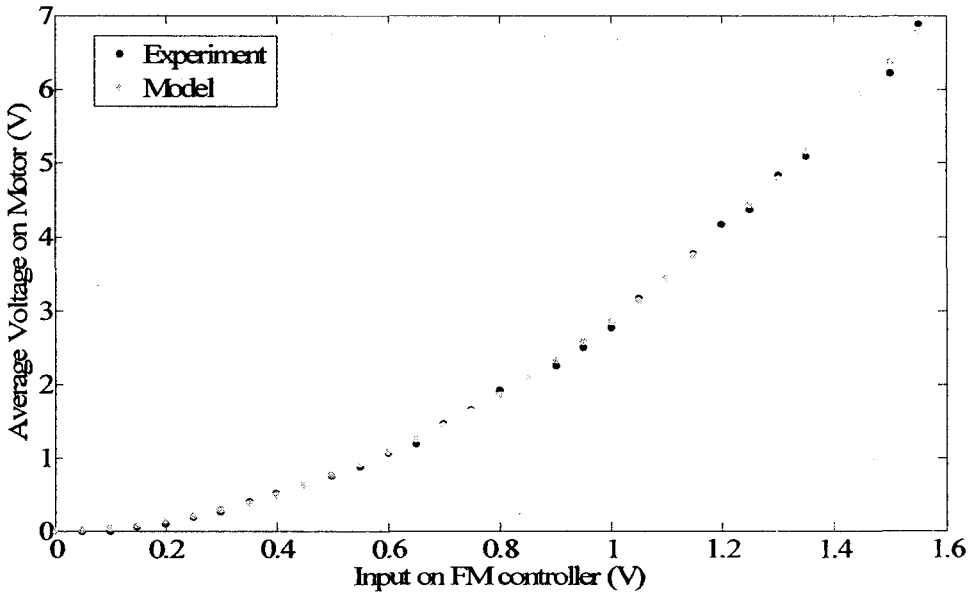

Fig.56. Average Voltage on main motor vs. input voltage on FM controller

4.2.2. Parameter Identification of the Hovercraft Vehicle

Unlike the actuators for the wheeled vehicle, there is no need to re-identify the parameters of the motors, because there are no extra loads or modifications applied on the actuators when they are installed on the hovercraft.

In order to identify the parameters of the hovercraft, equation (50) needs to be rewritten for the sake of reducing the number of parameters to be calculated.

$$\begin{aligned} \dot{r}_c &= p_1 U_{hT} - p_2 r_c - p_3 U_{hM} \\ \dot{u}_c &= p_4 U_{hM} - p_5 u_c + v_c r_c \\ \dot{v}_c &= -p_6 v_c - u_c r_c \end{aligned} \tag{54}$$

where $p_1 = \frac{l_{hT} a_T}{J_h}$, $p_2 = \frac{l_{hT} b_{hr}}{J_h}$, $p_3 = \frac{l_{hM} a_M}{J_h}$, $p_4 = \frac{a_{hM}}{m_h}$, $p_5 = \frac{b_{hu}}{m_h}$, $p_6 = \frac{b_{hv}}{m_h}$.

Following the procedures discussed in the above sections by using least square curve fit to the experimental data, (54) can be modified in the form of (55) and solved by using pseudoinverse approach.

$$\begin{aligned} &\mathbf{A}_{ls}\mathbf{x}_{ls} = \mathbf{b}_{ls} \\ &\mathbf{A}_{ls} = \begin{bmatrix} U_{hT} & -r_c & -U_{hM} & 0 & 0 & 0 \\ 0 & 0 & 0 & U_{hM} & -u_c & 0 \\ 0 & 0 & 0 & 0 & 0 & -v_c \end{bmatrix}, \\ &\mathbf{b}_{ls} = \begin{bmatrix} \dot{r}_c \\ \dot{u}_c - v_c r_c \\ \dot{v}_c + u_c r_c \end{bmatrix} \\ &\mathbf{x}_{ls} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \end{bmatrix}^T \end{aligned} \tag{55}$$

Four sets of experimental data with different initial conditions (ICs) are used in this process of the parameter identification. They all included step input responses with the same initial conditions, where $u_c = 0$, $v_c = 0$, $r_c = 0$ and $U_{hM}(t) = U_{hT}(t) = 0\,V$ for all $t \le 5s$. The first data set (IC#10) had a step magnitude of

$U_{hM}(t)=1.5V, U_{hT}(t)=0V$ when $t>5s$, while the second set (IC#11) had $U_{hM}(t)=1.7V, U_{hT}(t)=0V$ when $t>5s$. These two data sets were responsible for the translational motion of the hovercraft. The other two sets of step inputs consisted of two step inputs, in which $U_{hM}(t)=0V$, $U_{hT}(t)=1.2V$ (IC#12) and $U_{hM}(t)=0V$, $U_{hT}(t)=-1.2V$ (IC#13) for all $t>5s$, and they were responsible for the rotational motion of the hovercraft. The identification results are shown in Table 6, and their results are shown in Fig.57 to Fig.74. Please note the "0"s in Table 6 indicate either there is not enough data to identify that parameter, or the identified parameter is not reliable due to lack of information. It is apparent to see from these diagrams that, despite of minor disagreement with the experiment data, the overall performance of the system can be accurately reflected by the identified parameters.

Parameters	IC#10	IC#11	IC#12	IC#13
p_1	0	0	1.6017	1.5341
p_2	3.0079	2.4731	2.5128	2.2314
p_3	0.4	0.4	0	0
p_4	13.2313	16.312	16.5771	14.7901
p_5	1.0891	0.1369	0.1227	0.1437
p_6	2.6117	2.6034	2.6275	2.7591

Table 6. Estimated hovercraft parameters

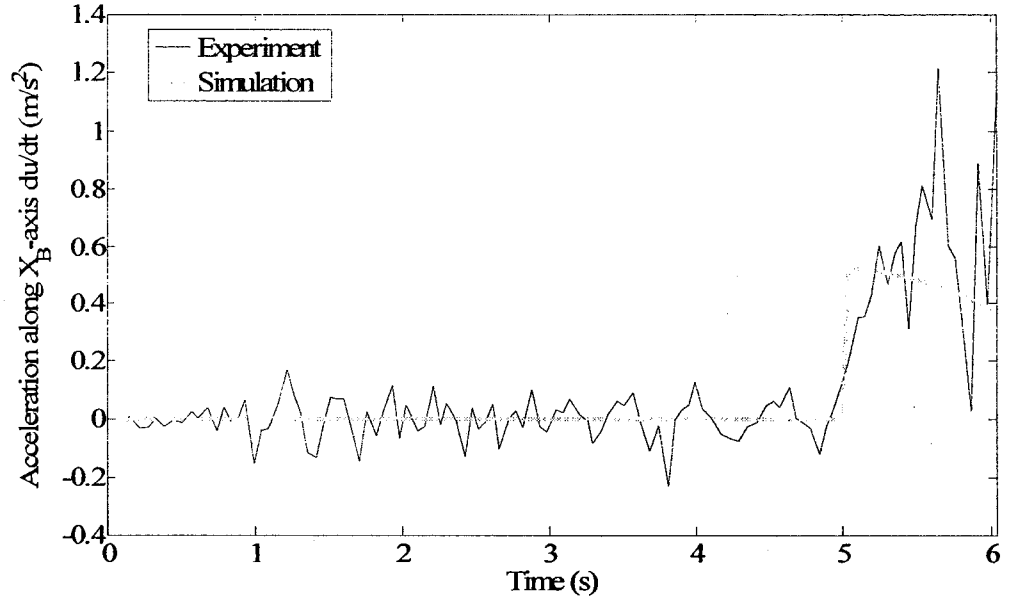

Fig.57. Acceleration along X_B-axis (IC#10)

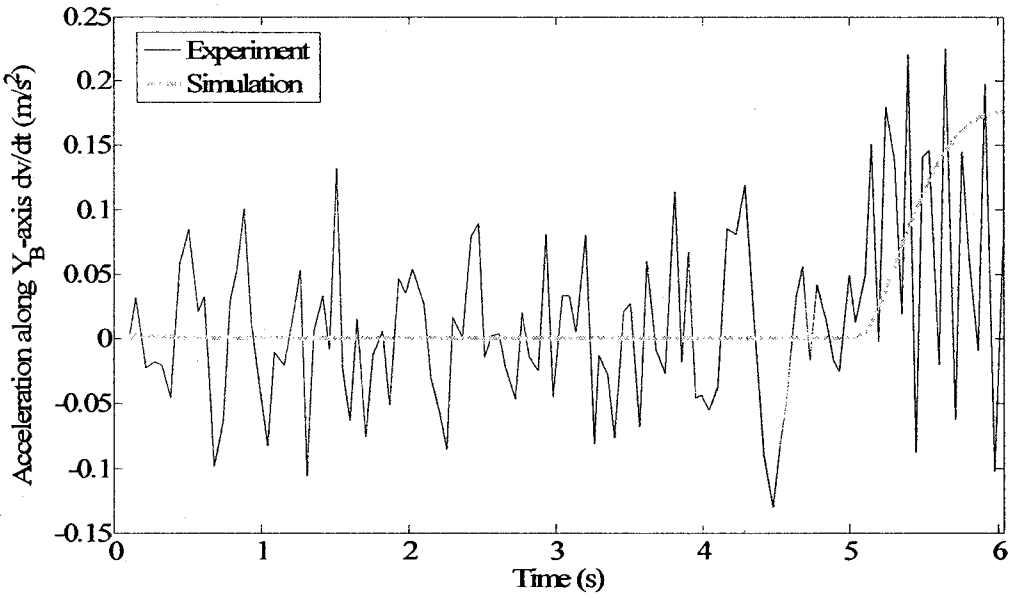

Fig.58. Acceleration along Y_B-axis (IC#10)

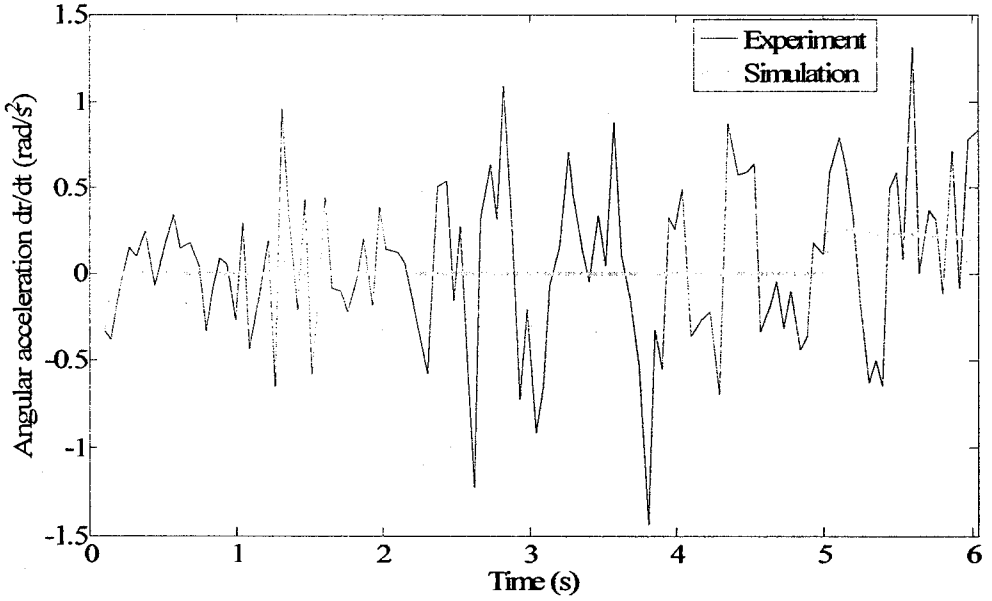

Fig.59. Angular Acceleration (IC#10)

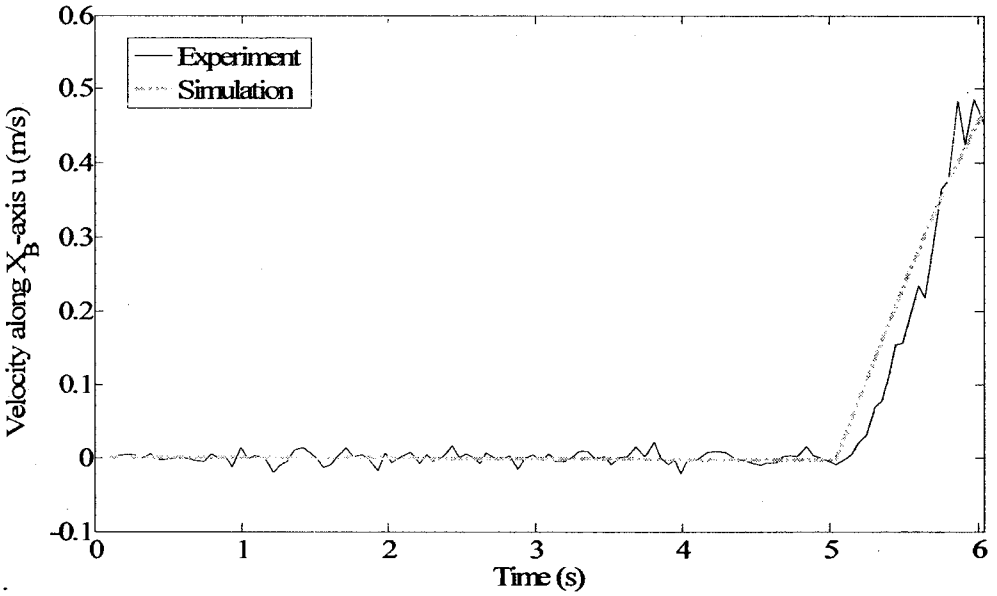

Fig.60. Velocity along X_B-axis (IC#10)

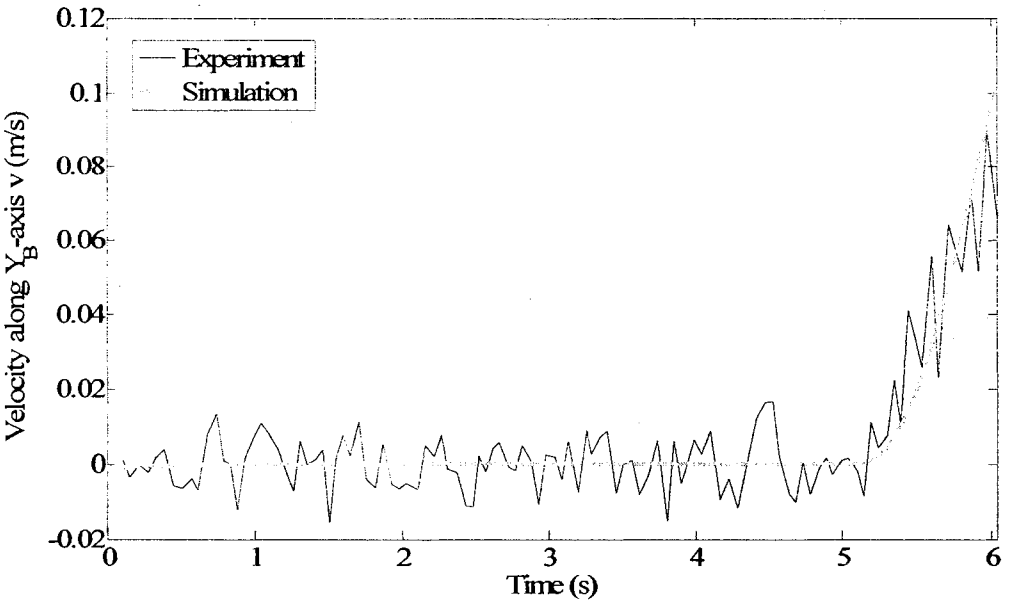

Fig.61. Velocity along Y_B-axis (IC#10)

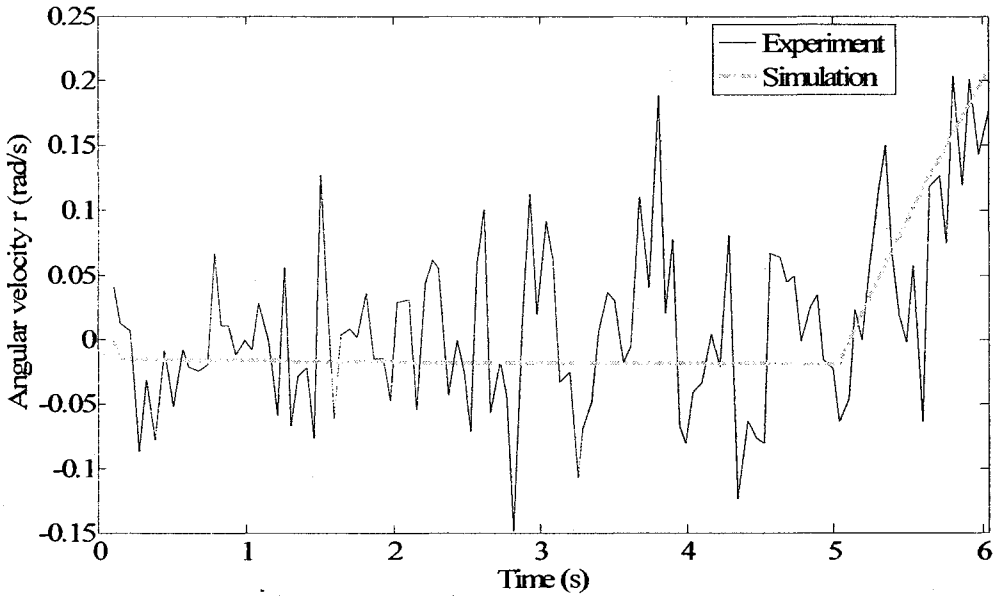

Fig.62. Angular Velocity (IC#10)

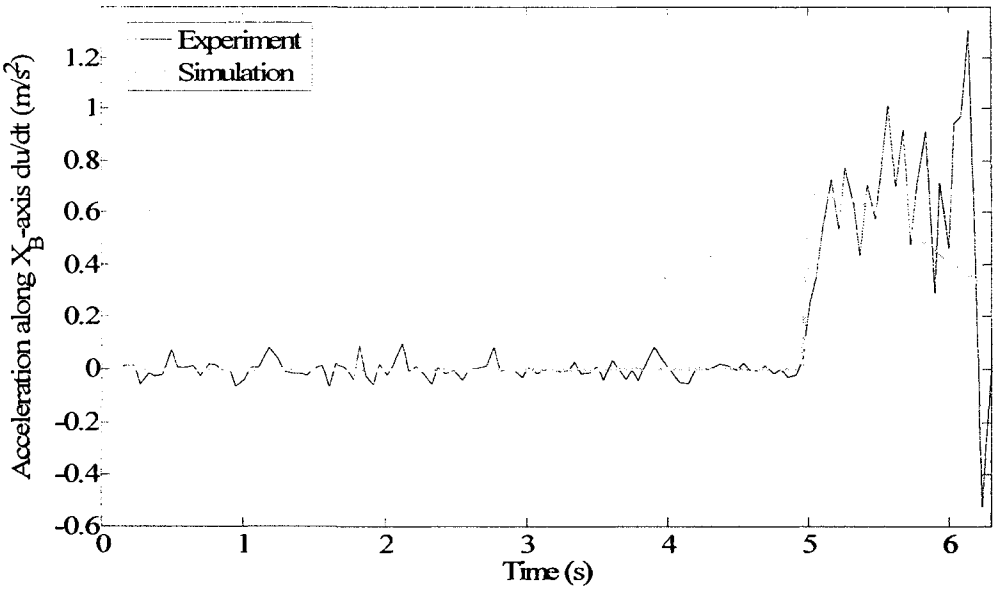

Fig.63. Acceleration along X_B-axis (IC#11)

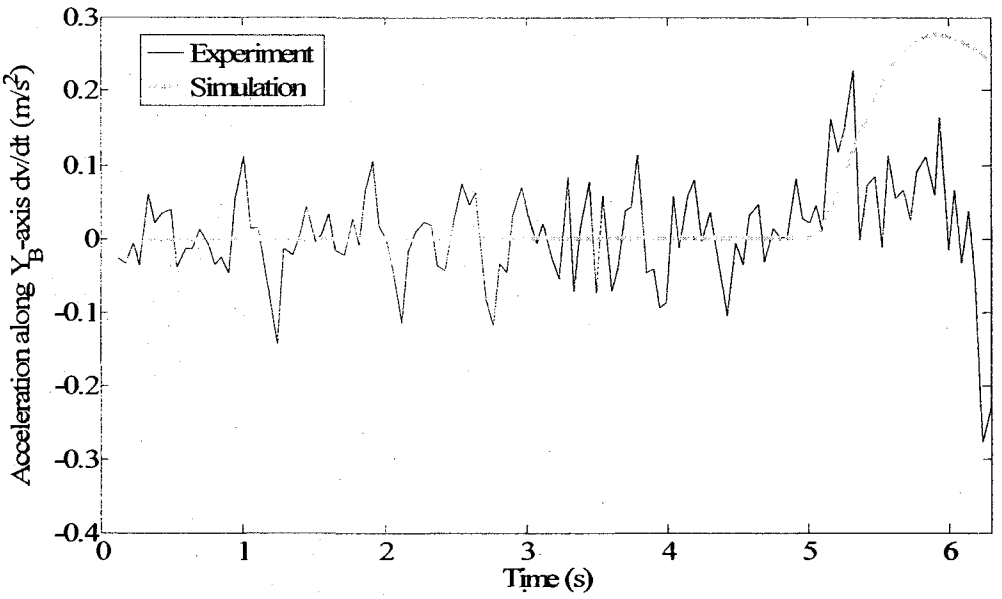

Fig.64. Acceleration along Y_B-axis (IC#11)

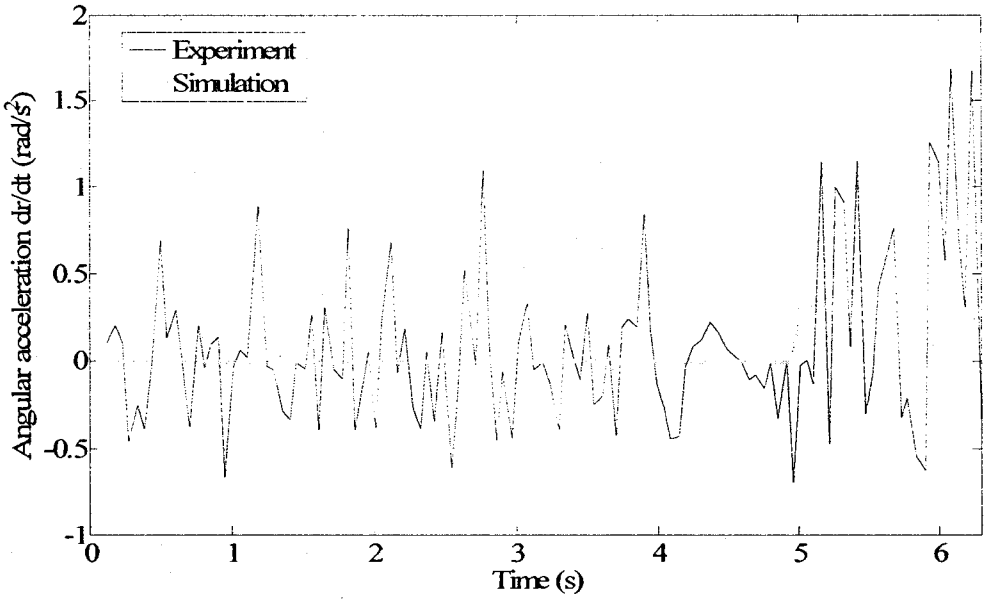

Fig.65. Angular Acceleration (IC#11)

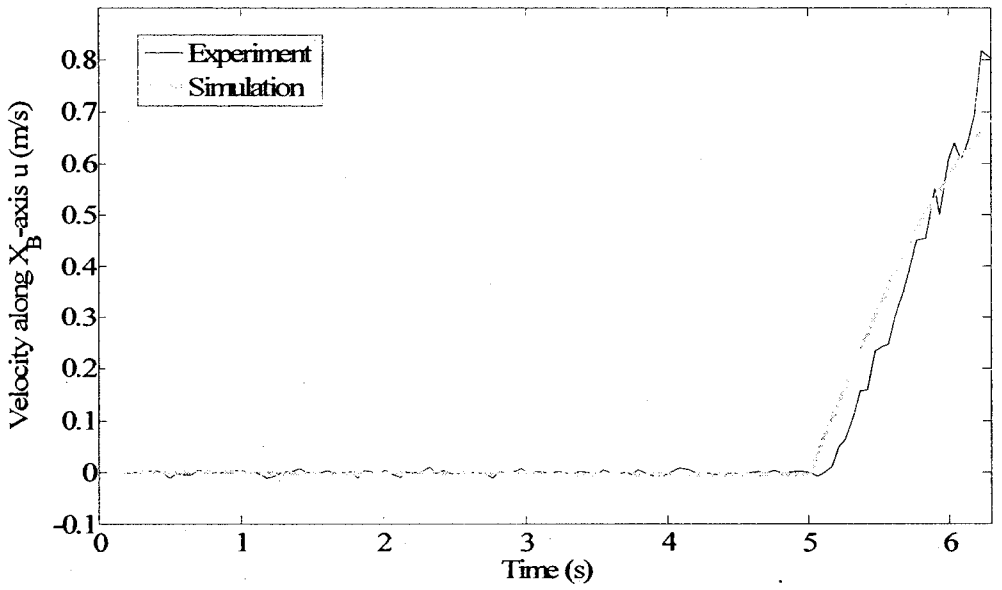

Fig.66. Velocity along X_B-axis (IC#11)

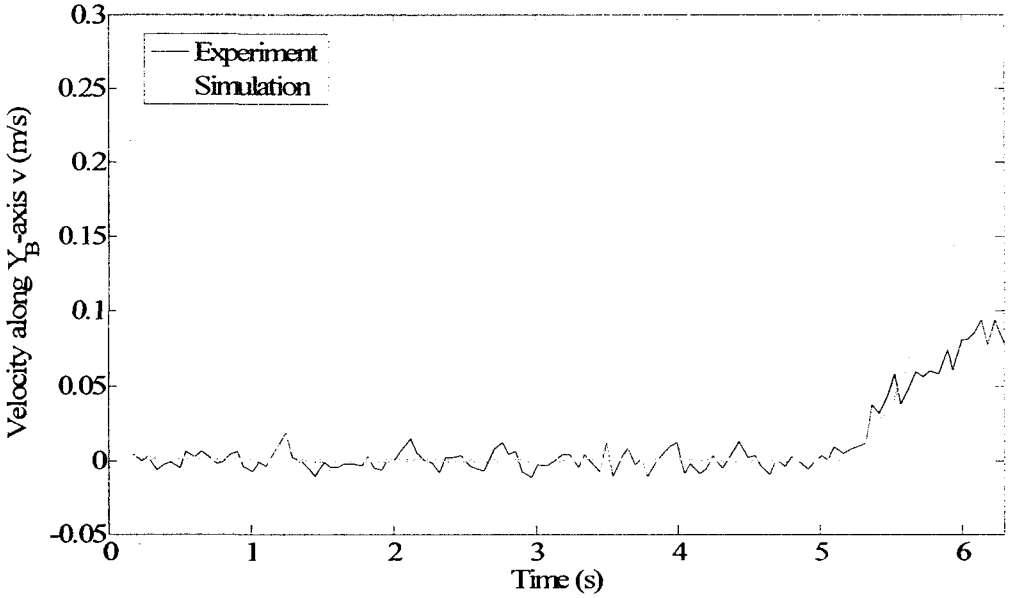

Fig.67. Velocity along Y_B-axis (IC#11)

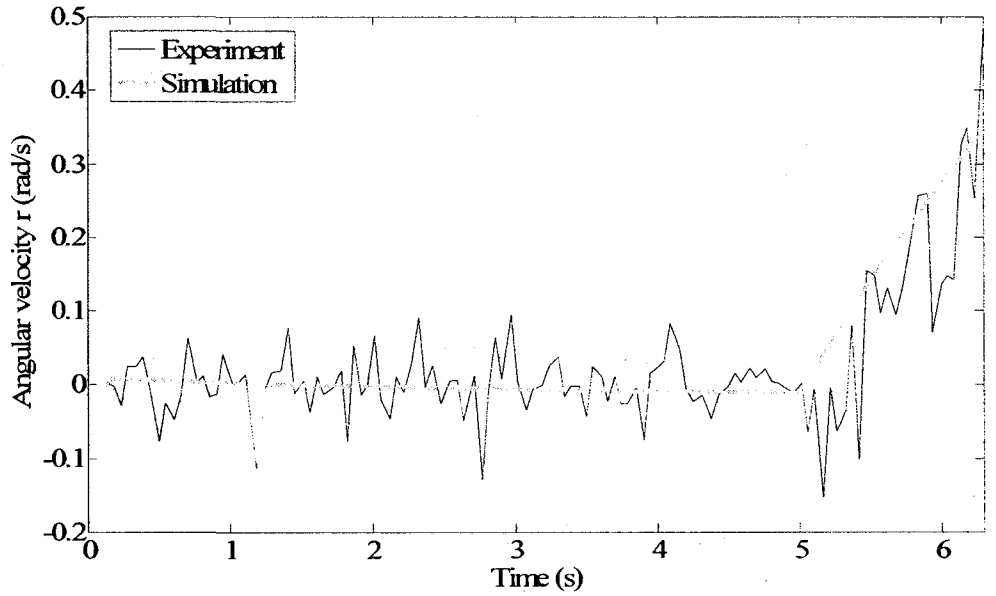

Fig.68. Angular Velocity (IC#11)

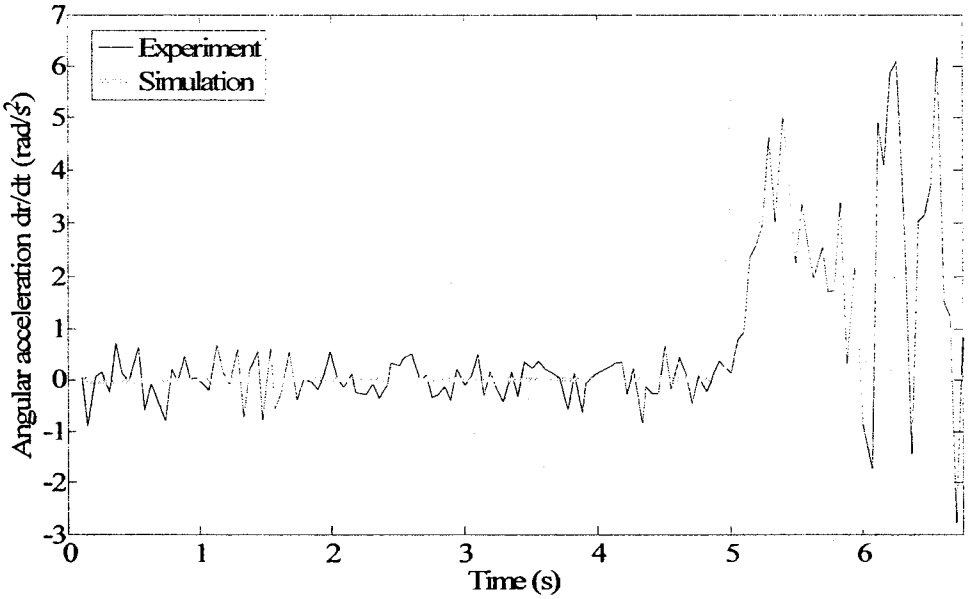

Fig.69. Angular Acceleration (IC#12)

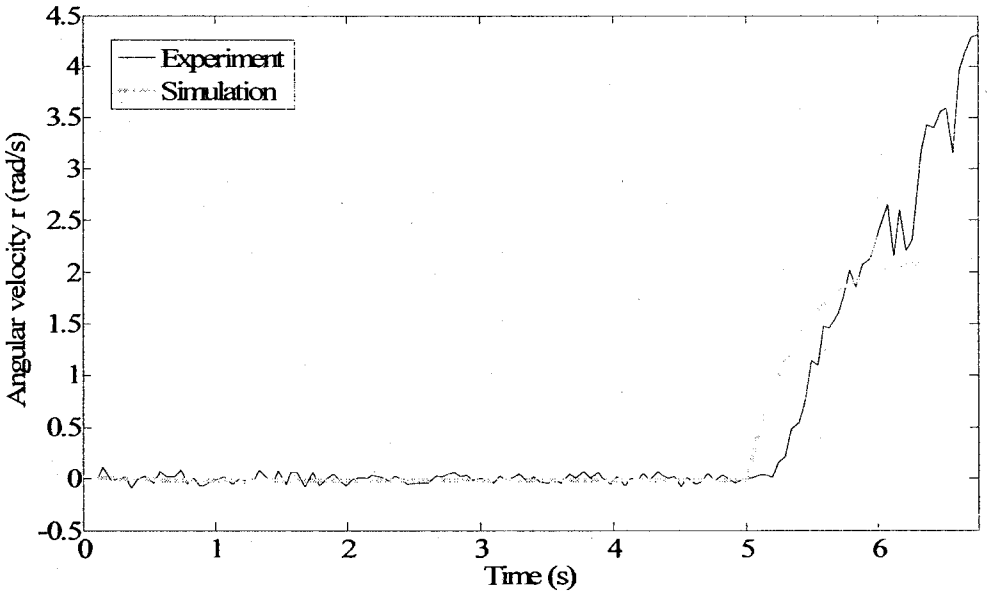

Fig.70. Angular Velocity (IC#12)

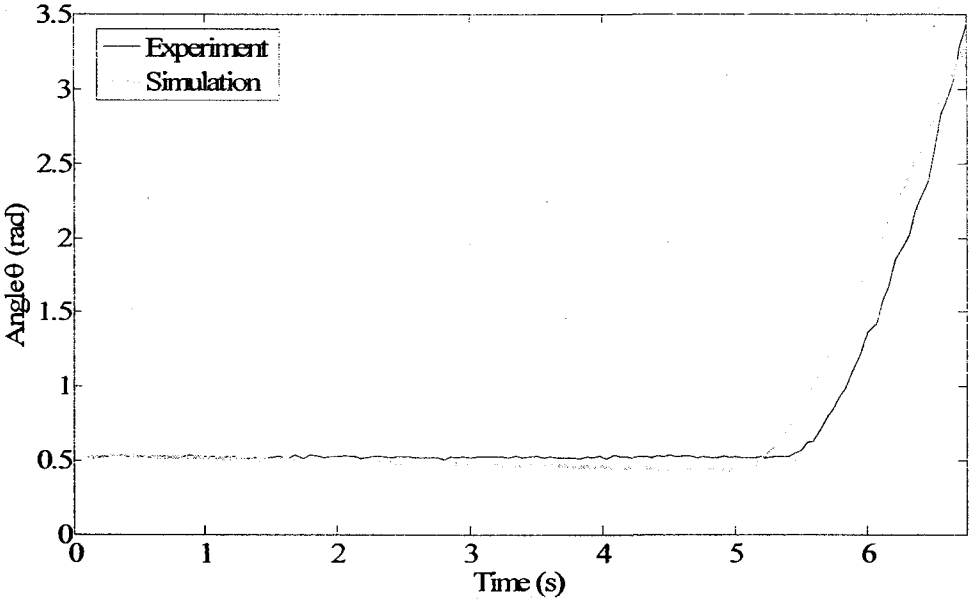

Fig.71. Angle (IC#12)

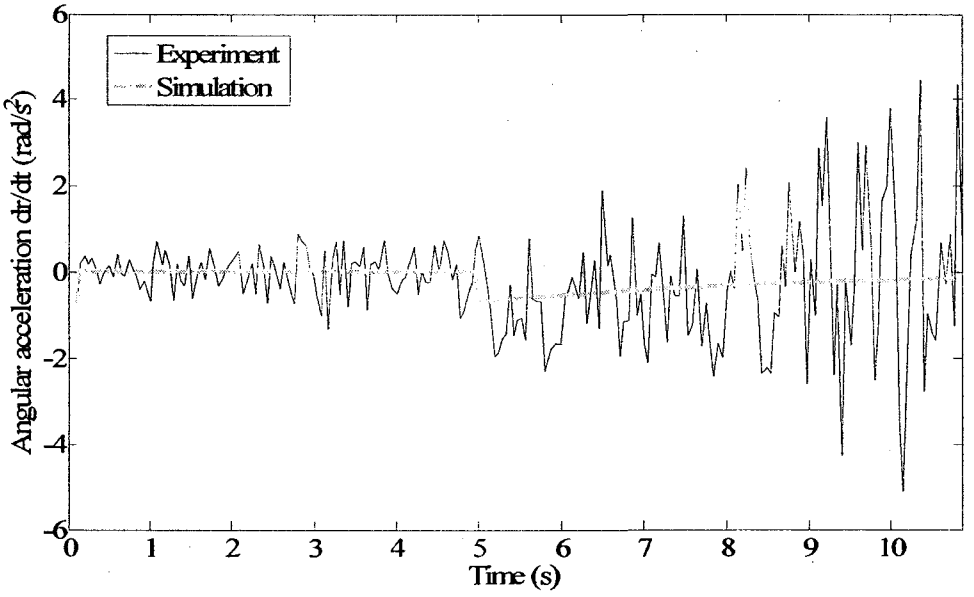

Fig.72. Angular Acceleration (IC#13)

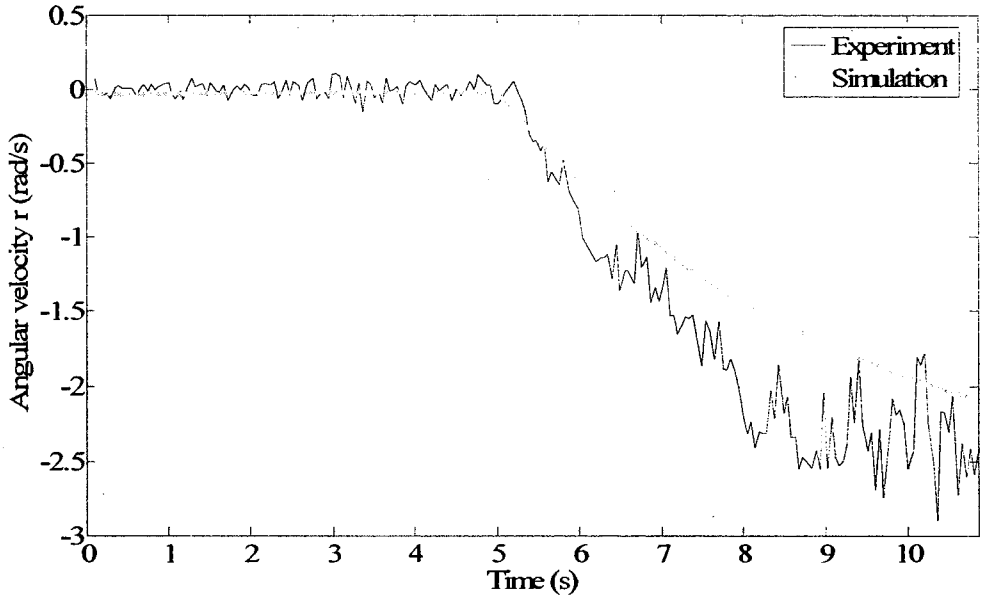

Fig.73. Angular Velocity (IC#13)

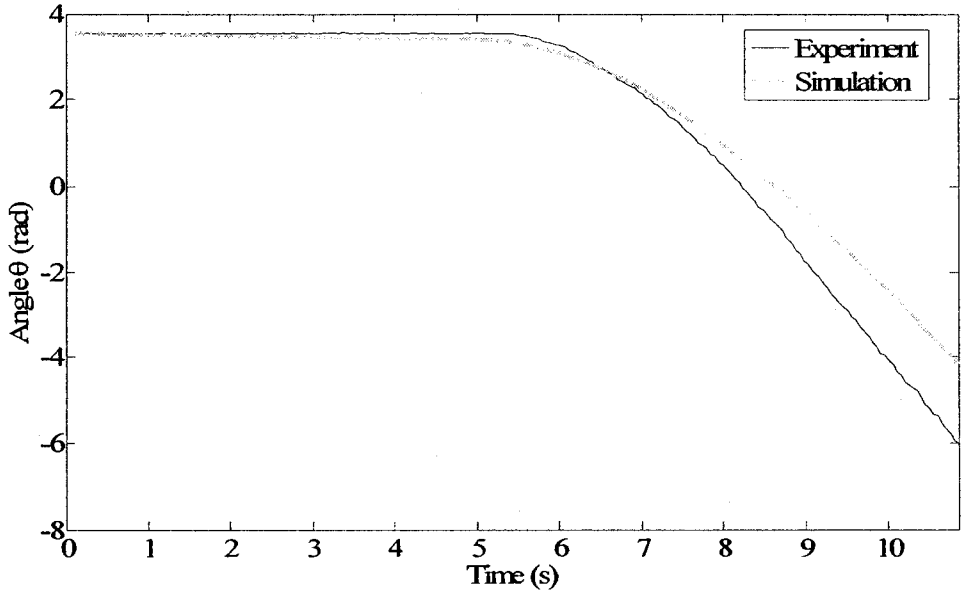

Fig.74. Angle (IC#13)

4.3. Model Verification

Similar to the section of wheeled vehicle parameter identification, the nominal model is validated by a set of inputs that was not used in the identifying process.

However, as stated earlier, the verification of the ducted fan system is not necessary, since Fig.55 and Fig.56 have already validated the accuracy.

The data set used in this section is IC#14, which was not used in the process of parameter identification. It included step input responses with the same initial conditions, where $u_c = 0$, $v_c = 0$, $r_c = 0$ and $U_{hM}(t) = U_{hT}(t) = 0\,V$ for all $t \le 5s$, and a step magnitude of $U_{hM}(t) = 1.4\,V$ and $U_{hT}(t) = 0.7\,V$ when $t > 5s$. The results are shown in the diagrams from Fig.75 to Fig.82. Same as the model validation section in the previous chapter, the validation process is necessary only for a short time, since the RHC experiments in the following chapters do not need very long optimization and execution horizons.

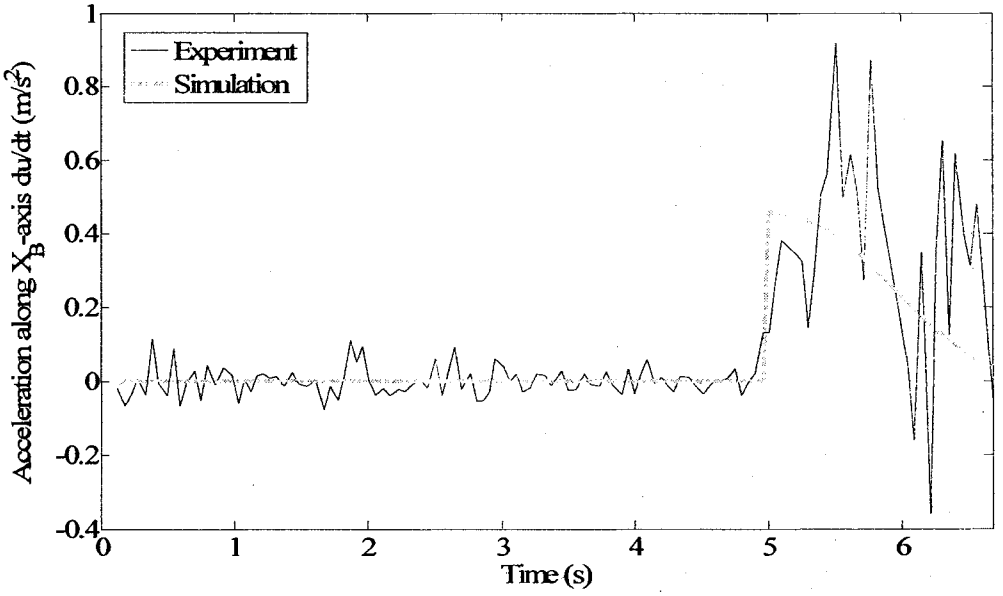

Fig.75. Acceleration along X_B-axis (IC#14)

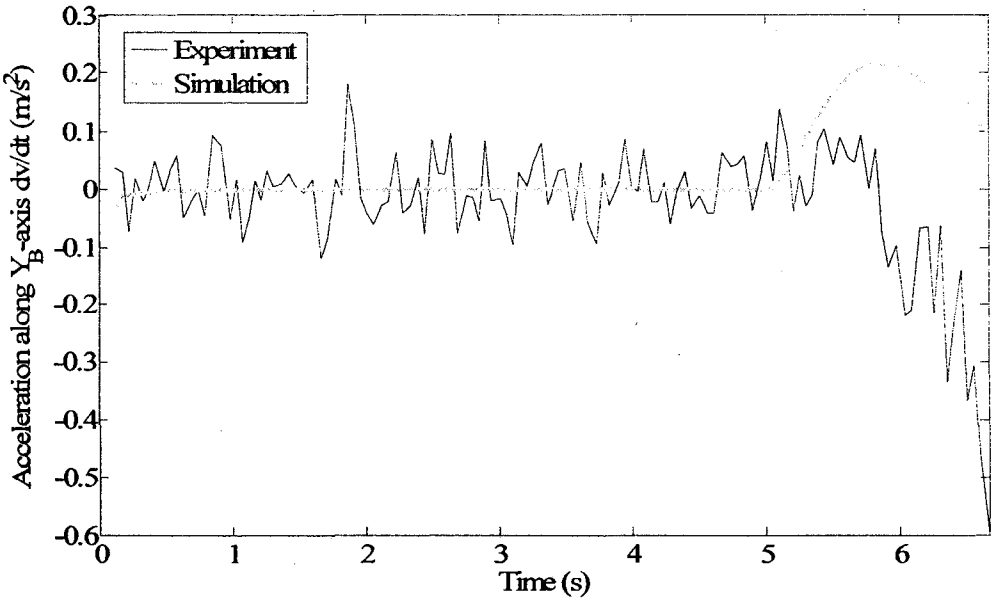

Fig.76. Acceleration along Y_B-axis (IC#14)

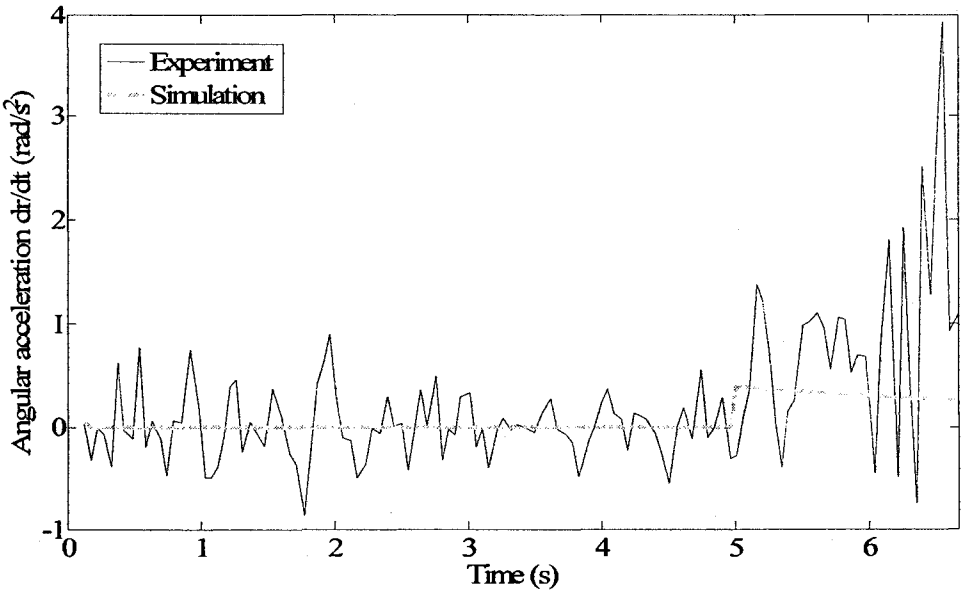

Fig.77. Angular Acceleration (IC#14)

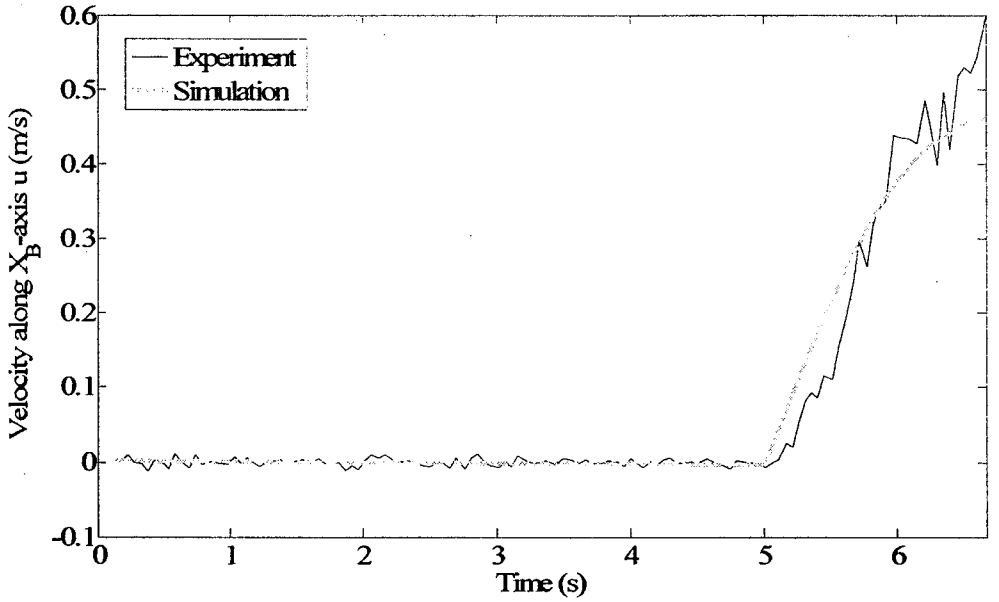

Fig.78. Velocity along X_B-axis (IC#14)

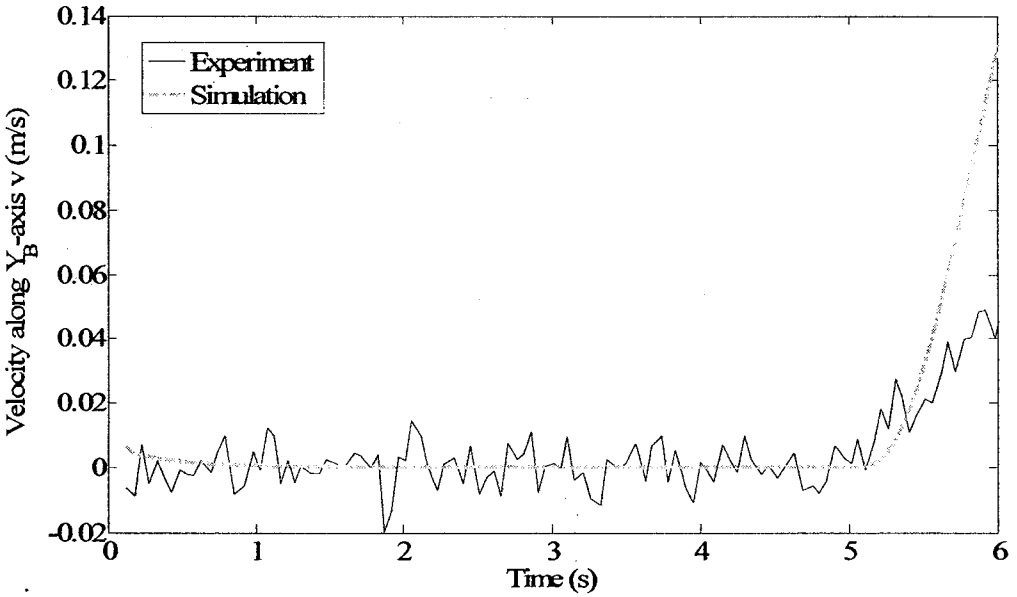

Fig.79. Velocity along Y_B-axis (IC#14)

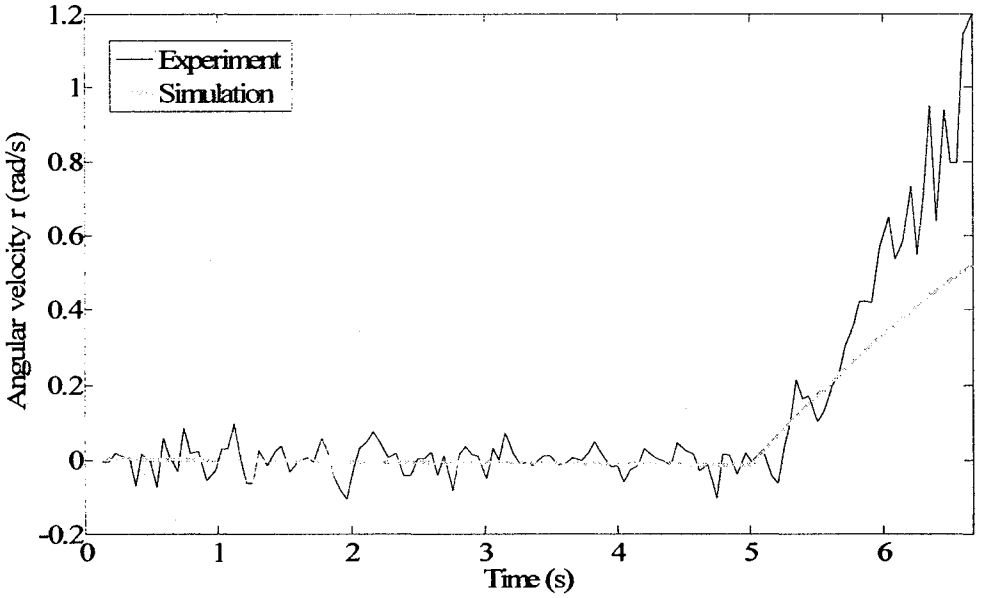

Fig.80. Angular velocity (IC#14)

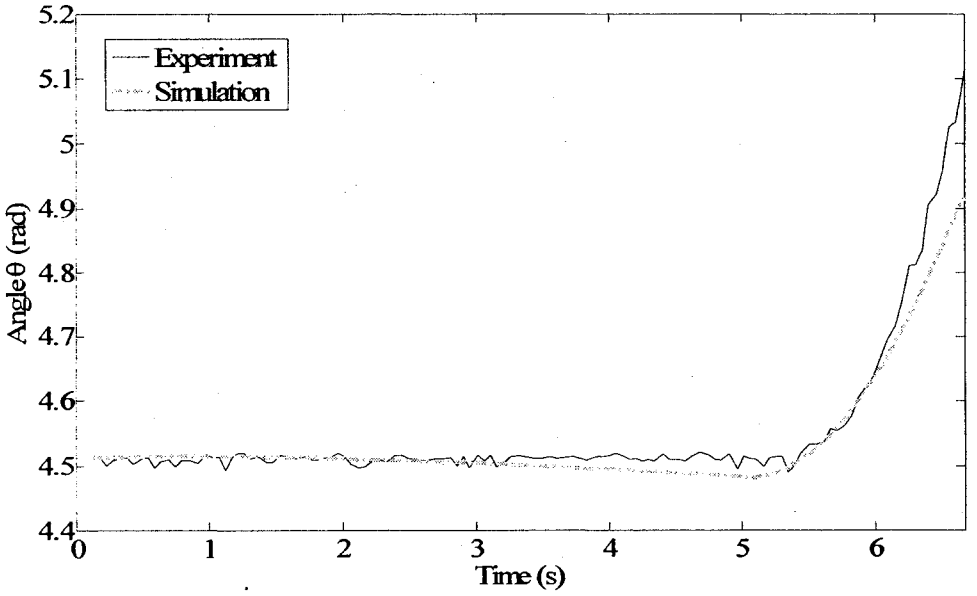

Fig.81. Angle vs. time (IC#14)

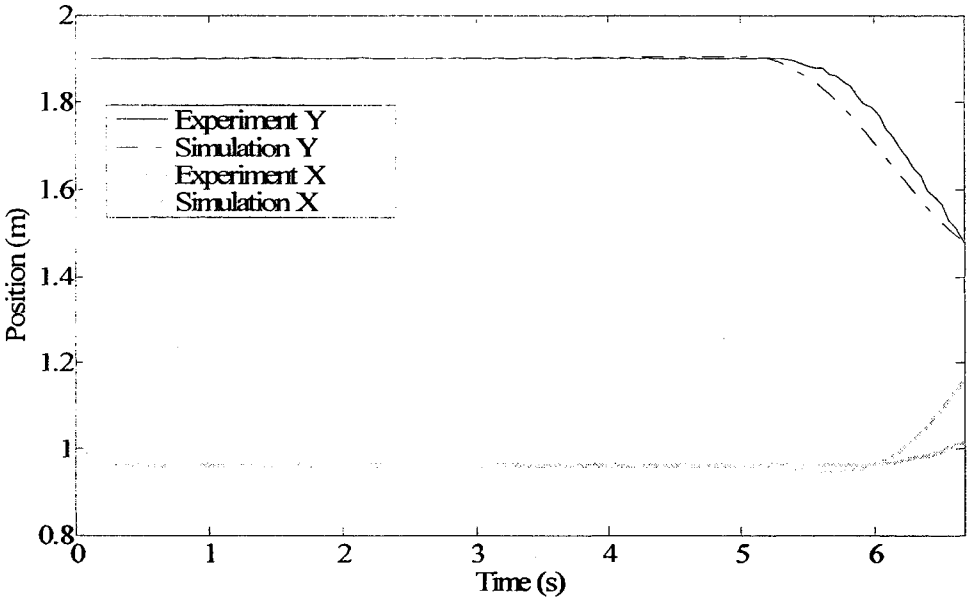

Fig.82. Position vs. time (IC#14)

The open loop control in the previous identification and validation processes are not asymptotically, but marginally stable, therefore, some deviations can be observed in the previous diagrams, such as Fig.76.

5. Application of Decentralized Receding Horizon Control to Wheeled Vehicles

Applying RHC and DRHC on wheeled vehicles is a fairly new concept, since there has been other control methods available for these type of nonholonomic systems, such as dynamic feedback linearization [88]. These methods have potentially faster sampling rates and guarantee stability [89]. However, none of them is able to easily handle input saturation and provide optimal performance. The RHC and DRHC methods can systematically address those critical issues.

In this chapter, DRHC will be applied to several simulations and experiments of multiple wheeled vehicle systems. The wheeled vehicle model used in this chapter has been obtained in (42). The procedure of designing a controller for the formation will follow the second method stated in Chapter 2.

5.1. Controller Design

In the case of single vehicle trajectory following, the cost function can be formulated as:

$$J^{fl}\left(\mathbf{x}(t),\mathbf{x}_D(t),t\right)=\int_t^{t+T}\left\|\mathbf{x}(\tau)-\mathbf{x}_D(\tau)\right\|_{\mathbf{Q}}^2 d\tau+\left\|\mathbf{x}(t+T)-\mathbf{x}_D(t+T)\right\|_{\mathbf{R}}^2, \qquad (56)$$

where according to (42), $\mathbf{x}(t)=\left[x_c(t)\quad y_c(t)\quad \theta_c(t)\quad \omega_{wm.L}\quad \omega_{wm.R}\right]^T$ denotes the state vector of the wheeled vehicle at time t, $\mathbf{x}_D(t)=\left[x_D\quad y_D\quad \theta_D\quad \omega_{D.L}\quad \omega_{D.R}\right]^T$ is the vector containing desired states, and $\mathbf{Q}$ and $\mathbf{R}$ are weighting matrices. Moreover,

(x_D, y_D) denote the desire position, θ_D is the desired angle, $\omega_{D,L}$ and $\omega_{D,R}$ denote the desired angular velocity of the left and right motor respectively.

When N_v ($N_v > 1$) vehicles added in the system, the formation of these $N_v + 1$ vehicles can be kept by the following approach. One of the vehicles is selected as the leader of the fleet that only follows the trajectory by using the cost function in (56); the rests are selected as followers, which keep certain distance from each other by using the cost function discussed below.

Let $\mathbf{x}_i(t)$ be the state vector of the i^{th} vehicle at time t. For the i^{th} vehicle there exists at least one j^{th} vehicle where $j \in A_i$, the set of i^{th} vehicle's neighbours. So for the i^{th} vehicle, there is a cost function (57):

$$J_i^n(\mathbf{x}_i(t), \underset{j\in A_i}{\mathbf{x}_j(t)}, t) = \int_t^{t+T} P\left(\sqrt{\|\mathbf{x}_i(\tau) - \mathbf{x}_j(t)\|_P^2} - r_{ij}\right)^2 d\tau + K\left(\sqrt{\|\mathbf{x}_i(t+T) - \mathbf{x}_j(t)\|_K^2} - r_{ij}\right)^2 \quad (57)$$

where r_{ij} is a scalar variable which denotes the nominal distance between the i^{th} and the j^{th} vehicle, **P** and **K** are weighting matrices and P and K are weighting scalars. How the weighting matrices and scalars are defined can be found in the next section.

The following parameters are selected for the RHC controller:

$$\begin{aligned} N_c &= 4 \\ N_i &= 50 \\ \delta &= 0.1\text{s} \\ T &= 1.0\text{s} \end{aligned} \quad (58)$$

5.2. Simulations

This section contains several simulations of trajectory following and formation control of wheeled vehicles for different cases.

Fig.83 shows a simulation of the tracking control of a wheeled vehicle. In this case, $\mathbf{x}(t)=\begin{bmatrix} x_c(t) & y_c(t) & \theta_c(t) & \omega_{wm.L} & \omega_{wm.R} \end{bmatrix}^T$, while $\mathbf{x}_D$ is chosen as:

$$\mathbf{x}_D(t)=\begin{bmatrix} 1.6+0.75\cos(t) \\ 1.2+0.75\sin(t) \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{59}$$

and **Q** and **R** are selected as:

$$\mathbf{Q}=\mathbf{R}=\begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{60}$$

In Fig.84, the formation of two vehicles is presented. The first vehicle follows the trajectory as in the first case, while a follower moves behind it and keeps a fixed distance from it. In this case, for the leader, $\mathbf{x}_1(t)=\begin{bmatrix} x_{c,1}(t) & y_{c,1}(t) & \theta_{c,1}(t) & \omega_{wm.L,1} & \omega_{wm.R,1} \end{bmatrix}^T$ and (60) remains unchanged; for the follower, $r_{ij}=0.1\text{m}$, $\mathbf{x}_2(t)=\begin{bmatrix} x_{c,2}(t) & y_{c,2}(t) & \theta_{c,2}(t) & \omega_{wm.L,2} & \omega_{wm.R,2} \end{bmatrix}^T$, and

$$\mathbf{P}=\mathbf{K}=\begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 3} \\ \mathbf{0}_{3\times 2} & \mathbf{0}_{3\times 3} \end{bmatrix} \tag{61}$$

$$P=K=1$$

In addition, two simulations for three and six vehicles keeping a triangular formation, while tracking a trajectory are performed and the results are presented in

Fig.86 and Fig.88, respectively. In this two cases, the definitions of $\mathbf{x}_i(t)$ and $\mathbf{x}_D(t)$ remain unchanged, (60) and (61) stay the same as the last simulation. r_{ij} can be set to any value, but in the following simulations, it remains 0.1 m.

Although, it can be observed that in Fig.85, and especially in Fig.87, the overshoot formation error (defined in (62)) between two vehicles are high at the beginning, due to the choice of initial conditions, but they quickly converge to their required steady state values as well. Same result is shown in Fig.89, however, for simplicity the figure only illustrates the formation error of the system, which is obtained by summing up the formation error between each pair of vehicles.

$$e_{ij}(t) = \left| \sqrt{\left\| \mathbf{x}_i(\tau) - \mathbf{x}_j(t) \right\|_P^2} - r_{ij} \right| \tag{62}$$

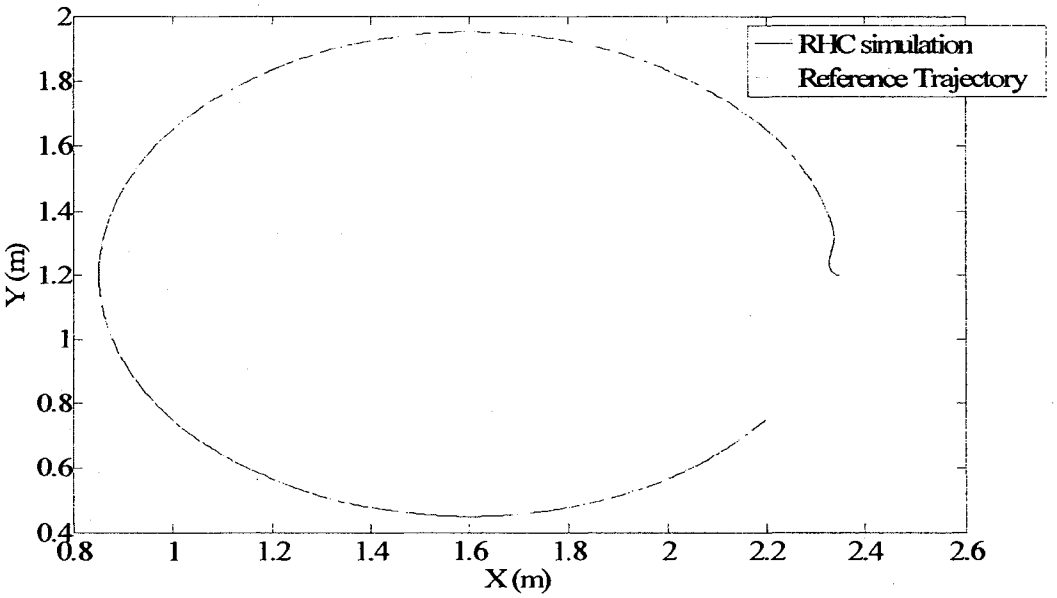

Fig.83. Simulation of trajectory following, for a single wheeled vehicle–

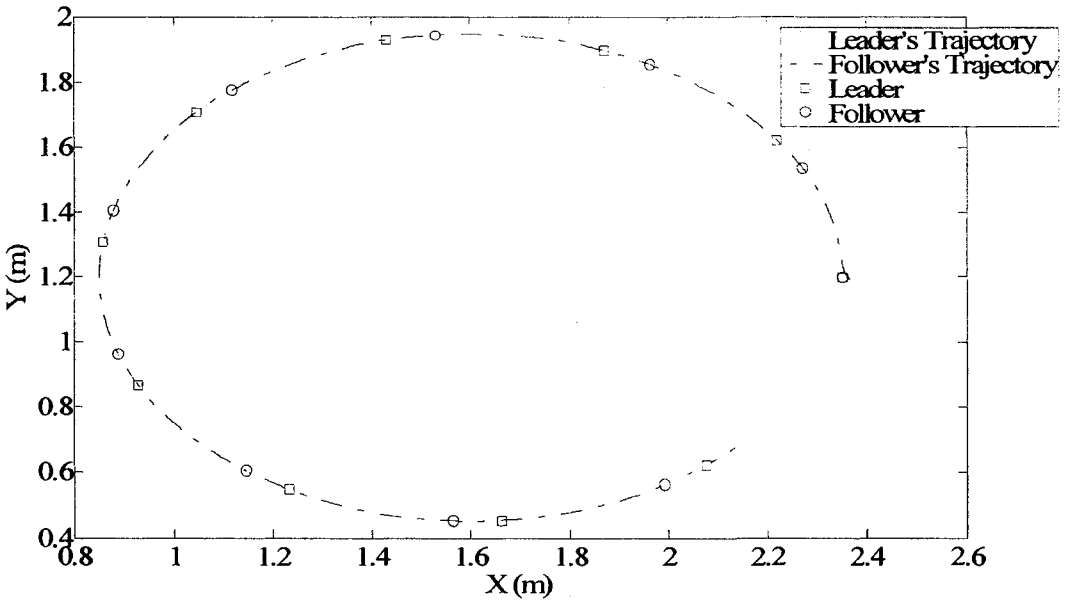

Fig.84. Simulation of trajectory following and formation control, for 2 wheeled vehicles

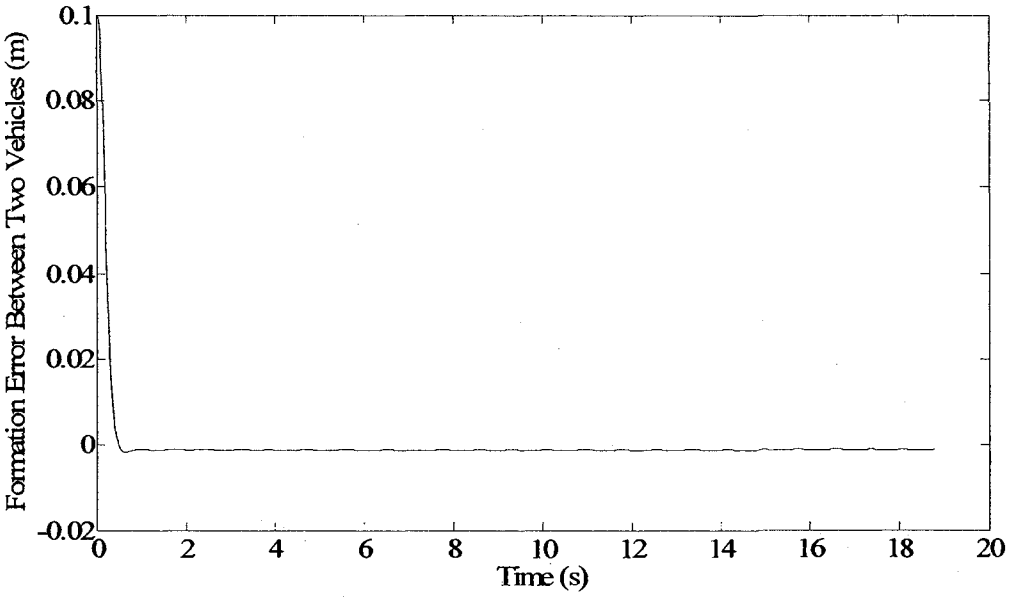

Fig.85. Distance between the two vehicles, for the case presented in Fig.84

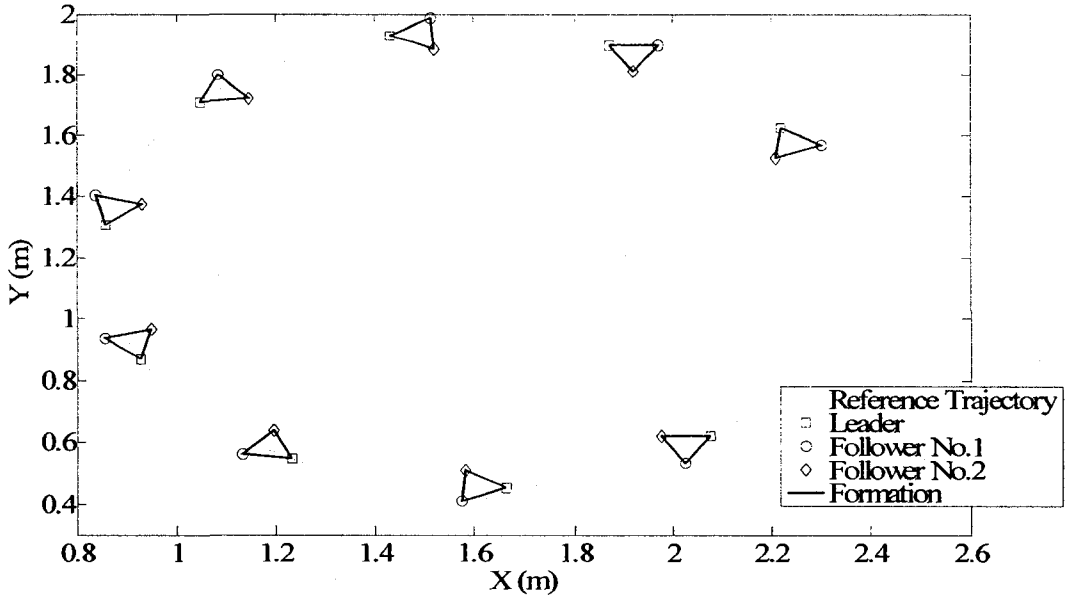

Fig.86. Simulation of trajectory following and formation control, for 3 wheeled vehicles

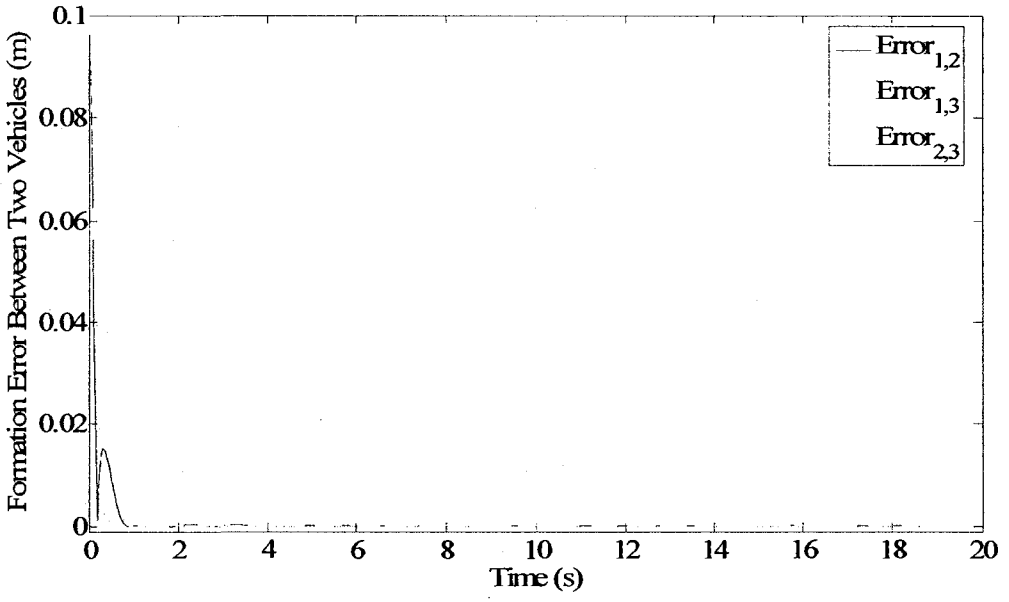

Fig.87. Distance between two of the three vehicle for the case presented in Fig.86

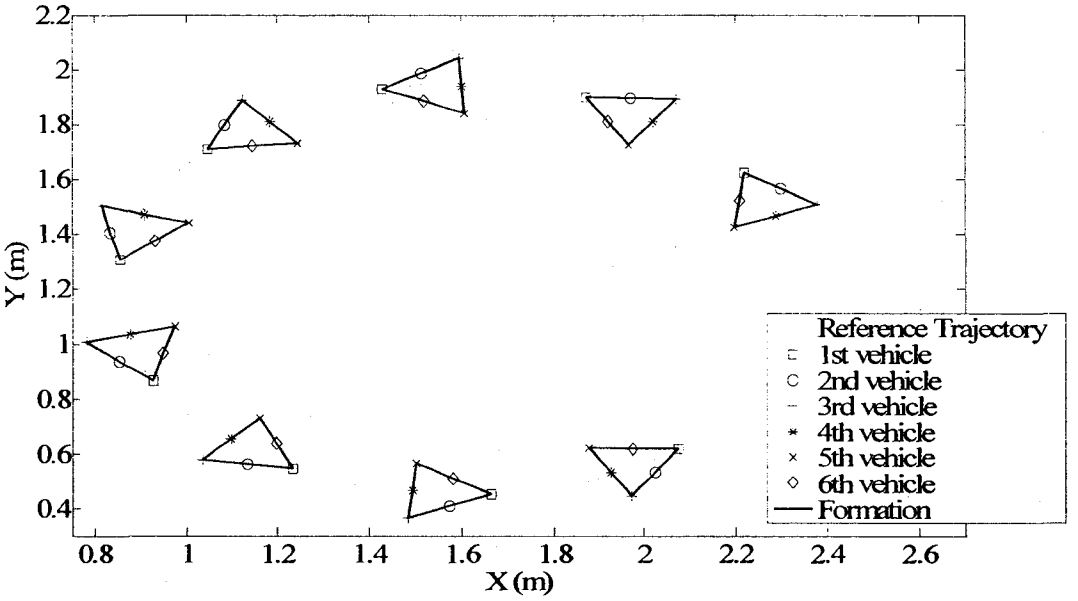

Fig.88. Simulation of trajectory following and formation control, for 6 wheeled vehicles

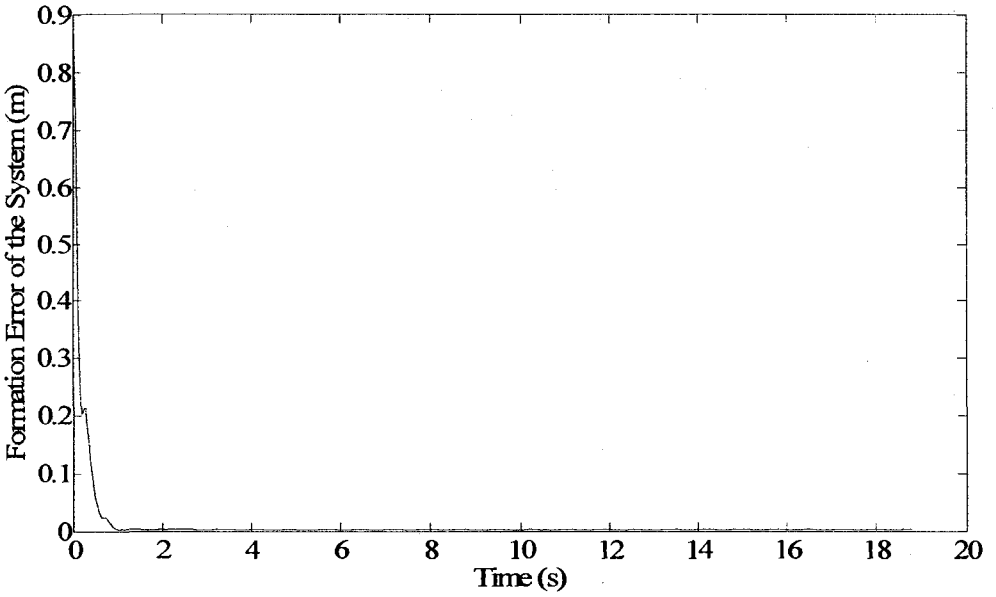

Fig.89. Formation error of the system for the case presented in Fig.88

5.3. Apparatus

In this section, the experimental apparatus will be briefly introduced. The apparatus consists of a vision feedback system and a controller computer. The vision

feedback system was introduced in chapter 3. It has nine web cameras pointing down to the testbed covering an area of approximately 5m by 5m. Each web camera is connected with a computer that processes the images acquired from the camera and sends the position of the targets to the controller computer at a frequency of 25Hz. This frequency is the maximum frequency the vision system can reach, thus causes the delay discussed in Chapter 3. Upon the reception of data from the vision system, the controller system calculates admissible inputs for the vehicle via a FM transmitter, which is connected to a D/A board. The D/A board is used to convert the digital control signals to analog signals need to be sent to the FM transmitter. A structural scheme of the apparatus is shown in Fig.90.

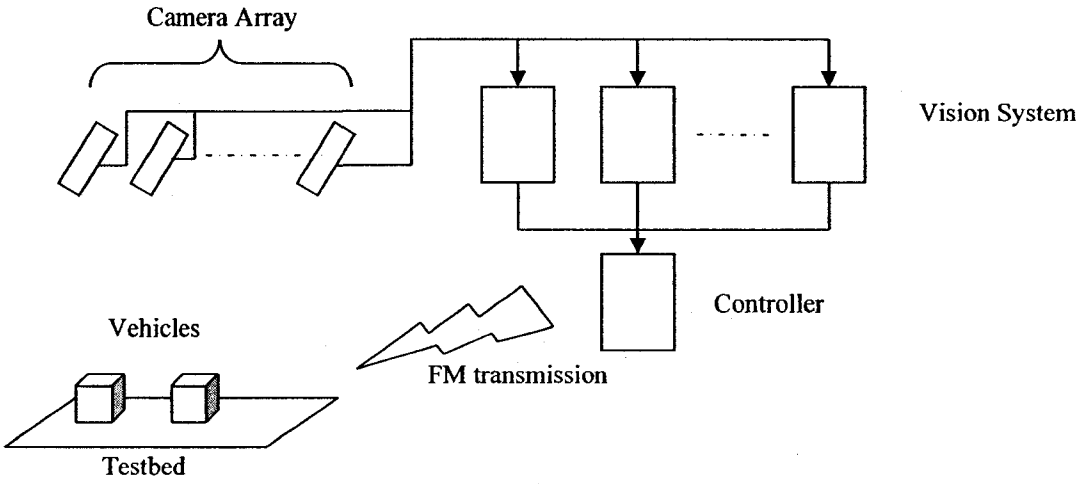

Fig.90. Structure of the apparatus

5.4. Checking the Constraint and Tuning the Parameters

A problem that researchers are usually faced is that no matter how perfect the output of the system would be in simulation, there would be some problem if the system was brought into reality. This problem persists in our RHC experiments as well.

Therefore, before moving into validating the previous simulations, we will discuss how to tune the parameters and check the constraints in order to obtain superior performance from the controller. A simple trajectory of line segments is introduced as an example showing how the procedure is undergone.

The trajectory is defined as:

$$\begin{aligned} x_D(t) &= 0.25 + 0.05t \\ y_D(t) &= 0.6 + 0.05t \end{aligned}, \text{ for all } 0 \le t \le 25s$$

$$\begin{aligned} x_D(t) &= 0.25 + 0.05t \\ y_D(t) &= y_D(25) \end{aligned} \text{ for all } t > 25s \tag{63}$$

where as before, $(x_D(t), y_D(t))$ denotes the desired position at time t. So if we follow the process explained in section 5.1, we can have the desired states for the controller as:

$$\mathbf{x}_D(t) = \begin{bmatrix} 0.25 + 0.05t \\ 0.6 + 0.05t \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{ for all } 0 \le t \le 25s$$

$$\mathbf{x}_D(t) = \begin{bmatrix} 0.25 + 0.05t \\ x_{D2,1}(25) \\ 0 \\ 0 \\ 0 \end{bmatrix}, \text{ for all } t > 25s \tag{64}$$

and by using the selection of RHC parameters in (60), a RHC controller is successfully constructed for this trajectory following problem, and its result is shown in Fig.91.

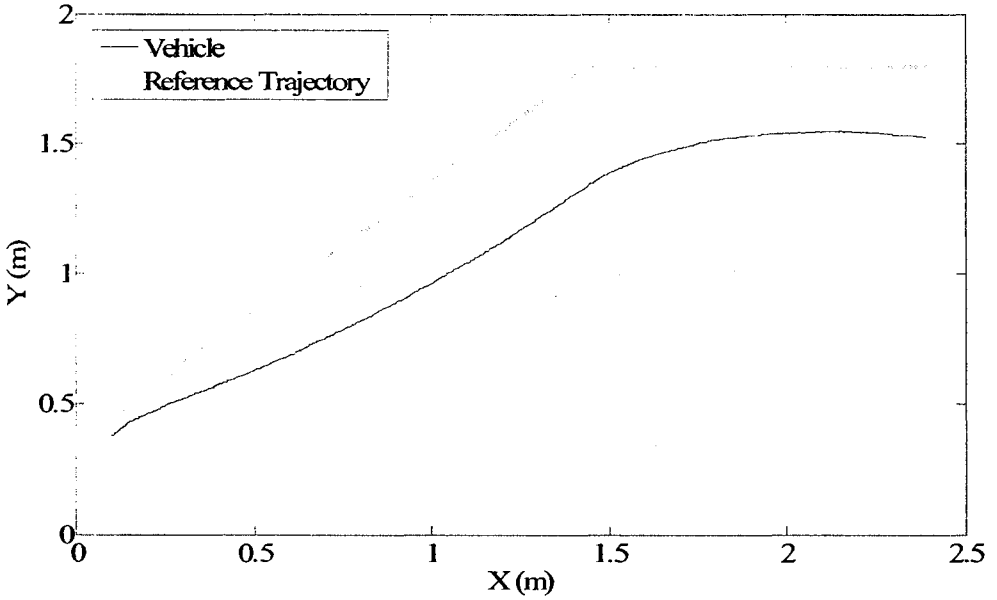

Fig.91. Trajectory following using wheeled vehicle, before tuning RHC

It is apparent that the performance of the controller is not satisfying, and the constraints and/or the parameters needs to be updated. The first step to determine whether the constraints or the parameters should be adjusted first is to carefully observe the output diagram. In Fig.91, it is obvious that the output trajectory has a trend towards the reference trajectory, but the offset is huge. In this case, it is recommended to check if there are other constraints that can be added in the controller to make the overall constraint strong enough to drive the system to the desired states.

In this example, it should not be difficult to see that a desired angle will help the system point to the desired position $(x_D(t), y_D(t))$ at time t, and if the system is able to do that, moving to that position can be easy for the vehicle. Thus, (64) is updated as:

$$\mathbf{x}_D(t)=\begin{bmatrix}0.25+0.05t\\0.6+0.05t\\\theta_D(t)\\0\\0\end{bmatrix},\ \text{for all } 0\le t\le 25s$$

$$\mathbf{x}_D(t)=\begin{bmatrix}0.25+0.05t\\x_{D2,J}(25)\\\theta_D(t)\\0\\0\end{bmatrix},\ \text{for all } t>25s \tag{65}$$

where

$$\theta_D(t)=\arctan 2\left(\left(y_D(t)-y_c(t)\right),\left(x_D(t)-x_c(t)\right)\right) \tag{66}$$

is the desired angle for the vehicle. Moreover the parameters associated with the RHC controller is updated as:

$$\mathbf{Q}=\mathbf{R}=\begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}_{3\times2}\\\mathbf{0}_{2\times3} & \mathbf{0}_{2\times2}\end{bmatrix} \tag{67}$$

and the result is shown in Fig.92. Please note that the parameters have not been tuned yet, and the change of parameters in (67) is just because of the need to bring $\theta_D(t)$ in the cost function.

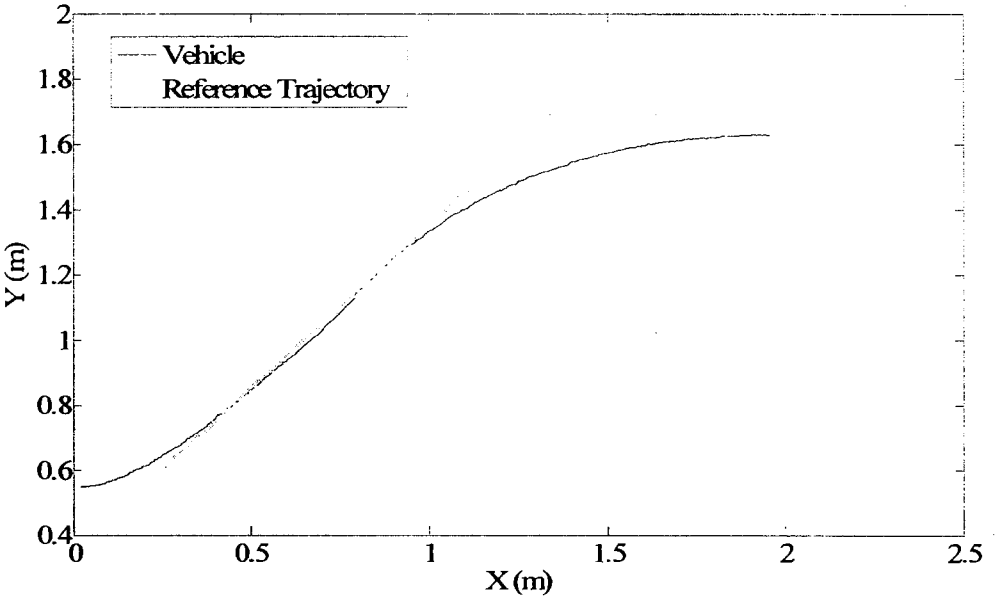

Fig.92. Trajectory following using wheeled vehicle, after adding constraint

Tuning the parameters can be summarised as increasing the value of the parameter corresponding to the most disagreement. For example, the cost function for Fig.92 can be derived from (56) as:

$$\begin{aligned} J(\mathbf{x}_c(t),\mathbf{x}_D(t)) = a_1 \int_t^{t+T} (x_c(\tau)-x_D(\tau))^2 d\tau + b_1 (x_c(t+T)-x_D(t+T))^2 \\ + a_2 \int_t^{t+T} (y_c(\tau)-y_D(\tau))^2 d\tau + b_2 (y_c(t+T)-y_D(t+T))^2 \\ + a_3 \int_t^{t+T} (\theta_c(\tau)-\theta_D(\tau))^2 d\tau + b_3 (\theta_c(t+T)-\theta_D(t+T))^2 \end{aligned} \quad (68)$$

where $a_1, a_2, a_3, b_1, b_2, b_3$ denote the parameters associated with the RHC controller for this problem. Since the disagreement between the actual and reference trajectory is primarily caused by the offset in Y direction, the first step is to increase the values of a_2 and b_2. After that, the new output should be checked to see if other parameters also need to be changed. The final result of this parameter tuning is

$$\mathbf{Q} = \mathbf{R} = \text{diag}([1.5 \quad 2.5 \quad 1.0 \quad 0.0 \quad 0.0]) \tag{69}$$

and the final output is shown in Fig.93. Please note that although it is possible to reduce huge offset simply by tuning the parameters, it is still recommended to check the missing constraints first, since the process of tuning is more complicated when compared with adding an important constraint.

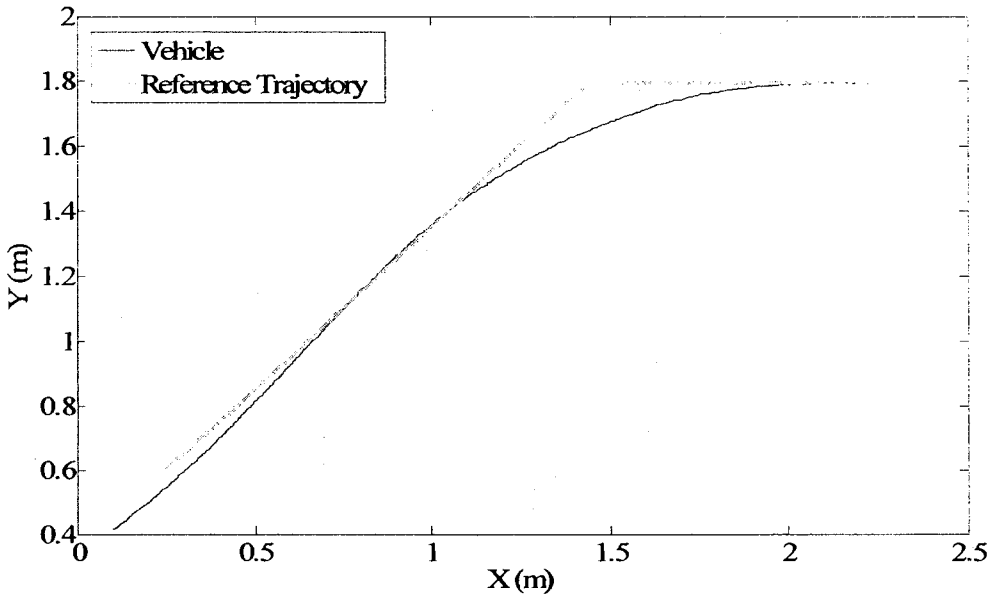

Fig.93. Trajectory following using wheeled vehicle, after adding constraint and tuning

And the initial condition for the case shown in Fig.91 is:

$$\mathbf{x}_1(0) = [0.1 \quad 0.37 \quad 0 \quad 0 \quad 0]^T, \tag{70}$$

for the case shown in Fig.92 is:

$$\mathbf{x}_1(0) = [0.021 \quad 0.55 \quad 0 \quad 0 \quad 0]^T, \tag{71}$$

for the case shown in Fig.93 is:

$$\mathbf{x}_1(0) = [0.012 \quad 0.41 \quad 0 \quad 0 \quad 0]^T. \tag{72}$$

5.5. Experimental Verification

In this section, two single vehicle trajectory following examples are presented to validate the algorithm in (56), followed by the triangular formation control of three vehicle. The later validation will be performed by using the decentralized RHC controller discussed in section 2.4 and a combination of actual experimental vehicle and simulation is employed. The experiments were run on the apparatus discussed in section 5.3.

Two experimental results are shown in Fig.94 and Fig.95, and their corresponding simulation result is shown in Fig.83. The initial condition for the first case (Fig.94) is as follows:

$$\mathbf{x}_1(0) = [2.051 \quad 0.815 \quad 0 \quad 0 \quad 0]^T. \tag{73}$$

and, the initial condition for second case (Fig.95) is:

$$\mathbf{x}_1(0) = [2.1021 \quad 1.0364 \quad 0 \quad 0 \quad 0]^T. \tag{74}$$

Please note that the initial positions of the vehicle in this experiment are different from the simulation presented in Fig.83. Besides, the selection of $\mathbf{Q}$ and $\mathbf{R}$ is updated to the following, as explained earlier:

$$\mathbf{Q} = \mathbf{R} = \text{diag}([1.0 \quad 1.2 \quad 0.5 \quad 0 \quad 0]) \tag{75}$$

while $\mathbf{x}_D(t)$ is modified as:

$$\mathbf{x}_D(t) = \begin{bmatrix} 1.6 + 0.75\cos(t) \\ 1.2 + 0.75\sin(t) \\ \theta_D(t) \\ 0 \\ 0 \end{bmatrix} \tag{76}$$

where the definition of $\theta_D(t)$ can be found in (66).

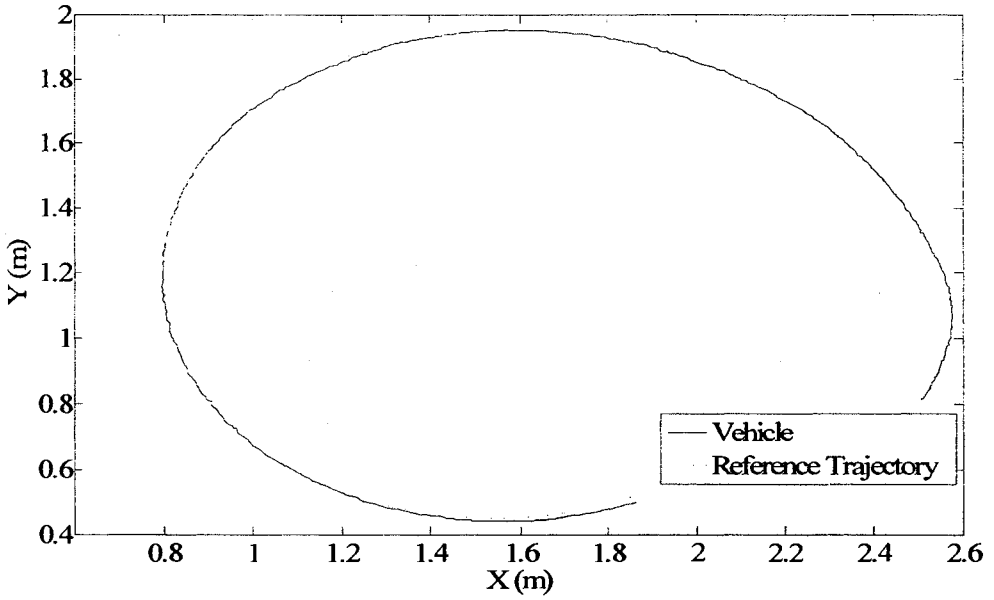

Fig.94. Experimental result of the first trajectory following example with a single wheeled vehicle

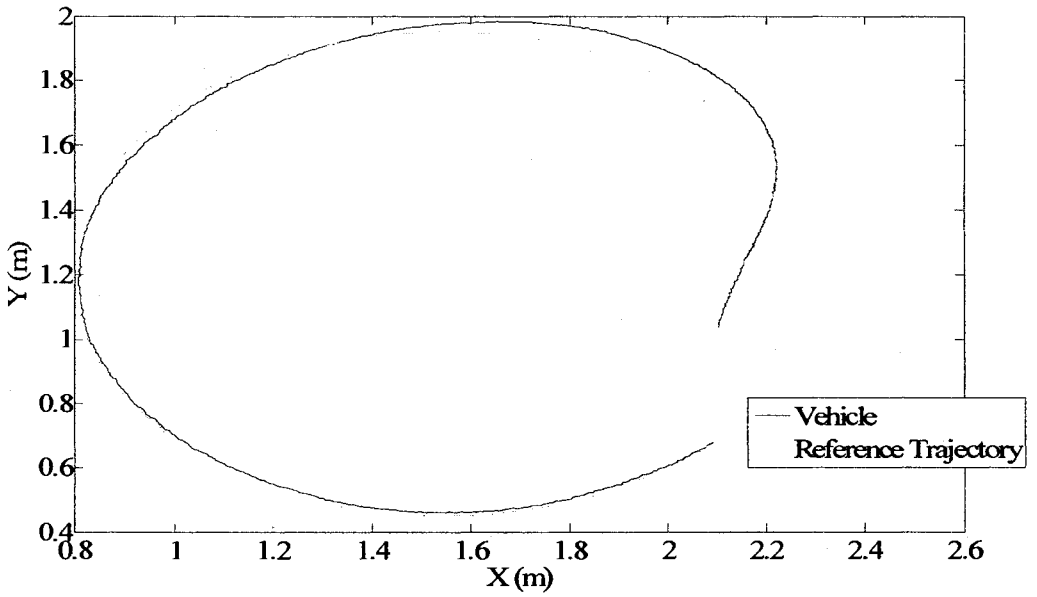

Fig.95. Experimental result of the second trajectory following example with a single wheeled vehicle

Then the case of three vehicle triangular formation control is validated. It should be indicated that the experimental results shown in Fig.96 to Fig.99 were run in a mixed reality fashion, in which only the leader is running in real world and the followers are

being simulated. This is an interesting case that real-time simulations are combined with the experimental results. However, this arrangement still closely follows the requirement of the DRHC environment, the members are under constraints of DRHC, and they need to use the second method in 2.4 as DRHC strategy, and On-the-Fly Computation method as actuation method. The initial conditions for the experiment shown in Fig.98 and Fig.99 are:

$$\mathbf{x}_1(0) = \mathbf{x}_2(0) = \mathbf{x}_3(0) = [2.37 \quad 0.88 \quad 0 \quad 0 \quad 0]^T \tag{77}$$

while the initial conditions for the experiment shown in Fig.100 and Fig.101 are:

$$\mathbf{x}_1(0) = \mathbf{x}_2(0) = \mathbf{x}_3(0) = [2.45 \quad 1.22 \quad 0 \quad 0 \quad 0]^T \tag{78}$$

Please note that the selection of $\mathbf{Q}$ and $\mathbf{R}$ remains as in (75), and $\mathbf{x}_D(t)$ stays the same as in (76). However, the selection of $\mathbf{P}$, $\mathbf{K}$, P, and K are updated as:

$$\begin{aligned} \mathbf{P} = \mathbf{K} &= \text{diag}([1.5 \quad 1.5 \quad 0.0 \quad 0.0 \quad 0.0]) \\ P = K &= 1.5 \end{aligned} \tag{79}$$

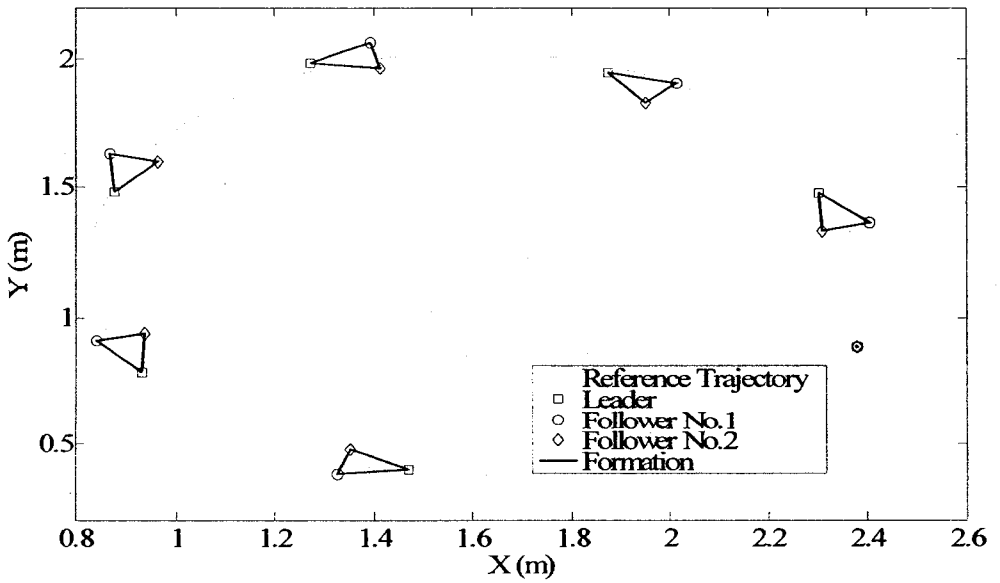

Fig.96. First formation control experiment with three wheeled vehicles

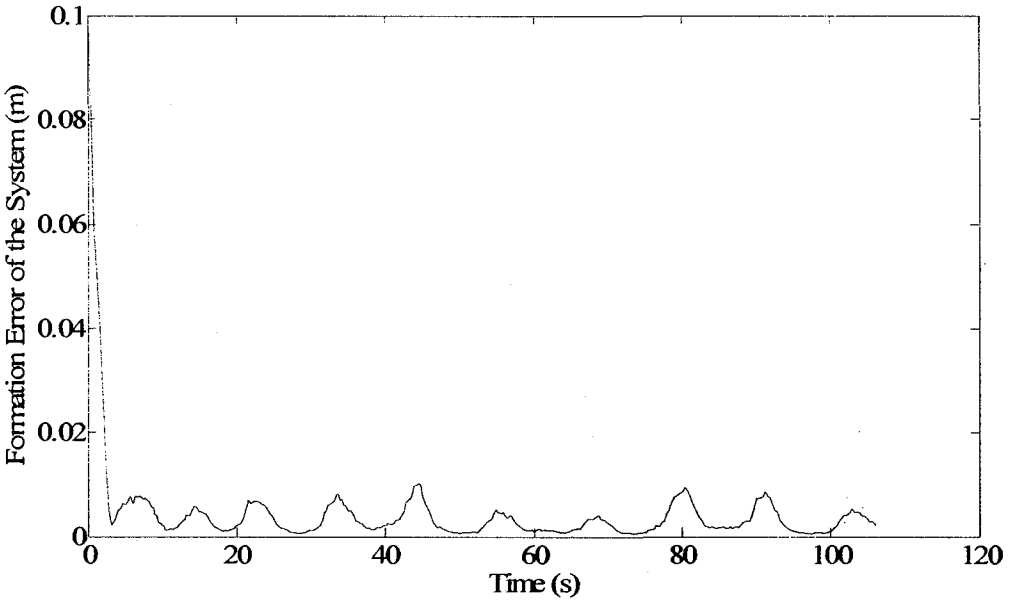

Fig.97. Formation error of the first formation control experiment

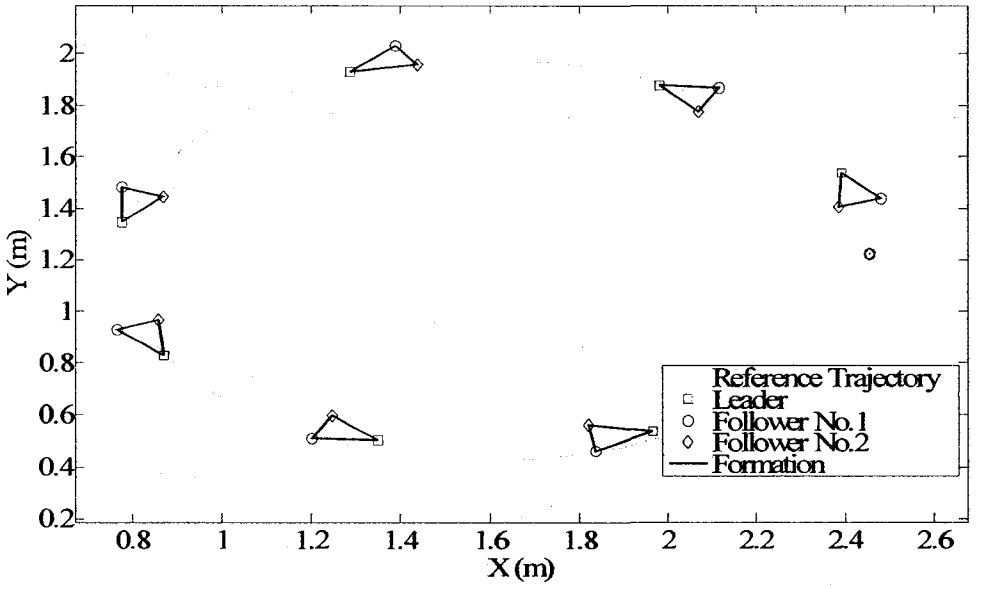

Fig.98. Second formation control experiment with three wheeled vehicles

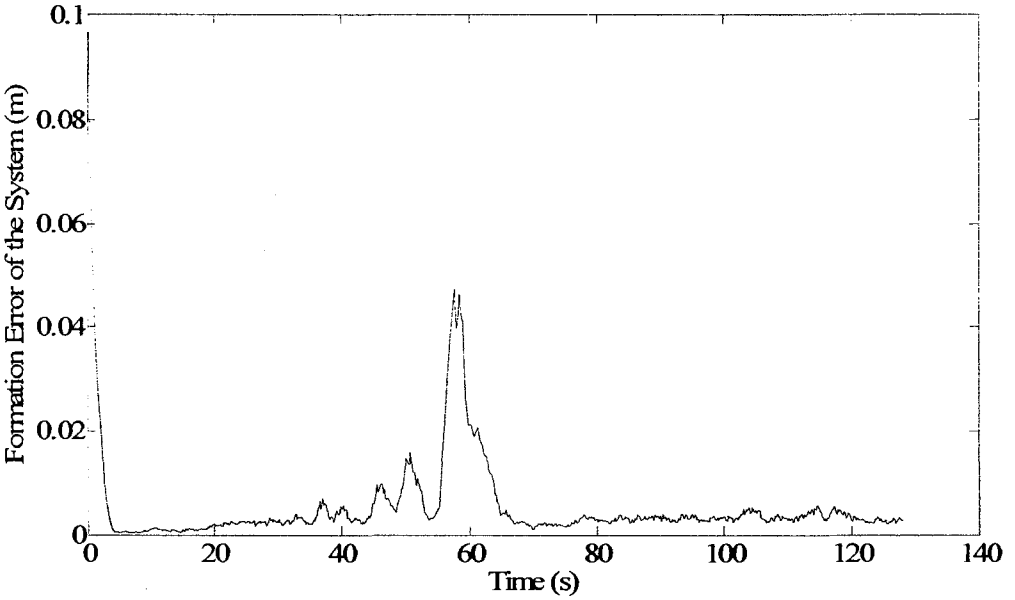

Fig.99. Formation error of the second formation control experiment

6. Application of Distributed DRHC to Hovercrafts

In this section, the hovercraft model, identified in the previous chapter, is used to implement multiple vehicle simulations and experiments. Similar to the last chapter, the goal in this section is also to focus on the performance of a fleet of multiple hovercrafts' trajectory following and formation behaviour by using decentralized RHC. By making the experiments a higher level, they will also be run in a distributed fashion, which means, instead of one computer calculating all the input for every vehicle, several computers connected via a high speed LAN share their data and work simultaneously to solve the control problem.

The number of vehicles for the simplest formation control could be as few as three. In order to discuss the implementation in this chapter, let us consider the simplest case. In this example, one of the hovercrafts is the leader of formation, which follows a trajectory resulted from a predefined path, and avoids an obstacle on its way, while the other two are followers. Their tasks are following the leader and keeping a specific distance from each other. Thus, a triangle formation is achieved as in Fig.100. Each vehicle is controlled by a single computer, and shares its motion data with the other two via a high speed LAN network using UDP/IP protocol. Since the model used in this implementation is the hovercraft model identified in the previous chapter, the i^{th} vehicle notion in chapter 2 will be also changed to i^{th} hovercraft, unless otherwise specified.

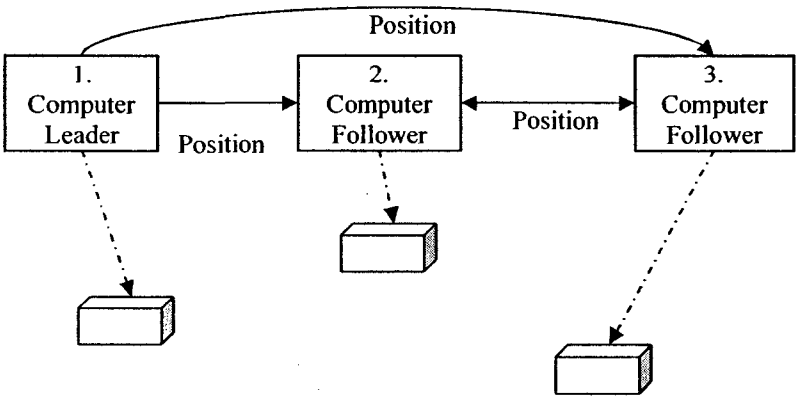

Fig.100. Layout of the three agents' formation

6.1. Distributed RHC System

The distributed RHC system consists of three computers and a high speed switch. As shown in Fig.100, each computer is responsible for solving the optimal problem of a hovercraft. In the following, different parts of this study are briefly explained.

6.1.1. User Datagram Protocol

The User Datagram Protocol (UDP) is selected as the data transmission protocol in this experiment. It is different from what is commonly used in the Internet today, Transmission Control Protocol (TCP). Because in UDP, sockets do not have to be connected before being used [59], datagram might arrive out of order, have duplicates, or even become missing. It is not a reliable protocol for some specific data transmission applications, such as web browsers and email clients. However, despite those properties, UDP is fast and ideal for the light communication, especially for the time sensitive systems like ours, as well as Voice Over Internet Protocol (VoIP) and online games.

6.1.2. Data Loss, Data Transmission Delay, Computation Time, and Time Synchronization

A. Data Loss

Because of the nature of UDP, data loss is inevitable in this process, for example, data has been sent from the sender computer, but the receiver computer has not prepared to obtain the data yet.

B. Data Transmission Delay and Computation Time

Data transmission delay is usually caused by the nature of hardware, such as resistance of network cables and the design of the switch circuit. Fig.101 shows the result of an experiment for calculating delays between two computers. In that experiment, computer A sent a set of data to computer B; immediately when B received the data it sent back a set of data to A. The delay is obtained by dividing the time used in this process by two. It seems to be fine, since the average delay is approximately 0.5×10^{-4} second. But if the number of computer rises, the delay will become relatively large for the whole system too.

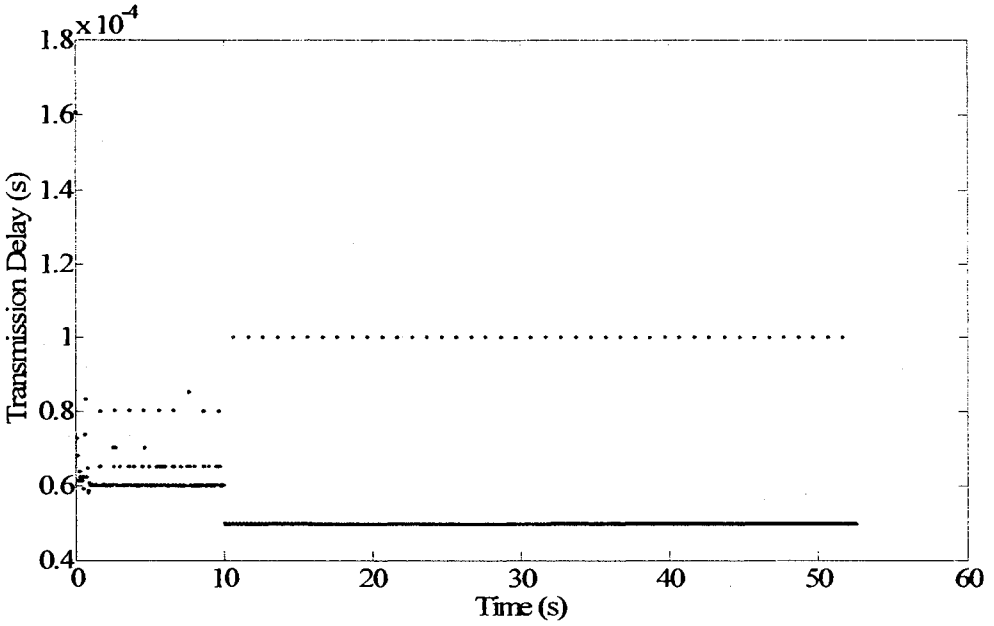

Fig.101. Data transmission delay between two computers.

Another main factor that affects the performance of the distributed system is the computation time on each individual computer. RHC is relatively time consuming when compared with other control methods, as we have discussed before. In the previous studies, both zero [82] and non-zero [81] computation time were assumed and studied. In this section, a novel method of dealing with computation time is introduced. That is, during the implementations, the computation time will be treated as a delay, similar to the data transmission delays. Fig.102 shows the computation time of the case where a trajectory following problem was solved on a single computer. Please note that there is no formation problem in this computation, and the following parameters are chosen:

$$\begin{aligned} N_c &= 4 \\ N_i &= 50 \\ \delta &= 0.1\text{s} \\ T &= 1.0\text{s} \end{aligned} \tag{80}$$

However, it should be noted that the computation time will vary if the above parameters are changed and/or different optimization methods are employed.

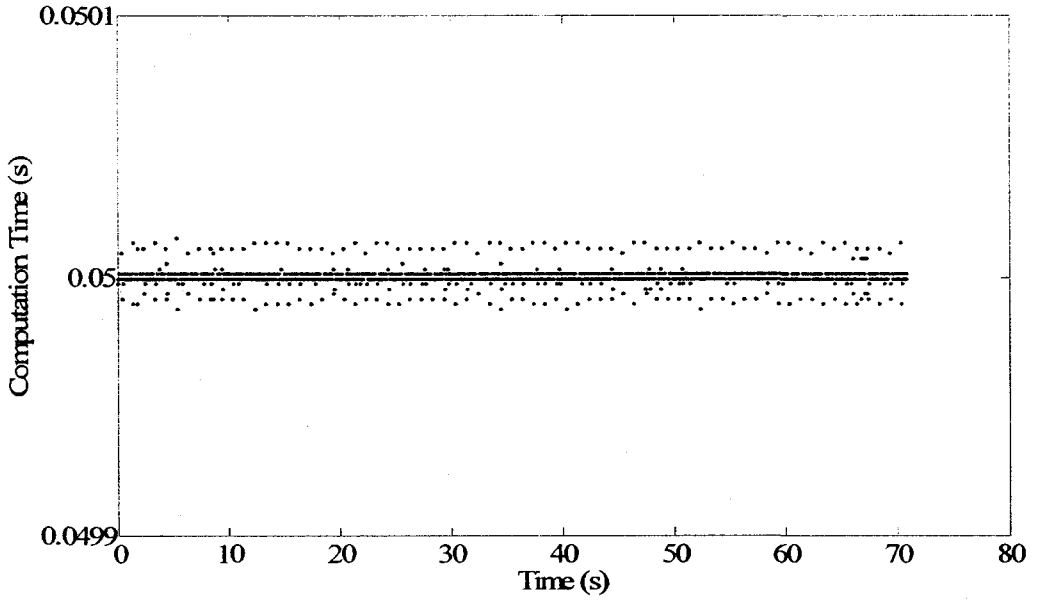

Fig.102. Computation time on a single computer.

C. Time Synchronization

Instead of simply exchanging updated position data of the vehicles among the computers in the system, local time of the leader computer will also be chosen as global time and sent to the follower computers at fixed periods. When they receive the global time, the followers will adjust their local time according to the difference between these two times. Fig.103 shows a flowchart of this procedure.

The synchronization offset among the subsystems can be obtained as follows. Let the data transmission time from one computer to another is ζ, and the calculation time for each simulation step is φ. By recalling the sensor delay t_{SD} and its upper bound T_{SD} obtained in (41), we can have the following:

$$\sigma = \zeta + \varphi + T_{SD} \tag{81}$$

where σ denotes the synchronization offset among the subsystems.

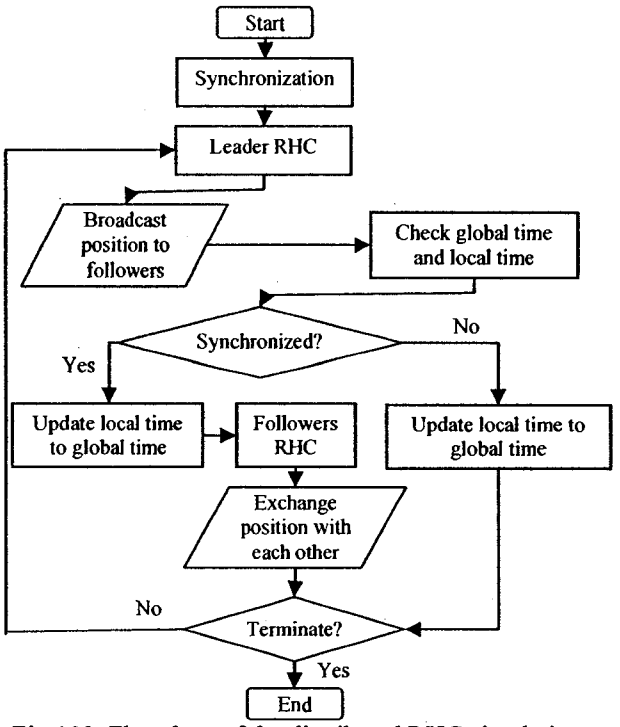

Fig.103. Flowchart of the distributed RHC simulation.

6.2. Controller Design

In this Chapter, the difficulty of controlling the fleet of vehicles is raised by adding an obstacle on the path. Combining the leader cost index presented in (56) with a penalty term for inputs, the following is assumed as the cost index for the leader without considering obstacle avoidance:

$$J_1^{fl}\left(\mathbf{x}_1(t), \mathbf{x}_D(t), t\right) = \int_t^{t+T} \left(\left\| \mathbf{x}_1(\tau) - \mathbf{x}_D(\tau) \right\|_{\mathbf{Q}}^2 + \left\| \mathbf{u}_1(\tau) \right\|_{\mathbf{C}}^2 \right) d\tau + \left\| \mathbf{x}_1(t+T) - \mathbf{x}_D(t+T) \right\|_{\mathbf{R}}^2 \tag{82}$$

where $\mathbf{C} = \mathbf{I}_{2\times 2}$ is a positive definite weighting matrix, and

$$\mathbf{x}_1(t) = \begin{bmatrix} u_{c,1} & v_{c,1} & r_{c,1} & x_{c,1} & y_{c,1} & \theta_{c,1} \end{bmatrix}^T \tag{83}$$

and

$$\mathbf{R} = \mathbf{Q} = \mathrm{diag}([0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]) \tag{84}$$

In addition, suppose that there is a stationary obstacle with a radius of R_O at (x_O, y_O). Let $\mathbf{z}_O = \begin{bmatrix} x_O & y_O \end{bmatrix}^T$, we could obtain the cost function for obstacle avoidance by adding a potential term as follows [80]:

$$J_1^{av}(\mathbf{x}_1(t), t) = \int_t^{t+T} P\left(\sqrt{\left\|\mathbf{x}_1(\tau) - \mathbf{C}_O \mathbf{z}_O\right\|_P^2} - R_O\right)^{-2} d\tau, \tag{85}$$

where $\mathbf{C}_O$, $\mathbf{P}$, and P are defined in the following:

$$\mathbf{C}_O = \begin{bmatrix} \mathbf{0}_{3\times 2} \\ \mathbf{I}_{2\times 2} \\ \mathbf{0}_{1\times 2} \end{bmatrix}, \quad \mathbf{P} = \mathrm{diag}([0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]), \text{ and } P = 1 \tag{86}$$

Therefore, by combining (82) and (85), the cost function for the leader is as follows:

$$\begin{aligned} J_1(\mathbf{x}_1(t), \mathbf{x}_D(t), t) &= \int_t^{t+T} P\left(\sqrt{\left\|\mathbf{x}_1(\tau) - \mathbf{C}_O \mathbf{z}_O\right\|_P^2} - R_O\right)^{-2} d\tau \\ &+ \int_t^{t+T} \left(\left\|\mathbf{x}_1(\tau) - \mathbf{x}_D(\tau)\right\|_Q^2 + \left\|\mathbf{u}_1(\tau)\right\|_C^2\right) d\tau + \left\|\mathbf{x}_1(t+T) - \mathbf{x}_D(t+T)\right\|_R^2 \end{aligned} \tag{87}$$

Therefore, by comparing (57) to (56) and (87), the following is assumed for the cost function of the followers:

$$\begin{aligned} J_{\substack{i \\ i=2,3;j=1,2,3;j\neq i}}^{fl}(\mathbf{x}_i(t), \mathbf{x}_j(t), t) &= \int_t^{t+T} K\left(\sqrt{\left\|\mathbf{x}_1(\tau + (i-1)\sigma) - \mathbf{x}_j(\tau + (i-1)\sigma)\right\|_K^2} - r_{ij}\right)^2 d\tau \\ &+ \int_t^{t+T} \left\|\mathbf{u}_i(\tau + (i-1)\sigma)\right\|_C^2 d\tau \end{aligned} \tag{88}$$

where σ is defined in (81) and $\mathbf{K}$, $\mathbf{C}$, and K are defined in the following:

$$\mathbf{K} = \mathrm{diag}([0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]),\ \mathbf{C} = \mathbf{I}_{2\times 2},\ \text{and}\ K = 1 \tag{89}$$

6.3. Distributed Simulation

The following terms are set and remain unchanged in the following simulations

$$r_{ij} = 0.1,\ \mathbf{z}_O = [1.5 \quad 1.25]^T,\ R_O = 0.4 \tag{90}$$

and all the weighting matrices are set to identity matrices and all the weighting scalars are set to 1.

The distributed simulation results are compared with the experimental results on a single computer in Fig.104 to Fig.107. There are two reference trajectories used in these examples. The first one (IC#11) is

$$\begin{aligned} \mathbf{x}_D(t) &= \begin{bmatrix} 2.5 + 0.1t \\ 2.75 + 0.1t \end{bmatrix}, \text{ for all } 0 \le t \le 20\,\mathrm{s} \\ \mathbf{x}_D(t) &= \begin{bmatrix} x_{D1,1}(20) \\ x_{D2,1}(20) + (t - 20) \end{bmatrix}, \text{ for all } t > 20\,\mathrm{s} \end{aligned} \tag{91}$$

and the other one (IC#12) is defined as:

$$\mathbf{x}_D(t) = \begin{bmatrix} 2.5 + 1.5\cos(0.01t) \\ 2.5 + 1.5\sin(0.01t) \end{bmatrix}, \text{ for all } t \ge 0\,\mathrm{s} \tag{92}$$

It is apparent that both single computer case and distributed case show promising results in trajectory following and obstacle avoidance.

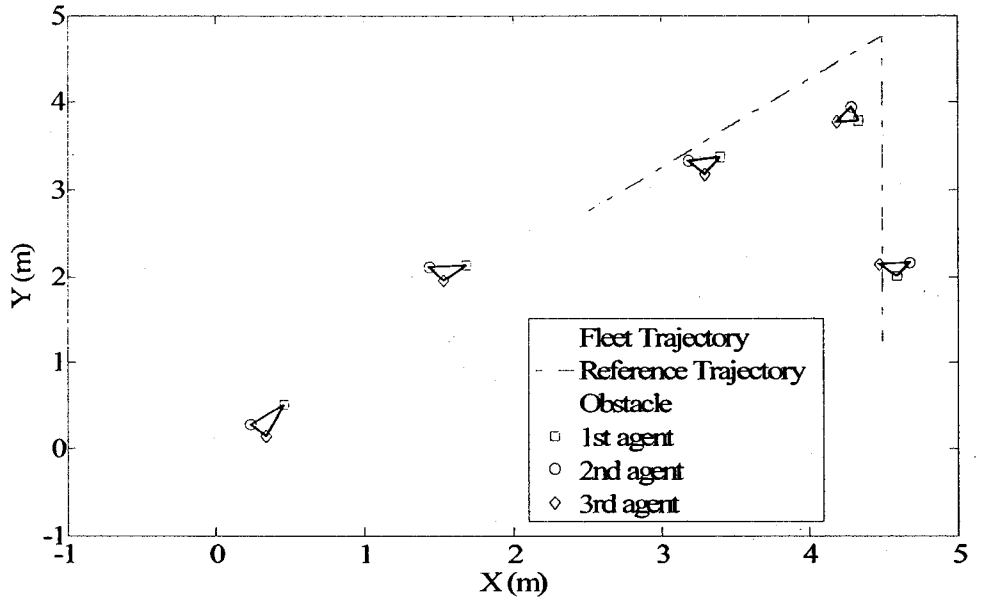

Fig.104. Trajectory following and obstacle avoidance for distributed simulation case (IC#11).

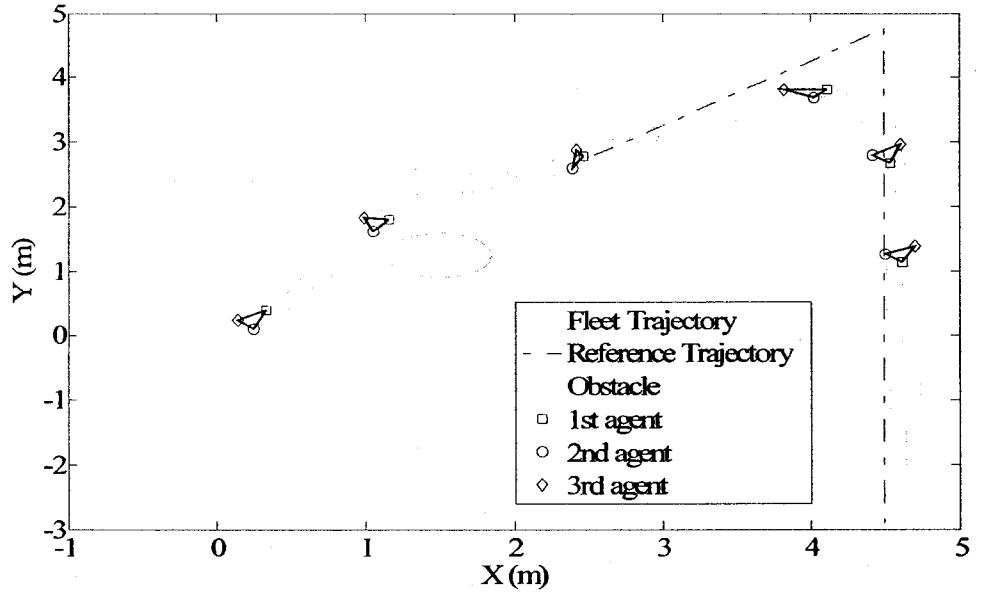

Fig.105. Trajectory following and obstacle avoidance experimental results for single computer case (IC#11).

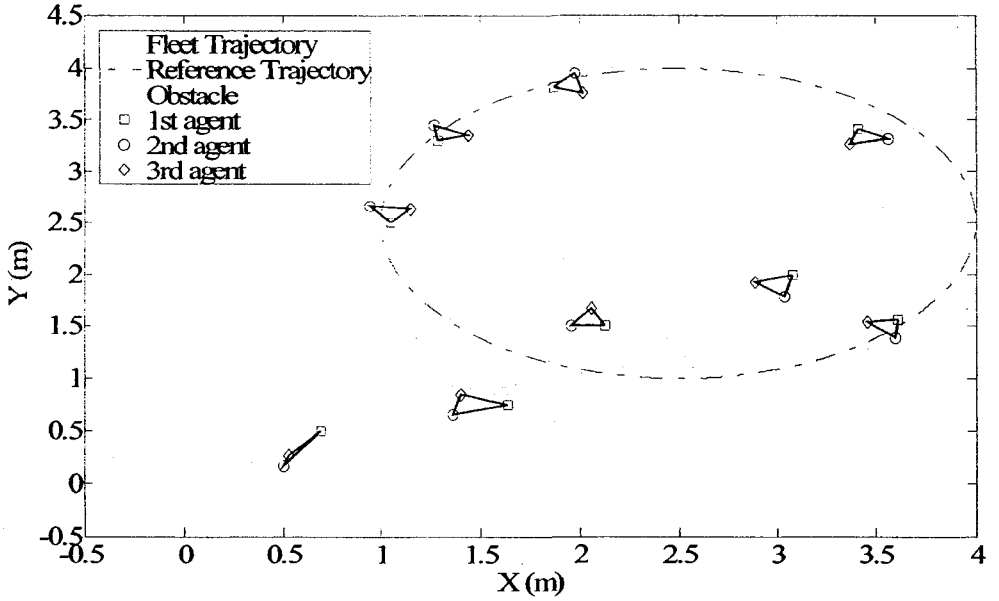

Fig.106. Trajectory following and obstacle avoidance for distributed simulation case (IC#12).

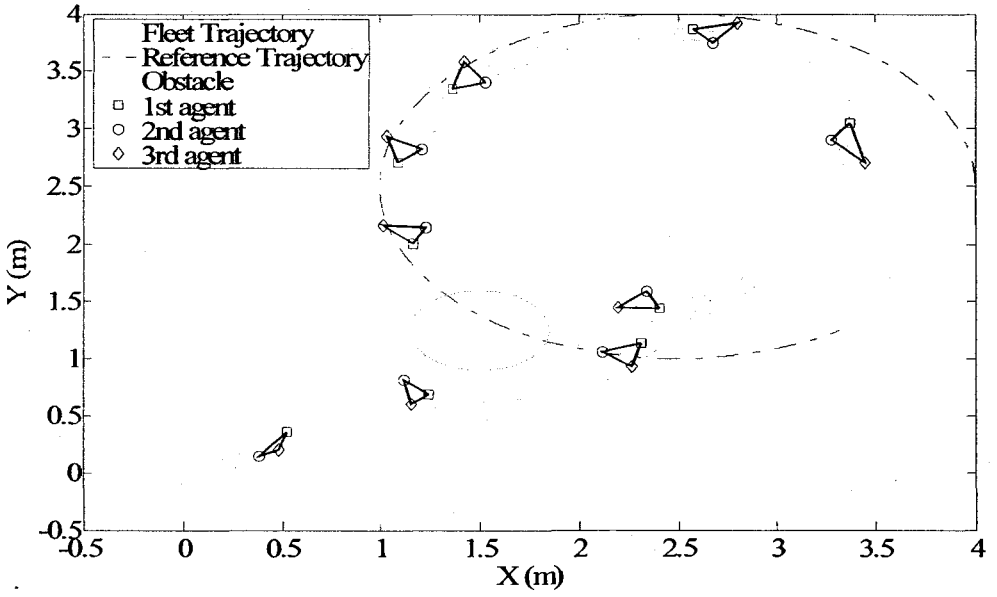

Fig.107. Trajectory following and obstacle avoidance experimental results for single computer case (IC#12).

6.4. Virtual Reality System

The foresaid distributed RHC system is connected to a virtual reality system. The new system is able to render the motion of the simulated hovercrafts. The output of this virtual reality system is shown in Fig.108. Please be advised that this section only contains some main ideas of the structure and mechanism of the system. See [69] for detailed instructions.

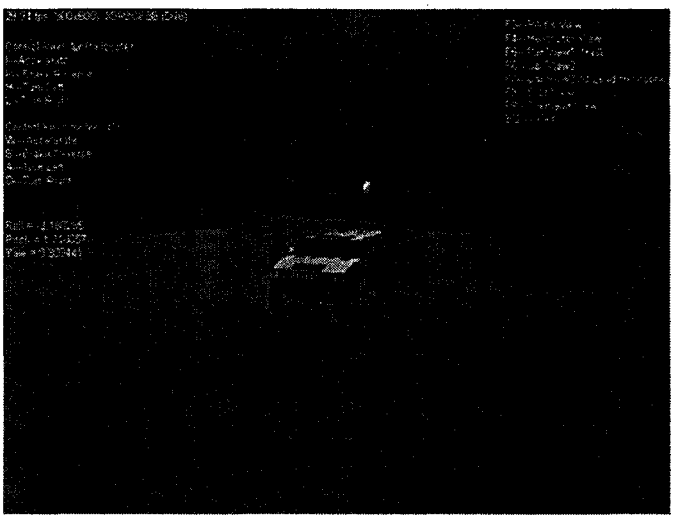

Fig.108. Screenshot of the virtual reality system rendering.

6.4.1. Battlefield Scene

Instead of simply rendering the motion of the hovercrafts, a battlefield scene is created. This scene brings more attractive features to the simulation, and makes the whole simulation much straightforward.

In this scene, the foresaid three hovercrafts are rendered as a fleet of helicopters. This is mainly due to the need of using land vehicles and buildings in the virtual world, and it would be too unusual to have a fleet of hovercrafts moving on land. Also the

structure of the hovercrafts are dynamically identical to helicopters in a two dimensional environment.

While patrolling in the virtual world, the fleet identifies an enemy land vehicle. Subsequently, they chase the enemy in their combat formation and avoid buildings on the way. When the helicopters move in a certain range from the vehicle, the leader of the fleet will launch a missile and shoot the enemy.

6.4.2. Classes Construction

In the field of virtual reality, Object Oriented Programming (OOP) is considered the most effective and suitable programming tool, because of its special way of organizing programs. Unlike the way of organizing data in structured programs, OOP is organized around data, with the principle of "data controlling access to code" [78]. Programs of this virtual reality rendering system are developed to fully take advantage of benefits offered by OOP; as a result, objects can be conveniently added, removed and modified, hence, considerably reducing maintenance costs. Fig.109 illustrates the class structure of the virtual system.

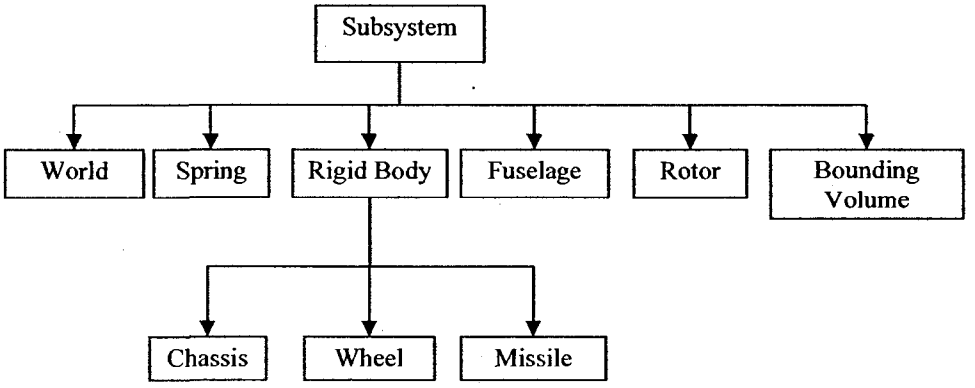

Fig.109. Inheritance of classes in the program.

The class *subsystem* is the base class for others that inherit the *subsystem* either directly or indirectly. The class *subsystem* only defines general variables, and all member functions have a prefix *virtual* which is designed to be overridden by derived classes. Although, the class *world* itself is a derived class of *subsystem*, it has functions similar to a "container" class for other classes that are derived from *subsystem*. The class *rigid body* includes all variables and functions for simulating a 6-DOF rigid body model in 3D environment. State variables involved are translational and rotational velocity, as well as quaternion variables due to their convenience. Note that *rigid body* is also the base class for the class *chassis*, *wheel* and *missile*. The class *chassis* is a derived class from *rigid body*, and it has its own variables which are added for modeling a vehicle dynamics. The class *chassis* has not only features from the *rigid body*, but it also inherits the class *subsystem*. In this way, multiple inheritances are achieved. The same condition can be found in the class *wheel* and *missile*.

6.4.3. Framework of the System

This subsection describes the framework of the virtual system, and delivers the idea about how the program generates a virtual environment. The flowchart in Fig.111 explains the major executions in the program, including both, the major function calls and communication.

In the program, the first step is *Initialization* which runs only once, since all variables defined here would not be changed after the simulation starts progressing. After the system is successfully initialized, *camera setting* module defines view points and

look-at points, and provides a set of perspectives for users who are allowed to freely switch among them as the simulation progresses. The enemy vehicle could not only move in a trajectory which is predefined by the user, but arbitrarily according to the data from *external inputs* module. Next, all the states of major models (the ground vehicle and the helicopters) are updated in the *model dynamic* module at each time step, and the Euler algorithm is applied for updating states.

This system adopts the basic bounding volume method for collision detection. It first automatically generates virtual spheres for each object, and those spheres should cover the object's entire shape. The final *Optimized Mesh Rendering* module loads all the meshes that are needed in the system, and defines some default parameters, such as program window size. Those meshes are not only assigned to visualize their corresponding objects, but they are also processed by a series of optimization algorithms provided by DirectX library.

6.5. Cockpit Simulator

The platform has 6 degree of freedom, 3 translational and 3 rotational motions. A real vehicle seat is mounted on the platform, which is shown in Fig.110. The platform is capable of capturing the full dynamics of the ground vehicle in the virtual reality system as the simulation progresses.

The interface has two routines: one is for the real-time hardware, the other is for graphics and data received from simulation. As the simulation is progressing, the two routines are communicating by the Inter-Process communication via Shared Memory provided by RTSS. By creating Shared Memory object, multiple processes can access the

region of memory with either a handle or a virtual address [62]. In this application, the Euler angles are stored in the shared memory for two routines to access. Thus, it is possible for real-time routine to access memory, and then transforms those angles into voltage signals as input to the hardware.

Fig.110. A driver seat mounted on the 6-DOF platform.

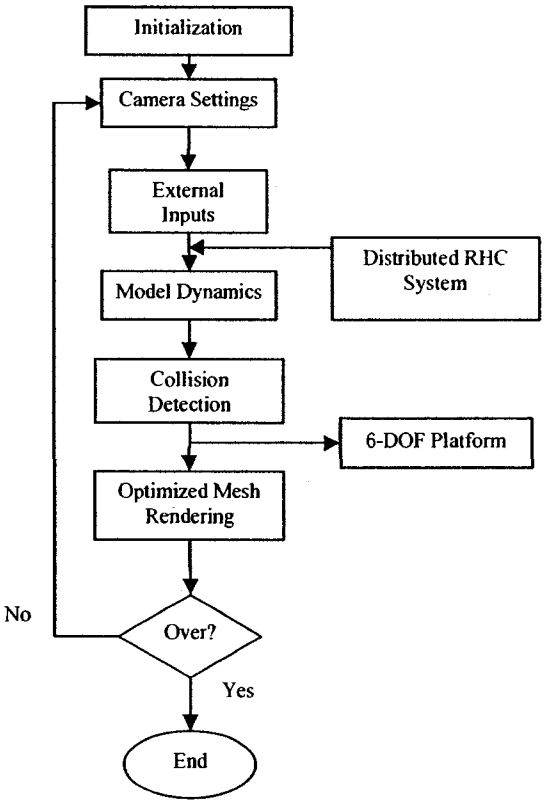

Fig.111. Flow chart for the virtual reality system and cockpit simulator.

7. Conclusions and Future Work

In this thesis, the decentralized receding horizon control method was investigated through numerous simulations and experiments. New algorithms and methods for trajectory following and formation control of multiple vehicle systems are evaluated and compared. Accurate models are developed, experimentally identified, and tested. It was found that the wheeled robot dynamics were best described by a combination of Coloumb and viscous friction, whereas the hovercraft dynamics could be adequately described using only a viscous friction model. Decentralized RHC is then applied to both types of vehicles through simulations and experiments.

A virtual reality simulation system with a 6 DOF cockpit is combined with the experiments to provide a higher level of capability to study more advanced DRHC problems. The results from the simulations and experiments indicate that the decentralized receding horizon control approach is well suited for meeting the requirements of complicated multi-vehicle control problems with low trajectory and formation errors. The combination with the virtual reality system brings more possibilities and scenarios that can be investigated for both civil and military applications.

Together, these results provide a new and useful framework for simulation and experimental testing of new decentralized RHC algorithms and other types of nonlinear control methods for multi-vehicle systems.

Future work includes generalizing the research for vehicles with senor, actuator and communication faults. The future research will also investigate the formulation of appropriate cost functions and predictive models for avoidance and interception of fast moving objects. Furthermore, the experimental apparatus and dynamic models will be

expanded to include more challenging types of vehicles such as model helicopters and robotic fish systems, which move significantly faster with more complex three dimensional dynamics.

References

[1] S. J. Qing and T. A. Badgwell, "An Overview of Industrial Model Predictive Control Technology," in *Chemistry Process Control-V,* vol. 93, no. 316, pp. 232-256, 1996.

[2] A. Bemporad and M. Morari, "Robust Model Predictive Control: A Survey," in *Robustness in identification and control Lecture Notes in Control and Information Sciences*, vol. 245: pp. 207-226, 1999.

[3] D. L. Yu, D. Williams, and J. B. Gomm, "On-line Implementation of a Model Predictive Controller on a Multivariable Chemical Process," *IEE Two-Day Workshop on Model Predictive Control: Techniques and Applications - Day 2 (Ref. No. 1999/096)*, pp. 2/1 – 2/5, April 1999.

[4] J. B. Rawling, "Tutorial: Model Predictive Control Technology," in *Proc. of the 1999 American Control Conference*, vol. 1, pp. 662-676, June 1999.

[5] C. E. Garcia, D. M. Prett, and M. Morari, "Model Predictive Control: Theory and Practice – a Survey," *Automatica*, vol. 25, no. 3, pp. 335–348, May 1989.

[6] P. Grieder, P. A. Parrilo, and M. Morari, "Robust Receding Horizon Control – Analysis & Synthesis," in *Proc. of 42nd IEEE Conference on Decision and Control*, vol.1, pp.941-946, December 2003.

[7] A. Azimi, B. Gholami, and B. W. Gordon, "Synthesis and implementation of single- and multi-vehicle systems guidance based on nonlinear control and optimization techniques," *Technical Report*, CIS Lab, Concordia University, 2006.

[8] D. E. Quevedo, J. A. De Dona, and G. C. Goodwin, "On the Dynamics of Receding Horizon Linear Quadratic Finite Alphabet control Loops," in *Proc. of the 41st IEEE Conference on Decision and Control*, vol. 3, pp. 2929–2934, December 2002.

[9] K. H. Lee, W. H. Kwon, and J. H. Lee, "Robust Receding Horizon Control for Linear Systems with Model Uncertainties," in *Proc. of the 35th IEEE*

Conference on Decision and Control, vol. 4, pp. 4002–4007, December 1996.

[10] H. Michalska and D. Q. Mayne, "Robust Receding Horizon Control of Constrained Nonlinear Systems," *IEEE Trans. Automatic Control*, vol. 38, no. 11, pp.1623–1633, November 1993.

[11] D. Q. Mayne and H. Michalska, "An Implementable Receding Horizon Controller For Stabilization of nonlinear Systems," in *Proc. of the 29th IEEE Conference on Decision and Control*, vol. 6, pp.3396–3397, December 1990.

[12] D. Q. Mayne and H. Michalska, "Receding Horizon Control of Nonlinear Systems," *IEEE Trans. Automatic Control*, vol. 35, no. 7, pp. 814–824, July 1990.

[13] B. Moerdyk, R. DeCarlo, D. Birdwell, M. Zefran, and J. Chiasson, "Hybrid Optimal Control for Load Balancing in a Cluster of Computer Nodes," in *Proc. of the 2006 IEEE International Conference on Control Applications,* pp. 1713–1718, October 2006.

[14] H. S. Chang and S. I. Marcus, "Receding Horizon Approach to Markov Games for Infinite Horizon Discounted Cost," in *Proc. of the 41st IEEE Conference on Decision and Control*, vol. 2, pp. 1380–1385, December 2002.

[15] C. Scheel and B. McInnis, "Parallel Processing of Optimal Control Problems by Dynamic Programming," *Information Sciences*, vol. 25, no.2, pp. 85-114, November 1981.

[16] T. -S. Chang, X. -X. Jin, P. B. Luh, and X. Miao, "Large-Scale Convex Optimal Control Problems: Time Decomposition, Incentive Coordination, and Parallel Algorithm," *IEEE Trans. Automatic Control*, vol. 35, no. 1, pp.108–114, January 1990.

[17] S. -C. Chang, T. -S. Chang, and P. B. Luh, "A Hierarchical Decomposition for Large-Scale Optimal Control Problems with Parallel Processing Structure," *Automatica*, vol. 25, no. 1, pp. 77–86, January 1989.

[18] S. -Y. Lin, "A Hardware Implementable Receding Horizon Controller for Constrained Nonlinear Systems," *IEEE Trans. Automatic Control*, vol. 39,

no. 9, pp. 1893–1899, September 1994.

[19] S. -Y. Lin, "A Hardware Implementable Two-Level Parallel Computing Algorithm for General Minimum-Time Control," *IEEE Trans. Automatic Control*, vol. 37, no. 5, pp.589–603, May 1992.

[20] M. Diehl, R. Findeisen, F. Allgower, H. G. bock, and J. P. Schloder, "Nominal Stability of Real-Time Iteration Scheme for Nonlinear Model Predictive Control," in *IEE Proc. of Control Theory and Applications*, vol. 152, pp. 296–308, May 2005.

[21] M. B. Milam, K. Mushambi, and R. M. Murray, "A New Computational Approach to Real-Time Trajectory Generation for Constrained Mechanical Systems," in *Proc. of the 39th IEEE Conference on Decision and Control*, vol. 1, pp. 845–851, December 2000.

[22] Y. Chen, Y. Zhuang, and W. Wang, "Cooperative Control for Formations of Mobile Robots under the Nonholonomic Constraints," in *Proc. of the 6th World Congress on Intelligent Control and Automation*, vol. 2, pp. 9042–9046, June 2006.

[23] N. Kohata, T. Yamaguchi, M. Takahide, T. Baba, and H. Hashimoto, "Dynamic Formation on Mobile Agents and its Evolutionary Parallel Computation," in *Proc. of IEEE Conference on Systems, Man and Cybernetics*, vol. 1, pp. 272–277, October 1999.

[24] Dan Henriksson and Johan Akesson, "Flexible Implementation of Model Predictive Control Using Sub-Optimal Solution," *Technical Report ISRN LUTFD2/TFRT- -7610- -SE.,* Department of Automatic Control, Lund Institute of Technology, 2004.

[25] Y. H. Dai and K. Schittkowski, "A Sequential Quadratic programming Algorithm with Non-Monotone Line Search," *Pacific Journal of Optimization,* vol. 4, pp. 335-351, 2005.

[26] K. Schittkowski, "NLPQLP: A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line Search - User's guide, Version 2.2," *Report*, Department of Computer Science, University of Bayreuth, 2006.

[27] J. -P. Richard, "Time-delay systems: an overview of some recent advances and open problems," *Automatica*, vol. 39, no. 10, pp. 1667-1694, October 2003.

[28] M. G. Earl and R. D'Andrea, "A decomposition approach to multi-vehicle cooperative control," *Robotics and Autonomous Systems*, vol. 55, no. 4, pp. 276-291, April 2007.

[29] M. Mercangoz and F. J. Doyle III, "Distributed model predictive control of an experimental four-tank system," *Journal of Process Control*, vol. 17, no. 3, pp. 297-308, March 2007.

[30] S. D. Canto, A. P. de Madrid, and S. Dormido, "Dynamic Programming on Clusters for Solving Control Problems," *4th Asian Control Conference*, 2002.

[31] G. Hassapis, "Implementation of model predictive control using real-time multiprocessing computing," *Microprocessors and Microsystems*, vol. 27, no. 7, pp. 327-340, August 2003.

[32] B. -I. Koh, A. D. George, R. T. Haftka, and B. J. Fregly, "Parallel asynchronous particle swarm optimization," *International Journal for Numerical Methods in Engineering*, vol. 67, no. 4, pp. 578–595, July 2006.

[33] E. V. Adutskevich and N. A. Likhoded, "Optimization of Data Exchange in Parallel Computers with Distributed Memory," *Cybernetics and Systems Analysis*, vol. 42, no. 2, pp. 298–310, March 2006.

[34] J. -P. Jiang, P. L. Ho, and J. C. Hinton, "Parallel Processing in Optimal Control of Robot Drives," *International Conference on Control*, vol. 1, no. 332, pp. 376–381, March 1991.

[35] N. Jin and Y. Rahmat-Samii, "Parallel Particle Swarm Optimization and Finite-difference Time-Domain (PSO/FDTD) Algorithm for Multiband and Wide-band Patch Antenna Designs," *IEEE Trans. Antennas and Propagation*, vol. 53, no. 11, pp. 3459–3468, November 2005.

[36] X. -B. Hu and W. -H. Chen, "Model predictive control for constrained systems with uncertain state-delays," *International Journal of Robust and Nonlinear Control*, vol. 14, no. 17, pp. 1421-1432, November 2004.

[37] M. J. Tenny, S. J. Wright, and J. B. Rawlings, "Nonlinear Model Predictive Control via Feasibility – Perturbed Sequential Quadratic Programming," *Computational Optimization and Applications*, vol. 28, no. 1, pp. 87-121, April 2004.

[38] A. Jadbabaie, J. Yu, and J. Hauser, "Stabilizing Receding Horizon Control of Nonlinear Systems: A Control Lyapunov Function Approach," in *Proc. of the American Control Conference*, vol. 3, pp. 1535-1539, June 1999.

[39] F. Borrelli, P. Falcone, and C. Del Vecchio, "Event-based receding horizon control for two-stage multi-product production plants," *Control Engineering Practice*, vol. 15, no. 12, pp. 1556-1568, December 2007.

[40] H. Sarimveis, P. Patrinos, C. D. Tarantilis, and C. T. Kiranoudis, "Dynamic modeling and control of supply chain systems: A review," *Computers and Operations Research*, vol. 35, no. 11, pp. 3530-3561, November 2008.

[41] G. C. Goodwin, M. M. Seron, R. H. Middleton, M. Zhang, B. F. Hennessy, P. M. Stone, and M. Menabde, "Receding horizon control applied to optimal mine planning," *Automatica*, vol. 42, no. 8, pp. 1337-1342, August 2006.

[42] D. Q. Mayne and H. Michalska, "Adaptive Receding Horizon Control For Constrained Nonlinear Systems," in *Proc. of the 32nd IEEE Conference on Decision and Control*, vol. 2, pp. 1286-1291, December 1993.

[43] A. Azimi, B. W. Gordon, and C. A. Rabbath, "Dynamic Scheduling of Decentralized Receding Horizon Controllers on Concurrent Processors for the Cooperative Control of Unmanned Systems," in *Proc. of the 46th IEEE Conference on Decision and Control*, pp. 518-523, December 2007.

[44] A. Azimi, B. W. Gordon, and C. A. Rabbath, "Dynamic Scheduling of Receding Horizon Controllers with Application to Multiple Unmanned Hovercraft Systems," *American Control Conference*, pp. 3324-3329, July 2007.

[45] E. Camponogara, D. Jia, B. H. Krogh, and S. Talukdar, "Distributed Model Predictive Control," *IEEE Control Systems Magazine*, vol. 22, no. 1, pp.44-52, February 2002.

[46] N. Motee and B. Sayyar-Rodsari, "Optimal Partitioning in Distributed Model

Predictive Control," in *Proc. of the 2003 American Control Conference*, vol. 6, pp.5300-5305, June 2003.

[47] D. Jia and B. H. Krogh, "Distributed Model Predictive Control," in *Proc. of the American Control Conference*, vol. 4, pp. 2767-2772, June 2001.

[48] W. B. Dunbar, "A Distributed Receding Horizon Control Algorithm for Dynamically Coupled nonlinear Systems," in *Proc. of the 44th IEEE Conference on Decision and Control and the 2005 European Control Conference,* pp. 6673-6679, December 2005.

[49] A. N. Venkat, J. B. Rawlings, and S. J. Wright, "Stability and optimality of distributed model predictive control," in *Proc. of the 44th IEEE Conference on Decision and Control and the 2005 European Control Conference*, pp. 6680-6685, December 2005.

[50] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 4, pp.549-558, April 2006.

[51] W. B. Dunbar and R. M. Murray, "Receding Horizon Control of Multi-Vehicle Formations: A Distributed Implementation," in *Proc. of the 33rd IEEE Conference on Decision and Control*, vol. 2, pp. 1995–2002; December 2004.

[52] X. Du, Y. Xi, and S. Li, "Distributed Model Predictive Control for Large-Scale Systems," in *Proc. of the American Control Conference*, vol. 4, pp. 3142-3143, June 2001.

[53] M. B. Milam, R. Franz, J. H. Hauser, and R. M. Murray, "Receding Horizon Control of Vectored Thrust Flight Experiment," *IEE Proc. of Control Theory and Applications*, vol. 152, no. 3, pp. 340-348, May 2005.

[54] L. Singh and J. Fuller, "Trajectory generation for a UAV in urban terrain, using nonlinear MPC," in *Proc. of American Control Conference*, vol. 3, pp. 2301-2308, June 2001.

[55] W. L. George and J. Scott, "ScreenSaver Science: Realizing Distributed Parallel Computing with Jini and JavaSpaces," presented in *11th Conference on Parallel Architectures and Compilation Techniques*, September 2002.

[56] J. Lu, "Design of Ethernet Based Real-time Distributed Systems," M.A.Sc thesis, Dept. Mech. & Ind. Eng., Concordia University, Montreal, Canada, 2004.

[57] M. J. Flynn, *Computer Architecture: Pipelined and Parallel Processor Design,* 1st ed.. Boston, MA: Jones and Bartlett, 1995, ch. 3.

[58] Y. Censor and S. A. Zenios, *Parallel Optimization: Theory, Algorithms, and Applications*. New York: Oxford, 1997, ch. 1.

[59] M. J. Donahoo and K. L. Calvert, *TCP/IP Sockets in C: Practical Guide for Programmers*. San Francisco, CA: Morgan Kaufmann, 2001, pp. 35-40.

[60] T. Keviczky, F. Borrelli, and G. J. Balas, "Decentralized receding horizon control for large scale dynamically decoupled systems," *Automatica*, vol. 42, no. 12, pp. 2105-2115, December 2006.

[61] T. Keviczky, "Decentralized Receding Horizon Control of Large Scale Dynamically Decoupled Systems," Ph.D. dissertation, Control Science and Dynamical Systems Center, Univ. of Minnesota, Twin Cities, USA, 2005.

[62] RTX 5.0 User's Guide, VenturCom Inc..

[63] W. H. Kwon, J. W. Kang, Y. S. Lee, and Y. S. Moon, "A simple receding horizon control for state delayed systems and its stability criterion," *Journal of Process Control*, vol. 13, no. 6, pp. 539-551, September 2003.

[64] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789-814, June 2000.

[65] H. Chen and F. Allgower, "A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability," *Automatica*, vol. 34, no. 10, pp. 1205-1217, October 1998.

[66] W. B. Dunbar and R. M. Murray, "Distributed receding horizon control for multi-vehicle formation stabilization," *Automatica*, vol. 42, no. 6, pp.549-558, April 2006.

[67] A. Regmi, R. Sandoval, R. Byrne, H. Tanner, and C. T. Abdallah, "Experimental Implementation of Flocking Algorithms in Wheeled Mobile Robots," in *Proc. of the American Control Conference*, vol. 7, pp.4917-4922,

June 2005.

[68] A. Azimi, B. Gholami, and B. W. Gordon, "Real-time Scheduling of Multiple Uncertain Receding Horizon Control Systems," *Technical Report*, CIS Lab, Concordia University, 2006.

[69] Y. Zhao, L. Bai, and B. Gordon, "Distributed Simulation and Virtual Reality Visualization of Multi-Robot Distributed Receding Horizon Control Systems," *IEEE International Conference on Robotics and Biomimetics*, pp. 1290-1295, December 2007.

[70] T. Keviczky, F. Borrelli, and G. J. Balas, "Stability Analysis of Decentralized RHC for the decoupled systems," *44th IEEE Conference on Decision and Control and the European Control Conference*, pp. 1689-1694, December 2005.

[71] A. Richards and J. How, "A decentralized algorithm for robust constrained model predictive control," *Proc. of American Control Conference*, vol. 5, pp. 4261-4266, July 2004.

[72] H. A. Izadi, B. W. Gordon, and C. A. Rabbath, "A variable communication approach for decentralized receding horizon control of multi-vehicle systems," *American Control Conference*, pp. 1781-1786, July 2007.

[73] H. Izadi, M. Pakmehr, and B. Gordon, "A receding horizon control approach for roll control of delta wing vortex-coupled dynamics," *2007 IEEE Aerospace Conference*, pp. 3377-3383, March 2007.

[74] A. Azimi, B. W. Gordon, and C. A. Rabbath, "Dynamic scheduling of multiple RHC systems with coupling and computational delay," *American Control Conference*, to be published.

[75] H. Izadi, B. W. Gordon, and C. A. Rabbath, "Decentralized control of multiple vehicles with limited communication bandwidth," *2008 IEEE International Conference on Systems, Man and Cybernetics*, to be published.

[76] H. Izadi, B. W. Gordon, and C. A. Rabbath, "Stability improvement for time varying decentralized receding horizon control systems," *13th IEEE IFAC International Conference on Methods and Models in Automation and Robotics*, to be published.

[77] M. Milam, "Real-time optimal trajectory generation for constrained dynamical system," PhD dissertation, Control and Dynamical Systems, California Inst. of Tech., Pasadena, USA, 2003.

[78] H. Schildt, *C++, The Complete Reference, 4th ed.* New York: McGraw-Hill, 2003.

[79] B. Gholami, "Receding Horizon Control of Uncertain Systems," M.A.Sc thesis, Dept. Mech. & Ind. Eng., Concordia University, Montreal, Canada, 2005.

[80] F.J. Lian and R. Murray, "Cooperative task planning of multi-robot systems with temporal constraints," in *Proc. of 2003 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 2504-2509, September 2003.

[81] B. Gholami, B. W. Gordon, and C. A. Rabbath, "Uncertain nonlinear receding horizon control systems subject to non-zero computation time," *44th IEEE Conference on Decision and Control, and the 2005 European Control Conference*, pp. 3765-3770, December 2005.

[82] J. Strizzi, I. M. Ross, and F. Fahroo, "Towards real-time computation of optimal controls for nonlinear systems," *AIAA Guidance, Navigation, and Control Conference and Exhibit*, pp. 1967-1976, August 2002.

[83] R. M. Murray, *Lecture Note on RHC Analysis*, California Institute of Technology, available: http://www.cds.caltech.edu/~murray/courses/cds110/wi08/L4-2_rhc.pdf

[84] B. Gholami, B. W. Gordon, and C. A. Rabbath, "Real-time scheduling of multiple uncertain receding horizon control systems," *Optimal Control Applications and Methods*, submitted for publication.

[85] A. E. Bryson and Y. -C. Ho, *Applied optimal control, optimization, estimation and control.* Washington: Hemisphere Pub. Corp, 1975.

[86] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C: the Art of Scientific Computing.* Cambridge [England]: Cambridge University Press, 1993.

[87] S. E. Gill, W. Murray, and M. A. Saunders, "User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming".

[88] G. Oriolo, A. De Luca, and M. Vendittelli, "WMR control via dynamic feedback linearization: design, implementation and experimental validation," *IEEE Trans. Control Systems Technology*, vol. 10, no. 6, pp.835–852, November 2002.

[89] A. De Luca, G. Oriolo, M. Vendittelli, "Control of wheeled mobile robots: An experimental overview," in *RAMSETE - Articulated and Mobile Robotics for Services and Technologies*, Springer-Verlag, 2001